HITACHI
Inspire the Next

ノンストップデータベース

HiRDB Version 10 システム導入・設計ガイド (UNIX(R)用)

解説・手引・操作書

3020-6-552-70

前書き

■ 対象製品

●適用 OS：AIX V7.1，AIX V7.2，AIX V7.3

P-1M62-35A1 HiRDB Server Version 10 10-07

P-1M62-1BA1 HiRDB/Run Time Version 10 10-07

P-1M62-1CA1 HiRDB/Developer's Kit Version 10 10-07

P-1M62-1DA1 HiRDB/Run Time Version 10(64) 10-07

P-1M62-1EA1 HiRDB/Developer's Kit Version 10(64) 10-07

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 7 (64-bit x86_64)，Red Hat Enterprise Linux 8 (64-bit x86_64)，Red Hat Enterprise Linux 9 (64-bit x86_64)

P-8462-35A1 HiRDB Server Version 10 10-07

P-8462-1DA1 HiRDB/Run Time Version 10(64) 10-07

P-8462-1EA1 HiRDB/Developer's Kit Version 10(64) 10-07

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 7 (64-bit x86_64)，Red Hat Enterprise Linux 8 (64-bit x86_64)，Red Hat Enterprise Linux 9 (64-bit x86_64)

P-8362-1BA1 HiRDB/Run Time Version 10 10-07

P-8362-1CA1 HiRDB/Developer's Kit Version 10 10-07

P-8362-3CA1 HiRDB Developer's Suite Version 10 10-07

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、外国為替及び外国貿易法の規制並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

HITACHI，HiRDB，Cosminexus，DABroker，DBPARTNER，HA モニタ，JP1，OpenTP1，TPBroker，uCosminexus，VOS3/LS，VOS3/US，VOS3/XS，XDM は，株式会社 日立製作所の商標または登録商標です。

Amazon Web Services，AWS，Powered by AWS ロゴ，Amazon EC2，Amazon Route 53 は，Amazon.com, Inc.またはその関連会社の商標です。

AMD は，Advanced Micro Devices, Inc.の商標です。

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

Hibernate は，米国およびその他の国における Red Hat, Inc.の登録商標です。

IBM，AIX，DataStage および PowerHA は，世界の多くの国で登録された International Business Machines Corporation の商標です。

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

Jboss は，米国およびその他の国における Red Hat, Inc.の登録商標です。

Linux は，Linus Torvalds 氏の米国およびその他の国における登録商標です。

Microsoft，Access，ActiveX，Azure，Excel，Visual Basic，Visual C++，Visual Studio，Windows，Windows Server は，マイクロソフト 企業グループの商標です。

Oracle(R)，Java 及び MySQL は，Oracle，その子会社及び関連会社の米国及びその他の国における登録商標です。

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

Red Hat は，米国およびその他の国における Red Hat, Inc.の登録商標です。

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

Red Hat Enterprise Linux は，米国およびその他の国における Red Hat, Inc.の登録商標です。

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

RHEL は，米国およびその他の国における Red Hat, Inc.の商標または登録商標です。

UNIX は，The Open Group の登録商標です。

Veritas および Veritas ロゴは，米国およびその他の国における Veritas Technologies LLC またはその関連会社の商標または登録商標です。

その他記載の会社名，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

2023 年 7 月　3020-6-552-70

変更内容

変更内容(3020-6-552-70)　HiRDB Version 10 10-07

追加・変更内容	変更個所
1 バックエンドサーバ当たりの最大起動プロセス数（pd_max_bes_process），及びユニット内の最大同時起動サーバプロセス数（pd_max_server_process）の上限値を拡大しました。これによって，構成の設計の自由度を高めることができます。	1.3.2(2)
クライアントーサーバ間の接続で新たな接続方式をサポートしました。これによって，クライアント側ファイアウォールでポート開放が不要となるほか，NAPT が設定されたネットワーク環境で接続できるようになります。	1.3.2(10)，1.3.2(11)，21.3.1，21.3.2，21.3.3，21.3.4
統計解析ユティリティ（pdstedit）で使用するファイルの容量の見積もり式を変更しました。	18.1.3
HiRDB の適用 OS に次の OS を追加しました。 • AIX V7.3 • Red Hat Enterprise Linux 9	－

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(3020-6-552-60)　HiRDB Version 10 10-06

追加・変更内容
データベース暗号化機能の暗号化表および暗号化列を使用している HiRDB をバージョンアップする際の注意事項を追加しました。
HiRDB/パラレルサーバの各ユニットが使用するメモリ所要量の計算式と変数の値を変更しました。
通常のデータディクショナリ用 RD エリアの容量の見積もり式を変更しました。
監査証跡ファイル用の HiRDB ファイルシステムの容量の見積もり式を変更しました。
統計解析ユティリティ（pdstedit）で使用するファイルの容量の見積もり式を変更しました。
Linux のオペレーティングシステムパラメタの見積もり式を変更しました。
HP-UX に関する説明を削除しました。

変更内容(3020-6-552-50)　HiRDB Version 10 10-05

追加・変更内容
表データ更新時に出力されるシステムログ量の，インデクスログ量の計算式に set pd_unique_check_mode の指定値ごとの見積もり式を追加しました。
セキュアシェルを使用する場合の見積もりに，次のパラメタの指定値の目安を追加しました。 • PerSourceMaxStartups

追加・変更内容
• ChrootDirectory
HiRDB の適用 OS に Windows Server 2022 を追加しました。

変更内容(3020-6-552-40)　HiRDB Version 10 10-04

追加・変更内容
これまで通信情報ファイルディレクトリはルートファイルシステム（/dev）下の固定のパスでしたが，任意のパスに変更できるようになりました。 これにより，通信情報ファイルディレクトリを/dev 以外に作成することで，ルートファイルシステムへの負荷を減らすことができます。 また，マルチ HiRDB 構成では，各 HiRDB システムで異なる HiRDB 管理者がそれぞれの通信情報ファイルディレクトリを管理することができます。
HiRDB/シングルサーバの，再開始用メモリサイズを求める計算に使用する，変数の値を変更しました。
HiRDB/シングルサーバの，サーバプロセスが使用するプロセス固有領域のメモリ所要量に，pd_work_buff_mode=pool2 を指定したときのメモリ所要量の計算式を追加しました。
HiRDB/シングルサーバの，ユニットコントローラが使用する共用メモリの計算式を変更しました。
HiRDB/パラレルサーバの，再開始用メモリサイズを求める計算に使用する，変数の値を変更しました。
HiRDB/パラレルサーバの，サーバプロセスが使用するプロセス固有領域のメモリ所要量に，pd_work_buff_mode=pool2 を指定したときのメモリ所要量の計算式を追加しました。
HiRDB/パラレルサーバの，ユニットコントローラが使用する共用メモリの計算式を変更しました。
AIX のオペレーティングシステムパラメタの，stack_hard の指定値の目安を変更しました。
Linux のオペレーティングシステムパラメタの，SEMMSL の指定値の目安を変更しました。
OS のオペレーティングシステムパラメタの見積もりに，セキュアシェルを使用する場合の見積もりを追加しました。

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 のシステムの構築方法，データベースの作成方法及びシステムとデータベースの設計方法について説明したものです。なお，ここに記載されていない前提情報については，マニュアル「HiRDB Version 10 解説」を参照してください。

■ 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／運用する方々を対象にしています。

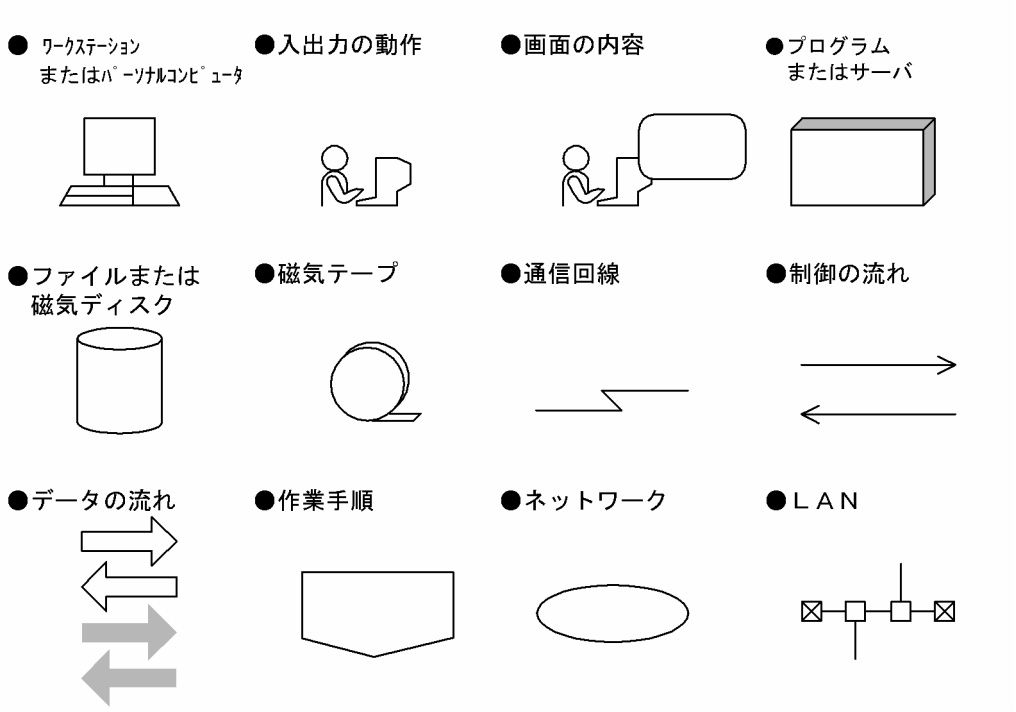
このマニュアルは次に示す知識があることを前提に説明しています。

- UNIX，又は Linux のシステム管理の基礎的な知識
- SQL の基礎的な知識

また，このマニュアルは，マニュアル「HiRDB Version 10 解説」を前提としていますので，あらかじめお読みいただくことをお勧めします。

■ 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。



目次

付録 1040

1 HiRDB のシステム構築の概要

この章では，HiRDB のシステム構築手順，HiRDB のファイル構成及びバージョンアップ手順について説明します。

1.1　オペランド省略時動作の概要

HiRDB では，HiRDB のバージョン，リビジョンごとに 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，及び SQL のオプションの省略値を見直して変更しています。省略値の変更に伴い，バージョン 09-50 以降，オペランド省略時動作として，推奨値を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提供しています。通常は，より安全なシステムを構築するために，指定が必要なオペランド数を大幅に削減した推奨モードの適用を検討してください。

09-50 より前のバージョンからバージョンアップを行う場合は，省略値変更によるメリット及びデメリットについて，「HiRDB システム定義のオペランド省略値の確認」，及び「そのほかの省略値の確認」で確認してください。確認の結果，旧バージョンとの互換性を重視する場合は，旧バージョンと同等の省略値になる互換モードを適用してください。ただし，この場合はすべてのオペランドが旧バージョンの省略値となりますので，各オペランドに推奨値を指定することを検討してください。

修正版 HiRDB への入れ替えを行う場合は，既に適用しているオペランド省略時動作を適用してください。

オペランド省略時動作は，HiRDB のセットアップで選択できます。また，pdsetenv コマンドで変更できます。

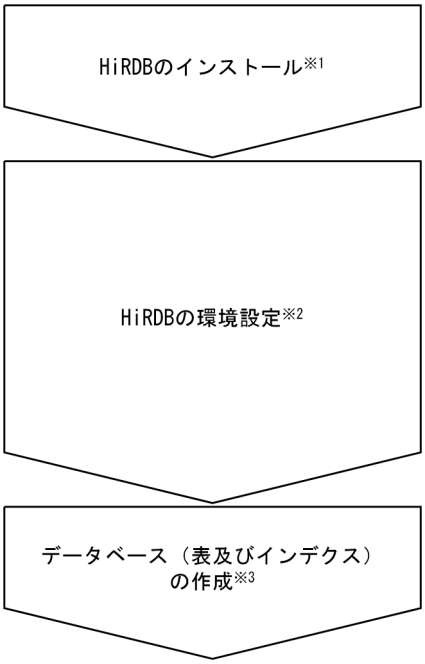
1.2　システム構築手順

ここでは，HiRDB を新規導入するときのシステム構築手順について説明します。

1.2.1　HiRDB を新規導入するときのシステム構築手順

HiRDB を新規導入するときのシステム構築手順を次の図に示します。

図 1-1　HiRDB を新規導入するときのシステム構築手順



注※1

手順については，「インストール」を参照してください。

注※2

HiRDB の環境設定は次に示す方法で実施します。

- コマンドを使用する方法

環境設定方法	概要
コマンドを使用する方法	HiRDB のコマンドを使用して HiRDB の環境設定を行います。 コマンドを使用する方法については，「コマンドによる環境設定」を参照してください。

注※3

手順については，「データベースの作成」を参照してください。

なお，HiRDB を 24 時間連続稼働するときにお勧めする運用方法，及び注意事項について，マニュアル「HiRDB システム運用ガイド」で説明しています。必要に応じて参照してください。

1.2.2　HiRDB の環境設定の概要

HiRDB 管理者は，コマンドを使用して HiRDB の環境設定をしてください。コマンドを使用して HiRDB の環境設定を行う方法については，「コマンドによる環境設定」を参照してください。なお，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。

1.2.3　ほかの製品と連携する場合の環境設定

ほかの製品と連携する場合の環境設定を次に示します。

(1)　レプリケーション機能を使用する場合

レプリケーション機能を使用するには，HiRDB Datareplicator，HiRDB Dataextractor が必要になります。レプリケーション機能の環境設定方法については，「レプリケーション機能との連携」を参照してください。

(2)　OLTP と連携する場合

OLTP と連携する場合の環境設定方法については，「OLTP との連携」を参照してください。

(3)　系切り替え機能を使用する場合

系切り替え機能を使用する場合は，クラスタソフトウェアが必要になります。クラスタソフトウェアはプラットフォームごとに異なります。クラスタソフトウェア，及び系切り替え機能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。

(4)　インナレプリカ機能を使用する場合

インナレプリカ機能を使用するには，HiRDB Staticizer Option が必要になります。環境設定方法については，「インナレプリカ機能との連携」を参照してください。

(5)　リアルタイム SAN レプリケーションを使用する場合（ディザスタリカバリ）

ログ同期方式のリアルタイム SAN レプリケーションを使用するには HiRDB Disaster Recovery Light Edition が必要になります。

リアルタイム SAN レプリケーションについては，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。

1.3　HiRDB のディレクトリ及びファイル構成

1.3.1　最初に作成するファイル

(1)　HiRDB 管理者が作成するディレクトリ及びファイル

HiRDB 管理者が作成するディレクトリ及びファイル構成を次の表に示します。

表 1–1　HiRDB 管理者が作成するディレクトリ及びファイル構成

ファイル又はディレクトリ名	説明
$PDDIR/conf/pdsys	システム共通定義を格納するファイル
$PDDIR/conf/pdutsys	ユニット制御情報定義を格納するファイル
$PDDIR/conf/pdsvrc	サーバ共通定義を格納するファイル
$PDDIR/conf/サーバ名	各サーバ定義を格納するファイル
$PDDIR/conf/pduapenv	UAP 環境定義を格納するディレクトリ
$PDDIR/conf/chgconf	システム構成変更用定義ファイルを格納するディレクトリ
任意※	通信情報ファイルを格納するディレクトリ

注※

pd_ipc_file_dir オペランドでディレクトリのパス名を指定します。

(2)　HiRDB が作成するディレクトリ及びファイル

HiRDB が作成するディレクトリ及びファイル構成を次の表に示します。

表 1–2　HiRDB が作成するディレクトリ及びファイル構成

ファイル又はディレクトリ名	説明
$PDDIR/bin	HiRDB のコマンド及びユティリティを格納するディレクトリ
$PDDIR/lib	HiRDB の共用ライブラリ，及びメッセージテキストファイルを格納するディレクトリ
$PDDIR/lib/sysconf	HiRDB システム定義の定義解析用ファイルを格納するディレクトリ
$PDDIR/lib/sysdef	
$PD DIR/lib/sysdef_r	
$PDDIR/lib/sysdef_v94	
$PDDIR/lib/servers	HiRDB サーバの実行ファイル及びライブラリを格納するディレクトリ

ファイル又はディレクトリ名	説明
$PDDIR/lib/save	pdmemsv コマンドでライブラリを共用化した場合に内容を保存しておくディレクトリ
$PDDIR/lib/chinese	EUC 中国語漢字コードパーサライブラリを格納するディレクトリ
$PDDIR/lib/chinese-gb18030	中国語漢字コードパーサライブラリを格納するディレクトリ
$PDDIR/lib/lang-c	単一バイト文字コードパーサライブラリを格納するディレクトリ
$PDDIR/lib/sjis	シフト JIS 漢字コードパーサライブラリを格納するディレクトリ
$PDDIR/lib/ujis	EUC 日本語漢字コードパーサライブラリを格納するディレクトリ
$PDDIR/lib/utf-8	Unicode（UTF-8）パーサライブラリを格納するディレクトリ
$PDDIR/lib/utf-8_ivs	Unicode（IVS 対応 UTF-8）パーサライブラリを格納するディレクトリ
$PDDIR/client/lib	HiRDB クライアントのライブラリを格納するディレクトリ
$PDDIR/client/xds/lib	バージョン 09-50 より前の HiRDB との互換のためのディレクトリ
$PDDIR/client/utl	HiRDB クライアントのコマンド及びユティリティを格納するディレクトリ
$PDDIR/include	UAP を作成するときに使用するヘッダ情報を格納するディレクトリ
$PDDIR/spool	HiRDB の作業ファイルを格納するディレクトリ
$PDDIR/spool/save※1	退避コアファイル，及びサーバプロセス異常終了情報ファイルを格納するディレクトリ
$PDDIR/spool/pdshmdump※1	共用メモリダンプファイルを格納するディレクトリ
$PDDIR/spool/pdlckinf※1	デッドロック・タイムアウト情報ファイル，及び排他資源管理テーブル情報ファイルを格納するディレクトリ
$PDDIR/spool/pdsysdump※1	システム共通の簡易ダンプファイルを格納するディレクトリ
$PDDIR/spool/pdsdsdump※1	シングルサーバ用の簡易ダンプファイルを格納するディレクトリ
$PDDIR/spool/pdfesdump※1	フロントエンドサーバ用の簡易ダンプファイルを格納するディレクトリ
$PDDIR/spool/pddicdump※1	ディクショナリサーバ用の簡易ダンプファイルを格納するディレクトリ
$PDDIR/spool/pdbesdump※1	バックエンドサーバ用の簡易ダンプファイルを格納するディレクトリ
$PDDIR/spool/pdstj1，pdstj2	統計ログファイル
$PDDIR/spool/pdlog1，pdlog2	メッセージログファイル
$PDDIR/spool/pdjnlinf	システムログ情報出力ファイルを格納するディレクトリ
$PDDIR/spool/pdjnlinf/errinf	システムログエラー情報出力ファイルを格納するディレクトリ
$PDDIR/spool/scdqid1，scdqid2	HiRDB 内部のスケジュールキュー情報を格納するファイル
$PDDIR/spool/oslmqid	メッセージキューの ID を格納するファイル
$PDDIR/spool/oslsmid	セマフォの ID を格納するファイル

ファイル又はディレクトリ名	説明
$PDDIR/spool/pdprcsts	prc ステータスファイル
$PDDIR/spool/.pdatmode	開始・終了用ステータスファイル
$PDDIR/spool/.pdipcid	セマフォの ID を管理するファイル
$PDDIR/spool/.pdommenv	共用メモリの情報を格納するファイル
$PDDIR/spool/cmdlog/cmdlog1，cmdlog2	実行したコマンドの履歴ファイル
$PDDIR/spool/cmdlog/cmdlogr1，cmdlogr2	リモートシェル／セキュアシェルの実行情報を格納するファイル
$PDDIR/spool/errlog/errlog1，errlog2	HiRDB の内部稼働履歴ファイル
$PDDIR/spool/olkfifs	スレッドロック用パイプファイルを格納するディレクトリ※7
$PDDIR/spool/olkrsfs	スレッドサスペンド／リジューム用パイプファイルを格納するディレクトリ※8
$PDDIR/spool/oslcntl	パイプファイル数管理ファイル
$PDDIR/spool/cnctusrinf	正常停止又は計画停止コマンド実行時に HiRDB に接続しているユーザがいた場合，接続ユーザ情報を格納するファイル
$PDDIR/spool/cnctusrdtl	正常停止又は計画停止コマンド実行時に HiRDB に接続しているユーザがいた場合，pdls -d act，pdls -d prc，pdls -d trn のコマンド実行結果を格納するファイル
$PDDIR/spool/pdsqldump※1	アクセスパス情報ファイルを格納するディレクトリ
$PDDIR/spool/pdtrninf	リアルタイム SAN レプリケーション使用時のトランザクション情報ファイル出力ディレクトリ
$PDDIR/spool/pdprf	PRF トレースファイル出力ディレクトリ
$PDDIR/spool/pdeeinf	バージョン 09-50 より前の HiRDB との互換のためのディレクトリ
$PDDIR/spool/pduaperr	SQL エラーレポートファイルディレクトリ
$PDDIR/spool/pdcwwrn	SQL 実行時間警告情報ファイルディレクトリ
$PDDIR/spool/utlrpt	処理性能情報ファイル出力ディレクトリ
任意※2	RPC トレースファイル
/dev/HiRDB/pth/※3	通信情報ファイルを格納するディレクトリ
$PDDIR/tmp※4	HiRDB 内部用ワークディレクトリ
$PDDIR/tmp/pdommenv	共用メモリの情報を格納するファイル
$PDDIR/tmp/home/HiRDB が管理する識別子のディレクトリ	カレントワーキングディレクトリ
$PDDIR/conf	HiRDB システム定義ファイルを格納するディレクトリ
$PDDIR/conf/backconf	システム構成変更コマンド実行時の，変更前の HiRDB システム定義を格納するディレクトリ

ファイル又はディレクトリ名	説明
$PDDIR/conf/Inittab	/etc/inittab 退避ディレクトリ
$PDDIR/.dbenv	HiRDB データベース環境情報ファイルを格納するディレクトリ
$PDCLTPATH/pdsql1.trc，pdsql2.trc※5	UAP が実行した SQL のトレース情報を格納するファイル
$PDCLTPATH/pderr1.trc，pderr2.trc※5	UAP とサーバ間の通信エラー情報を格納するファイル
$PDDIR/plugin	HiRDB プラグイン運用ディレクトリを統括するディレクトリ
$PDDIR/plugin/.sys	HiRDB 内部用ワークディレクトリ
$PDDIR/plugin/lib	プラグインライブラリを格納するディレクトリ
$PDDIR/plugin/プラグイン名称	プラグイン運用ディレクトリ
$PDDIR/plugin/プラグイン名称/.sys	HiRDB 内部用ワークディレクトリ
$PDDIR/plugin/プラグイン名称/bin	プラグイン専用コマンドを格納するディレクトリ
$PDDIR/plugin/プラグイン名称/etc	各プラグイン共通に必要なファイルを格納するディレクトリ
$PDDIR/plugin/プラグイン名称/conf	プラグイン用コンフィグレーションファイルを格納するディレクトリ
$PDDIR/jre※6	Java 実行環境
$PDDIR/renew	修正版 HiRDB の入れ替え用ディレクトリ
$PDDIR/renew_bak	修正版 HiRDB に入れ替えるときの，稼働中の HiRDB のバックアップ用ディレクトリ
$PDDIR/.pdlogprgid	syslog のプログラム ID を管理するファイル
●Linux（Linux 6 以前）の場合 /etc/init.d/ /etc/rc0.d/ /etc/rc1.d/ /etc/rc2.d/ /etc/rc3.d/ /etc/rc5.d/ /etc/rc6.d/ ●Linux（Linux 7 以降）の場合 なし	OS 起動時，又は OS シャットダウン時に動作するスクリプトファイルを格納するディレクトリ ファイル名： ●Linux（Linux 6 以前）の場合 HiRDB/シングルサーバは HiRDB_S，K09HiRDB_S，又は S91HiRDB_S HiRDB/パラレルサーバは HiRDB_P，K09HiRDB_P，又は S91HiRDB_P
$PDDIR/pdistup	簡易セットアップツールを格納するディレクトリ
$PDDIR/spool/tmp	作業用一時ファイル格納ディレクトリ
$PDDIR/spool/pdcmdact1，pdcmdact2	コマンド実行権限変更機能適用時に，コマンド実行権限の変更操作とコマンド実行操作の内容を記録するファイル

注※1

このディレクトリは，HiRDB がトラブルシュート情報を出力するディレクトリで，容量が増え続ける可能性があります。そのため，定期的に削除する必要があります。pdcspool コマンドで定期的に削除してください。

なお，次のオペランドでトラブルシュート情報を定期的に削除する設定ができます。オペランドの詳細は，マニュアル「HiRDB システム定義」を参照してください。

- pd_spool_cleanup_interval
- pd_spool_cleanup_interval_level
- pd_spool_cleanup
- pd_spool_cleanup_level

注※2

pd_rpc_trace_name オペランドでファイル名を指定します。

注※3

サーバマシン中のすべての HiRDB サーバを停止した状態の場合，このディレクトリ下のファイルを削除できます。

なお，pdsetup コマンドに-I オプションを指定した場合には作成されません。

注※4

HiRDB が内部的に使用するディレクトリです。このディレクトリ下にディレクトリ及びファイルを作成しないでください。また，HiRDB がファイルを作成するディレクトリにこのディレクトリを指定しないでください（例えば，pd_rpc_trace_name オペランドなど）。このディレクトリはユニットを開始するときに毎回削除及び新規作成されます。

注※5

このファイルは，PDCLTPATH で指定したディレクトリに二つ出力されます。PDCLTPATH の指定がない場合，UAP を起動したときのカレントディレクトリ（OpenTP1 から起動される UAP の場合，$DCDIR/tmp/home/**サーバ名 xx** のディレクトリ）下に出力されます。

作成されるファイル名は，X/Open に従った API（TX_関数）の使用の有無によって異なります。TX_関数の使用時に作成されるファイル名は次のようになります。

- pdsql**xxxxx**-1.trc，pdsql**xxxxx**-2.trc
- pderr**xxxxx**-1.trc，pderr**xxxxx**-2.trc

（凡例）**xxxxx**：UAP 実行時のプロセス ID

ファイル名がプロセス ID になるため，UAP 実行時のサーバプロセス数分のファイルが出力される可能性があるので注意が必要です。

注※6

バージョン 07-03 より前の場合は作成されます。バージョン 07-03 以降の場合は，JRE が同梱されないため，作成されません。

注※7

このディレクトリ下には，pd_max_server_process オペランドの値×2 ＋ 100 個のパイプファイルが作成されます。

注※8

このディレクトリ下に作成されるパイプファイルの個数の概算式を次に示します。

- HiRDB/シングルサーバの場合
  pd_max_server_process の値＋ 127 ＋ pd_max_users の値×4
- HiRDB/パラレルサーバの場合
  pd_max_server_process の値＋ 127 ＋ a
  a：次の計算式の値になります。
  フロントエンドサーバの場合：pd_max_users の値×2
  ディクショナリサーバの場合：pd_max_dic_process の値×35
  バックエンドサーバの場合：pd_max_bes_process の値×35
  ユニット内にあるサーバの分を計算して合計してください。

1.3.2 単調増加ファイル

HiRDB を使用すると単調増加するファイルを情報の種別ごとに示します。

なお，*は任意の英数字です。また，ファイル名又はディレクトリ名には，標準のパス名を記載しています。システムによっては異なる場合があります。

また，サポートバージョンとは，サポートを開始したバージョンを示します。例えば，サポートバージョンが初期の場合は，全バージョンでサポートしていることになります。

表中の凡例を次に示します。

○：オプションなどで最大サイズを制限できます。
×：最大サイズを制限できません。
S：HiRDB/シングルサーバ
P：HiRDB/パラレルサーバ
DK：HiRDB/Developer's Kit
RT：HiRDB/Run Time

(1) 簡易ダンプ

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
1	$PDDIR/spool/pdfesdump/*	フロントエンドサーバ用の簡易ダンプファイルです。pdfes プロセスセグメンテーション障害，又はアボート時に生成されます。 自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。	数 MB（不定）	ディレクトリ： $PDDIR/spool/pdfesdump，pdfesdump1，pdfesdump2 でディレクトリ 3 世代のループ ファイル： ディレクトリ下に無制限	×	P	初期
2	$PDDIR/spool/pddicdump/*	ディクショナリサーバ用の簡易ダンプファイルです。pddic プロセスセグメンテーション障害，アボート，ディクショナリ用 RD エリア閉塞時に生成されます。 自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。	数 MB（不定）	ディレクトリ： $PDDIR/spool/pddicdump，pddicdump1，pddicdump2 でディレクトリ 3 世代のループ ファイル： ディレクトリ下に無制限	×	P	初期
3	$PDDIR/spool/	バックエンドサーバ用の簡易ダンプファイルです。pdbes プロセスセグメンテーション障害，アボート，ユーザ用 RD エリア閉塞時に生成されます。	数 MB（不定）	ディレクトリ： $PDDIR/spool/pdbesdump，	×	P	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	pdbesdump/*	自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。		pdbesdump1，pdbesdump2 でディレクトリ 3 世代のループ ファイル： ディレクトリ下に無制限			
4	$PDDIR/spool/pdsdsdump/*	シングルサーバ用の簡易ダンプファイルです。pdsds プロセスセグメンテーション障害，アボート，ユーザ用 RD エリア閉塞時に生成されます。 自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。	数 MB（不定）	ディレクトリ： $PDDIR/spool/pdsdsdump，pdsdsdump1，pdsdsdump2 でディレクトリ 3 世代のループ ファイル： ディレクトリ下に無制限	×	S	初期
5	$PDDIR/spool/pdsysdump/*	システム共通の簡易ダンプファイルです。HiRDB システムを制御するプロセスの異常終了時に生成されます。 自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level	数～数十 MB（不定）	ディレクトリ： $PDDIR/spool/pdsysdump，pdsysdump1，pdsysdump2 でディレクトリ 3 世代のループ ファイル： ディレクトリ下に無制限	×	S，P	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
		手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。					

(2) エラー情報

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
1	$PDDIR/spool/pdlckinf/*出力日時	デッドロック・タイムアウト情報ファイルです。排他制御エラー発生時に生成されます。 自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。	数 KB	無制限	×	S，P	初期
2	$PDDIR/spool/pdlckinf/出力日時.mem	排他資源管理テーブル情報ファイルです。排他資源管理テーブル不足発生時に生成されます。 自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level	数 KB	無制限	×	S，P	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
		• pd_spool_cleanup • pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。					
3	$PDDIR/spool/pdtrninf/*	トランザクション情報ファイルです。リアルタイム SAN レプリケーション使用時に生成されます。 自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。	数 MB	無制限	×	S，P	07-01
4	$PDDIR/spool/pdjnlinf/*	システムログファイルの空き容量監視機能のシステムログファイルの状態情報ファイルです。システムログファイルの空き容量が警告値未満になったときに 1 ファイル生成されます。 自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。	2729～3521 バイト程度	サーバごとに 1 ファイル	×	S，P	07-00

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
5	$PDDIR/spool/pdjnlinf/errinf/*	システムログエラー情報出力ファイルです。リラン時，システムログ読み込みエラー時に生成されます。 自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。又は pdstart コマンド実行時（オプションなし，-i オプション指定時，又は dbdestroy オプション指定時）にも削除されます。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。	最大値は pd_log_max_data_size オペランドの指定値	ログ世代数分（有限個）	×	S，P	初期
6	/tmp/pdskipsql_*サーバ名*_*プロセスID*	pdorend コマンド実行時の追い付き反映 SQL スキップ情報ファイルです。pdorend コマンド実行時の追い付き反映処理で SQL スキップ対象エラーが発生したときに生成されます。 OS の rm コマンドなどを実行して手動で削除します。	((400 +((1500 ×マッピングキー構成列数）×2) [+ 50 +（Σ（50 +↑（更新要素数※1÷9) ↑×80)]※2）× スキップした SQL 件数）× 1.2	pdorend コマンドの対象 RD エリアが存在するサーバ数（コマンドのプロセス ID ごと，かつサーバ単位）	×	S，P	07-01
7	$PDCLTPATH/	クライアントエラー情報ファイルです。UAP 実行時に生成されます。OS	クライアント環境変数	2 ファイルをラップで使用	○	S，P，DK，RT	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	pderr1.trc, pderr2.trc	の rm コマンドなどを実行して手動で削除します。	PDUAPERLOGで指定（省略値は 4096 バイト）				
8	$PDCLTPATH/pderrxxxxx-1.trc, pderrxxxxx-2.trc（xxxxx：UAP実行時のプロセスID）	クライアントエラー情報ファイルです(X/Open に従った API（TX_関数）を使用した場合)。UAP 実行時に生成されます。OS の rm コマンドなどを実行して手動で削除します。	クライアント環境変数 PDUAPERLOGで指定（省略値は 4096 バイト）	UAP のプロセス ID ごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	07-01

注※1

更新要素数の詳細を次に示します。

・UPDATE SET（要素指定）：更新対象の要素数

・UPDATE ADD：1

・UPDATE DELETE：削除対象の要素数

注※2

制御文に mvcelmwarn=ignore を指定し，かつ繰返し列が存在する場合に[]内の値を加算します。

Σ：繰返し列数分を加算します。

(3) チューニング情報

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
1	$PDDIR/spool/pdsqldump/*	アクセスパス情報ファイルです。クライアント環境変数 PDVWOPTMODE に 1 以上を指定している場合，SQL 文実行時（前処理ごと）に生成されます。 自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。	（数 KB～数百 KB）×トランザクション内 SQL 数	トランザクションごと	×	S，P	04-03

(4) トラブルシュート情報

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
1	$PDDIR/spool/save/abcode.*	サーバプロセス異常終了情報（アボートコードなど）です。サーバプロセスの異常終了時に生成されます。 自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup	32 ビットモードの場合：40 バイト 64 ビットモードの場合：72 バイト	無制限	×	S，P	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
		• pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。					
2	$PDDIR/spool/save/*	サーバプロセス異常終了情報（コアファイル）です。サーバプロセスの異常終了時に生成されます。 自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。	数 MB～数十 MB 注　次の場合は，アタッチしていた共用メモリダンプも core に含まれるため，数 GB になることがあります。 • AIX，かつ HiRDB のバージョンが 09-50 より前で CORE_NOSHM 環境変数を指定していな	ダウンしたサーバ名[1-3]でファイル 3 世代のループ （サーバ名が不明な場合，サーバ名が数字となる場合があります）	×	S，P	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
			い場合 • Linux のバージョンが 5.2 より前で，かつ HiRDB のバージョンが 08-04 より前の場合				
3	$PDDIR/spool/save/*.cancel	サーバプロセス異常終了情報（コアファイル）です。pdcancel -d コマンドによるサーバプロセス強制終了時，又は pd_client_waittime_over_abort オペランドに Y を指定し，PDCWAITTIME オーバに伴うサーバプロセスのキャンセル時に生成されます。 自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。	数 MB～数十 MB 注　次の場合は，アタッチしていた共用メモリダンプも core に含まれるため，数 GB になることがあります。 • AIX，かつ HiRDB の	キャンセルしたプロセス数分	×	S，P	04-02

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
		なお，$PDDIR 下が容量不足となった場合はユニットダウンします。	バージョンが09-50より前でCORE_NOSHM環境変数を指定していない場合 • Linuxのバージョンが5.2より前で，かつHiRDBのバージョンが08-04より前の場合				
4	$PDDIR/spool/	サーバプロセス異常終了情報です。 サーバプロセスの異常終了時に生成されます。	数MB～十数MB	ダウンしたサーバ名[1-3].debでファイル3世代のループ	×	S，P	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	save/*.deb	自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。		(サーバ名が不明な場合，サーバ名が数字となる場合があります)			
5	$PDDIR/spool/save/*.ext.deb	サーバプロセス異常終了情報です。サーバプロセスの異常終了時に生成され，バックトレースを出力します。 自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。	数十 KB	ダウンしたサーバ名[1-3].deb でファイル 3 世代のループ※1 (サーバ名が不明な場合，サーバ名が数字となる場合があります)	×	S，P	09-03
6	$PDDIR/spool/save/*コマンド名*.txt*	pdload，pdrorg，pdreclaim，pdpgbfon コマンドのトラブルシュート情報です。コマンドのタイムアウト時に生成されます。 自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level	数 KB～数十 KB (そのときに実行していた HiRDB のプロセス数や，排他待ち資源数によって変化します)	コマンドのプロセス ID ごとに 1 ファイル	×	S，P	06-02

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
		手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。					
7	$PDDIR/spool/save/*.cwaitover	サーバプロセスの異常終了情報です。pd_client_waittime_over_abort オペランドに Y を指定している場合，サーバプロセス異常終了時に生成されます。 自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。	数 MB～十数 MB	無制限	×	S，P	04-05
8	$PDDIR/spool/save/*.shmdump	共用メモリダンプファイルです。pd_client_waittime_over_abort オペランドに Y を指定している場合，HiRDB を起動してから，初回の PDCWAITTIME オーバに伴うサーバプロセス異常終了時に生成されます。 自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。	数百 MB～数 GB（pdls -d mem コマンドで共用メモリを使用するプロセスの属性が"MANAGER"と表示された共用メモリセグメントのサイズ	HiRDB を起動してから初回の PDCWAITTIME オーバに伴うサーバプロセス異常終了時に 1 個生成します。※2	×	S，P	04-05

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
			とほぼ同じです）				
9	$PDDIR/spool/pdshmdump/*	共用メモリダンプファイルです。サーバプロセス異常終了時，ユニット異常終了時に生成されます。 自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。	数百 MB～数 GB（pdls -d mem コマンドで共用メモリを使用するプロセスの属性が"MANAGER"と表示された共用メモリセグメントのサイズとほぼ同じです）	shmdump と shmdump.old のファイル 2 世代でループ	×	S，P	初期
10	$PDDIR/spool/pdshmdump/*.ucb.*	共用メモリダンプファイルです。サーバプロセス異常終了時に生成されます。 自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。	数 KB	無制限	×	S，P	初期
11	$PDDIR/spool/	共用メモリダンプファイルです。サーバプロセス異常終了時に生成されます。	数 KB	無制限	×	S，P	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	pdshmdump/*.rmb.*	自動で削除する場合： 次のオペランドを指定します。 • pd_spool_cleanup_interval • pd_spool_cleanup_interval_level • pd_spool_cleanup • pd_spool_cleanup_level 手動で削除する場合： pdcspool コマンドを実行します。 なお，$PDDIR 下が容量不足となった場合はユニットダウンします。					

注※1

pdcancel -d コマンドでプロセスを強制終了した場合，又は pd_client_waittime_over_abort オペランドに Y を指定し，PDCWAITTIME オーバに伴うサーバプロセスの異常終了が発生した場合，次のファイルがキャンセルしたプロセス数分生成されます。

$PDDIR/spool/save/サーバ名.プロセス ID.cancel.ext.deb

注※2

pd_clt_waittime_over_dump_level オペランドに shm_fesonly を指定した場合，クライアントが接続した FES ユニットで共用メモリダンプが生成されます。

(5) 通信用スペシャルファイル

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
1	/dev/HiRDB/pth/*※	通信用スペシャルファイルです。 HiRDB 開始時，HiRDB サーバ起動時，コマンド実行時，UAP 接続時に生成されます。 HiRDB サーバプロセスが正常終了すると自動で削除されます。 手動で削除する場合は，OS の rm コマンドで削除します（サーバマシンの	0	最大 196605	×	S，P	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
		すべての HiRDB サーバを停止した状態で削除できます)。 なお，$PDDIR 下が容量不足となった場合，ユニットが起動できません。					

注※

通信情報ファイルディレクトリ変更機能を適用時（10-04 以降）には，pd_ipc_file_dir オペランドに指定されたディレクトリに作成されます。

(6) インデクス作成用一時ファイル

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
1	*pd_plugin_ixmk_dir 指定ディレクトリ/インデクス名称.RD エリア名称*（pd_plugin_ixmk_dir にディレク	プラグインインデクスのインデクス情報ファイルです。SQL 文実行時に生成されます。 プラグインインデクスの一括作成が正常終了したとき，プラグインインデクスの再作成を実行したときに，自動で削除されます。 手動で削除する場合は OS の rm コマンドなどを実行します。	マニュアル「HiRDB システム運用ガイド」の「プラグインインデクスの遅延一括作成」の「注意事項」を参照してください。	プラグイン数×インデクス格納 RD エリア数（更新の発生した RD エリア数）	×	S，P	05-06

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	トリを指定した場合）						

(7) 運用コマンド作業用一時ファイル

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
1	$PDDIR/tmp/CMr*	運用コマンド一時ファイルです。運用コマンド実行時に生成されます。 運用コマンド実行終了時に自動的に削除されます。	MAX（1024バイト×グローバルバッファ数，512バイト×RDエリア数）	コマンドのプロセスIDごとに1ファイル	○	S，P	初期
2	$PDDIR/tmp/CMs*	運用コマンド一時ファイルです。運用コマンド実行時に生成されます。 運用コマンド実行終了時に自動的に削除されます。	128バイト×RDエリア数	コマンドのプロセスIDごとに1ファイル	○	S，P	初期
3	$PDDIR/tmp/CMb*	pdbufls コマンドの差分情報ファイルです。pdbufls コマンド（-k sts 指定，又はオプション指定なし）実行時に生成されます。 HiRDB 開始時に自動で削除されます。手動で削除する場合は，pdbufls コマンドを実行していないときに OS の rm コマンドなどを実行します。	16バイト＋124バイト×グローバルバッファ数	コマンドのプロセスIDごとに1ファイル	○	S，P	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
4	$PDDIR/tmp/pdcmd*	運用コマンド結果ファイルです。運用コマンド実行時に生成されます。運用コマンド実行終了時に自動的に削除されます。 なお，このファイルは HiRDB/パラレルサーバでディクショナリサーバとシステムマネジャが別ノードの場合にだけ使用されます。	MAX（1024 バイト×グローバルバッファ数，512 バイト×RD エリア数）	コマンドのプロセス ID ごとに 1 ファイル	○	P	初期
5	/tmp/pddefrev.exp.*（pd_tmp_directory オペランドは無効です）	pddefrev コマンド作業用一時ファイルです。pddefrev コマンド実行開始時に生成されます。 pddefrev コマンド実行終了時に自動的に削除されます。手動で削除する場合は，OS の rm コマンドなどを実行します。	搬出対象の資源数×70 バイト	コマンドのプロセス ID ごとに 1 ファイル	×	S，P	04-01
6	$PDDIR/spool/tmp/*	運用コマンド一時ファイルです。運用コマンド実行時に生成されます。運用コマンド実行終了時に自動的に削除されます。	最大 1024 バイト	コマンドのプロセス ID ごとに 1 ファイル	×	S，P	09-04
7	$PDDIR/tmp/CMp*	pdchpathf コマンド作業用一時ファイルです。pdchpathf コマンド実行時に作成されます。 pdchpathf コマンド実行終了時に自動的に削除されます。手動で削除する場合は，OS の rm コマンドなどを実行します。	制御文に指定した area 数×524 ＋制御文に指定した HiRDB ファイルシステム領域内の HiRDB ファイル	pdchpathf の制御文数＋ 1	×	S，P	09-65

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
			数×72＋330				

(8) ユティリティ結果ファイル

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
1	$TMPDIR/pdcp1*，/tmp/pdcp1*，-pオプション指定ディレクトリ/pdcp1*（08-02以降でpd_tmp_directoryオペランド指定時は	pdcopy結果ファイルです。pdcopy実行時に生成されます。 OSのrmコマンドなどを実行して手動で削除します。	3500バイト×RDエリア数	pdcopyの-pオプションなしの場合に毎回異なるファイル名で1ファイル	○	S，P	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	pd_tmp_directory ディレクトリ下)						
2	$TMPDIR/pdrs1*，/tmp/pdrs1*，-wオプション指定ディレクトリ/pdrs1*(08-02以降でpd_tmp_directory オペランド指定時はpd_tmp_directory ディレク	pdrstr 結果一時ファイルです（ログ指定回復時のロールバックログ格納一時ファイル)。pdrstr 実行時に生成されます。 ロールバック完了時に自動で削除されます。手動で削除する場合は，OS の rm コマンドなどを実行します。	ロールバックログ量に依存します。	pdrstr の対象 RD エリアが属するサーバ数	×	S，P	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	トリ下)						
3	$TMPDIR/pdrs2*，/tmp/pdrs2*，-pオプション指定ディレクトリ/pdrs2*(08-02以降でpd_tmp_directoryオペランド指定時はpd_tmp_directoryディレクトリ下)	pdrstr結果ファイルです。pdrstr実行時に生成されます。 OSのrmコマンドなどを実行して手動で削除します。	3500バイト×RDエリア数	pdrstrの-pオプションなしの場合に毎回異なるファイル名で1ファイル	○	S，P	初期
4	$TMPDIR/	pdrstr結果一時ファイルです（メッセージをコンソールに出力するための	256バイト	1（コマンド実行ノードだけ）	○	S，P	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	pdrs4，/tmp/pdrs4（08-02 以降で pd_tmp_directory オペランド指定時は pd_tmp_directory ディレクトリ下）	一時ファイル）。pdrstr 実行時に生成されます。 pdrstr 実行終了時に自動で削除されます。手動で削除する場合は，OS の rm コマンドなどを実行します。					
5	/tmp/REPORT*（バージョン 08-02 以降で pd_tmp_directory オペランド指定時は pd_t	pdrbal 処理結果ファイルです。 pdrbal 実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	「リバランスユティリティ（pdrbal）実行時のファイルの容量」を参照してください。	report 制御文指定なしの場合に毎回異なるファイル名で 1 ファイル	×	S，P	06-00

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	mp_directory ディレクトリ下)						
6	/tmp/CONSTCK-REPORT-* (08-02 以降で pd_tmp_directory オペランド指定時は pd_tmp_directory ディレクトリ下)	pdconstck 処理結果ファイルです。pdconstck 実行開始時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	「整合性チェックユティリティ (pdconstck) 実行時のファイルの容量」を参照してください。	コマンドのプロセス ID ごとに 1 ファイル	×	S，P	07-03

(9) ユティリティ作業用一時ファイル

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
1	$TMPDIR/pdcp3*，/tmp/pdcp3*（08-02 以降で pd_tmp_directory オペランド指定時は pd_tmp_directory ディレクトリ下）	pdcopy 結果一時ファイル（メッセージをコンソールに出力するための一時ファイル）です。pdcopy 実行時に生成されます。 pdcopy 実行終了時に自動的に削除されます。手動で削除する場合は，OS の rm コマンドなどを実行します。	256 バイト	1（コマンド実行ノードだけ）	○	S，P	初期
2	$TMPDIR/pdcp4*，/tmp/pdcp4*（08-02 以降で pd_tmp_d	pdcopy 結果一時ファイル（メッセージ格納一時ファイル）です。pdcopy 実行開始時に生成されます。 pdcopy 実行終了時に自動的に削除されます。手動で削除する場合は，OS の rm コマンドなどを実行します。	256 バイト×pdcopy 対象 RD エリア数	pdcopy の対象 RD エリアが属するサーバ数＋ 2	○	S，P	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	irectory オペランド指定時は pd_tmp_directory ディレクトリ下）						
3	$TMPDIR/pdrs5*，/tmp/pdrs5*（08-02 以降で pd_tmp_directory オペランド指定時は pd_tmp_directory ディレクトリ下）	pdrstr 結果一時ファイル（メッセージ格納一時ファイル）です。pdrstr 実行開始時に生成されます。 pdrstr 実行終了時に自動的に削除されます。手動で削除する場合は，OS の rm コマンドなどを実行します。	256 バイト×pdrstr 対象 RD エリア数	対象 RD エリアが属するサーバ数＋2 ただし，対象サーバ数が 2 以上，又は回復対象ユニット以外に存在するログを入力する場合には，さらに＋1。	○	S，P	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
4	/tmp/ERROR*（08-02 以降で pd_tmp_directory オペランド指定時は pd_tmp_directory ディレクトリ下）	pdload 入力データ格納情報の結果ファイル（エラー情報ファイル）です。pdload 実行時に生成されます。OS の rm コマンドなどを実行して手動で削除します。	「データベース作成ユティリティ（pdload）実行時のファイルの容量」を参照してください。	source 文で error オペランドを指定しない場合，毎回異なるファイル名で 1 ファイル	×	S，P	初期
5	/usr/tmp/*，/var/tmp/*，/tmp/*（08-02 以降で pd_tmp_directory オペランド指定時は pd_tmp_d	pdload エラー情報ファイル作成用一時ファイルです。pdload 実行時に生成されます。OS の rm コマンドなどを実行して手動で削除します。 なお，このファイルは HiRDB/パラレルサーバの場合だけ使用されます。	「データベース作成ユティリティ（pdload）実行時のファイルの容量」を参照してください。	データロード対象表格納サーバ数	×	S，P	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	irectory ディレクトリ下)						
6	/tmp/LOBMID*(08-02 以降で pd_tmp_directory オペランド指定時は pd_tmp_directory ディレクトリ下)	pdload BLOB 列ロード用ワークファイルです。pdload 実行開始時に生成されます。OS の rm コマンドなどを実行して手動で削除します。	「データベース作成ユティリティ（pdload）実行時のファイルの容量」を参照してください。	オプション-k に d 以外を指定し，lobmid 文を省略して BLOB 列を持つ表にデータロードする場合に，毎回異なるファイル名で 1 ファイル	×	S，P	03-00
7	/tmp/INDEX*, *idxwork 制御文指定ディレクトリ*/INDE	pdload，pdrorg，pdrbal のインデクス情報ファイルです。pdload，pdrorg，pdrbal 実行時に生成されます。 インデクスロード完了時に自動的に削除されます。手動で削除する場合は，OS の rm コマンドなどを実行します。	次の箇所を参照してください。 「データベース作成ユティリティ（pdload）実行時のファイ	インデクス一括作成モード選択時，毎回異なるファイル名でインデクス数×インデクス格納 RD エリア数分	×	S，P	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	X*（08-02 以降で pd_tmp_directory オペランド指定時は pd_tmp_directory ディレクトリ下）		ルの容量」 「データベース再編成ユティリティ（pdrorg）実行時のファイルの容量」 「リバランスユティリティ（pdrbal）実行時のファイルの容量」				
8	/tmp/rs*, *sort 制御文指定ディレクトリ*/rs*（08-02 以降で pd_tmp_directory オペランド指定時は pd_t	pdload，pdrorg，pdrbal のソート用ワークファイルです。pdload，pdrorg，pdrbal 実行時に生成されます。 インデクスロード完了時，又はプロセス終了時に自動的に削除されます。手動で削除する場合は，OS の rm コマンドなどを実行します。	次の箇所を参照してください。 「データベース作成ユティリティ（pdload）実行時のファイルの容量」 「データベース再編成ユティリティ（pdrorg）実行時のファイ	インデクス一括作成モード選択時，毎回異なるファイル名でインデクス格納サーバ数分	×	S，P	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	mp_directory ディレクトリ下)		ルの容量」 「リバランスユティリティ(pdrbal)実行時のファイルの容量」				
9	/tmp/*.dbst.data，/tmp/*.dbst.msg (08-02 以降で pd_tmp_directory オペランド指定時は pd_tmp_directory ディレクトリ下)	pddbst コマンド一時ファイルです。pddbst 実行開始時に生成されます。 pddbst 実行終了時に自動的に削除されます。手動で削除する場合は，OS の rm コマンドなどを実行します。	「データベース状態解析ユティリティ(pddbst)実行時のファイルの容量」を参照してください。	コマンドのプロセス ID ごとにそれぞれ 1 ファイル	×	S，P	初期
10	/tmp/*.syi, *.syo,	pdstedit コマンド作業用一時ファイルです。pdstedit 実行開始時に生成されます。	「統計解析ユティリティ(pdstedi	コマンドのプロセス ID ごと，かつ解析対象の情報ごとに 1 ファイル	×	S，P	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	*.uai, *.uao, *.sqi, *.sqo, *.pci, *.pco, *.soi, *.soo, *.doi, *.doo, *.bui, *.buo, *.fii, *.fio, *.dfi, *.dfo, *.ixi, *.ixo, *.isi, *.iso, *.cni, *.cno, *.qhi, *.qho, *.shi, *.sho, *.obi, *.obo, *.fsi, *.fso, *.hbi, *.hbo, *-w* オプション指	pdstedit 実行終了時に自動的に削除されます。手動で削除する場合は，OS の rm コマンドなどを実行します。	t）実行時のファイルの容量」を参照してください。				

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	定ディレクトリ/***（08-02 以降で pd_tmp_directory オペランド指定時は pd_tmp_directory ディレクトリ下）						

(10) トレース情報

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
1	$PDCLTPATH/pdsql1.trc,	SQL トレース情報（クライアント環境変数 PDSQLTRCFMT に 1 を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	クライアント環境変数 PDSQLTRACE で指定	2 ファイルをラップで使用	○	S，P，DK，RT	初期

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	pdsql2.trc						
2	$PDCLTPATH/pdsqlxxxxx-1.trc,pdsqlxxxxx-2.trc xxxxx：UAP実行時のプロセスID）	SQLトレース情報（X/Openに従ったAPI（TX_関数を使用，かつクライアント環境変数PDSQLTRCFMTに1を指定した場合）を使用した場合）です。SQL文実行時に生成されます。 OSのrmコマンドなどを実行して手動で削除します。	クライアント環境変数PDSQLTRACEで指定	UAPのプロセスIDごとに2ファイルを作成し，ラップで使用	○	S，P，DK，RT	07-01
3	$PDCLTPATH/pdjsqlxxxxxxxx_ppppp_1.trc,pdjsqlxxxxxxxx_ppppp_2.trc xxxxxxxx：接続した	SQLトレース情報（Type4 JDBCドライバを使用，かつクライアント環境変数PDSQLTRCFMTに1を指定した場合）です。SQL文実行時に生成されます。 OSのrmコマンドなどを実行して手動で削除します。	クライアント環境変数PDSQLTRACEで指定	接続したサーバ名とクライアント側の受信ポート番号ごとに2ファイルを作成し，ラップで使用	○	S，P，DK，RT	08-00

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	サーバ名 ppppp：クライアント側の受信ポート番号						
4	$PDTRCPATH/pdjsqlxxxxxxxx_ppppp_1.trc, pdjsqlxxxxxxxx_ppppp_2.trc xxxxxxxx：接続したサーバ名 ppppp：クライアント側の受信ポー	動的 SQL トレース（クライアント環境変数 PDSQLTRCFMT に 1 を指定，pdtrcmgr コマンドを使用，かつ Type4 JDBC ドライバを使用した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	pdtrcmgr -s コマンドで指定	接続したサーバ名とクライアント側の受信ポート番号ごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	08-00

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	ト番号						
5	$PDJDBFILEDIR/pdexc1.trc, pdexc2.trc	Exception トレースログです。Type4 JDBC ドライバ内で例外発生時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	マニュアル「HiRDB UAP 開発ガイド」の「Exception トレースログ」を参照してください。	2 ファイルをラップで使用	○	S，P，DK，RT	08-00
6	ユーザ指定（DriverManager クラスの setLogWriter メソッド，DataSource インタフェースの setLogWriter メソッドで指定）	JDBC インタフェースメソッドトレースです。Type4 JDBC ドライバ内で例外発生時，Connection.close メソッド実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	↑180×n×m÷1024 ↑　キロバイト n：接続時に指定するユーザプロパティの TRC_NO で指定（省略値は 500） m：接続から切断までの例外発生回数＋ 1	1 ファイル	○	S，P，DK，RT	08-00

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
7	$PDTRCPATH/pdcHHMMSSmmm_XXX_1.trc, pdcHHMMSSmmm_XXX_2.trc HHMMSSmmm：CONNECT時間 XXX：CONNECT通番	動的 SQL トレース（pdtrcmgr コマンドを使用，かつクライアント環境変数 PDSQLTRCFMT に 1 を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	pdtrcmgr -s コマンドで指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	06-00
8	$PDCLTPATH/pdrcnct1.trc, pdrcnct2.trc	再接続トレースです。自動再接続機能で自動的に接続したときに生成されます。 OS の rm コマンドなどを実行して手動で削除します。	クライアント環境変数 PDRCTRACE で指定	2 ファイルをラップで使用	○	S，P，DK，RT	07-01
9	ユーザ指定	pdorend コマンド実行時の追い付き反映 SQL トレース情報ファイルです。	追い付き反映制御	pdorend 反映プロセスごとに 2 ファイル	○	S，P	08-02

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	（pdorend コマンドの追い付き反映制御ファイルで指定）	pdorend コマンド実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	ファイルで指定	を作成し，ラップで使用			
10	$PDJDBFILEDIR/pdjdataerr1.trc, pdjdataerr2.trc	不正電文トレースです。Type4 JDBC ドライバ内で不正電文受信時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	マニュアル「HiRDB UAP 開発ガイド」の「不正電文トレース」を参照してください。	2 ファイルをラップで使用	○	S，P，DK，RT	09-60
11	$PDCLTPATH/pdXXXXXXXyyyyyyyyy_1.trc, pdXXXXXXXyyyyyyyyy_2.trc	動的 SQL トレース（pdclttrc コマンドを使用，かつクライアント環境変数 PDSQLTRCFMT に 1 を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	pdclttrc コマンドの-o で指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	07-01

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	XXXXXXXX：接続したサーバ名 yyyyyyyy：サーバプロセス ID						
12	$PDCLTPATH/pdjsqlxxxxxxxx_ppppp_1.trc, pdjsqlxxxxxxxx_ppppp_2.trc xxxxxxxx：接続したサーバ名 ppppp：クライアント側の受	動的 SQL トレース（pdclttrc コマンドを使用，Type4 JDBC ドライバを使用，かつクライアント環境変数 PDSQLTRCFMT に 1 を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	pdclttrc コマンドの-o で指定	接続したサーバ名とクライアント側の受信ポート番号ごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	08-00

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	信ポート番号						
13	$PDCLTPATH/pdjsql1.trc, pdjsql2.trc	SQL トレース情報（Type4 JDBC ドライバを使用しサーバと接続できなかった場合，かつクライアント環境変数 PDSQLTRCFMT に 1 を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	クライアント環境変数 PDSQLTRACE で指定	2 ファイルをラップで使用	○	S，P，DK，RT	08-00
14	$PDCLTPATH/pd2sql1.trc, pd2sql2.trc	SQL トレース情報（クライアント環境変数 PDSQLTRCFMT に 2 を指定し，サーバと接続できなかった場合又はクライアント環境変数 PDXATRCFILEMODE に LUMP を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	クライアント環境変数 PDSQLTRACE で指定	2 ファイルをラップで使用	○	S，P，DK，RT	09-50
15	$PDCLTPATH/pd2sqlHHMMSfff_XXXXXX_YYYYYYYYY_1.trc, pd2sqlHHMMSfff_XXX	SQL トレース情報（クライアント環境変数 PDSQLTRCFMT に 2 を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	クライアント環境変数 PDSQLTRACE で指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	09-50

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	XXXXX_YYYYYYYYY_2.trc HHMMSSfff：コネクト要求時間（HH：時，MM：分，SS：秒，fff：ミリ秒） XXXXXXXX：接続したサーバ名 YYYYYYYYYY：コネクト通番						
16	$PDTRCPATH/pd2csqlHHMMSSfff_	動的 SQL トレース（pdtrcmgr コマンドを使用，かつクライアント環境変数 PDSQLTRCFMT に 2 を指定した場合）です。SQL 文実行時に生成されます。	pdtrcmgr -s コマンドで指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	09-50

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	XXXXXXXX_YYYYYYYYY_1.trc, pd2csqlHHMMSSfff_XXXXXXXX_YYYYYYYYY_2.trc HHMMSSfff：コネクト要求時間（HH：時，MM：分，SS：秒，fff：ミリ秒） XXXXXXXX：接続したサーバ名	OS の rm コマンドなどを実行して手動で削除します。					

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	YYYYYYYYYY：コネクト通番						
17	$PDCLTPATH/pd2XXXXXXXXyyyyyyyyyy_1.trc, pd2XXXXXXXXyyyyyyyyyy_2.trc XXXXXXXX：接続したサーバ名 yyyyyyyyyy：サーバプロセスID	動的SQLトレース（pdclttrcコマンドを使用，かつクライアント環境変数PDSQLTRCFMTに2を指定した場合）です。SQL文実行時に生成されます。 OSのrmコマンドなどを実行して手動で削除します。	pdclttrcコマンドの-oで指定	コネクトごとに2ファイルを作成し，ラップで使用	○	S，P，DK，RT	09-50
18	$PDCLTPATH/pd2js	SQLトレース情報（Type4 JDBCドライバを使用しサーバと接続できなかった場合，かつクライアント環境変数PDSQLTRCFMTに2を指定した	クライアント環境変数PDSQL	2ファイルをラップで使用	○	S，P，DK，RT	09-50

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	ql1.trc, pd2jsql2.trc	場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	TRACE で指定				
19	$PDCLTPATH/pd2jsqlHHMMSSfff_XXXXXXXX_YYYYYYYYY_1.trc, pd2jsqlHHMMSSfff_XXXXXXXX_YYYYYYYYY_2.trc HHMMSSfff：コネクト要求時間（HH：時，MM：分，SS：	SQL トレース情報（Type4 JDBC ドライバを使用，かつクライアント環境変数 PDSQLTRCFMT に 2 を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	クライアント環境変数 PDSQLTRACE で指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	09-50

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	秒，fff：ミリ秒 XXXXXXXX：接続したサーバ名 YYYYYYYY：コネクト通番						
20	$PDTRCPATH/pd2jcsqlHHMMSSfff_XXXXXXXX_YYYYYYYY_1.trc，pd2jcsqlHHMMSSfff_XXXXXXXX_YYYYYYYY_2.trc	動的 SQL トレース（pdtrcmgr コマンドを使用，Type4 JDBC ドライバを使用，かつクライアント環境変数 PDSQLTRCFMT に 2 を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	pdtrcmgr -s コマンドで指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	09-50

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	HHMMSSfff：コネクト要求時間（HH：時，MM：分，SS：秒，fff：ミリ秒） XXXXXXXX：接続したサーバ名 YYYYYYYYYY：コネクト通番						
21	$PDCLTPATH/pd2jXXXXXXXXyyyyyyyyyy_1.trc，pd2jXXXXXX	動的 SQL トレース（pdclttrc コマンドを使用，Type4 JDBC ドライバを使用，かつクライアント環境変数 PDSQLTRCFMT に 2 を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	pdclttrc コマンドの-o で指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	09-50

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	XXyyyyyyyyyy_2.trc XXXXXXXX：接続したサーバ名 yyyyyyyyyy：サーバプロセス ID						
22	$PDCLTPATH/pdrcnct_ppppp_1.trc, pdrcnct_ppppp_2.trc ppppp：クライアント側の受信ポート番号	再接続トレース（Type4 JDBC ドライバを使用した場合）です。自動再接続機能で自動的に接続したときに生成されます。 OS の rm コマンドなどを実行して手動で削除します。	クライアント環境変数 PDRCTRACE で指定	クライアント側の受信ポート番号ごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	08-00

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
23	$PDWRTLNPATH/pdwrtln1.trc, pdwrtln2.trc	WRITE LINE 文の値式の値を出力するファイルです。WRITE LINE 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	クライアント環境変数 PDWRTLNFILSZ で指定	2 ファイルをラップで使用	○	S，P，DK，RT	07-00
24	$PDWRTLNPATH/pdwrtlnxxxxx-1.trc, pdwrtlnxxxxx-2.trc xxxxx：クライアントプロセス ID	WRITE LINE 文の値式の値を出力するファイル（X/Open に従った API（TX_関数）を使用した場合）です。WRITE LINE 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	クライアント環境変数 PDWRTLNFILSZ で指定	クライアントプロセスごとに 2 ファイルを作成しラップで使用	○	S，P，DK，RT	07-00
25	$PDWRTLNPATH/pdwrtlnXXXXXX_ppppp_1.trc, pdwrt	WRITE LINE 文の値式の値を出力するファイル（Type4 JDBC ドライバを使用した場合）です。WRITE LINE 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	クライアント環境変数 PDWRTLNFILSZ で指定	接続したサーバ名とクライアント側の受信ポート番号ごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	08-00

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	lnXXXXXXX_ppppp_2.trc XXXXXXXX：接続したサーバ名 ppppp：クライアント側の受信ポート番号						
26	$PDCLTPATH/pdjsqlHHMMSSfff_XXXXXXXX_YYYYYYYYY_1.trc, pdjsqlHHMMSSfff_XXX	SQL トレース情報（Type4 JDBC ドライバを使用，クライアント環境変数 PDSQLTRCFMT に 1 を指定，かつクライアント環境変数 PDCONTYPE に ACTIVE を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	クライアント環境変数 PDSQLTRACE で指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	10-07

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	XXXXX_YYYYYYYYY_2.trc HHMMSSfff：コネクト要求時間（HH：時，MM：分，SS：秒，fff：ミリ秒 XXXXXXXX：接続したサーバ名 YYYYYYYYY：コネクト通番						
27	$PDTRCPATH/pdjsqlHHMMSSfff_	動的 SQL トレース（pdtrcmgr コマンドを使用，Type4 JDBC ドライバを使用，クライアント環境変数 PDSQLTRCFMT に 1 を指定，かつクライアント環境変数 PDCONTYPE に ACTIVE を指定した場合）です。SQL 文実行時に生成されます。	pdtrcmgr -s コマンドで指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	10-07

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	XXXXXXXX_YYYYYYYYY_1.trc, pdjsqlHHMMSSfff_XXXXXXXX_YYYYYYYYY_2.trc HHMMSSfff：コネクト要求時間（HH：時，MM：分，SS：秒，fff：ミリ秒） XXXXXXXX：接続したサーバ名	OS の rm コマンドなどを実行して手動で削除します。					

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	YYYYYYYY：コネクト通番						
28	$PDCLTPATH/pdjsqlHHMMSSfff_XXXXXXXX_YYYYYYYY_1.trc, pdjsqlHHMMSSfff_XXXXXXXX_YYYYYYYY_2.trc HHMMSSfff：コネクト要求時間（HH：時，MM：分，	動的 SQL トレース（pdclttrc コマンドを使用，Type4 JDBC ドライバを使用，クライアント環境変数 PDSQLTRCFMT に 1 を指定，かつクライアント環境変数 PDCONTYPE に ACTIVE を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	pdclttrc コマンドの-o で指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	10-07

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	SS：秒，fff：ミリ秒） XXXXXXXX：接続したサーバ名 YYYYYYYY：コネクト通番						
29	$PDCLTPATH/pdrcnct_HHMMSSfff_XXXXXXXX_YYYYYYYY_1.trc，pdrcnct_HHMMSSfff_XXXXXXXX_YYYYYYYY	再接続トレース（Type4 JDBC ドライバを使用，かつクライアント環境変数 PDCONTYPE に ACTIVE を指定した場合）です。自動再接続機能で自動的に接続したときに生成されます。 OS の rm コマンドなどを実行して手動で削除します。	クライアント環境変数 PDRCTRACE で指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	10-07

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	Y_2.trc HHMMSSfff：コネクト要求時間（HH：時，MM：分，SS：秒，fff：ミリ秒） XXXXXXXX：接続したサーバ名 YYYYYYYY：コネクト通番						
30	$PDWRTLNPATH/pdwrtlnHHMMSSfff_XXXXXXXX_Y	WRITE LINE 文の値式の値を出力するファイル（Type4 JDBC ドライバを使用，かつクライアント環境変数 PDCONTYPE に ACTIVE を指定した場合）です。WRITE LINE 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	クライアント環境変数 PDWRTLNFILSZ で指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	10-07

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	YYYYYYYY_1.trc, pdwrtlnHHMMSSfff_XXXXXXXX_YYYYYYYY_2.trc HHMMSSfff：コネクト要求時間（HH：時，MM：分，SS：秒，fff：ミリ秒） XXXXXXXX：接続したサーバ名 YYYYYYYY：コネ						

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	クト通番						

(11) 統計情報

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
1	$PDREPPATH/pdHHMMSSmmm_XXX_1.trc, pdHHMMSSmmm_XXX_2.trc HHMMSSmmm：CONNECT時間 XXX：CONNEC	UAP 統計レポート（クライアント環境変数 PDSQLTRCFMT に 1 を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	クライアント環境変数 PDSQLTRACE で指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	06-00

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	T 通番						
2	$PDREPPATH/pdjsqlxxxxxxxx_ppppp_1.trc, pdjsqlxxxxxxxx_ppppp_2.trc xxxxxxxx：接続したサーバ名 ppppp：クライアント側の受信ポート番号	UAP 統計レポート（クライアント環境変数 PDSQLTRCFMT に 1 を指定，かつ Type4 JDBC ドライバを使用した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	クライアント環境変数 PDSQLTRACE で指定	接続したサーバ名とクライアント側の受信ポート番号ごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	08-00
3	$PDREPPATH/pdXXXXXXyyyy	動的 UAP 統計レポート（pdclttrc コマンドを使用，かつクライアント環境変数 PDSQLTRCFMT に 1 を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	pdclttrc コマンドの-o で指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	07-01

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	yyyyy_1.trc, pdXXXXXXXXyyyyyyyyy_2.trc XXXXXXXX：接続したサーバ名 yyyyyyyyy：サーバプロセス ID						
4	$PDREPPATH/pdjsqlxxxxxxxx_ppppp_1.trc, pdjsqlxxxxxxxx_ppppp_2.trc xxxxxxxx：接	動的 UAP 統計レポート（pdclttrc コマンドを使用，クライアント環境変数 PDSQLTRCFMT に 1 を指定，かつ Type4 JDBC ドライバを使用した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	pdclttrc コマンドの-o で指定	接続したサーバ名とクライアント側の受信ポート番号ごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	08-00

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	続したサーバ名 ppppp：クライアント側の受信ポート番号						
5	$PDREPPATH/pd2sqlHHMMSSfff_XXXXXXXX_YYYYYYYYY_1.trc, pd2sqlHHMMSSfff_XXXXXXXX_YYYYYYYYY_2.trc HHMMSSfff：コ	UAP 統計レポート（クライアント環境変数 PDSQLTRCFMT に 2 を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	クライアント環境変数 PDSQLTRACE で指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	09-50

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	ネクト要求時間（HH：時，MM：分，SS：秒，fff：ミリ秒） XXXXXXXX：接続したサーバ名 YYYYYYYY：コネクト通番						
6	$PDREPPATH/pd2XXXXXXXyyyyyyyy_1.trc，pd2XXXXXXXyyyyyyyy	動的 UAP 統計レポート（pdclttrc コマンドを使用，かつクライアント環境変数 PDSQLTRCFMT に 2 を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	pdclttrc コマンドの-o で指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	09-50

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	yyy_2.trc XXXXXXXX：接続したサーバ名 yyyyyyyyyy：サーバプロセス ID						
7	$PDREPPATH/pd2jsqlHHMMSSfff_XXXXXXXX_YYYYYYYYYY_1.trc, pd2jsqlHHMMSSfff_XXXXXXXX_YYYYYYYYYY_2.trc	UAP 統計レポート（Type4 JDBC ドライバを使用，かつクライアント環境変数 PDSQLTRCFMT に 2 を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	クライアント環境変数 PDSQLTRACE で指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	09-50

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	HHMMSSfff：コネクト要求時間（HH：時，MM：分，SS：秒，fff：ミリ秒） XXXXXXXX：接続したサーバ名 YYYYYYYYYY：コネクト通番						
8	$PDREPPATH/pd2jXXXXXXXXyyyyyyyyyy_1.trc，pd2jXXXXXX	動的 UAP 統計レポート（pdclttrc コマンドを使用，Type4 JDBC ドライバを使用，かつクライアント環境変数 PDSQLTRCFMT に 2 を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	pdclttrc コマンドの-o で指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	09-50

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	XXyyyyyyyyyy_2.trc XXXXXXXX：接続したサーバ名 yyyyyyyyyy：サーバプロセス ID						
9	$PDREPPATH/pdjsqlHHMMSfff_XXXXXXXX_YYYYYYYYY_1.trc, pdjsqlHHMMSfff_XXXXXXXX_YYYYYYYYY	UAP 統計レポート（Type4 JDBC ドライバを使用，クライアント環境変数 PDSQLTRCFMT に 1 を指定,かつクライアント環境変数 PDCONTYPE に ACTIVE を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	クライアント環境変数 PDSQLTRACE で指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	10-07

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	Y_2.trc HHMMSSfff：コネクト要求時間（HH：時，MM：分，SS：秒，fff：ミリ秒） XXXXXXXX：接続したサーバ名 YYYYYYYY：コネクト通番						
10	$PDREPPATH/pdjsqlHHMMSSfff_XXXXXXXX_YYYY	動的 UAP 統計レポート（pdclttrc コマンドを使用，Type4 JDBC ドライバを使用，クライアント環境変数 PDSQLTRCFMT に 1 を指定，かつクライアント環境変数 PDCONTYPE に ACTIVE を指定した場合）です。SQL 文実行時に生成されます。 OS の rm コマンドなどを実行して手動で削除します。	pdclttrc コマンドの-o で指定	コネクトごとに 2 ファイルを作成し，ラップで使用	○	S，P，DK，RT	10-07

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	YYYYY_1.trc, pdjsqlHHMMSSfff_XXXXXXXX_YYYYYYYYY_2.trc HHMMSSfff：コネクト要求時間（HH：時，MM：分，SS：秒，fff：ミリ秒） XXXXXXXX：接続したサーバ名 YYYYYYYYY：コネ						

項番	ファイル名又はディレクトリ名	説明	ファイルサイズの概算値	ファイル数	最大サイズの制限可否	出力される製品種別	サポートバージョン
	クト通番						

1.4　HiRDB のバージョンアップ

HiRDB をバージョンアップする方法について説明します。

バージョンアップとは，HiRDB のバージョン又はリビジョンをアップグレードすること（VV-RR-ZZ の形式で示す HiRDB のバージョン番号のうち，VV 又は RR が上がること）を指します。

HiRDB/シングルサーバをバージョンアップする場合は，それに対応するユティリティ専用ユニットもバージョンアップしてください。HiRDB/シングルサーバとそれに対応するユティリティ専用ユニットのバージョンを合わせてください。

HiRDB/パラレルサーバをバージョンアップする場合は，すべてのユニットをバージョンアップして，HiRDB/パラレルサーバを構成するすべてのユニット間のバージョンを合わせてください。

注意事項

- HiRDB をバージョンアップするときに，**旧バージョンの HiRDB をアンインストールしないでください。**新バージョンの HiRDB を上書きインストールしてください。
- セキュリティ監査機能を使用している場合にバージョンアップするときは，注意事項があります。注意事項については，マニュアル「HiRDB システム運用ガイド」を参照してください。
- Java ストアドプロシジャ及び Java ストアドファンクションを使用している場合に 07-03 以降にバージョンアップするときは，注意事項があります。注意事項については，「Java ストアドプロシジャ及び Java ストアドファンクションを使用する場合」を参照してください。
- 暗号化列を含む表を使用している場合，10-05 以前から 10-06 以降へのバージョンアップについては，「暗号化列を含む表を使用している場合」を参照してください。

1.4.1　バージョンアップ前にすること

バージョンアップをする前に，次に示す内容を必ず実施してください。

また，セキュリティ監査機能を使用している場合は，マニュアル「HiRDB システム運用ガイド」の「バージョンアップ時の注意事項」も参照してください。

なお，マルチ HiRDB でライブラリの共用化を行っている場合，すべての運用ディレクトリに対して，次の作業を実施し，ライブラリの共用化を解除してください。

- HiRDB がオンライン状態であるかどうかの確認
- HiRDB の状態確認
- ライブラリの共用化の解除

(1) 空き領域の確認

データベース状態解析ユティリティ（pddbst）で，データディクショナリ用 RD エリアに必要な空き領域があるかどうかを確認してください。空き領域がない場合は，次に示すどちらかの方法で空き領域を確保してください。

- データベース再編成ユティリティ（pdrorg）でディクショナリ表を再編成します。
- データベース構成変更ユティリティ（pdmod）でデータディクショナリ用 RD エリアを拡張します。

この空き領域の確認は，旧バージョンと新バージョンを入れ替えるときにだけ必要です。修正版 HiRDB への入れ替えの場合は不要です。

ユティリティを実行する方法については，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。

バージョンアップに必要な空き領域

バージョンアップをする前の HiRDB のバージョンに応じて，次の表に示す空き領域があるかどうかを確認してください。空き領域が不足していると，バージョンアップ後の HiRDB 開始時又は pdvrup コマンド実行時に容量不足でエラーとなることがあります。

表 1-3　バージョンアップするのに必要な空き領域

データディクショナリ用 RD エリアに格納しているディクショナリ表	データディクショナリ用 RD エリアに必要な空きセグメント数			
	07-00 以降からバージョンアップする場合	08-00 以降からバージョンアップする場合	09-00 以降からバージョンアップする場合	10-00 以降からバージョンアップする場合
SQL_COLUMNS 表	↑71÷S↑	↑33÷S↑	↑22÷S↑	－
SQL_INDEXES 表	↑9÷S↑	↑5÷S↑	↑4÷S↑	－
SQL_INDEX_COLINF 表	↑11÷S↑	↑4÷S↑	↑4÷S↑	↑4÷S↑
SQL_VIEWS 表	↑30÷S↑	↑23÷S↑	↑16÷S↑	↑4÷S↑
SQL_VIEW_DEF 表	↑234÷S↑	↑107÷S↑	↑69÷S↑	↑5÷S↑
SQL_VIEW_TABLE_USAGE 表	↑6÷S↑	↑5÷S↑	↑5÷S↑	－
SQL_TABLES 表	↑14÷S↑	↑8÷S↑	↑8÷S↑	－
SQL_TABLE_PRIVILEGES 表	↑10÷S↑	↑5÷S↑	↑2÷S↑	－
SQL_USER 表	↑2÷S↑	↑2÷S↑	↑2÷S↑	↑2÷S↑
SQL_ACCESS_SECURITY 表	↑2÷S↑	↑2÷S↑	↑2÷S↑	↑2÷S↑
SQL_ROUTINES 表※	↑14÷S↑	↑14÷S↑	－	－

データディクショナリ用 RD エリアに格納している ディクショナリ表	データディクショナリ用 RD エリアに 必要な空きセグメント数			
	07-00 以降 から バージョン アップ する場合	08-00 以降 から バージョン アップ する場合	09-00 以降 から バージョン アップ する場合	10-00 以降 から バージョン アップ する場合
SQL_ROUTINE_PARAMS 表※	↑14÷S↑	↑10÷S↑	－	－

（凡例）

－：該当しません。

S：対象表を格納するデータディクショナリ用 RD エリアのセグメントサイズ

注※

データディクショナリ LOB 用 RD エリアを定義していない場合は不要です。

(2) システム用 RD エリアのバックアップの取得

データベース複写ユティリティ（pdcopy）で，次に示す RD エリアのバックアップを取得してください。

- 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
- 監査証跡表を格納している RD エリア（セキュリティ監査機能を使用している場合）

ただし，バージョンアップに成功した後にバージョンダウンを行う場合（例えば，テストのために一度バージョンアップした後，バージョンダウンして元の運用に戻す場合など）は，全 RD エリアのバックアップを取得しておく必要があります。

バックアップの取得手順

1. pdstop コマンドで HiRDB を正常終了させます。
2. pdstart -r コマンドで HiRDB を開始します。
3. データベース複写ユティリティ（pdcopy）で RD エリアのバックアップを取得します。このとき，参照・更新不可能モード（-M x 指定）を指定してください。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」又は「HiRDB コマンドリファレンス」を参照してください。

(3) HiRDB がオンライン状態であるかどうかの確認

pdls コマンドですべてのユニットが ACTIVE と表示されているかどうかを確認してください。ACTIVE と表示されている場合，pdstop コマンドで正常終了させてください。

(4) HiRDB の正常終了

バージョンアップする前に HiRDB を正常終了させてください。HiRDB/パラレルサーバの場合はシステムマネジャがあるマシンから終了させてください。HiRDB が既に終了している場合は，次に示す情報を参照して HiRDB が正常終了しているかどうかを確認してください。

- メッセージログファイル又は syslog ファイル

正常終了していない場合は，pdstart コマンドでいったん HiRDB を開始して，pdstop コマンドで正常終了させてください。

(5) HiRDB の状態確認

HiRDB をバージョンアップするユニットの状態を確認するため，pdls -d ust コマンドを実行します。

終了ステータスが 4 の場合（ユニットの状態が STARTING 又は STOPPING）：

HiRDB が開始処理の途中，又は停止処理の途中です。処理が終了してから pdls -d ust コマンドを再度実行してください。

終了ステータスが 8 の場合（ユニットの状態が PAUSE）：

障害によって，プロセスサービスの再起動を中断した状態です。KFPS00715-E メッセージ及びこのメッセージ以前の syslog ファイルに出力されたメッセージを参照して障害の原因を取り除いてから，pdrpause コマンドを実行してください。その後，ユニットを再開始して，pdstop コマンドで正常終了させてください。

(6) ライブラリの共用化の解除

マルチ HiRDB でライブラリを共用化しているときは，pdmemsv -d コマンドでライブラリの共用化を解除してください。そして，バージョンアップ後に，再度 pdmemsv コマンドでライブラリを共用化してください。

(7) コマンド，ユティリティ，アプリケーション，及び HiRDB と連携している製品の停止

コマンド，ユティリティ，アプリケーションは終了させてください。また，HiRDB Datareplicator，HiRDB Dataextractor，及び JP1/PFM などの HiRDB にアクセスする連携製品についても，あらかじめ停止しておいてください。これらを停止しないと，実行形式ファイルや共用ライブラリの削除に失敗し，バージョンアップに失敗することがあります。

(8) HiRDB システム定義のオペランド省略値の確認

HiRDB では，HiRDB のバージョン，リビジョンごとに HiRDB システム定義の省略値を見直して変更しています。バージョン 09-50 以降，オペランド省略時動作として，推奨値を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提供しています。通常は，より安全なシステムを構築するために，指定が必要なオペランド数を大幅に削減した推奨モードの適用を検討してください。バージョン

09-50 以降，特定のバージョンの省略値を仮定する場合は互換モードを適用し，pd_sysdef_default_option オペランドは指定しないでください。

- **バージョン 09-50 より前のバージョンからバージョンアップを行う場合**
  省略値変更によるメリット及びデメリットについて，マニュアル「HiRDB システム定義」の「バージョン，リビジョンによる HiRDB システム定義の変更点」で確認してください。確認の結果，旧バージョンとの互換性を重視する場合は，旧バージョンと同等の省略値になる互換モードを適用してください。ただし，この場合はすべてのオペランドが旧バージョンの省略値となりますので，各オペランドに推奨値を指定することを検討してください。
- **バージョン 07-00 以前からのバージョンアップを行う場合，又は pd_sysdef_default_option オペランドを指定している場合**
  前述の「バージョン 09-50 より前のバージョンからバージョンアップを行う場合」の変更に加えて，マニュアル「HiRDB システム定義」の「バージョン 09-50 より前のバージョンで省略値が変更になったオペランド，及び指定不要になったオペランド」のメリットの説明を参照して，各オペランドの指定値に推奨値を指定することを検討してください。

(9) そのほかの省略値の確認

HiRDB では，ユティリティのオプション，及び SQL のオプションを HiRDB のバージョン，リビジョンごとに見直して変更しています。

バージョン 09-50 より前のバージョンからバージョンアップを行う場合は，省略値変更によるメリット及びデメリットを次の箇所で確認してください。確認の結果，旧バージョンとの互換性を重視する場合は，旧バージョンと同等の省略値になる互換モードを適用してください。ただし，この場合はすべてのオペランドが旧バージョンの省略値となりますので，各オペランドに推奨値を指定することを検討してください。

- マニュアル「HiRDB コマンドリファレンス」の「バージョンアップによって省略値が変更，又は指定不要になったオプション及び制御文」
- マニュアル「HiRDB SQL リファレンス」の「バージョン，リビジョンによる SQL 構文の省略時解釈の変更点」

(10) メモリ所要量の確認

HiRDB をバージョンアップすると，HiRDB のメモリ所要量が増えることがあります。「HiRDB のメモリ所要量」を参照して HiRDB のメモリ所要量を確認してください。

(11) ステータスファイルの容量の確認

HiRDB をバージョンアップすると，HiRDB のステータスファイル容量が増えることがあります。「ステータスファイルの容量の見積もり」を参照して HiRDB のステータスファイルの容量を確認してください。

(12)　シンクポイントダンプファイルの容量の確認

HiRDB をバージョンアップすると，HiRDB のシンクポイントダンプファイルの容量が増えることがあります。「シンクポイントダンプファイルの容量の見積もり」を参照して HiRDB のシンクポイントダンプファイルの容量を確認してください。

(13)　OS のオペレーティングシステムパラメタの確認

HiRDB をバージョンアップすると，OS のオペレーティングシステムパラメタ（カーネルパラメタ）の値が変更になることがあります。オペレーティングシステムパラメタの見積もり方法については，「OS のオペレーティングシステムパラメタの見積もり」を参照してください。

備考

次に示す条件をすべて満たす場合はオペレーティングシステムパラメタの値が増えるため，必ず見積もり直してください。

- バージョン 06-00 以前から 06-01 以降にバージョンアップする場合
- システムログファイルを 31 グループ以上作成する場合

(14)　システムログファイルの総レコード数の確認

バージョンアップする場合は，上書きできる状態のシステムログファイルの総レコード数を確認してください。次の表に示す総レコード数より少ないと，バージョンアップに失敗することがあります。

表 1-4　バージョンアップするのに必要なシステムログファイルの総レコード数

対象バージョン	総レコード数※		
	システムログファイルのレコード長		
	1024 の場合	2048 の場合	4096 の場合
07-00 以降から バージョンアップする場合	4700	2440	1340
08-00 以降から バージョンアップする場合	2640	1360	730
09-00 以降から バージョンアップする場合	1670	870	480
10-00 以降から バージョンアップする場合	280	150	90

なお，HiRDB/パラレルサーバの場合は，ディクショナリサーバのシステムログファイル（上書きできる状態）の総レコード数を確認してください。

注※

次に示すどちらかの方法でシステムログファイルの総レコード数を確認してください。

- pdloginit コマンドの-n オプションの指定値の合計が総レコード数となります。
- pdlogls -d sys -s サーバ名 -e コマンドを実行してください。実行結果の Recode-count の先頭部分に出力されたレコード数（16 進数）の合計が総レコード数となります。

(15) HiRDB 運用ディレクトリ下のファイルのバックアップの取得

バージョンアップの失敗に備えて，HiRDB 運用ディレクトリ下（$PDDIR/conf 下）のファイルのバックアップを取得してください。取得したバックアップは，新バージョンの動作確認後に削除してください。HiRDB 運用ディレクトリのバックアップの取得方法については，「HiRDB 運用ディレクトリのバックアップの取得」を参照してください。

(16) 付加 PP のバージョンアップ

バージョンアップ前の HiRDB で付加 PP を使用していた場合，HiRDB と同じバージョンの付加 PP をインストールする必要があります。付加 PP については，「付加 PP のインストール」を参照してください。

(17) 追加された予約語の確認

SQL の拡張に伴って，HiRDB の各バージョンで次の予約語を追加しています。SQL 文中に，予約語と同じ名前を引用符で囲まないで使用している場合は，バージョンアップ後に SQL 文が文法エラーになることがあります。

HiRDB のバージョン	追加した予約語
06-00	GET_JAVA_STORED_ROUTINE_SOURCE, IS_USER_CONTAINED_IN_HDS_GROUP
06-01	なし
06-02	BIT_AND_TEST
07-00	CONDITION, EXIT, HANDLER, TIMESTAMP_FORMAT, VARCHAR_FORMAT
07-01	FREE, LOCATOR
07-02	なし
07-03	OVER
08-00	ENCRYPT
08-01	なし
08-02	COUNT_FLOAT, SQLCODE_OF_LAST_CONDITION,

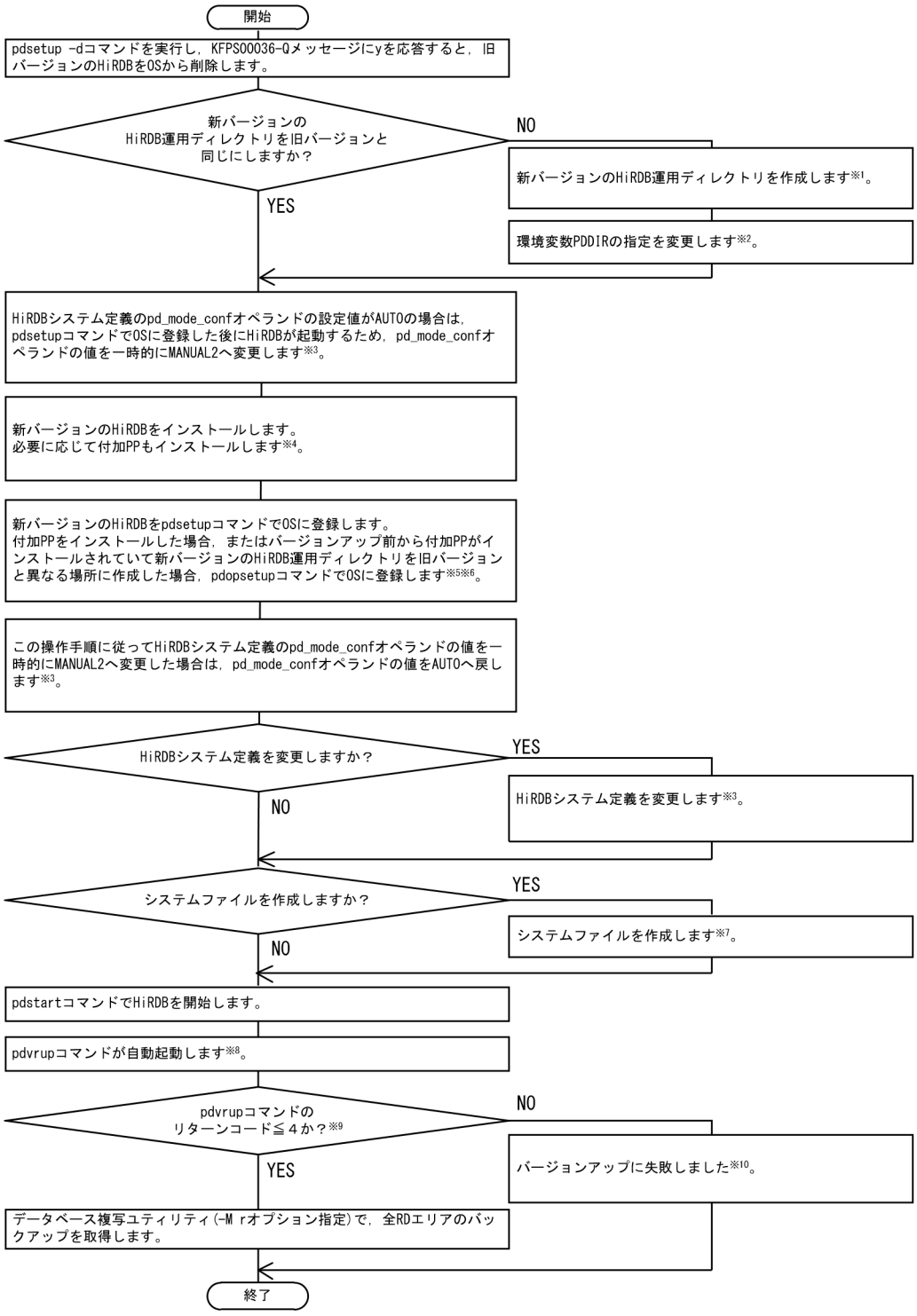
HiRDB のバージョン	追加した予約語
	SQLERRM_OF_LAST_CONDITION, XML, XMLAGG, XMLEXISTS, XMLQUERY, XMLSERIALIZE
08-03	なし
08-04	XMLPARSE
08-05	なし
09-00	なし
09-01	なし
09-02	なし
09-03	COMPRESSED
09-04	TRUNCATE

引用符で囲んでいない名前が，追加された予約語と重複している場合は，マニュアル「HiRDB SQL リファレンス」の「SQL の予約語と重複したときの対応」を参照して，対処してください。

1.4.2 旧バージョンと新バージョンを入れ替える場合

旧バージョンと新バージョンを入れ替える場合の操作手順を次の図に示します。

図 1-2　旧バージョンと新バージョンを入れ替える場合の操作手順



注※1

手順は，「HiRDB 運用ディレクトリの作成」を参照してください。

注※2

手順は，「環境変数の設定」を参照してください。

注※3

手順については，「HiRDB システム定義（UAP 環境定義を除く）の変更方法」を参照してください。

注※4

それぞれの手順については，「HiRDB のインストール」及び「付加 PP のインストール」を参照してください。

注※5

それぞれの登録手順については，「HiRDB 及び付加 PP の OS への登録」を参照してください。

注※6

リビジョンをアップグレードする場合は，バージョンアップ前から付加 PP がインストールされていても pdopsetup コマンドを実行する必要はありません。新バージョンと旧バージョンの運用ディレクトリが同じであれば付加 PP はそのまま引き継がれます。

注※7

手順については，「システムファイルの作成」を参照してください。

注※8

- システム共通定義で pd_auto_vrup = N を指定すると，pdvrup コマンドは自動起動しません。この場合，KFPS05203-Q メッセージ（pdvrup コマンドの入力要求メッセージ）が出力されたら，HiRDB 管理者が pdvrup コマンドを入力してください。
- 修正版 HiRDB への入れ替えの場合，pdvrup コマンドは自動起動しません。次の手順に進んでください。

注※9

pdvrup コマンドの実行結果は，メッセージログファイル又は syslog ファイルにある KFPX24404-I メッセージを検索して確認してください。

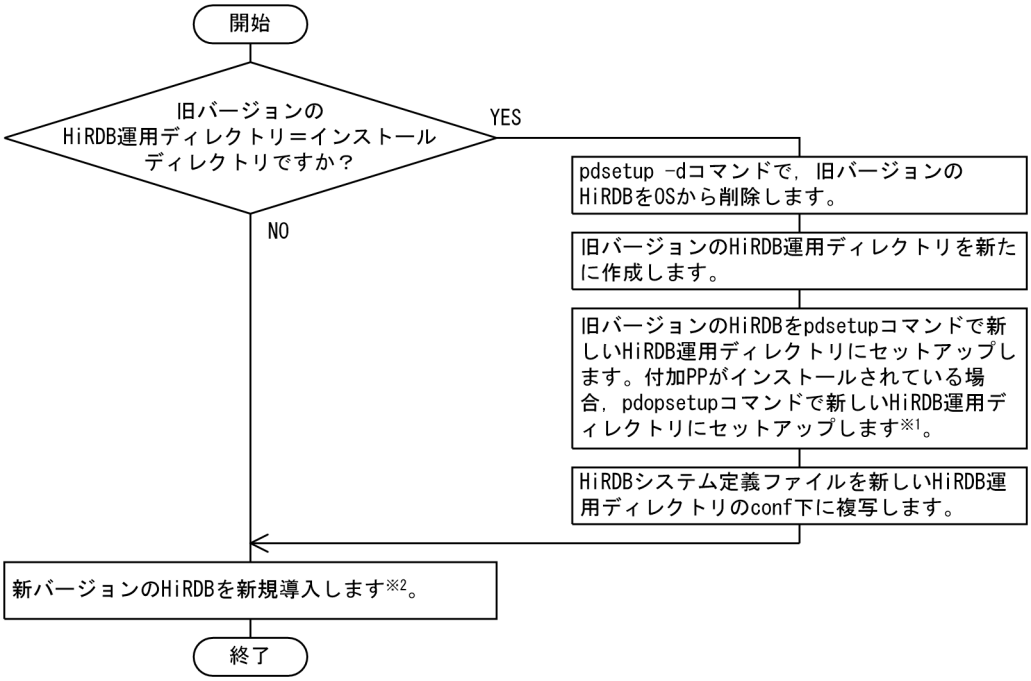
注※10

「バージョンアップに失敗した場合」を参照してください。

1.4.3 旧バージョンを残して，新バージョンを導入する場合

旧バージョンを残して，新バージョンを導入する場合（旧バージョンと新バージョンとでマルチ HiRDB を構成する場合）の操作手順を次の図に示します。

図 1-3　旧バージョンを残して，新バージョンを導入する場合（旧バージョンと新バージョンとでマルチ HiRDB を構成する場合）の操作手順



注※1

リビジョンをアップグレードする場合は，バージョンアップ前から付加 PP がインストールされていても pdopsetup コマンドを実行する必要はありません。インストールディレクトリと旧バージョンの HiRDB 運用ディレクトリが同じであれば付加 PP はそのまま引き継がれます。

注※2

手順については，「システム構築手順」を参照してください。

1.4.4　HiRDB のプラグインをバージョンアップする場合

HiRDB をバージョンアップするときは，プラグインもバージョンアップする必要がある場合があります。プラグインの前提になる HiRDB のバージョンとプラグインのバージョンアップの手順については，該当するプラグインのマニュアル及び「プラグインのバージョンアップ」を参照してください。

1.4.5　Java ストアドプロシジャ及び Java ストアドファンクションを使用する場合

Java ストアドプロシジャ及び Java ストアドファンクションを使用する場合の注意事項を次に示します。

- 使用前に JRE を入手しておく必要があります（各プラットフォームのベンダのホームページから JRE に関する情報を入手できます）。Java ストアドプロシジャ及び Java ストアドファンクションを使用す

るために必要な JRE のバージョンについては，マニュアル「HiRDB システム運用ガイド」を参照してください。

- 使用する JRE のルートディレクトリを pd_java_runtimepath オペランドに指定する必要があります。また，必要に応じて Java 仮想マシンのライブラリが格納されているディレクトリを pd_java_libpath オペランドに指定します。pd_java_runtimepath オペランド及び pd_java_libpath 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。

1.4.6 バージョンアップに失敗した場合

ここでは，次に示す現象が発生した場合の対処方法について説明します。

- pdvrup コマンドを入力して 5 以上のリターンコードが返ってきた
- HiRDB 開始時にリターンコードが 5 以上の KFPX24404-I メッセージが出力された

この場合，一緒に出力されたメッセージを参照して対策してください。

(1) HiRDB を終了させなくてもよいとき

失敗の原因を取り除いた後に，再度 pdvrup コマンドを入力してください。

(2) HiRDB を終了させないといけないとき

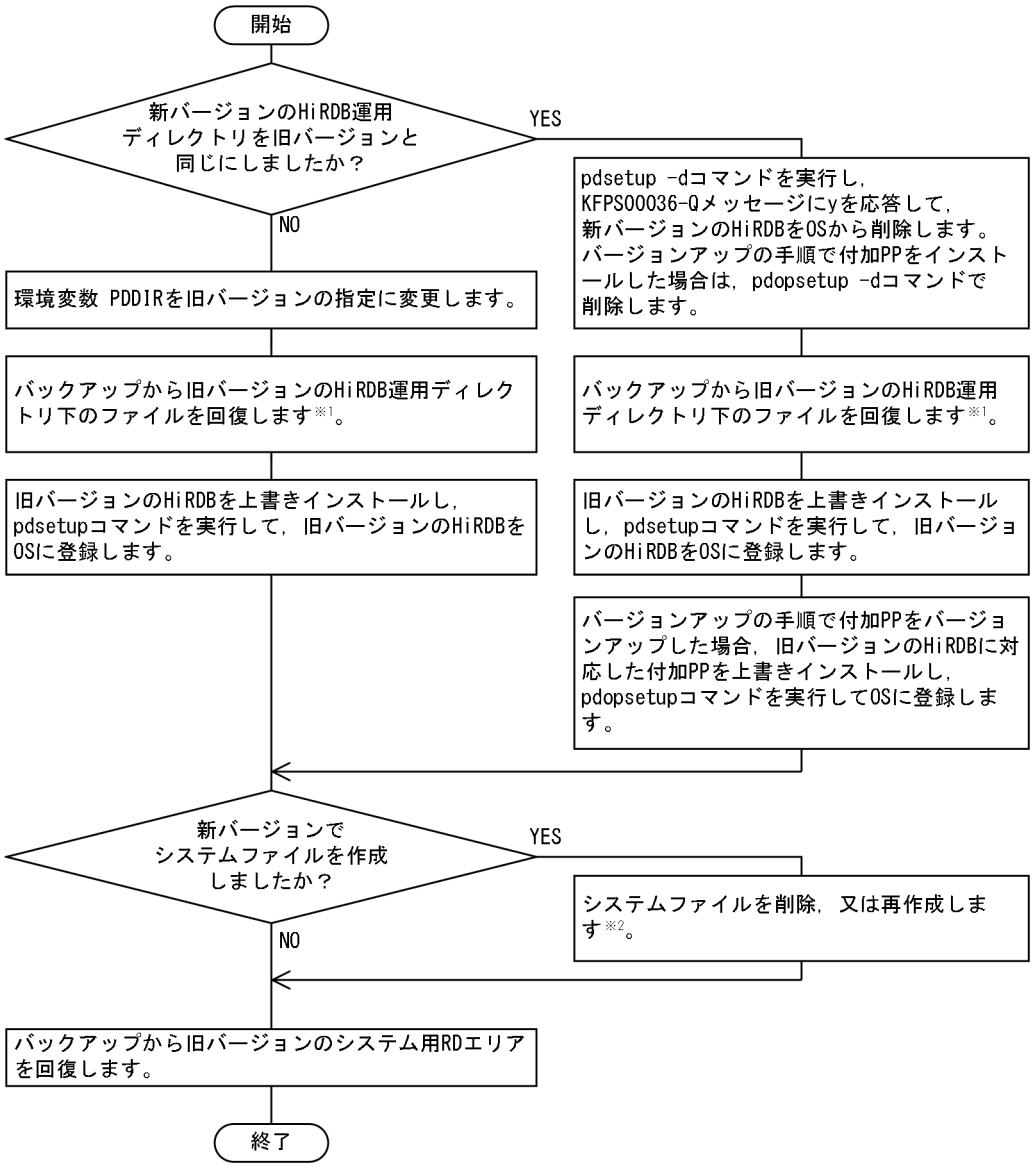
HiRDB を終了しないと，失敗の原因が取り除けない場合は，いったん HiRDB を終了してください。そして，失敗の原因を取り除いた後に，pdstart コマンドで HiRDB を開始してください。開始すると，pdvrup コマンドの入力要求メッセージ（KFPS05203-Q）が出力されるので，再度 pdvrup コマンドを入力してください。

(3) HiRDB を旧バージョンに戻さないといけないとき

失敗の原因によっては，HiRDB を旧バージョンに戻して対処する必要があります。例えば，データディクショナリ用 RD エリアの容量が不足していてバージョンアップに失敗した場合は，HiRDB を旧バージョンに戻してデータベース構成変更ユティリティ（pdmod）で対策する必要があります。このような場合は，いったん HiRDB を旧バージョンに戻して，失敗の原因を取り除き，その後で再度バージョンアップをしてください。

HiRDB を旧バージョンに戻す手順を次の図に示します。

図 1-4　HiRDB を旧バージョンに戻す手順（バージョンアップに失敗した場合）



注※1

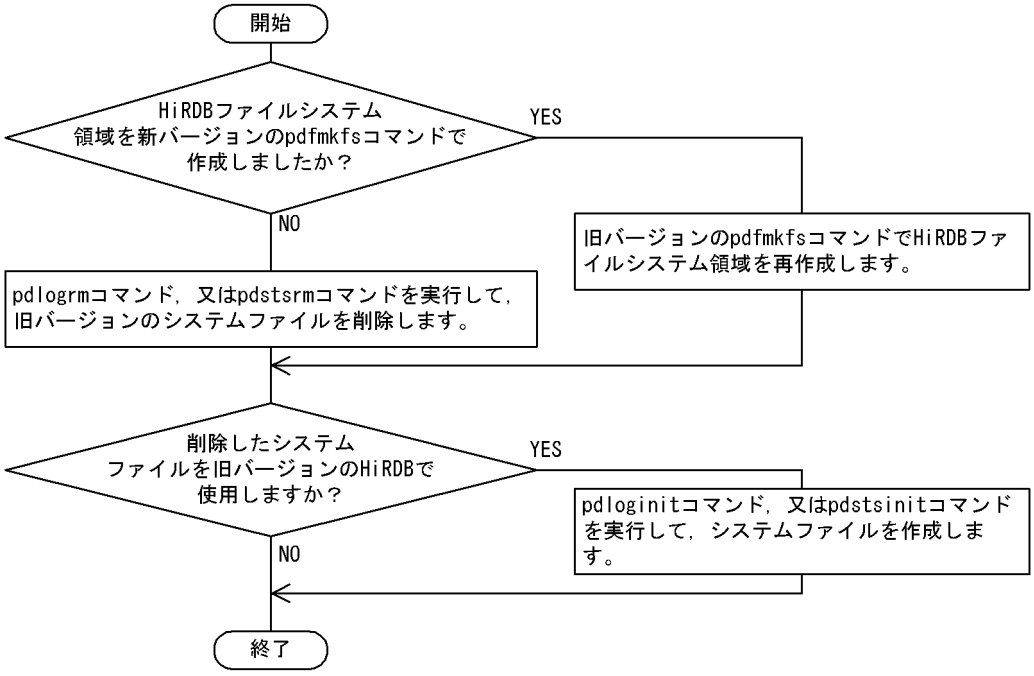
バックアップがない場合でも，$PDDIR/conf 下のシステム定義ファイルが残っていれば，それらのファイルをそのまま使用できます。バックアップがなく，かつ$PDDIR/conf 下のファイルも削除してしまった場合は，システム定義ファイルを作り直してください。

注※2

手順については，次の図を参照してください。

新バージョンで作成したシステムファイルの削除，又は再作成する手順を次の図に示します。

図 1-5　新バージョンで作成したシステムファイルの削除又は再作成手順



1.4.7　HiRDB を旧バージョンに戻す場合

HiRDB のバージョンアップに成功した後にバージョンダウンを行う場合（例えば，テストなどで一度 HiRDB をバージョンアップした後に，バージョンダウンして元の運用に戻したい場合など）は，旧バージョンで再構築する必要があります。

(1)　前提条件

旧バージョンで再構築する場合には，バージョンアップ前にすべての RD エリアのバックアップ，及びシステム定義ファイルのバックアップを取得しておく必要があります。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。

(2)　再構築の手順

再構築の手順は，最初に HiRDB をインストール及び環境設定する手順と基本的に同じです。旧バージョンで再構築する場合の手順を次に示します。

1. HiRDB を停止します（pdstop コマンド）。
2. HiRDB を旧バージョンに戻すことで付加 PP の前提バージョンを満たさなくなる場合，付加 PP を OS から削除します（pdopsetup -d コマンドを実行）。
3. 新バージョンの HiRDB を OS から削除します（pdsetup -d コマンドを実行し，Y を応答）。
4. 新バージョンの HiRDB をアンインストールします。※1

   手順 2.で付加 PP を OS から削除した場合，付加 PP をアンインストールします。

5. 旧バージョンをインストールします。手順 4.で付加 PP をアンインストールした場合，旧バージョンの付加 PP をインストールします。
6. 旧バージョンの HiRDB を OS に登録します（pdsetup コマンドを実行）。
7. 手順 5.で旧バージョンの付加 PP をインストールした場合，付加 PP を OS に登録します（pdopsetup コマンドを実行）。
8. 旧バージョンの環境設定を行います。

   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）。※2

   HiRDB ファイルシステム領域に，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）を作成します（pdloginit コマンド，及び pdstsinit コマンドを実行）。

   なお，新バージョンの HiRDB でシステム定義を変更した場合，バージョンアップ前に取得した旧バージョンのシステム定義ファイルに変更する（旧バージョンの HiRDB のシステム定義に戻す）必要があります。
9. データベース回復ユティリティ（pdrstr）を起動するため，pdstart -r コマンドで HiRDB を開始します。
10. データベース回復ユティリティ（pdrstr）で，バージョンアップ前に取得したバックアップファイルからデータベースを回復します（全 RD エリアのリストア）。このとき，バージョンアップ後の HiRDB で更新したログを含むアンロードログファイルは使用しないでください。
11. HiRDB を停止します（pdstop コマンド）。
12. HiRDB を開始します（pdstart コマンド）。

注※1

マルチ HiRDB の構成で複数の HiRDB システムを運用している場合は，HiRDB をアンインストールすると，サーバマシン内のすべての HiRDB が削除されます。そのため，新バージョンの HiRDB をアンインストールしないで，新バージョンの HiRDB に旧バージョンの HiRDB を上書きインストールしてください。

注※2

新バージョンの HiRDB で pdfmkfs コマンドを実行していない場合，旧バージョンでの HiRDB ファイルシステム領域の作成は不要です。

(3) 注意事項

旧バージョンに戻す場合の注意事項を次に示します。

1. 系切り替え機能を使用している場合

   現用系，予備系ともに旧バージョンに戻してください。
2. リアルタイム SAN レプリケーションを使用している場合

   すべてのデータ反映方式で，メインサイト，リモートサイトを同時に旧バージョンに戻してください。また，ログ同期方式のときは，再構築の手順 12.の後でシステムログ適用化を実行する必要があります。
3. セキュリティ監査機能を使用している場合

   再構築の手順 8.で，システムファイルを作成するとき，監査証跡ファイルも作成してください。

4. HiRDB Datareplicator とのデータ連動機能を適用している場合

- 再構築の手順 1.～7.の間に HiRDB Datareplicator も同時に旧バージョンに戻す必要があります。
- 再構築の手順 12.の後に HiRDB，HiRDB Datareplicator 連動環境を再初期化する必要があります。

1.4.8 バージョンアップ時の留意事項

HiRDB のバージョンアップでディクショナリ表は最新になりますが，既存のリソースによってディクショナリ表の容量が増えるケースがあります。このような場合，ディクショナリ表はメンテナンスされません。しかし，ディクショナリ表をメンテナンスすると，性能向上するケースがあるため，次のケースに該当する場合は，後述するディクショナリ表メンテナンスの方法を参照し，実施してください。

- インナレプリカ機能を使用している場合

  インナレプリカ機能を使用し，かつ RD エリアの数が多い環境で，次の表のどれかの操作を実施する場合に性能向上します。

表 1-5 メンテナンスが必要なケースとディクショナリの表メンテナンス内容

#	メンテナンスが必要なケース	ディクショナリ表のメンテナンスの内容		
		機能項目	表名	内容
1	次のどれかの定義系 SQL を実施 • CREATE TABLE • CREATE INDEX • ALTER TABLE • DROP TABLE • DROP INDEX	インデクスの追加	SQL_RDAREAS	次のインデクスを追加（インデクス名(種別)） • SQLINDEXM106(151) • SQLINDEXS46(152)
2	データベース構成変更ユティリティ(pdmod)で RD エリアの再初期化(initialize rdarea 文)を実施			
3	次のどれかのコマンドを実施 • pdorbegin • pdorchg • pdorend • pddbchg			

(1) ディクショナリ表のメンテナンスの方法

HiRDB のバージョンアップ後に，データベース再編成ユティリティ（pdrorg）を実行することで，ディクショナリ表のメンテナンスを行います。運用手順の詳細は，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」の「ディクショナリ表のメンテナンス」を参照してください。

(2) ディクショナリ表のメンテナンスの前にすること

ディクショナリ表のメンテナンスをする前に，次に示す内容を必ず確認してください。

1. データディクショナリ用 RD エリアの容量の確認

   ディクショナリ表のメンテナンスでインデクスを追加することで，データディクショナリ用 RD エリアの容量が増えます。表「メンテナンスが必要なケースとディクショナリの表メンテナンス内容」の該当するディクショナリ表について，「データディクショナリ用 RD エリアの容量の見積もり」の「インデクスの格納ページ数の計算方法」を参照してください。

2. システムログファイルの容量の確認

   ディクショナリ表のメンテナンスでインデクスを追加することで，システムログを出力します。システムログ量について，「ディクショナリ表のメンテナンス実行時に出力されるシステムログ量」を参照してください。

3. 作業表用ファイルの容量の確認

   ディクショナリ表のメンテナンスでインデクスを追加することで，シングルサーバ又はディクショナリサーバに，インデクスデータの一時的な情報を格納するための領域が必要となります。作業表用ファイルの容量について，「ディクショナリ表のメンテナンスが使用する作業表用ファイルの容量」を参照してください。

4. 排他資源要求数の確認（システム定義 pd_lck_pool_size 及び pd_fes_lck_pool_size の値の確認）

   ディクショナリ表のメンテナンスでインデクスを追加することで，シングルサーバ又はディクショナリサーバで，排他資源を使用します。排他資源要求数について，マニュアル「HiRDB システム定義」の「排他資源数の見積り」の「ディクショナリ表のメンテナンス（pdrorg -k maintenance -c dic)」を参照してください。

(3) ディクショナリ表のメンテナンス実行要否の確認方法

ディクショナリ表のメンテナンスの実行要否は，次に示す方法で確認してください。

インデクスの有無の確認方法

DBA 権限者が，ディクショナリ表 SQL_INDEXES を，表「メンテナンスが必要なケースとディクショナリの表メンテナンス内容」の該当するインデクス名で検索して，情報の有無によって確認します。情報がある場合は，ディクショナリ表のメンテナンスを実行する必要はありません。

（例）インデクス SQLINDEXM106 の有無を確認する

```
SELECT INDEX_NAME FROM MASTER.SQL_INDEXES
WHERE INDEX_NAME = 'SQLINDEXM106'
```

(4) ディクショナリ表のメンテナンスの注意事項

インデクスの削除について

ディクショナリ表のメンテナンスでインデクスを追加すると，追加したインデクスは削除できません。インデクスを追加する前の状態に戻すには，ディクショナリ表メンテナンス実行前に取得したバック

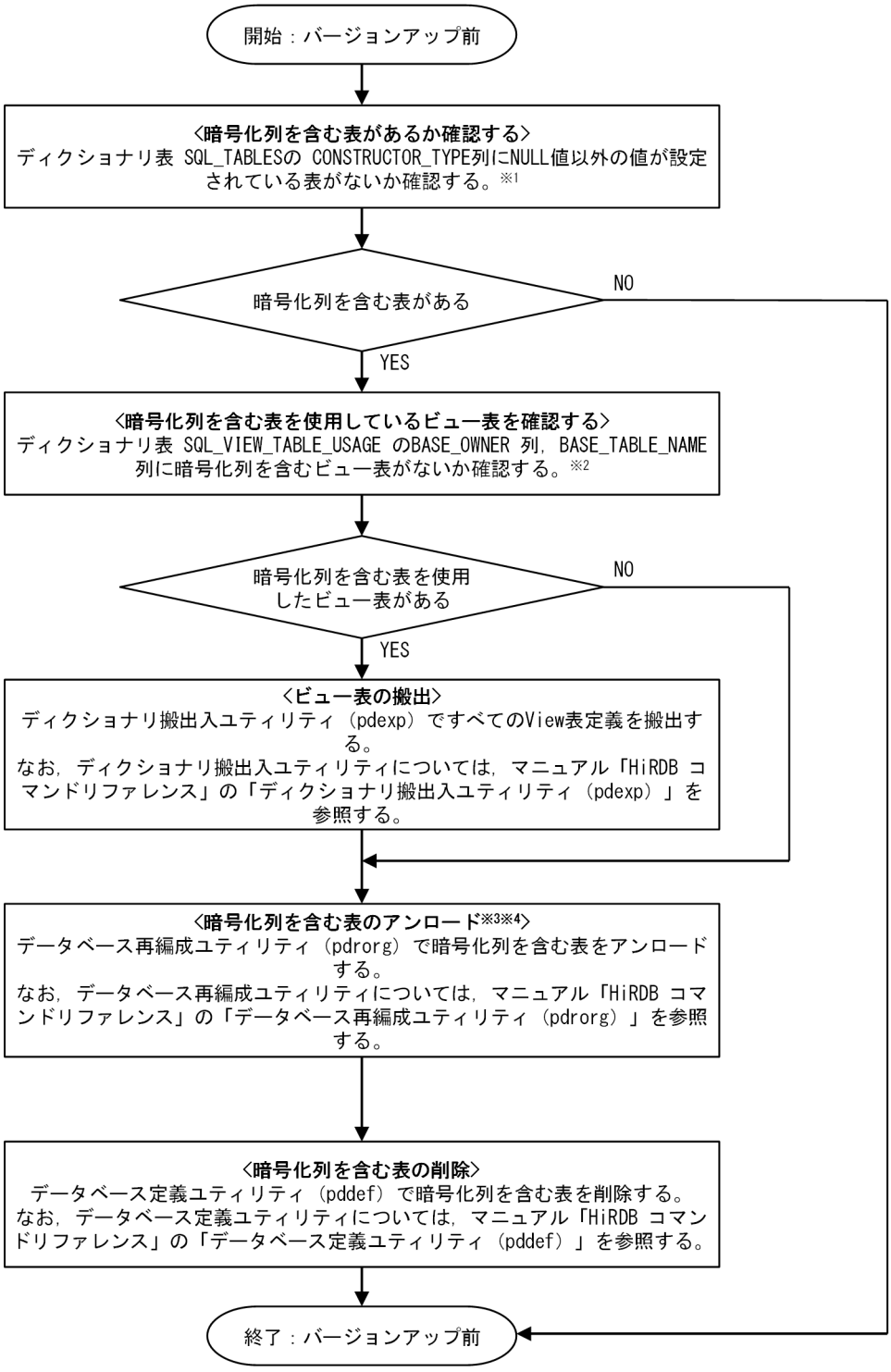
アップを使用して回復してください。詳細はマニュアル「HiRDB コマンドリファレンス」の「データベース再編成ユティリティ」の「ディクショナリ表のメンテナンス」を参照してください。

1.4.9 暗号化列を含む表を使用している場合

暗号化列を含む表が HiRDB に定義されていると，pdvrup コマンドが異常終了します。暗号化列を含む表を使用している場合，10-05 以前から 10-06 以降に HiRDB をバージョンアップする際，バージョンアップ前に，暗号化列を含む表はすべて搬出し暗号化列を含む表を削除してください。バージョンアップ後に，搬出した表を再定義し，データを搬入してください。

暗号化列を含む表を使用している場合の，HiRDB 10-05 以前から 10-06 以降にバージョンアップする前，及びバージョンアップ後に必要な手順を次の図に示します。

(1) バージョンアップ前の手順



注※1

次の SQL 文で暗号化列を含む表の一覧が取得できます。

```
SELECT TABLE_SCHEMA,TABLE_NAME FROM MASTER.SQL_TABLES WHERE CONSTRUCTOR_TYPE IS NOT NULL
WITHOUT LOCK NOWAIT;
```

注※2

次の SQL 文で暗号化列を含む表を使用しているビュー表の一覧が取得できます。

```
SELECT VIEW_SCHEMA,VIEW_NAME FROM MASTER.SQL_VIEW_TABLE_USAGE AS V INNER JOIN MASTER.SQL_
TABLES AS T ON V.BASE_OWNER=T.TABLE_SCHEMA AND V.BASE_TABLE_NAME= T.TABLE_NAME WHERE CONS
TRUCTOR_TYPE IS NOT NULL WITHOUT LOCK NOWAIT;
```

注※3

暗号化列を含む表を使用した次の定義についても，バージョンアップ後に再定義が必要になります。

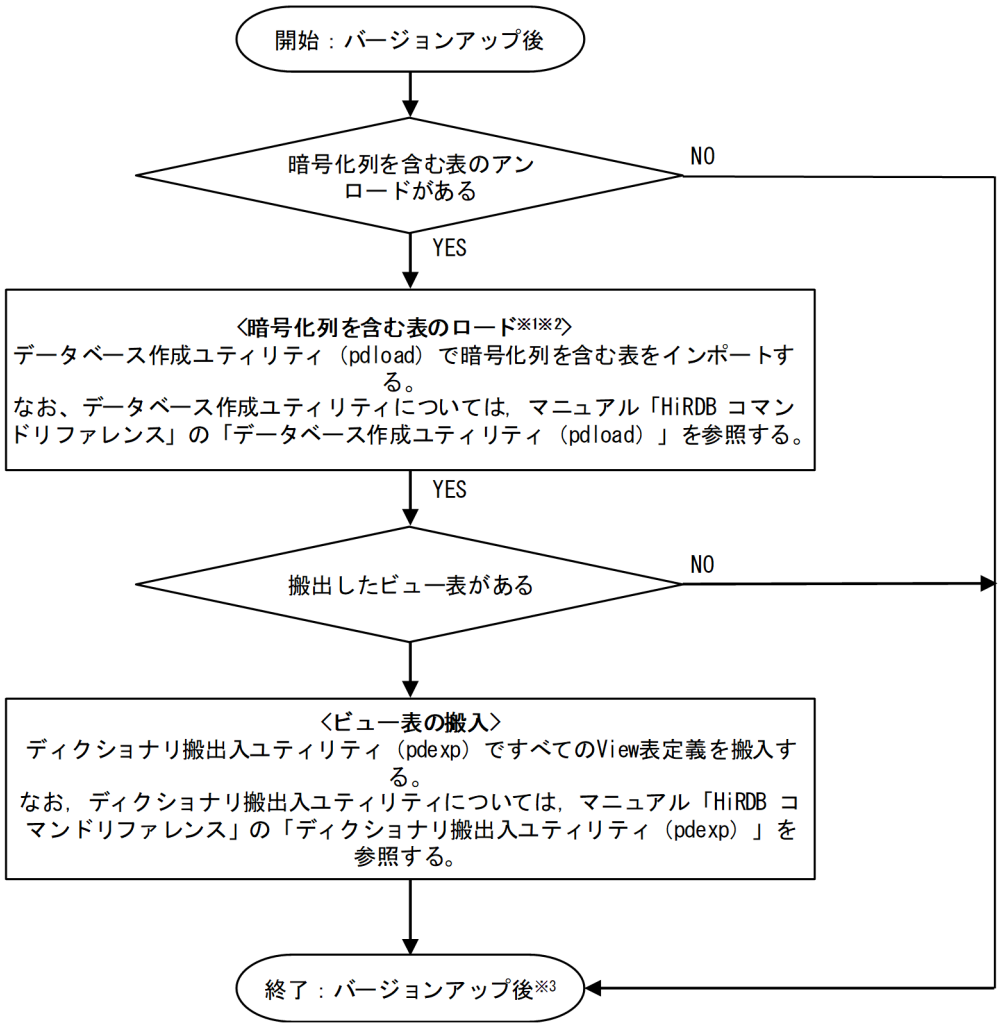
- 暗号化列を含む表を操作している手続き
- 暗号化列を含む表の更新で動作するトリガー
- 暗号化列を含む表を操作しているトリガー
- 暗号化列を含む表を参照制約で使用している表

注※4

次のオプションを指定して表移行用アンロードファイルを作成してください。

```
pdrorg -k unld -W bin -w -t［暗号化列を含む表名］［コントロールファイル］
```

(2)　バージョンアップ後の手順



注※1

暗号化列を含む表を使用した次の定義についても，バージョンアップ後に再定義が必要になります。

- 暗号化列を含む表を操作している手続き
- 暗号化列を含む表の更新で動作するトリガー
- 暗号化列を含む表を操作しているトリガー
- 暗号化列を含む表を参照制約で使用している表

注※2

次のオプションを指定して表移行用アンロードファイルをインポートしてください。

```
pdload -W -w all -b ［暗号化列を含む表名］［コントロールファイル］
```

注※3

「バージョンアップ後の手順」後，再定義していたルーチンの SQL オブジェクトを再登録するため，ALTER ROUTINE 文で再登録を行ってください。

1.5　修正版 HiRDB への入れ替え

修正版 HiRDB とは，07-02-01 のように，稼働中の HiRDB とバージョン番号及びリビジョン番号が同じで，「-mn」のコードがあるものです（下線部がコード）。07-02 より前のバージョンの場合，m は/，英字（I，O，P～T を除く），又は数字，n は A～Z のアルファベットです。07-02 以降は m，n は共に数字です。

修正版 HiRDB への入れ替えは，既存の HiRDB を終了しないで，稼働中に実行できます。

HiRDB を終了して修正版 HiRDB を入れ替える場合は，既に適用しているオペランド省略時動作を適用してください。

1.5.1　修正版 HiRDB への入れ替え方法

修正版 HiRDB への入れ替えには，次の方法があります。

- HiRDB を終了して入れ替え
- HiRDB の稼働中に入れ替え

それぞれについて説明します。

1.5.2　HiRDB を終了して入れ替え

HiRDB を終了して入れ替えるには，次の方法があります。

- インストーラによる入れ替え
  インストーラを使用して修正版 HiRDB と入れ替えます。
- 修正パッチを Web から入手して適用
  稼働中の HiRDB と，バージョン及びリビジョン番号が同じ場合，修正パッチを適用することで最新版に入れ替えできます。修正パッチは，Web からダウンロードして入手できます。

なお，どの方法についても，修正版 HiRDB の入れ替え後に pdopsetup コマンドを実行して付加 PP を再登録する必要はありません。

(1)　入れ替えの手順

(a)　インストーラによる入れ替え

インストーラを使用して修正版 HiRDB と入れ替えます。

入れ替え方法は「旧バージョンと新バージョンを入れ替える場合」と同じです。次の箇所を参照して，修正版への入れ替えを行ってください。

- 「HiRDB のバージョンアップ」の説明及び注意
- 「旧バージョンと新バージョンを入れ替える場合」の操作手順※

  注※　手順の中の pdvrup コマンドに関する操作は不要です。

また，修正版への入れ替え前に次のことを行ってください。

1. HiRDB がオンライン状態であるかどうかの確認

   pdls コマンドですべてのユニットが ACTIVE と表示されているかどうかを確認してください。ACTIVE と表示されている場合，手順 2.に進んでください。HiRDB が既に終了している場合は，手順 3.へ進んでください。

2. HiRDB の終了

   HiRDB を任意の終了モードで停止してください。HiRDB/パラレルサーバの場合はすべての HiRDB のユニットを停止してください。

3. HiRDB のユニットの状態確認

   HiRDB のユニットの状態を確認するため，pdls -d ust コマンドを実行してください。終了ステータスが 4 の場合（ユニットの状態が STARTING 又は STOPPING），HiRDB が開始処理の途中，又は停止処理の途中です。処理が終了してから pdls -d ust コマンドを再度実行してください。

   HiRDB/パラレルサーバの場合はすべての HiRDB のユニットで pdls -d ust コマンドを実行して，HiRDB のユニットの状態を確認してください。

4. ライブラリの共用化の解除

   「バージョンアップ前にすること」の「ライブラリの共用化の解除」を行ってください。

5. コマンド，ユティリティ，アプリケーション，及び HiRDB と連携している製品の停止

   「バージョンアップ前にすること」の「コマンド，ユティリティ，アプリケーション，及び HiRDB と連携している製品の停止」を行ってください。

(b)　修正パッチを Web から入手して適用

稼働中の HiRDB と，バージョン及びリビジョン番号が同じ場合，修正パッチを適用することで最新版に入れ替えできます。修正パッチは，Web からダウンロードして入手できます。

適用手順については，修正パッチに添付されている RELEASE.TXT 又は RELEASE.EUC を参照してください。

1.5.3　HiRDB の稼働中に入れ替え

HiRDB の稼働中に入れ替えるには，次の方法があります。

- インストーラによる入れ替え

  インストーラを使用して修正版 HiRDB と入れ替えます。

- 修正パッチを Web から入手して適用

  稼働中の HiRDB と，バージョン及びリビジョン番号が同じ場合，修正パッチを適用することで最新版に入れ替えできます。修正パッチは，Web からダウンロードして入手できます。

なお，どの方法についても，修正版 HiRDB の入れ替え後に pdopsetup コマンドを実行して付加 PP を再登録する必要はありません。

(1) 前提条件

HiRDB の稼働中に入れ替えができるのは，次の条件を満たしている場合です。

- バージョン，HiRDB サーバの種別，アドレッシングモード

  修正版 HiRDB と稼働中の HiRDB とで次に示す項目が同じである必要があります。なお，次の項目は pdadmvr コマンドで確認できます。

  - バージョン番号，リビジョン番号
  - HiRDB サーバの種別（HiRDB/シングルサーバか，HiRDB/パラレルサーバか）
  - アドレッシングモード（32 ビットモードか，64 ビットモードか）
- OS のオペレーティングシステムパラメタ

  入れ替え後の HiRDB が必要な OS のオペレーティングシステムパラメタの見積もりが，現在の設定されているカーネルパラメタの値の範囲内である必要があります。
- HiRDB 運用ディレクトリ

  インストールディレクトリと稼働中の HiRDB 運用ディレクトリが異なるディレクトリである必要があります。
- ライブラリの共用化

  マルチ HiRDB の場合，ライブラリを共用化していない必要があります。
- ディスクの空き容量

  HiRDB 運用ディレクトリに，現在稼働中の HiRDB と修正版 HiRDB の両方が格納できる程度のディスクの空き容量が必要です。修正版 HiRDB を格納するのに必要な空き容量はリリースノートを参照してください。
- HiRDB クライアント

  オンライン業務をしている HiRDB クライアントは，入れ替えをする HiRDB サーバ以外で稼働している必要があります。HiRDB クライアントが入れ替えをする HiRDB サーバ上で稼働している場合，HiRDB クライアントを停止し，オンライン業務を停止する必要があります。
- クライアントライブラリ

  オンライン業務をしている HiRDB クライアントが利用している HiRDB/Developer's Kit 及び HiRDB/Run Time のバージョンは 07-00 以降である必要があります。07-00 より前のバージョンを使用している場合，入れ替えの途中で接続中の HiRDB クライアントとの接続が切断されます。
- 自動再接続機能の適用

HiRDB に接続する HiRDB クライアントは，自動再接続機能（PDAUTORECONNECT=YES）を使用する必要があります。自動再接続機能を使用していない場合，入れ替えの途中で接続中のクライアントとの接続が切断されます。自動再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。

(2) 入れ替えの手順

修正版 HiRDB への入れ替え手順を次に示します。

1. 修正版 HiRDB のインストール

   修正版 HiRDB を上書きインストールします。HiRDB/パラレルサーバの場合，全ユニットで修正版 HiRDB を上書きインストールしてください。インストーラによる入れ替えの場合，インストール方法については「インストール」を参照してください。修正パッチを Web から入手して適用する場合，適用手順については修正パッチに添付されている RELEASE.TXT 又は RELEASE.EUC を参照してください。

2. 入れ替え用ディレクトリに修正版 HiRDB をコピー

   "修正版 HiRDB のインストールディレクトリ/bin/pdprgcopy 稼働中の HiRDB 運用ディレクトリ"コマンドを実行して，インストールディレクトリに格納されている修正版 HiRDB を HiRDB 運用ディレクトリ下の入れ替え用ディレクトリ（$PDDIR/renew）にコピーしてください。HiRDB/パラレルサーバの場合はシステムマネジャがあるユニットで pdprgcopy コマンドを実行してください。

3. HiRDB がオンライン状態であるかどうかの確認

   pdls コマンドで全ユニットが「ACTIVE」と表示されていることを確認してください。

4. 修正版 HiRDB への入れ替え

   "修正版 HiRDB のインストールディレクトリ/bin/pdprgrenew 稼働中の HiRDB 運用ディレクトリ"コマンドを実行して，HiRDB の入れ替えをします。このコマンドを実行すると，稼働中の HiRDB をバックアップ用ディレクトリ（$PDDIR/renew_bak）に退避した後，稼働中の HiRDB を 2.でコピーした入れ替え用ディレクトリの修正版 HiRDB と入れ替えます。HiRDB/パラレルサーバの場合はシステムマネジャがあるユニットで pdprgrenew コマンドを実行してください。

(3) 系切り替え機能使用時の入れ替え手順

系切り替え機能を使用している場合，HiRDB の稼働中に入れ替えができるのは次の場合です。

- スタンバイ型系切り替えの場合

  実行系が現用系で稼働中のときだけです。実行系が予備系で稼働中のときは，系を切り替えてコマンドを実行してください。

- スタンバイレス型系切り替えの場合

  すべての正規 BES が稼働中のときだけです。代替中は入れ替えできません。

系切り替え機能使用時の修正版 HiRDB への入れ替え手順を次に示します。

- サーバモードで運用している場合

  スタンバイ型系切り替え

  1. 予備系が実行系の場合，現用系が実行系になるように系を切り替えてください。
  2. 待機系の HiRDB を停止してください。
  3. 実行系で修正版 HiRDB への入れ替えを実行してください。
  4. 待機系の HiRDB を pdsetup -d コマンドで OS から削除してください（KFPS00036-Q メッセージに y を応答）。
  5. 待機系に修正版 HiRDB を上書きインストールしてください。
  6. 待機系の HiRDB を pdsetup コマンドで OS に登録してください。
  7. 1.で停止させた待機系の HiRDB を再起動します。待機系の HiRDB で pdstart コマンド（HiRDB/パラレルサーバの場合は pdstart -q コマンド）を実行してください。

  1：1 スタンバイレス型系切り替え

  1. 代替 BES が稼働中の場合，系を切り戻してください。
  2. 代替 BES ユニットの代替部の待機状態を解除してください。解除方法については，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. 実行系の正規 BES ユニットで修正版 HiRDB への入れ替えを実行してください。

     2.で待機状態を解除した代替部は pdprgrenew コマンドを実行すると自動的に待機状態になるため，これ以降の操作は不要です。

  影響分散スタンバイレス型系切り替え

  1. 受け入れユニットのゲスト BES が稼働中の場合，正規ユニットに系を切り戻してください。
  2. HA グループに属する，すべての稼働していないゲスト BES の受け入れ可能状態を解除してください。
  3. 実行系の正規ユニットで修正版 HiRDB への入れ替えを実行してください。

     2.で受け入れ可能状態を解除したゲスト BES は，pdprgrenew コマンドを実行すると自動的に受け入れ可能状態になるため，これ以降の操作は不要です。

- モニタモードで運用している場合

  1. 予備系が実行系の場合，現用系が実行系になるように系を切り替えてください。
  2. 実行系で修正版 HiRDB への入れ替えを実行してください。
  3. 待機系の HiRDB を pdsetup -d コマンドで OS から削除してください（KFPS00036-Q メッセージに y を応答）。
  4. 待機系に修正版 HiRDB を上書きインストールしてください。
  5. 待機系の HiRDB を pdsetup コマンドで OS に登録してください。

(4) 注意事項

- 修正版 HiRDB へ入れ替えの実行不可

  HiRDB の稼働状況によっては，修正版 HiRDB の入れ替えが実行できないことがあります。詳細については，マニュアル「HiRDB コマンドリファレンス」の pdprgrenew コマンドの説明を参照してください。

- UAP のレスポンス遅延

  pdprgrenew コマンドを実行している時間は，UAP のレスポンスが遅くなります。このため，比較的トラフィックが低い時間帯に実行することをお勧めします。

- 定義の変更

  修正版 HiRDB への入れ替えに伴って，必要となるメモリサイズが変わり，システム定義の変更が必要となる場合があります。その場合は事前にシステム構成変更コマンド（pdchgconf コマンド）で HiRDB システムの定義を変更する必要があります。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。

- 運用コマンド，ユティリティの実行

  pdprgrenew コマンド実行中は運用コマンドやユティリティを実行しないでください。実行すると，HiRDB が停止している旨のエラーが表示されたり，HiRDB の入れ替え作業が失敗することがあります。

- 系切り替え機能使用不可

  修正版への入れ替え中は系切り替え機能は使用できません。

- ホールダブルカーソルの無効

  修正版への入れ替えではカーソル保持ができないので，ホールダブルカーソルを使用する UAP は入れ替え前後で実行できません。そのため，UAP はエラーとなります。

- UNTIL DISCONNECT 指定の LOCK 文が無効

  修正版への入れ替えでは UNTIL DISCONNECT 指定の排他を保持できないため，UNTIL DISCONNECT 指定の LOCK 文を使用する UAP は修正版への入れ替え前後で実行できません。そのため，UAP はエラーとなります。

(5) 運用上の注意事項

修正版 HiRDB へ入れ替えるときの運用上の注意事項を次に示します。

- データベース構成変更ユティリティ（pdmod）で割り当てたグローバルバッファが無効になります。そのため，入れ替え後に再度グローバルバッファを割り当てる必要があります。
- pd_spool_cleanup_interval オペランドの時間のカウント開始時点が入れ替え時の時刻にリセットされます。
- pd_spool_cleanup オペランドに normal，又は force を指定している場合，入れ替え時に出力済みのトラブルシュート情報は削除されます。
- pdstbegin コマンドや pdstend コマンドを使用して，統計情報の取得条件を pd_statistics オペランドや pdstbegin オペランドの指定と異なる値に変更していた場合，次のようになります。

- pd_statistics オペランドや pdstbegin オペランドを指定しないで起動した環境で pdstbegin コマンドで統計情報を取得していた場合，入れ替え後には統計情報を取得しなくなります。そのため，再度 pdstbegin コマンドを実行する必要があります。
- pd_statistics オペランドに A 若しくは Y を指定，又は pdstbegin オペランドを指定して起動した環境で pdstend コマンドで統計情報取得を中止した場合や，pdstbegin コマンドを実行して取得する統計情報の種類を変更していた場合，入れ替え後にはシステム共通定義に記述した指定で統計情報を取得します。そのため，再度 pdstend コマンドや pdstbegin コマンドを実行する必要があります。

- 絞込み検索で使用しているリストがなくなるため，入れ替え後に ASSIGN LIST 文でリストを再作成する必要があります。
- pdchprc コマンドで変更した常駐プロセス数は HiRDB システム定義で指定した常駐プロセス数に戻るため，入れ替え後に再度 pdchprc コマンドを実行する必要があります。
- 修正版 HiRDB への入れ替えをすると，システムログファイルが切り替わります。入れ替え前に，スワップできるシステムログファイルがあることを確認し，不足している場合は次の対処をしてください。
  - スワップできるシステムログファイルがない場合

    アンロード待ち状態のファイルがあれば，アンロードしてください。アンロード待ち状態のファイルがなければ，システム構成変更コマンド（pdchgconf コマンド）を使用してスワップできるログファイルを追加してください。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。

    スワップできるシステムログファイルがない状態で入れ替えを実行すると，KFPS01256-E メッセージを出力後，Psjnf07 又は Psjn381 のコードでアボートして HiRDB が停止します。この場合は，スワップできるファイルを準備してから pdstart コマンドで HiRDB を起動してください。
  - スワップできるシステムログファイルが一つだけの場合

    修正版 HiRDB への入れ替えはできますが，入れ替え中にスワップできるファイルがないことを示す KFPS01224-I メッセージが出力されます。入れ替え後にスワップできるシステムログファイルを準備してください。
- 修正版 HiRDB への入れ替えをすると，メッセージログファイルが切り替わります。ただし，メッセージログファイルの切り替えを知らせるメッセージ（KFPS01910-I など）は表示されません。メッセージログファイル中のメッセージを保存したい場合は，入れ替え前にメッセージログファイルをバックアップしてください。

(6) 関連製品の制限及び注意事項

- プラグイン

  プラグインを利用している場合も HiRDB の稼働中に修正版 HiRDB へ入れ替えができます。ただし，プラグインの入れ替えはできません。
- HiRDB Datareplicator 連携機能

  HiRDB Datareplicator を使用してデータ抽出中の HiRDB に対して pdprgrenew コマンドを実行しないでください。オンライン業務を停止しないで修正版 HiRDB への入れ替えをする場合，抽出側の

HiRDB Datareplicator を終了させる必要があります。ただし，HiRDB Datareplicator 連携は中止させないでください（pdrplstop コマンドは実行しないでください）。HiRDB Datareplicator 連携を中止させると，抽出側 DB と反映側 DB が不整合になることがあります。

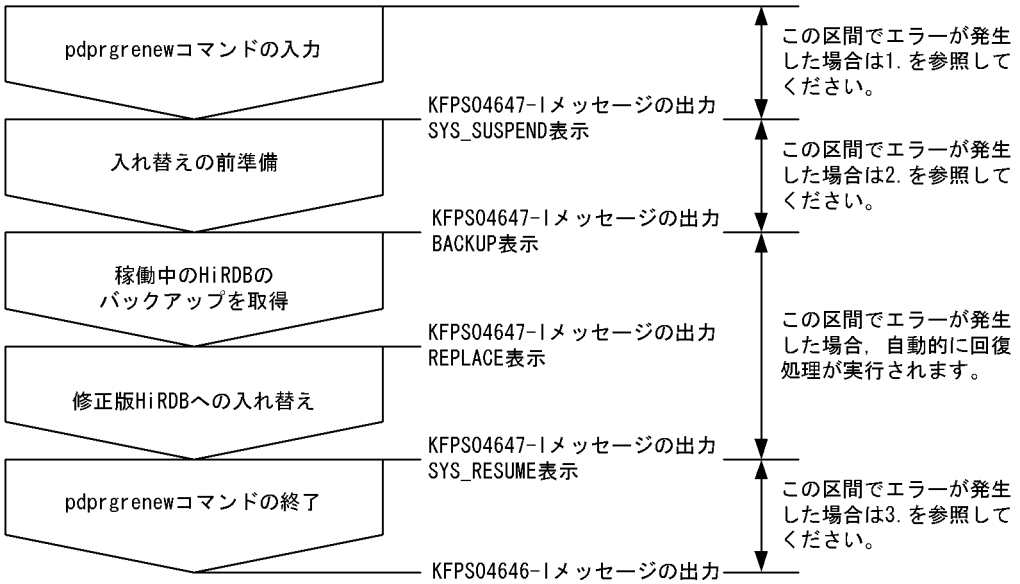
(7) 障害時の運用

(a) エラー発生時の対処

修正版 HiRDB への入れ替え実行中にエラーが発生した場合，pdprgrenew コマンドは自動的に入れ替え前の HiRDB に戻して HiRDB を動作させようとします。エラー発生後に，コマンドがリターンコード 12 の KFPS04646-I メッセージを出力して終了した場合，HiRDB を入れ替え前に戻す作業が失敗しています。そのため，標準エラー出力や syslogfile に出力されたエラーメッセージと KFPS04647-I メッセージを参照して対処してください。

修正版 HiRDB への入れ替え中にエラーが発生した場合の対処方法を次の図に示します。

図 1-6 修正版 HiRDB への入れ替え中にエラーが発生した場合の対処方法



1. pdprgrenew コマンドがエラーになった要因を取り除いて，pdprgrenew コマンドを再度実行してください。
2. HiRDB のサーバプロセスがあれば，pdstop -f コマンドで HiRDB を強制終了してから pdprgrenew -b コマンドを実行してください。HiRDB のサーバプロセスがなければ，pdprgrenew -b コマンドを実行してください。pdprgrenew -b コマンドを実行すると，回復処理として，入れ替え前の HiRDB で再開始しようとします。

   また，HiRDB の終了処理失敗に関するエラーメッセージ及びアボートコードが表示されることがあります。メッセージの対処方法に従って，入れ替え前の HiRDB の稼働環境を確認し，対処してください。
3. HiRDB のサーバプロセスがあれば，pdstop -f コマンドで HiRDB を強制終了してから pdprgrenew -b コマンドを実行してください。HiRDB のサーバプロセスがなければ，pdprgrenew -b コマンドを

実行してください。pdprgrenew -b コマンドを実行すると，回復処理として，修正版 HiRDB を入れ替え用ディレクトリに戻します。

また，HiRDB の開始処理失敗に関するエラーメッセージ及びアボートコードが表示されることがあります。入れ替え後の修正版 HiRDB が動作するための環境に問題がある可能性があるので，表示されるメッセージに従って対処してください。

(b) 入れ替え失敗時に HiRDB が入れ替え前の状態に戻っているかどうかの確認

修正版 HiRDB への入れ替えに失敗したとき，入れ替え前の HiRDB に戻っているかどうかは，次の項目をチェックして確認できます。次の項目を満たしていれば，入れ替え前の HiRDB に戻っています。

- pdadmvr -s コマンドで表示されるバージョンが入れ替え前の HiRDB バージョンと一致する
- HiRDB がオンライン状態（pdls コマンドの結果，全ユニットが「ACTIVE」と表示）である
- バックアップ用ディレクトリ（$PDDIR/renew_bak）がない

1.6　64 ビットモードの HiRDB への移行方法

同一マシンで，32 ビットモードの HiRDB から 64 ビットモードの HiRDB に移行する方法について説明します。

1.6.1　64 ビットモードに移行する際の考慮点

(1)　互換性がないファイル

32 ビットモードの HiRDB で使用していたファイルは，基本的に 64 ビットモードの HiRDB で使用できます。ただし，次に示すファイルは互換性がないため，64 ビットモードの HiRDB で使用できません。

- バックアップファイル
- マスタディレクトリ用 RD エリア及びデータディレクトリ用 RD エリアを構成する HiRDB ファイル

(2)　省略値が変わるオペランド

HiRDB を 32 ビットモードから 64 ビットモードにすると，一部の HiRDB システム定義のオペランドの省略値が変わります。マニュアル「HiRDB システム定義」の「32 ビットモードと 64 ビットモードで省略値が異なるオペランド」を参照して，変更内容を確認してください。

(3)　メモリ所要量の違い

HiRDB を 32 ビットモードから 64 ビットモードにすると，メモリ所要量が増えます。メモリ所要量の計算方法については，「HiRDB のメモリ所要量」を参照してください。

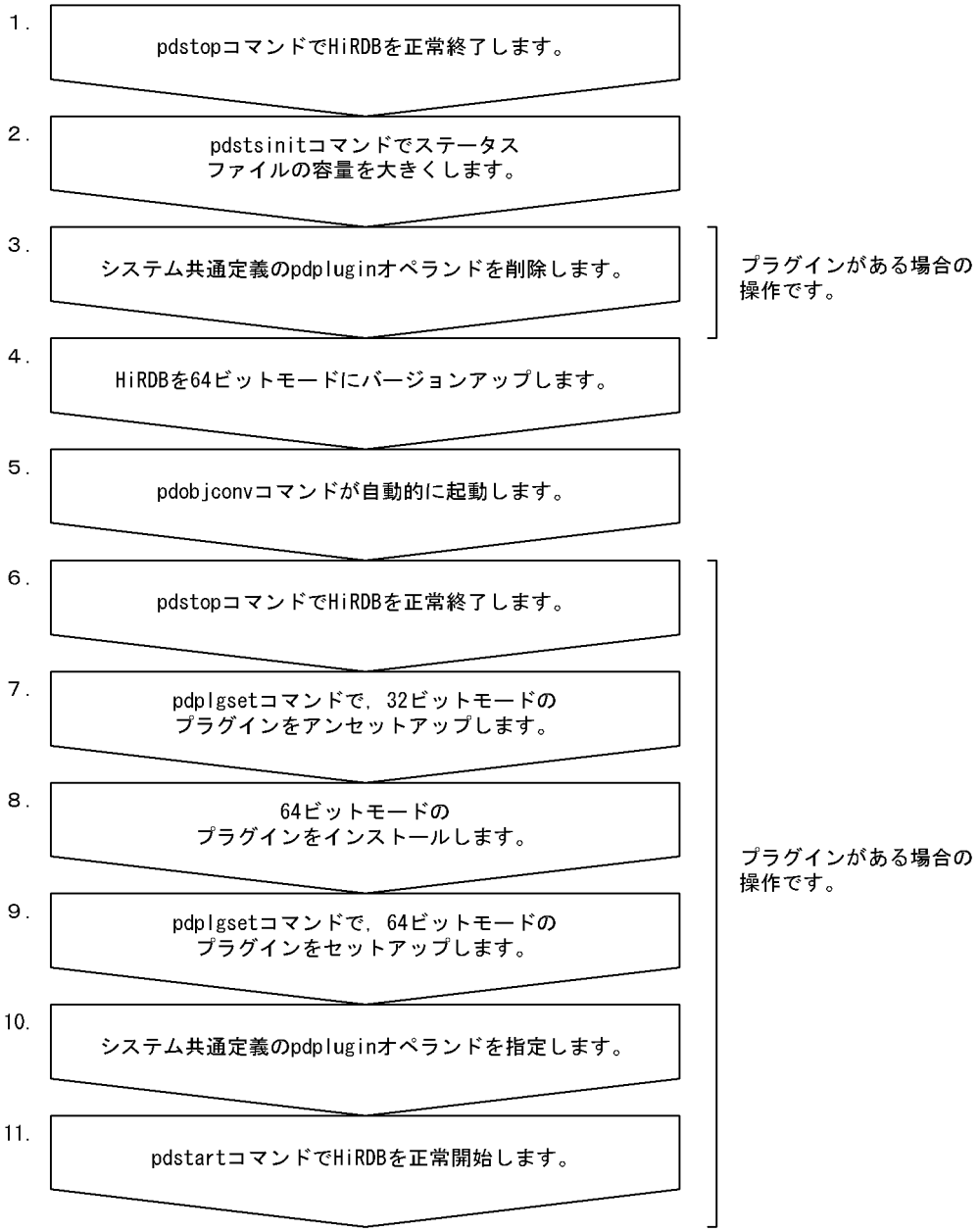
(4)　UOC インタフェースの違い

HiRDB を 64 ビットモードにすると，データベース作成ユティリティ（pdload）及びデータベース再編成ユティリティ（pdrorg）の UOC インタフェースが変わるため，UOC を作成し直す必要があります。UOC インタフェースについては，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。

1.6.2　64 ビットモードへの移行手順

64 ビットモードへの移行手順を次の図に示します。

図 1-7　64 ビットモードへの移行手順



注　上図の手順の左にある数字は，この後の操作手順の番号に対応しています。例えば，上図 ３.の操作は，操作手順 3.で説明しています。

1. pdstop コマンドで HiRDB を正常終了します

2. pdstsinit コマンドでステータスファイルの容量を大きくします

「ステータスファイルの容量の見積もり」を参照して，ステータスファイルの容量を見積もり直してください。必要があれば，pdstsinit コマンドでステータスファイルの容量を大きくしてください。

3. システム共通定義の pdplugin オペランドを削除します

システム共通定義の pdplugin オペランドを削除してください。この操作をしないと，HiRDB を 64 ビットモードにバージョンアップした後，正常開始できなくなります。

4. HiRDB を 64 ビットモードにバージョンアップします

64 ビットモードの HiRDB にバージョンアップしてください。バージョンアップの方法については，「HiRDB のバージョンアップ」を参照してください。

64 ビットモードの HiRDB にバージョンアップする前に，データディクショナリ用 RD エリアの空き領域を調べます。このとき，「バージョンアップ前にすること」に記載されている空き領域に加えて，次の表に示す空き領域が必要になります。

表 1-6　64 ビットモードにバージョンアップするのに必要な空き領域

データディクショナリ用 RD エリアに格納している ディクショナリ表	データディクショナリ用 RD エリアに 必要な空きセグメント数
SQL_TABLES 表	↑3÷S↑
SQL_VIEW_DEF 表	↑214÷S↑
SQL_ROUTINE_PARAMS 表※	↑3÷S↑

（凡例）

S：対象表を格納するデータディクショナリ用 RD エリアのセグメントサイズ

注※

データディクショナリ LOB 用 RD エリアを定義していない場合は不要です。

5. pdobjconv コマンドが自動的に起動します

バージョンアップ操作の中で，pdvrup コマンドの実行があります。この pdvrup コマンドが正常終了すると，pdobjconv コマンド※1 が自動的に実行されます。このコマンドのリターンコード※2 が 0 又は 4 ならば，64 ビットモードへの移行は終了です。0 又は 4 以外の場合は，「SQL オブジェクトの移行に失敗した場合」に示す方法で，64 ビットモードへの移行作業を継続してください。

注※1

32 ビットモードで作成したビュー表，手続き及び関数の SQL オブジェクトを 64 ビットモードでも使用できるようにするコマンドです。

注※2

KFPX21002-I メッセージにリターンコードが表示されます。このメッセージはシステムログファイル及び syslogfile に出力されます。リターンコードが 8 又は 12 の場合は，標準エラー出力にも出力されます。リターンコードの意味を次に示します。

0：

pdobjconv コマンドが正常終了しました。

4：

警告レベルのエラーはありますが，pdobjconv コマンドを正常終了します。

8：

一部の SQL オブジェクトの移行に失敗しました。メッセージ又は pdobjconv コマンドの処理結果情報（SQL オブジェクト移行情報）を参照して，エラーとなった要因を調査して取り除いてください。

又は，ユティリティ実行上のエラーが発生しました。

12：

pdobjconv コマンドが異常終了しました。メッセージ又は pdobjconv コマンドの処理結果情報（SQL オブジェクト移行情報）を参照して，エラーとなった要因を調査して取り除いてください。

pdcancel コマンドで pdobjconv コマンドをキャンセルしたり，pdobjconv コマンドのプロセスで異常が発生したりすると，リターンコードが 12 になります。

6. pdstop コマンドで HiRDB を正常終了します

7. pdplgset コマンドで 32 ビットモードのプラグインをアンセットアップします

「pdplgset -d プラグイン名」の形式でコマンドを実行します。

プラグインによっては，アンセットアップする前にプラグインのバックアップを取得する必要があります。該当するプラグインのマニュアルを参照して，バックアップが必要かどうかを確認してください。

8. 64 ビットモードのプラグインをインストールします

64 ビットモードのプラグインをインストールします。インストールの方法については，該当するプラグインのマニュアルを参照してください。

9. pdplgset コマンドで 64 ビットモードのプラグインをセットアップします

「pdplgset プラグイン名 プラグインインストールディレクトリ名」の形式でコマンドを実行します。

10. システム共通定義の pdplugin オペランドを指定します

64 ビットモードのプラグインのプラグイン名称をシステム共通定義の pdplugin オペランドに指定してください。

11. pdstart コマンドで HiRDB を正常開始します

1.6.3 SQL オブジェクトの移行に失敗した場合

ここでは，pdobjconv コマンドのリターンコードが 8 又は 12 の場合の処置について説明します。

(1) リターンコードが 8 の場合

一部の SQL オブジェクトの移行に失敗しました。SQL オブジェクト移行情報を参照して，移行に失敗した SQL オブジェクトを確認してください。SQL オブジェクト移行情報の見方については，マニュアル「HiRDB コマンドリファレンス」の pdobjconv コマンドを参照してください。

移行に失敗した SQL オブジェクトを移行する場合は，出力されたメッセージを参照して失敗の原因を取り除いてください。その後，pdobjconv コマンドを実行してください。なお，HiRDB をいったん終了した場合は，HiRDB を開始した後に pdobjconv コマンドを実行してください。

(2)　リターンコードが 12 の場合

pdobjconv コマンドが異常終了しました。出力されたメッセージを参照して異常終了の原因を取り除いてください。その後，pdobjconv コマンドを実行してください。なお，HiRDB をいったん終了した場合は，HiRDB を開始した後に pdobjconv コマンドを実行してください。

1.6.4　64 ビットモードへの移行に失敗した場合（旧バージョンに戻す場合）

64 ビットモードへの移行に失敗して，HiRDB を旧バージョンに戻す場合の方法については，「バージョンアップに失敗した場合」を参照してください。

なお，「バージョンアップに失敗した場合」の方法を実施した後に，pdstsinit コマンドで全ステータスファイルを初期化してください。この操作をしないと，HiRDB を正常開始できません。

2 インストール

この章では，インストール前後に必要な作業，HiRDB のインストール手順，付加 PP インストール時の注意及び HiRDB のアンインストールについて説明します。

2.1　インストール前の作業

ここでは，製品をインストールする前の作業について説明します。

なお，HiRDB/パラレルサーバの場合，すべてのサーバマシンで同じバージョンのプラットフォームを使用してください。

2.1.1　OS のオペレーティングシステムパラメタの確認・変更

実行者　スーパユーザ

HiRDB が使用するメッセージキュー及びセマフォ所要量を見積もり，必要に応じて OS のオペレーティングシステムパラメタ（カーネルパラメタ）を変更する必要があります。オペレーティングシステムパラメタの見積もりについては，「OS のオペレーティングシステムパラメタの見積もり」を参照してください。

なお，AIX 版の場合の注意事項を次に示します。

- /etc/security/limits ファイルを編集し，root ユーザ及び HiRDB 管理者の OS のシステム資源の制限値を変更してから OS を再起動しておく必要があります。
  例えば，AIX の場合，デフォルトの通常ファイルの上限は 1 ギガバイトです。ユティリティ実行時に出力されるワークファイルが 1 ギガバイト以上になると，HiRDB のユティリティは異常終了します。このため，OS のシステム資源の制限値を変更しておく必要があります。
- syslogfile を出力する設定になっているか確認してください。

インストール時にエラーメッセージのエラーの理由に'No-space'（書き込むファイルに十分な容量がありません）が出力された場合，次の原因が考えられます。

- ディスク容量が十分な状態でこのエラーになる場合は，HiRDB ファイルシステム領域をラージファイルとして定義していないか，又は OS のカーネルパラメタの制限に該当している可能性があります。
  OS のカーネルパラメタの制限については，「OS のオペレーティングシステムパラメタの見積もり」を参照してください。

2.1.2　HiRDB 管理者の登録

実行者　スーパユーザ

HiRDB を管理するユーザを各サーバマシンの OS に登録してください。登録する情報を次に示します。

- ログイン名
  英字で始まる 8 文字以下の英数字。
  「ALL」，「HiRDB」，「MASTER」及び「PUBLIC」は使用できません。

HiRDB/パラレルサーバの場合，又は HiRDB/シングルサーバで系切り替え機能を使用する場合，OS に登録する HiRDB 管理者用のログイン名は HiRDB を起動するすべてのサーバマシンで同じにしてください。

- ユーザ ID（数値）

  系切り替え機能を使用する場合，OS に登録する HiRDB 管理者用のユーザ ID は HiRDB を起動するすべてのサーバマシンで同じにしてください。

- グループ ID（数値）

  「HiRDB グループの設定」を参照してください。

- ホームディレクトリ

  任意

- ログインシェル

  任意

注意事項

ユーザ ID を登録した後に，必ずパスワードを登録してください。ただし，パスワードを同じにする必要はありません。

ここで登録したユーザ ID でログインするユーザを **HiRDB 管理者**といいます。HiRDB 管理者には，次に示す権限が与えられます。

1. HiRDB の各種システムファイル及びディレクトリの所有者としてのアクセス権が与えられます。これによって，ほかのユーザからの書き込みを禁止できます。
2. HiRDB の運用コマンド及びユティリティを実行できます。

マルチ HiRDB の場合

HiRDB ごとにそれぞれ HiRDB 管理者を登録してください。

2.1.3 HiRDB グループの設定

実行者　スーパユーザ

HiRDB 専用のグループを各サーバマシンの OS に設定してください。グループ名は，英字で始まる 8 文字以下の英数字にしてください。系切り替え機能を使用する場合，OS に登録する HiRDB 管理者用のグループ ID は HiRDB を起動するすべてのサーバマシンで同じにしてください。

HiRDB グループを設定すると，グループ以外のユーザが HiRDB ファイルシステム領域，HiRDB 運用ディレクトリ下に作成されるファイルなどにアクセスすることを拒否できます。そのため，セキュリティを強化できます。

通信情報ファイルディレクトリ変更機能を適用している環境で HiRDB サーバと同一ホストから UAP や一部の運用コマンド，ユティリティを実行する場合は，HiRDB グループに属しているユーザで実行する必要

があります。運用コマンドとユティリティの実行権限の詳細は，マニュアル「HiRDB コマンドリファレンス」の「運用コマンド一覧」及び「ユティリティ一覧」を参照してください。

マルチ HiRDB の場合

HiRDB ごとに別々のグループを設定すると，HiRDB ごとに利用者を区別できます。

2.1.4 インストールディレクトリの作成

実行者　スーパユーザ

ルートパーティションを圧迫しないように，HiRDB をインストールする前に HiRDB のインストールディレクトリをあらかじめ作成しておきます。このインストールディレクトリは，ファイルシステムを圧迫しないように専用のディスクパーティションに作成することをお勧めします。

HiRDB は，ここで作成したインストールディレクトリ下にインストールされます。

ディスクパーティションについては，OS のマニュアルを参照してください。

HiRDB を新規にインストールする場合，各サーバマシンにインストールディレクトリを作成してください。HiRDB のインストールディレクトリ名を次の表に示します。

表 2-1　HiRDB のインストールディレクトリ

HiRDB の種類	AIX，又は Linux 版の場合
HiRDB/シングルサーバの場合	/opt/HiRDB_S
HiRDB/パラレルサーバの場合	/opt/HiRDB_P

2.1.5 ホスト名の登録

HiRDB が使用するホスト名（システム定義及びクライアント環境定義で指定するホスト名）を hosts ファイル又は DNS などに登録し，名前解決してください。HiRDB が使用するホスト名は 32 文字以内にしてください。

システム定義及びクライアント環境定義中にホスト名を指定する場合，ホスト名，IP アドレス，又は FQDN のどれかの形式で指定します。

また，HiRDB/シングルサーバだけで HiRDB システムが構成されている※場合，システム定義及びクライアント環境定義中にループバックアドレスを指定できます。ループバックアドレスを指定すると，ホスト名の登録が不要になります。

注※

HiRDB/シングルサーバだけで HiRDB システムが構成されているとは，次に示す条件をすべて満たすことをいいます。

- HiRDB クライアントと HiRDB サーバが同一マシンにある（HiRDB クライアントが別マシンにない）
- ユティリティ専用ユニットがない

> **参考**
>
> ループバックアドレスとは，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 を指定する場合はホスト名を登録し，名前解決しておく必要があります。

ホスト名は，次の表に示すシステム定義及びクライアント環境定義に指定できます。システム定義の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。クライアント環境定義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 2-2　ホスト名を指定するシステム定義

システム定義のオペランド	指定するホスト名※1	
	標準ホスト名※2	標準ホスト名以外のホスト名※3
pdunit -x オプション	○	○
pdunit -c オプション	○	○
pdstart -x オプション	○	○
pdstart -m オプション	○	○
pdstart -n オプション	○	○
pdstbegin -x オプション	○	○
pd_hostname	○	×
pd_security_host_group	○	○

（凡例）

○：指定できます

×：指定できません

注※1

別名は指定できません。

注※2

標準ホスト名とは，コマンドプロンプトで hostname コマンドを実行して表示されたホスト名を指します。

注※3

標準ホスト名以外のホスト名とは，複数起動した IP アドレスに対して設定した，標準ホスト名とは異なるホスト名を指します。

表 2-3　ホスト名を指定するクライアント環境定義

クライアント環境定義のオペランド	指定するホスト名※1	
	標準ホスト名※2	標準ホスト名以外のホスト名※3
PDHOST	○	○
PDFESHOST	○	○
PDCLTRCVADDR	○	○
HiRDB_PDHOST	○	○
PDASTHOST	○	○

（凡例）

○：指定できます

注※1

別名は指定できません。

注※2

標準ホスト名とは，コマンドプロンプトで hostname コマンドを実行して表示されたホスト名を指します。

注※3

標準ホスト名以外のホスト名とは，複数起動した IP アドレスに対して設定した，標準ホスト名とは異なるホスト名を指します。

2.2　HiRDB のインストール手順

ここでは，HiRDB のインストール手順について説明します。

2.2.1　HiRDB のインストール

実行者　スーパユーザ

日立 PP インストーラを使用してサーバマシンごとに HiRDB をインストールします。

HiRDB をバージョンアップする場合

HiRDB をインストールする前に，インストールディレクトリ下に稼働中の HiRDB がないかどうかを（OS の ps コマンドなどで）確認してください。稼働中の HiRDB がある場合は，その HiRDB を pdstop コマンドで正常終了した後，pdsetup -d コマンドを実行し，KFPS00036-Q メッセージに y で応答して HiRDB を OS から削除してください。その後，インストールを実行してください。

HiRDB/パラレルサーバの場合

HiRDB/パラレルサーバを構成するすべてのサーバマシンで，同じバージョンの製品をインストールしてください。

マルチ HiRDB の場合

複数の HiRDB/シングルサーバ又は複数の HiRDB/パラレルサーバをインストールする場合，デフォルトではインストールディレクトリが同じになるので注意してください。インストールを続けて実行すると，インストールディレクトリが同じため，前にインストールした HiRDB を後からインストールした HiRDB が上書きしてしまいます。

したがって，HiRDB をインストールしたら，HiRDB 運用ディレクトリを作成し，HiRDB を OS に登録してください（「HiRDB 運用ディレクトリの作成」と「HiRDB 及び付加 PP の OS への登録」の操作を実行してください）。その後，ほかの HiRDB をインストールしてください。

2.2.2　付加 PP のインストール

実行者　スーパユーザ

HiRDB の付加 PP を使用する場合は，付加 PP をインストールします。付加 PP の機能と，HiRDB/パラレルサーバの場合に付加 PP をインストールするサーバを次の表に示します。

表 2-4　付加 PP の機能とインストール先サーバ

製品名	付加 PP を導入すると使用できる機能	インストール先サーバマシン
HiRDB Staticizer Option	• インナレプリカ機能 • 更新可能なオンライン再編成	すべてのサーバマシン

製品名	付加 PP を導入すると使用できる機能	インストール先サーバマシン
HiRDB Advanced High Availability	• 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	
	グローバルバッファの動的変更（pdbufmod コマンド）	
	システム構成変更コマンド（pdchgconf コマンド）	
	表のマトリクス分割	
	分割格納条件の変更（ALTER TABLE）	
HiRDB Non Recover FES	回復不要 FES	
HiRDB Accelerator	インメモリデータ処理によるバッチ高速化機能	
HiRDB Data Convert Type1 Option	暗号化機能※	
HiRDB Disaster Recovery Light Edition	ログ同期方式のリアルタイム SAN レプリケーション	業務サイトの HiRDB システムを構成するすべてのサーバマシンとログ適用サイトの HiRDB システムを構成するすべてのサーバマシン

注※

インストール方法については，マニュアル「HiRDB データベース暗号化機能」の「インストール」を参照してください。

2.2.3 プラグインのインストール

実行者　スーパユーザ

プラグインを使用する場合は，該当するプラグインを OS のインストーラを使用してインストールします。インストール方法については，該当するプラグインのマニュアルを参照してください。

2.3　インストール後の作業

ここでは，製品をインストールした後の作業について説明します。

2.3.1　HiRDB 運用ディレクトリの作成

実行者　HiRDB 管理者

HiRDB 運用ディレクトリを各サーバマシンに作成してください。HiRDB 運用ディレクトリとは，HiRDB を実行するディレクトリのことです。ここで作成した HiRDB 運用ディレクトリ下に HiRDB の各種ディレクトリ及びファイルが格納されます。

なお，HiRDB 運用ディレクトリはインストールディレクトリと同じにしないでください。インストールディレクトリを HiRDB 運用ディレクトリとすると，ディスク圧迫によるトラブルや，インストールの失敗などの可能性があります。また，インストールするたびにインストールディレクトリの所有者を root から HiRDB 管理者に変更して，グループとモードを(1)のように変更する必要があります。

(1)　HiRDB 運用ディレクトリに設定する情報

HiRDB 運用ディレクトリの名称は任意ですが，次に示す情報を設定してください。

- ディレクトリ名：任意
- 所有者　：HiRDB 管理者
- グループ：HiRDB グループ
- モード　：0755

また，次のことに注意してください。

- 「/」で始まり，次の要素で構成される文字列で指定してください。
  - 英数字
  - _（下線）
  - .（ピリオド）
  - パス区切りの「/」
- 「/」だけの指定はできません。
- パス名の長さは 128 文字（バイト）以内にしてください。ただし，Linux 版の場合は，118 文字（バイト）以内にしてください。

(2) HiRDB 運用ディレクトリ作成時の考慮点

1. HiRDB 運用ディレクトリを作成するには，2 ギガバイト以上のディスクの空き容量が必要です。ただし，システムの規模や HiRDB 運用ディレクトリ下に出力されるトラブルシュート情報ファイル（$PDDIR/spool 下のファイル）によって大きく変動するため，目安として次の計算式以上の空き容量を用意してください。

   （2ギガバイト＋共用メモリダンプファイル※のサイズ×2）×1.2（余裕値）

   注※

   $PDDIR/spool/pdshmdump/shmdump ファイルのことです。

   HiRDB が出力するトラブルシュート情報ファイルの一覧，及び各ファイルのサイズについては，「単調増加ファイル」を参照してください。

2. HiRDB 運用ディレクトリはルートディスク以外に作成することをお勧めします。HiRDB 運用ディレクトリ下にはトラブルシュート情報のファイル（$PDDIR/spool 下のファイル）が作成されます。これらのファイルを pdcspool コマンドなどで定期的に削除しないと，ディスク容量不足によって OS の動作に影響を与えることがあります。

   HiRDB 運用ディレクトリをルートディスクに作成する場合は，ルートディレクトリとは異なるパーティションに HiRDB 運用ディレクトリを作成することをお勧めします。

3. HiRDB 運用ディレクトリはローカルディスクに作成してください。また，/etc/checklist 又は/etc/fstab に指定するファイルシステムのマウントタイプには nosuid を指定しないでください。

4. HiRDB 運用ディレクトリはシンボリックリンクしないでください。

5. マルチ HiRDB の場合は，HiRDB ごとに異なる名称の HiRDB 運用ディレクトリを作成してください。

6. HiRDB/パラレルサーバで 2：1 系切り替え構成又は相互系切り替え構成をする場合は，全サーバマシンで HiRDB 運用ディレクトリを同じにできません。このときの HiRDB 運用ディレクトリの設定方法については，マニュアル「HiRDB システム運用ガイド」の「HiRDB/パラレルサーバのシステム構成例」の「相互系切り替え構成の例」を参照してください。

(3) HiRDB 運用ディレクトリ下のファイルの削除

サーバプロセス，又はクライアントの強制終了時などに，HiRDB は$PDDIR/spool 下にトラブルシュート情報を出力します。また，ワークファイルの出力先を特に指定していない場合にコマンド又はユティリティを［Ctrl ＋ C］キーを押すなどして途中終了させると，$PDDIR/tmp 下にコマンド又はユティリティが出力した作業用一時ファイルが削除されずに残ります。これらのファイルを残しておくと，HiRDB 運用ディレクトリがあるディスクの容量を圧迫する原因になります。HiRDB 運用ディレクトリがあるディスクの容量が不足すると HiRDB が異常終了することがあるため，HiRDB は次に示すファイルを定期的に削除します。

- トラブルシュート情報ファイル（$PDDIR/spool 下のファイル）
- 作業用一時ファイル（$PDDIR/tmp 下のファイル）
- pd_tmp_directory オペランドに指定したディレクトリ下のファイル

これらの単調増加するファイルについては，「単調増加ファイル」を参照してください。

なお，通常はこれらのファイルを 24 時間ごとに削除します。この削除間隔を pd_spool_cleanup_interval オペランドで変更できます。また，pd_spool_cleanup_interval_level オペランドで指定した日より前に出力されたファイルだけを削除するという指定ができます。

このほかにも，トラブルシュート情報（$PDDIR/spool 下のファイル）を一括して削除する方法があります。

- pdcspool コマンドでトラブルシュート情報ファイルを削除できます。作業用一時ファイル（$PDDIR/tmp 下のファイル）も削除できます。
- HiRDB の開始時に自動的にトラブルシュート情報ファイルを削除します。pd_spool_cleanup オペランドでトラブルシュート情報ファイルを削除するかどうかを指定します。このオペランドの省略値は「削除する」です。また，pd_spool_cleanup_level オペランドで指定した日より前に出力されたトラブルシュート情報ファイルだけを削除するという指定ができます。

備考

pdcspool コマンドのオプション，pd_spool_cleanup_level，又は pd_spool_cleanup_interval オペランドの指定で，削除するトラブルシュート情報を選択できます。

(4) HiRDB 運用ディレクトリのバックアップの取得

HiRDB 運用ディレクトリがあるディスクの障害などに備えて，HiRDB 運用ディレクトリ下のファイル（$PDDIR/conf 下のファイル）のバックアップを取得してください。HiRDB 運用ディレクトリを回復するには，$PDDIR/conf 下のファイルのバックアップが必要になります。$PDDIR/conf 下には，HiRDB システム定義のファイルが格納されています。HiRDB システム定義を変更した後に，$PDDIR/conf 下のファイルのバックアップを取得してください。

また，HiRDB 運用ディレクトリ下にユーザがファイルを作成する場合，そのファイルのバックアップも取得してください。HiRDB 運用ディレクトリの回復時にそのファイルのバックアップが必要になります。

HiRDB 運用ディレクトリ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。

●HiRDB 運用ディレクトリがあるディスクのバックアップを取得する場合

HiRDB 運用ディレクトリがあるディスクのバックアップを取得する必要が生じた場合は，次に示す手順でバックアップを取得してください。

〈手順〉

1. pdstop コマンドで HiRDB を終了します。
2. pdsetup -d コマンドを実行します。n を応答してください。※
3. OS の機能（コマンド）で，HiRDB 運用ディレクトリを格納しているディスクのバックアップを取得します。
4. pdsetup コマンドを実行します。

5. pdstart コマンドで HiRDB を開始します。

注※

バックアップの中に回復後の不正動作の元となるファイルを残さないために pdsetup -d コマンドを実行します。また，回復後も必要となるファイルを残すために n と応答します。

2.3.2　ワークファイル出力先ディレクトリの作成

実行者　HiRDB 管理者

HiRDB が出力するワークファイルの出力先となるディレクトリを作成してください。コマンドやユティリティの実行時に生成される様々なワークファイルの出力先として，ここで作成したディレクトリを指定します。これによって，出力先がユニット単位で 1 か所になるため，煩雑になりやすいワークファイルの管理が容易になります。ワークファイル出力先ディレクトリを作成しない運用もできますが，その場合はワークファイルの出力先が一定にならないため，pdcspool コマンドによるワークファイルの削除ができなくなります。そのため，ワークファイル出力先ディレクトリを作成しておくことをお勧めします。

(1)　ワークファイル出力先ディレクトリの容量の見積もり

ワークファイル出力先ディレクトリ下の空き領域は，次の値以上に設定してください。ワークファイル出力中に HiRDB 又はコマンドが異常終了した場合，ワークファイルは削除されません。そのため，pdcspool コマンドを実行する前にディスク容量が不足しないように，ワークファイル出力先ディレクトリの空き領域には十分に余裕のある値を設定してください。

ワークファイル出力先ディレクトリの容量（単位：キロバイト）＝

178224 ＋ a ＋ b ＋ c ＋ d ＋ e ＋ f ＋ g ＋ h

変数	説明	計算式，又は計算式の参照箇所
a	pdconstck 実行時の処理結果ファイルの容量	「整合性チェックユティリティ（pdconstck）実行時のファイルの容量」
b	pddbst 実行時の次のファイルの容量 • ワーク用ファイル • ソート用ワークファイル	「データベース状態解析ユティリティ（pddbst）実行時のファイルの容量」
c	pdload 実行時の次のファイルの容量 • インデクス情報ファイル • エラー情報ファイル • エラー情報ファイル作成用一時ファイル • LOB 中間ファイル • ソート用ワークファイル	「データベース作成ユティリティ（pdload）実行時のファイルの容量」
d	pdorend 実行時	2400×マッピングキー列数×スキップした SQL 件数＋ 100
e	pdrbal 実行時の次のファイルの容量 • インデクス情報ファイル	「リバランスユティリティ（pdrbal）実行時のファイルの容量」

変数	説明	計算式，又は計算式の参照箇所
	• ソート用ワークファイル	
f	pdrorg 実行時の次のファイルの容量 • インデクス情報ファイル • ソート用ワークファイル	「データベース再編成ユティリティ（pdrorg）実行時のファイルの容量」
g	pdrstr -w 実行時のソート用ワークディレクトリの容量	マニュアル「HiRDB コマンドリファレンス」
h	pdstedit 実行時の次のファイルの容量 • ワーク用ファイル • ソート用ワークファイル • DAT 形式ファイル	「統計解析ユティリティ（pdstedit）実行時のファイルの容量」

(2) ワークファイル出力先ディレクトリの指定

ワークファイルの出力先を 1 か所にするには，pd_tmp_directory オペランドに作成したディレクトリを指定します。

pd_tmp_directory オペランドを指定していない場合，HiRDB は各コマンド及びユティリティによって決められたディレクトリにワークファイルを出力します。なお，ワークファイル出力先は次の順番で決定されます。

1. コマンドのオプション，又はユティリティの制御文で指定した出力先
2. 1.の指定がない場合，pd_tmp_directory オペランドで指定した出力先
3. 2.の指定がない場合，環境変数 TMPDIR で指定した出力先※
4. 3.の指定がない場合，/tmp ディレクトリ

注※

コマンドがサーバ側で動作する場合はプロセスサーバプロセス（pdprcd）に設定される環境変数 TMPDIR になります。

(3) ワークファイルの削除

HiRDB は通常，24 時間ごとにワークファイルを削除します。この削除間隔を pd_spool_cleanup_interval オペランドで変更できます。また，pd_spool_cleanup_interval_level オペランドで指定した日より前に出力されたファイルだけを削除するという指定ができます。このとき，pd_tmp_directory オペランドで指定したワークファイル出力先ディレクトリ下のファイルを削除します。

また，コマンド又はユティリティが出力したワークファイルのうち，HiRDB が削除しないものについては，pdcspool コマンドによって定期的に削除する必要があります。この場合にも，pd_tmp_directory オペランドで指定したワークファイル出力先ディレクトリ下のファイルを削除します。

ワークファイルの削除については，「HiRDB 運用ディレクトリ下のファイルの削除」を参照してください。

2.3.3　通信情報ファイルディレクトリの作成

実行者　HiRDB 管理者

通信情報ファイルディレクトリ変更機能を適用する場合，HiRDB が作成する通信情報ファイルの格納先ディレクトリを作成してください。このとき，所有者，グループなどを次のようにしてください。

表 2-5　通信情報ファイルディレクトリの作成条件

項目	内容
アクセス権	0770
ユーザ	HiRDB 管理者
グループ	HiRDB グループ
パス	• 絶対パスのパス長が 80 バイト以下 • ルートディレクトリではないパス

通信情報ファイルディレクトリは NFS のような共有ファイルシステムではなく，ローカルディスクのファイルシステムに作成してください。また，ほかの用途で使用しているディレクトリ（pd_tmp_directory オペランドで指定したワークファイル出力先ディレクトリなど）を通信情報ファイルディレクトリとしないでください。

通信情報ファイルディレクトリ変更機能の詳細は，マニュアル「HiRDB システム運用ガイド」の「通信情報ファイルディレクトリの変更方法」を参照してください。

2.3.4　HiRDB 及び付加 PP の OS への登録

実行者　スーパユーザ

(1)　OS への登録方法

(a)　pdsetup コマンドの実行

pdsetup コマンドで，HiRDB を OS に登録してください。pdsetup コマンドを実行すると，インストールディレクトリ下に作成されたディレクトリ及びファイルが，HiRDB 運用ディレクトリ下に複写されます。HiRDB/パラレルサーバの場合，pdsetup コマンドはサーバマシンごとに実行してください。

OS への登録手順を次に示します。

〈手順〉

1. pdsetup コマンドを実行します。
2. 終了ステータスを確認します。

pdsetup コマンドの終了ステータスが 0 でない場合，障害が起きている可能性があります。syslog ファイルに出力されたメッセージを参照して，障害原因を取り除いてから，OS の登録を再度実行してください。

pdsetup コマンドについては，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。

注意事項

- pdsetup コマンドは，HiRDB のインストールディレクトリ下の bin ディレクトリにあります。
- pdsetup コマンドを実行すると，HiRDB のインストールディレクトリ下の conf ディレクトリに Inittab というディレクトリが自動的に作成されます。ここには，/etc/inittab ファイルのバックアップが作成されます。万一 pdsetup コマンドの実行中又は実行後に/etc/inittab ファイルが破壊されたときには，このバックアップを参照して OS を起動してください。
- pdsetup コマンドを実行した後で，HiRDB 運用ディレクトリ下で HiRDB が作成したファイル及びディレクトリを削除したり，所有者やアクセス権の変更をしたりしないでください。これらの操作によって，HiRDB を開始できなくなることがあります。
- pdsetup コマンドは，インストールされた HiRDB のロードモジュールを，指定された HiRDB 運用ディレクトリにコピー又は上書きします。ほかのプログラムのデータやユーザデータの損失を防ぐため，HiRDB 運用ディレクトリには，次のどちらかを指定してください。

  ・HiRDB 専用に作成したディレクトリ

  ・HiRDB のインストールディレクトリ（HiRDB/シングルサーバの場合は/opt/HiRDB_S，HiRDB/パラレルサーバの場合は/opt/HiRDB_P）

(b) pdsetup コマンドで選択するオペランド省略時動作

オペランド省略時の動作を pdsetup コマンドの-v オプションで選択します。pdsetup コマンドを実行して OS への登録をした後にオペランド省略時動作を変更したい場合は，pdsetenv コマンドを実行してください。また，オペランド省略時の動作で適用したモードは，pdadmvr コマンド，又は HiRDB 開始時に出力される KFPS01826-I メッセージで確認できます。

〈規則〉

- 系切り替え構成の場合，現用系と予備系でオペランド省略時動作の指定を同一にしてください。
- HiRDB/パラレルサーバでは，各ユニットのオペランド省略時動作の指定をすべて同一にしてください。

(c) pdsetup コマンドで指定する文字コード

●サーバ側の指定

HiRDB で使用する文字コードを pdsetup コマンドの-c オプションで指定します。HiRDB で使用できる文字コードを次の表に示します。

表 2-6　HiRDB で使用できる文字コード

文字コード	適用 OS	
	AIX	Linux
シフト JIS 漢字コード	◎	○
EUC 中国語漢字コード	○	○
中国語漢字コード（GB18030）	○	○
単一バイト文字コード	○	○
EUC 日本語漢字コード	○	◎
Unicode（UTF-8）	○	○
Unicode（IVS 対応 UTF-8）	○	○

（凡例）

◎：使用できる文字コード（pdsetup コマンドの -c オプションを省略すると仮定される文字コード）

○：使用できる文字コード

なお，どの文字コードを使用しているかは，pdadmvr -c コマンドで確認できます。

●クライアント側の指定

クライアント側では，サーバの文字コードに応じてクライアント環境定義の LANG 又は PDCLTLANG オペランドに文字コードを指定します。クライアント環境定義は，UAP の作成又は実行時に参照されます。サーバの文字コードに応じて指定できる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。

(d)　POSIX ライブラリ版を使用する場合の pdsetup コマンドの指定

POSIX ライブラリ版を使用する場合，pdsetup コマンドの -l オプションを指定します。次に示す機能を使用する場合に POSIX ライブラリ版を使用します。

- Java ストアドプロシジャ及び Java ストアドファンクション※

Java ストアドプロシジャ及び 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。

ただし，Linux 版は，特別なセットアップをしなくても，Java ストアドプロシジャ及び Java ストアドファンクションを使用できます。

注※

Java ストアドプロシジャ及び Java ストアドファンクションは次に示す HiRDB で使用できます。

- AIX 版（32 ビットモードの POSIX ライブラリ版）
- Linux 版（32 ビットモード），及び Linux（EM64T）版

(e) コマンド実行権限変更機能を適用する場合の pdsetup コマンドの指定

コマンド実行権限変更機能を適用する場合，pdsetup コマンドの-S オプションを指定します。コマンド実行権限変更機能については，マニュアル「HiRDB システム運用ガイド」を参照してください。

〈規則〉

- 系切り替え構成の場合，現用系と予備系でコマンド実行権限変更機能の適用有無を同一にしてください。
- HiRDB/パラレルサーバでは，各ユニットのコマンド実行権限変更機能の適用有無をすべて同一にしてください。

(f) 通信情報ファイルディレクトリ変更機能を適用する場合の pdsetup コマンドの指定

通信情報ファイルディレクトリ変更機能を適用する場合は，-I オプションを指定します。通信情報ファイルディレクトリ変更機能については，マニュアル「HiRDB システム運用ガイド」の「通信情報ファイルディレクトリの変更方法」を参照してください。

〈規則〉

- 系切り替え構成の場合，現用系と予備系で通信情報ファイルディレクトリ変更機能の適用有無を同一にしてください。
- HiRDB/パラレルサーバでは，各ユニットの通信情報ファイルディレクトリ変更機能の適用有無をすべて同一にしてください。

(2) 付加 PP の登録

付加 PP をインストールした場合は，pdopsetup コマンドで付加 PP を OS に登録してください。

pdopsetup コマンドについては，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。

●HiRDB/パラレルサーバの場合

全ユニットに同じ付加 PP を登録する必要があるため，各サーバマシンで pdopsetup コマンドを実行してください。

●系切り替え機能を使用する場合

- 現用系で付加 PP を登録した場合，同じ付加 PP を予備系でも登録してください（予備系でも pdopsetup コマンドを実行してください）。
- 相互系切り替え構成などで一つのサーバマシンに二つ以上の HiRDB を OS に登録している場合，HiRDB（HiRDB/パラレルサーバの場合はユニット）ごとに pdopsetup コマンドを実行してください。例えば，相互系切り替え構成で，一つのサーバマシンに二つの HiRDB（現用系と予備系）がある場合，そのサーバマシンでは pdopsetup コマンドを 2 回実行する必要があります（現用系の HiRDB と予備系の HiRDB にそれぞれ実行する必要があります）。

●リアルタイム SAN レプリケーションを使用する場合

メインサイトで付加 PP を登録した場合，同じ付加 PP をリモートサイトでも登録してください（リモートサイトでも pdopsetup コマンドを実行してください)。

2.3.5　環境変数の設定

(1)　HiRDB 管理者が設定する必要がある環境変数

HiRDB 管理者の環境に，次の表に示す環境変数を設定してください。

環境変数は，各サーバマシンのログインシェルに合わせて次のファイルに設定してください。

- Bourne シェルのとき ：$HOME/.profile
- C シェルのとき：$HOME/.cshrc

表 2–7　HiRDB 管理者の環境変数に設定する内容

環境変数	設定する内容
PDDIR※1	HiRDB 運用ディレクトリを絶対パス名で指定します。
PDCONFPATH※2	HiRDB システム定義ファイルを格納するディレクトリを絶対パス名で指定します。ただし，ユニット制御情報定義は，この指定の内容にかかわらず，$PDDIR/conf 下のファイルを使用します。また，ユニット制御情報定義で PDCONFPATH オペランドを指定する場合は，PDCONFPATH オペランドと同じ内容にしてください。
PATH	$PDDIR/bin を追加指定します。
SHLIB_PATH※3	$PDDIR/lib を追加指定します。

注※ 1

PDDIR の絶対パス名の長さは，次に示すバイト以内で設定してください。

- AIX の場合は，128 バイト
- Linux の場合は，118 バイト

注※ 2

PDCONFPATH の絶対パス名の長さは，213 バイト以内で設定してください。

注※ 3

Linux 版の場合は LD_LIBRARY_PATH に，AIX 版の場合は LIBPATH になります。

LANG 環境変数，及び PDLANG 環境変数の設定については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

備考

pd_tmp_directory オペランドを指定している場合は，環境変数 TMPDIR を指定する必要はありません。

pd_tmp_directory オペランドを指定していない場合は，環境変数 TMPDIR を省略すると，pd_tmp_directory オペランドを省略したときと同様のディレクトリ下に一時ファイル（テンポラリファイル）を作成します。また環境変数 TMPDIR を設定するときは，ディレクトリパス長を 512 バイト以内で設定してください。TMPDIR を指定すると，HiRDB の運用コマンド又はユティリティを中断した場合，HiRDB は TMPDIR に指定したディレクトリ下に"pdcmd"又は"plcmd"で始まるファイルを作成することがあります。HiRDB の運用コマンド又はユティリティが終了した後も"pdcmd"及び"plcmd"で始まるファイルが削除されない場合は，OS の rm コマンドなどでこれらのファイルを削除してください。

(2) UAP を実行するユーザが設定する必要がある環境変数

UAP を実行する場合，実行するユーザごとにクライアント環境定義を環境変数に設定する必要があります。クライアント環境の設定形式は，クライアントマシンの OS や使用するシェルによって異なります。クライアント環境定義の設定内容については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(3) 表やインデクスを定義するユーザが設定する必要がある環境変数

表やインデクスを定義する場合，実行するユーザごとに次の環境変数を設定しておく必要があります。設定する環境変数を Bourne シェルの形式で次に示します。

- PDHOST=HiRDB サーバのホスト名［，予備系 HiRDB サーバのホスト名］

  接続する HiRDB サーバのホスト名を指定します。HiRDB/パラレルサーバの場合は，システムマネジャを定義したサーバマシンのホスト名を指定します。また，PDFESHOST を指定している場合は，PDFESHOST のホスト名を指定できます。PDFESHOST のホスト名を指定した場合，システムマネジャユニットに障害が発生しても，HiRDB サーバに接続できます。

- PDUSER=認可識別子/パスワード

  認可識別子，パスワードを指定します。ここで指定したユーザに対して，スキーマの定義権限を与えておく必要があります。また，スキーマの定義（CREATE SCHEMA）で指定する認可識別子は，この環境変数で指定した認可識別子を指定します。なお，認可識別子，パスワードに英小文字を使用する場合は，「'"認可識別子"/"パスワード"'」の形式で指定してください。

- PDNAMEPORT=HiRDB サーバのポート番号

  HiRDB サーバのポート番号を指定します。PDHOST に指定したホストにアクセスする HiRDB システムのポート番号を指定してください。

2.3.6 リモートシェル実行環境の設定

実行者　HiRDB 管理者

次に示す場合に，HiRDB は各ユニットのホスト間でコマンドのリモート実行やファイル転送を行います。

- HiRDB/パラレルサーバの場合

- IP アドレスを引き継がない系切り替え機能を使用する場合
- ユティリティ専用ユニットを使用する場合

このため，対象となるユニットの間で相互にログインを許可する設定が必要です。次の表に設定が必要となる条件と対象を示します。複数の条件に該当する場合は，それぞれの条件をすべて満たすように設定します。

表 2-8　リモートシェル実行環境の設定が必要になる場合

設定が必要となる条件	相互にログインを許可する設定が必要となる対象
HiRDB/パラレルサーバの場合	HiRDB/パラレルサーバを構成する各ユニットの間※
IP アドレスを引き継がない系切り替え機能を使用する場合	現用系を構成するサーバマシンと，予備系を構成するサーバマシンとの間
ユティリティ専用ユニットを使用する場合	HiRDB/シングルサーバとユティリティ専用ユニットとの間

注※

システム定義（pdunit -x オプション，pdunit -c オプション）へ指定したすべてのホストの間で相互にログインを許可する設定が必要です。

(1)　事前準備

HiRDB を構築する環境に合わせて，HiRDB がリモート実行やファイル転送に使用するコマンドの種類を選びます。次の表に HiRDB が使用するコマンドを示します。

表 2-9　HiRDB がリモート実行やファイル転送に使用するコマンド

分類	HiRDB が使用するコマンド	コマンドを配置するパス	備考
リモートシェル	rsh，rcp	/usr/bin※	－
セキュアシェル	ssh，scp		サーバマシンで，SSH プロトコルを利用可能にする必要があります。

（凡例）

－：備考はありません。

注※

上記のパスにコマンドが配置されていない場合は，配置してください。

HiRDB を構築するすべてのサーバマシンで，リモート実行やファイル転送に使用するコマンドを統一してください。

(2) 環境設定（リモートシェル/セキュアシェルの設定）

対象のサーバマシン間，又は各ユニットのホスト間で相互にログインできるように環境設定を行います。ログイン時にパスワードやパスフレーズの入力など，プロンプトへの入力をしないでログインできるように設定してください。詳細は OS のマニュアルを参照してください。

ログインを許可するホスト名は，システム定義（pdunit -x オプション，pdunit -c オプション，pdstart -x オプション）へ指定したホスト名を指定してください。

なお，$HOME/.cshrc ファイルを作成する場合，登録端末がないときは標準出力又は標準エラーファイルにデータを出力しない構造にしてください。

(3) 環境設定（システム定義の設定）

HiRDB がリモート実行やファイル転送に使用するコマンドにセキュアシェルを選択した場合，システム共通定義で pd_cmd_rmode=ssh を指定してください。

2.3.7 HiRDB の運用コマンドをバックグラウンドで実行する場合の注意

HiRDB の運用コマンドをバックグラウンドモードで実行する場合は，端末ポート用のオプション設定でバックグラウンドジョブの端末への出力を抑止していないことを確認してください。

バックグラウンドジョブの端末への出力を抑止したままでコマンドを入力すると，コマンドが終了しないで HiRDB 下のプロセスが不当に残ってしまいます。

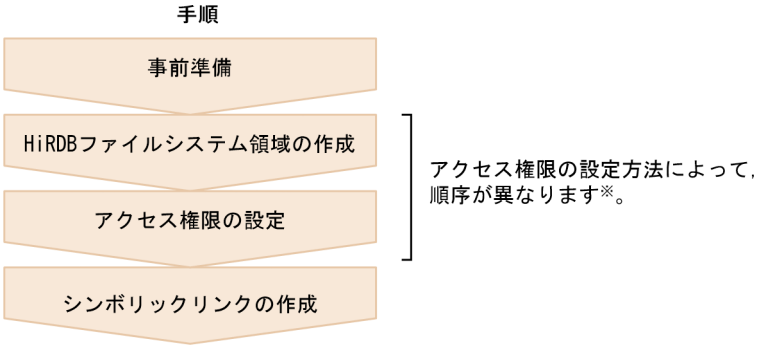
端末ポート用のオプション設定については，stty コマンドの tostop オプションで指定します。stty コマンドの tostop オプションの指定については，OS のマニュアルを参照してください。

2.3.8 HiRDB ファイルシステム領域を作成する準備

(1) 通常ファイル上に HiRDB ファイルシステム領域を作成する場合

通常ファイル上に HiRDB ファイルシステム領域を作成する場合の手順を次の図に示します。

図 2-1　通常ファイル上に HiRDB ファイルシステム領域を作成する場合の手順



注※

詳細は，「アクセス権限の設定」を参照してください。

(a)　事前準備

実行者　スーパユーザ

次の作業を行います。

- ハードディスクを初期化します。
- 初期化したハードディスクにパーティションを設定します。
- 設定したパーティションを UNIX のファイルシステムとして初期化します。

これらの作業方法については，OS のマニュアルを参照してください。

(b)　HiRDB ファイルシステム領域の作成

実行者　HiRDB 管理者

pdfmkfs コマンドを実行して，UNIX ファイルシステム領域上に HiRDB ファイルシステム領域を作成します。初期値のファイルモードは 8 進数表記で 660 です。

(c)　アクセス権限の設定

実行者　HiRDB 管理者

権限を持たないユーザからの不当なアクセスを防止するには，作成した HiRDB ファイルシステム領域のファイルモードを変更します。

ファイルモードの変更は，umask コマンド，又は chmod コマンドで行います。umask コマンドは HiRDB ファイルシステム領域を作成する前に，chmod コマンドは HiRDB ファイルシステム領域を作成した後に実行します。

これらのコマンドについては，OS のマニュアルを参照してください。

なお，アクセス制限の設定値については，「HiRDB ファイルシステム領域のアクセス権の考え方」を参照してください。

(d)　シンボリックリンクの設定

実行者　HiRDB 管理者

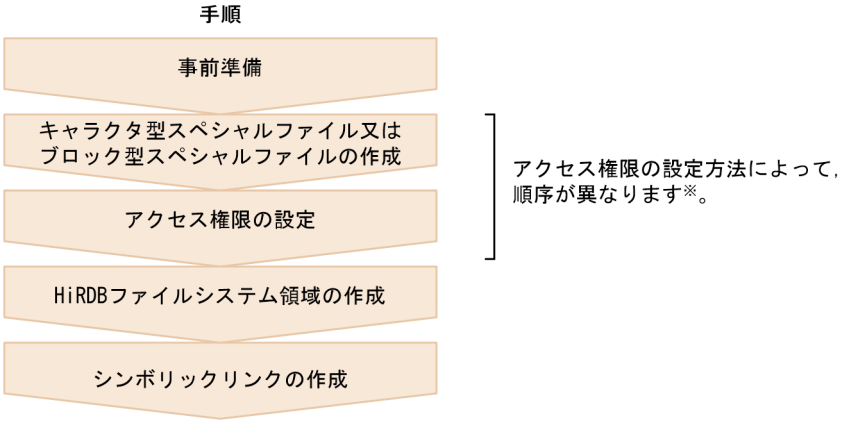
HiRDB ファイルシステム領域の名称には，通常ファイルの実体名称をそのまま使用しないで，OS の ln コマンドで実体名称に**シンボリックリンク**した名称を使用することをお勧めします。

なお，ln コマンドについては，OS のマニュアルを参照してください。

(2)　キャラクタ型スペシャルファイル又はブロック型スペシャルファイル上に HiRDB ファイルシステム領域を作成する場合

キャラクタ型スペシャルファイル又はブロック型スペシャルファイル上に HiRDB ファイルシステム領域を作成する場合の手順を次の図に示します。

図 2-2　キャラクタ型スペシャルファイル又はブロック型スペシャルファイル上に HiRDB ファイルシステム領域を作成する場合の手順



注※

詳細は，「アクセス権限の設定」を参照してください。

(a)　事前準備

実行者　スーパユーザ

次の作業を行います。

- ハードディスクを初期化します。
- 初期化したハードディスクにパーティションを設定します。

これらの作業方法については，OS のマニュアルを参照してください。

(b)　キャラクタ型スペシャルファイル又はブロック型スペシャルファイルの作成

実行者　スーパユーザ

キャラクタ型スペシャルファイル又はブロック型スペシャルファイルの作成，及びモードの設定をします。

＜ファイルの作成＞

●キャラクタ型スペシャルファイルの場合

mknod コマンドを実行して，ディスクパーティションに対応するキャラクタ型スペシャルファイルを作成します。

Linux の場合は，mknod コマンド以外の方法で作成します。

OS ごとのキャラクタ型スペシャルファイルの作成方法を次の表に示します。なお，各コマンド及び機能の詳細については，OS のマニュアルを参照してください。

表 2-10　キャラクタ型スペシャルファイルの作成方法

OS	ディスクパーティション	作成方法	備考
Linux	LV	raw コマンドを実行します。ディスクパーティションを有効化するために，LV の認識後に raw コマンドを実行してください。	ディスクパーティションを有効化するために，OS 再起動時には再度 raw コマンドを実行してください。OS 起動時に自動的に raw コマンドを実行する場合は，/etc/rc.local に raw コマンドを記述してください。
	LV 以外	udev 機能を利用します。	キャラクタ型スペシャルファイルを定義した udev 機能用のルールファイルを作成し，適切な場所に配置してください。
上記以外	全種別	mknod コマンドを実行します。	なし。

●ブロック型スペシャルファイルの場合

OS のコマンド（fdisk，parted，又は mknod）を使用して，ディスクパーティションをブロック型スペシャルファイルとして使用します。

各コマンド及び機能の詳細については，OS のマニュアルを参照してください。

＜モードの設定＞

作成したキャラクタ型スペシャルファイル又はブロック型スペシャルファイルのモードを次のように設定します。

所有者，アクセス権		設定する情報
所有者	ユーザ ID	HiRDB 管理者
	グループ ID	HiRDB 管理者のグループ ID
アクセス権	所有者	rw（読み書きができます）

所有者，アクセス権		設定する情報
	グループ	rw（読み書きができます）
	その他	--（アクセスできません）

(c)　アクセス権限の設定

実行者　スーパユーザ

権限を持たないユーザからの不当なアクセスを防止するには，作成したキャラクタ型スペシャルファイル又はブロック型スペシャルファイルのファイルモードを変更します。

ファイルモードの変更は，umask コマンド，又は chmod コマンドで行います。umask コマンドはキャラクタ型スペシャルファイル又はブロック型スペシャルファイルを作成する前に，chmod コマンドは作成した後に実行します。

これらのコマンドについては，OS のマニュアルを参照してください。

また，Linux 6 以降の場合，通常の設定ではブロック型スペシャルファイルのアクセス権を HiRDB 管理者に固定することができないため，別途 udev の設定が必要になります。udev の設定については，OS のマニュアルを参照してください。

なお，アクセス制限の設定値については，「HiRDB ファイルシステム領域のアクセス権の考え方」を参照してください。

(d)　HiRDB ファイルシステム領域の作成

実行者　HiRDB 管理者

pdfmkfs コマンドを実行して，キャラクタ型スペシャルファイル又はブロック型スペシャルファイル上に HiRDB ファイルシステム領域を作成します。

(e)　シンボリックリンクの設定

実行者　HiRDB 管理者

HiRDB ファイルシステム領域の名称には，キャラクタ型スペシャルファイル又はブロック型スペシャルファイルの実体名称をそのまま使用しないで，OS の ln コマンドで実体名称に**シンボリックリンク**した名称を使用することをお勧めします。

なお，ln コマンドについては，OS のマニュアルを参照してください。

(3)　HiRDB ファイルシステム領域のアクセス権の考え方

HiRDB ファイルシステム領域のアクセス権を設定するときの考え方について説明します。

(a) 設定値の考え方

HiRDB は，機密保護を強化するため，HiRDB 管理者と同じ OS 上のグループ ID を持つユーザ群を設けることで，グループ以外のユーザからのアクセスを制限することを推奨しています。このことから，HiRDB ファイルシステム領域のアクセス権についても，領域の所有者とグループに対してだけ読み書きのアクセス権を与えることを推奨します。

例えば，マルチ HiRDB 構成の場合，それぞれの HiRDB グループを設定することで，HiRDB ごとのアクセスを切り分けることができます。これによって，誤って別の HiRDB へアクセスしてしまうことを防ぐことができます。

なお，HiRDB グループについては，「HiRDB グループの設定」を参照してください。

(b) アクセス権の変更

HiRDB ファイルシステム領域のファイルモードを変更する場合の注意事項を，アクセス権の設定値ごとに示します。

アクセス権の設定値（8 進数表記）	説明及び注意事項
660	推奨値です。
640	HiRDB 管理者と同じグループ ID を持つユーザの更新権限をなくします。HiRDB 管理者以外の実行を許可している運用コマンド又はユティリティなどが HiRDB 管理者以外のユーザ権限で実行できない場合があります。
600	HiRDB 管理者だけがアクセス権を持ちます。HiRDB 管理者以外の実行を許可している運用コマンド又はユティリティなどが HiRDB 管理者以外のユーザ権限で実行できない場合があります。
上記以外	変更しないでください。

(c) umask の設定値

HiRDB ファイルシステム領域，及びキャラクタ型スペシャルファイル又はブロック型スペシャルファイルのファイルモードを変更する場合は，(b)の説明を参照して umask の設定を行ってください。

なお，pdfmkfs コマンドで作成した HiRDB ファイルシステム領域のファイルモードの初期値は，所有者及びグループに読み書きの権限を与えています（8 進数表記で 660）。OS の mknod コマンドで作成したキャラクタ型スペシャルファイル又はブロック型スペシャルファイルのファイルモードの初期値については，OS のマニュアルを参照してください。

(d) HiRDB ファイルのアクセス権

HiRDB ファイルシステム領域は，上記で説明した OS 上のアクセス権が有効になりますが，HiRDB ファイルに対しては OS 上のアクセス権は無効です。また，HiRDB では，HiRDB ファイルのアクセス権の制御は行っていません。したがって，HiRDB ファイルごとにアクセスを制限したい場合は，HiRDB ファイルシステム領域を切り分けて，HiRDB ファイルシステム領域ごとにアクセス権を変更してください。

(4) OS 又はデバイスドライバの機能で，物理ボリューム及び論理ボリュームの入出力エラーを検知するまでの時間設定

実行者　スーパユーザ

入出力処理が無応答状態になると，そのほかの UAP，ユティリティ，又は運用コマンドもその影響を受けて停滞します。

OS 又はデバイスドライバの機能※で物理ボリューム又は論理ボリュームの入出力エラーを検知するまでの時間を設定できる場合は，入出力処理が無応答にならないよう，設定を行ってください。

注※

詳細については，OS 又はデバイスドライバのマニュアルを参照してください。

入出力処理が無応答となった場合，UAP やティリティの強制終了も停滞する可能性があるため，設定値は，オペランド又はオプションで指定できる UAP 又はユティリティの監視時間より小さくしてください。

オペランド又はオプションで指定できる UAP 又はユティリティの監視時間については，マニュアル「HiRDB システム運用ガイド」の「UAP 又はユティリティの実行時間の監視（無応答障害時の影響を抑える方法)」を参照してください。

2.4 HiRDB のアンインストール

実行者　スーパユーザ及び HiRDB 管理者

HiRDB のアンインストールは，今後このサーバマシンで HiRDB を使用しないときだけ実施してください。それ以外でアンインストールを実施することはお勧めしません。

HiRDB をアンインストールする場合の注意事項

HiRDB をアンインストールする場合は，コマンド，ユティリティ，アプリケーション，HiRDB Datareplicator，及び HiRDB Dataextractor はあらかじめ停止しておいてください。これらを停止しないと，実行形式ファイルや共用ライブラリの削除に失敗することがあります。

HiRDB/シングルサーバのアンインストールを行うと，そのサーバマシン上で，次のディレクトリ下に存在する HiRDB の実行に必要なファイル及びディレクトリが削除されます。

- /opt/HiRDB_S
- ユーザが pdsetup コマンドで OS に登録したすべての HiRDB 運用ディレクトリ※

HiRDB/パラレルサーバのアンインストールを行うと，そのサーバマシン上で，次のディレクトリ下に存在する HiRDB の実行に必要なファイル及びディレクトリが削除されます。

- /opt/HiRDB_P
- ユーザが pdsetup コマンドで OS に登録したすべての HiRDB 運用ディレクトリ※

注※

pdsetup -d コマンドで HiRDB システムを OS から削除したときに，KFPS00036-Q メッセージに n で応答していた HiRDB 運用ディレクトリも削除対象になります。

HiRDB のアンインストール手順を次に示します。HiRDB/パラレルサーバの場合，HiRDB/パラレルサーバを構成するすべてのサーバマシンでアンインストール作業を実施してください。

〈手順〉

1. HiRDB を停止します。

   pdls -d ust コマンドを実行し，HiRDB が稼働中の場合は pdstop コマンドで
   HiRDB を停止します。

   稼働中の HiRDB が存在する状態でアンインストールを実行してしまった場合は，HiRDB プロセス及び HiRDB が使用していた OS 資源が居残ります。この場合は，OS を一度リブートすることによって，HiRDB プロセスの停止と OS 資源の解放をしてください。

2. HiRDB を OS から削除します。

   pdsetup -d コマンドを実行し，KFPS00036-Q メッセージに y で応答をして HiRDB を OS から削除します。

   pdsetup -d コマンドの終了ステータスが 0 でない場合，障害が起きているおそれがあります。syslog ファイルに出力されたメッセージを参照して，障害原因を取り除いてから，OS からの登録削除を再度実行してください。

3. アンインストールを実行します。

日立 PP インストーラを使用してサーバマシンごとに HiRDB をアンインストールします。このとき，日立 PP インストーラの画面で結果を確認してください。アンインストールが成功していない場合は，syslog ファイルに出力されたメッセージを参照して，障害原因を取り除いてから，アンインストールを再度実行してください。

3 コマンドによる環境設定

この章では，コマンドを使用して HiRDB の環境を設定する方法について説明します。

3.1　コマンドによる環境設定の概要

3.1.1　コマンドによる環境設定手順

(1)　環境設定の前に設計する項目

HiRDB の環境設定をする前に，システムの構成を設計してください。次に示す項目について設計します。

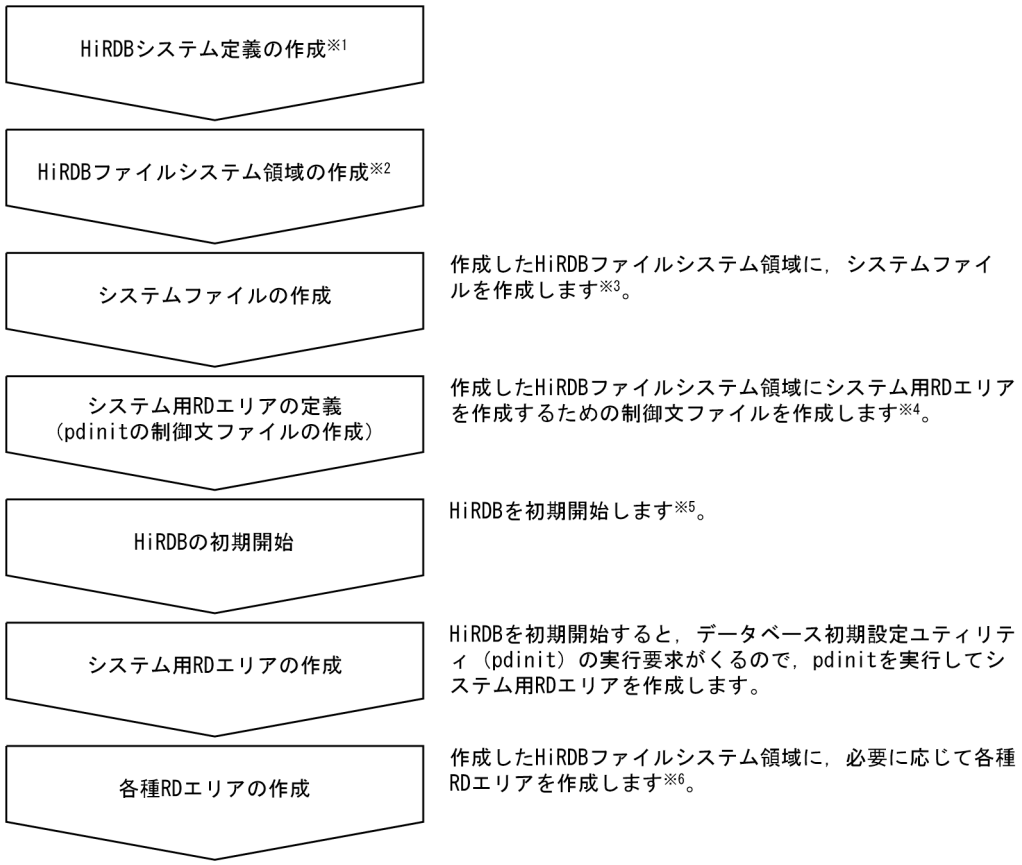
- ユニット及びサーバ構成
- HiRDB ファイルシステム領域の構成
- システムファイルの構成
- 作業表用ファイルの構成
- RD エリアの構成

上記の項目の構成を「HiRDB/シングルサーバの設計」又は「HiRDB/パラレルサーバの設計」を参照して決めてください。その後，4.2 以降の説明に従って HiRDB の環境を設定してください。

(2)　環境設定手順

コマンドによる HiRDB の環境設定手順を次の図に示します。

図 3-1　コマンドによる HiRDB の環境設定手順



注※1

詳細は，「HiRDB システム定義の作成」を参照してください。

注※2

詳細は，「HiRDB ファイルシステム領域の作成」を参照してください。

注※3

詳細は，「システムファイルの作成」を参照してください。

注※4

詳細は，「システム用 RD エリアの作成」を参照してください。

注※5

詳細は，「HiRDB の初期開始」を参照してください。

注※6

詳細は，「ユーザ用 RD エリアの作成」以降を参照してください。

設定する内容

- データベース初期設定ユティリティ（pdinit）でシステム用 RD エリア（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 エリア及びデータディクショナリ用 RD エリア）を作成して，HiRDB をまず開始できるようにします。

- その後，データベース構成変更ユティリティ（pdmod）で，必要な RD エリア（ユーザ用 RD エリア，データディクショナリ LOB 用 RD エリア，ユーザ LOB 用 RD エリア又はリスト用 RD エリア）を追加します。

なお，ユーザ用 RD エリア，データディクショナリ LOB 用 RD エリア，ユーザ LOB 用 RD エリア及びリスト用 RD エリアは，データベース初期設定ユティリティ（pdinit）でシステム用 RD エリアと一緒に作成することもできます。

3.2 HiRDB システム定義の作成

実行者　HiRDB 管理者

設計したシステム構成及び稼働環境に従って HiRDB システム定義を作成します。ここで説明する項目を次に示します。

- HiRDB システム定義の作成
- HiRDB システム定義ファイルの共用化（HiRDB/パラレルサーバ限定）
- HiRDB システム定義の変更方法
- UAP 環境定義の追加又は変更方法

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。

留意事項

- 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してください。このコマンドは 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。pdconfchk コマンドでチェックでき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。
- HiRDB システム定義ファイルのパーミッションは，ファイルの所有者（HiRDB 管理者）にだけ，読み込み権限及び書き込み権限を持たせるように設定，維持するようにしてください。
- HiRDB システム定義を変更した後に，$PDDIR/conf 下のファイルのバックアップを取得してください。HiRDB 運用ディレクトリがあるディスクの障害などに備えて，HiRDB 運用ディレクトリ下のファイル（$PDDIR/conf 下のファイル）のバックアップを取得します。HiRDB 運用ディレクトリを回復するには，$PDDIR/conf 下のファイルのバックアップが必要になります。

3.2.1 HiRDB システム定義の作成（HiRDB/シングルサーバの場合）

(1) システム共通定義の作成

システム共通定義には HiRDB の構成及び共通情報を定義します。システム共通定義を作成して次に示すファイルに格納します。

- $PDDIR/conf/pdsys

システム共通定義では，ユニット構成，サーバ構成，及びグローバルバッファの定義をします。

なお，HiRDB のコマンドやユティリティは，この定義ファイルの内容に従って動作するため，HiRDB のコマンド，又はユティリティを実行するユーザ（OS 上のユーザ）に対して，この定義ファイルに対する読み込み権限（r）を与えてください。

(2) ユニット制御情報定義の作成

ユニット制御情報定義にはユニットの実行環境を定義します。ユニット制御情報定義を作成して次に示すファイルに格納します。

- $PDDIR/conf/pdutsys

ユニット制御情報定義ではユニット用ステータスファイルの定義をします。

なお，HiRDB のコマンドやユティリティは，この定義ファイルの内容に従って動作するため，HiRDB のコマンド，又はユティリティを実行するユーザ（OS 上のユーザ）に対して，この定義ファイルに対する読み込み権限（r）を与えてください。

(3) シングルサーバ定義の作成

シングルサーバ定義にはシングルサーバの実行環境を定義します。シングルサーバ定義を作成して次に示すファイルに格納します。

- $PDDIR/conf/サーバ名※

HiRDB のコマンドやユティリティは，この定義ファイルの内容に従って動作するため，HiRDB のコマンド，又はユティリティを実行するユーザ（OS 上のユーザ）に対して，この定義ファイルに対する読み込み権限（r）を与えてください。

シングルサーバ定義で指定する項目の例を次に示します。

- システムログファイル
- シンクポイントダンプファイル
- サーバ用ステータスファイル
- 作業表用ファイル

なお，ユティリティ専用ユニットの場合は，シングルサーバ定義は必要ありません。

注※

システム共通定義の pdstart オペランドの-s オプションに指定するサーバ名と同じにしてください。「pdstart -s sds1」と指定した場合は，次に示すファイルに格納してください。

- $PDDIR/conf/sds1

(4) UAP 環境定義の作成（任意）

UAP の実行環境を定義します。必要に応じて UAP 環境定義を作成して次に示すファイルに格納します。

- $PDDIR/conf/pduapenv/任意の名称※

なお，HiRDB 管理者は UAP 環境定義を使用するユーザに対して，$PDDIR/conf/pduapenv ディレクトリの読み込み権限（r）と実行権限（x）を与えてください。また，UAP 環境定義ファイルには読み込み権限（r）を与えてください。

また，HiRDB のコマンドやユティリティは，この定義ファイルの内容に従って動作するため，HiRDB のコマンド，又はユティリティを実行するユーザ（OS 上のユーザ）に対して，この定義ファイルに対する読み込み権限（r）を与えてください。

UAP 環境定義で指定する項目の例を次に示します。

- ローカルバッファを使用してアクセスする RD エリア又はインデクスがほかのユーザに使用されている場合の UAP の動作
- UAP が使用するローカルバッファ

注※

ファイル名称は先頭がアルファベットの英数字列（最大 8 文字）にしてください。

(5) SQL 予約語定義の作成（任意）

SQL 予約語削除機能を使用する場合，UAP ごとに削除する予約語を定義します。必要に応じて SQL 予約語定義を作成して次に示すファイルに格納します。

- $PDDIR/conf/pdrsvwd/任意の名称※

なお，HiRDB 管理者は SQL 予約語定義を使用するユーザに対して，$PDDIR/conf/pdrsvwd ディレクトリの読み込み権限（r）と実行権限（x）を与えてください。また，SQL 予約語削除ファイルには読み込み権限（r）を与えてください。

また，HiRDB のコマンドやユティリティは，この定義ファイルの内容に従って動作するため，HiRDB のコマンド，又はユティリティを実行するユーザ（OS 上のユーザ）に対して，この定義ファイルに対する読み込み権限（r）を与えてください。

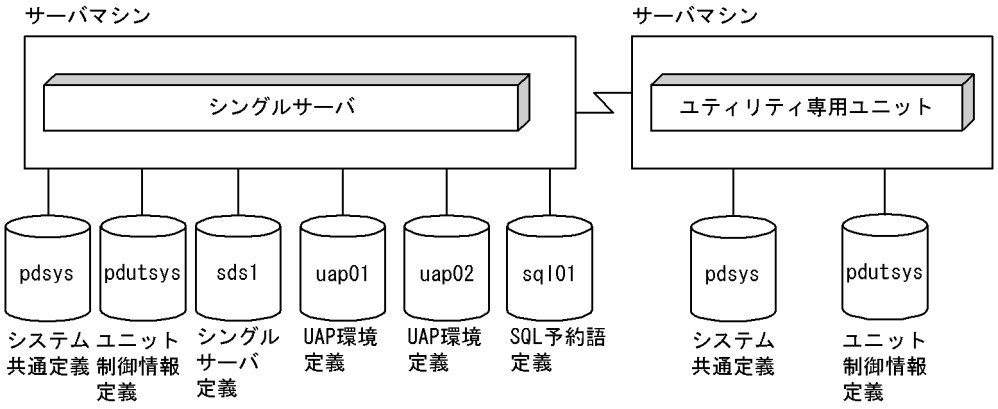
注※

ファイル名称は先頭がアルファベットの英数字列（最大 8 文字）にしてください。

(6) HiRDB システム定義ファイルの構成例

HiRDB システム定義ファイルの構成例を次の図に示します。

図 3-2　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，UAP環境定義，及びSQL予約語定義は必要ありません。

3.2.2　HiRDB システム定義の作成（HiRDB/パラレルサーバの場合）

(1)　システム共通定義の作成

システム共通定義には HiRDB の構成及び共通情報を定義します。システム共通定義を作成して次に示すファイルに格納します。

- $PDDIR/conf/pdsys

各サーバマシンに**同じ内容のシステム共通定義**を作成してください。

システム共通定義では，ユニット構成，サーバ構成，及びグローバルバッファの定義をします。

なお，HiRDB のコマンドやユティリティは，この定義ファイルの内容に従って動作するため，HiRDB のコマンド，又はユティリティを実行するユーザ（OS 上のユーザ）に対して，この定義ファイルに対する読み込み権限（r）を与えてください。

(2)　ユニット制御情報定義の作成

ユニット制御情報定義にはユニットの実行環境を定義します。ユニット制御情報定義を作成して次に示すファイルに格納します。

- $PDDIR/conf/pdutsys

各サーバマシンにユニット制御情報定義を作成してください。

ユニット制御情報定義では，ユニット用ステータスファイルの定義をします。

なお，HiRDB のコマンドやユティリティは，この定義ファイルの内容に従って動作するため，HiRDB のコマンド，又はユティリティを実行するユーザ（OS 上のユーザ）に対して，この定義ファイルに対する読み込み権限（r）を与えてください。

(3) サーバ共通定義の作成（任意）

サーバ共通定義には，(4)～(6)で説明するサーバ定義のオペランドの省略値を定義します。必要に応じて各サーバマシンにサーバ共通定義を作成して次に示すファイルに格納します。

- $PDDIR/conf/pdsvrc

次に示す場合にサーバ共通定義を作成すると便利です。

- 1 サーバマシンに定義するサーバ数が多い場合
- 各サーバの定義内容に共通部分が多い場合

サーバ共通定義で指定した内容は，そのサーバマシンに定義したすべてのサーバに対して有効となります。各サーバの定義内容に共通部分が多い場合は，共通部分をサーバ共通定義で指定し，異なる部分を各サーバ定義で指定することをお勧めします。

特に，HiRDB システム定義ファイルを共用化する場合は，サーバ共通定義を作成することをお勧めします。

また，HiRDB のコマンドやユティリティは，この定義ファイルの内容に従って動作するため，HiRDB のコマンド又はユティリティを実行するユーザ（OS 上のユーザ）に対して，この定義ファイルに対する読み込み権限（r）を与えてください。

(4) フロントエンドサーバ定義の作成

フロントエンドサーバ定義にはフロントエンドサーバの実行環境を定義します。フロントエンドサーバ定義を作成して次に示すファイルに格納します。

- $PDDIR/conf/サーバ名※

フロントエンドサーバを定義するサーバマシンに，フロントエンドサーバ定義を作成してください。

フロントエンドサーバ定義で指定する項目の例を次に示します。

- フロントエンドサーバ用のシステムログファイル
- フロントエンドサーバ用のシンクポイントダンプファイル
- フロントエンドサーバ用のステータスファイル

注※

システム共通定義の pdstart オペランドの-s オプションに指定するサーバ名と同じにしてください。例えば，「pdstart -s f001」と指定した場合は，次に示すファイルに格納してください。

- $PDDIR/conf/f001

なお，HiRDB のコマンドやユティリティは，この定義ファイルの内容に従って動作するため，HiRDB のコマンド又はユティリティを実行するユーザ（OS 上のユーザ）に対して，この定義ファイルに対する読み込み権限（r）を与えてください。

(5) ディクショナリサーバ定義の作成

ディクショナリサーバ定義にはディクショナリサーバの実行環境を定義します。ディクショナリサーバ定義を作成して次に示すファイルに格納します。

- $PDDIR/conf/サーバ名※

ディクショナリサーバを定義するサーバマシンに，ディクショナリサーバ定義を作成してください。

ディクショナリサーバ定義で指定する項目の例を次に示します。

- ディクショナリサーバ用のシステムログファイル
- ディクショナリサーバ用のシンクポイントダンプファイル
- ディクショナリサーバ用のステータスファイル
- 作業表用ファイル

注※

システム共通定義の pdstart オペランドの-s オプションに指定するサーバ名と同じにしてください。例えば，「pdstart -s dic」と指定した場合は，次に示すファイルに格納してください。

- $PDDIR/conf/dic

なお，HiRDB のコマンドやユティリティは，この定義ファイルの内容に従って動作するため，HiRDB のコマンド又はユティリティを実行するユーザ（OS 上のユーザ）に対して，この定義ファイルに対する読み込み権限（r）を与えてください。

(6) バックエンドサーバ定義の作成

バックエンドサーバ定義にはバックエンドサーバの実行環境を定義します。バックエンドサーバ定義を作成して次に示すファイルに格納します。

- $PDDIR/conf/サーバ名※

バックエンドサーバを定義するサーバマシンに，バックエンドサーバ定義を作成してください。

バックエンドサーバ定義で指定する項目の例を次に示します。

- バックエンドサーバ用のシステムログファイル
- バックエンドサーバ用のシンクポイントダンプファイル
- バックエンドサーバ用のステータスファイル
- 作業表用ファイル

注※

システム共通定義の pdstart オペランドの-s オプションに指定するサーバ名と同じにしてください。例えば，「pdstart -s b001」と指定した場合は，次に示すファイルに格納してください。

- $PDDIR/conf/b001

なお，HiRDB のコマンドやユティリティは，この定義ファイルの内容に従って動作するため，HiRDB のコマンド又はユティリティを実行するユーザ（OS 上のユーザ）に対して，この定義ファイルに対する読み込み権限（r）を与えてください。

(7) UAP 環境定義の作成（任意）

UAP の実行環境を定義します。必要に応じて UAP 環境定義を作成して次に示すファイルに格納します。

- $PDDIR/conf/pduapenv/任意の名称※

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。

なお，HiRDB 管理者は UAP 環境定義を使用するユーザに対して，$PDDIR/conf/pduapenv ディレクトリの読み込み権限（r）と実行権限（x）を与えてください。また，UAP 環境定義ファイルには読み込み権限（r）を与えてください。

また，HiRDB のコマンドやユティリティは，この定義ファイルの内容に従って動作するため，HiRDB のコマンド又はユティリティを実行するユーザ（OS 上のユーザ）に対して，この定義ファイルに対する読み込み権限（r）を与えてください。

UAP 環境定義で指定する項目の例を次に示します。

- ローカルバッファを使用してアクセスする RD エリア又はインデクスがほかのユーザに使用されている場合の UAP の動作
- UAP が使用するローカルバッファ

注※

ファイル名称は先頭がアルファベットの英数字列（最大 8 文字）にしてください。

(8) SQL 予約語定義の作成（任意）

SQL 予約語削除機能を使用する場合，UAP ごとに削除する予約語を定義します。必要に応じて SQL 予約語定義を作成して次に示すファイルに格納します。

- $PDDIR/conf/pdrsvwd/任意の名称※

SQL 予約語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。

なお，HiRDB 管理者は SQL 予約語定義を使用するユーザに対して，$PDDIR/conf/pdrsvwd ディレクトリの読み込み権限（r）と実行権限（x）を与えてください。また，SQL 予約語削除ファイルには読み込み権限（r）を与えてください。

また，HiRDB のコマンドやユティリティは，この定義ファイルの内容に従って動作するため，HiRDB のコマンド又はユティリティを実行するユーザ（OS 上のユーザ）に対して，この定義ファイルに対する読み込み権限（r）を与えてください。

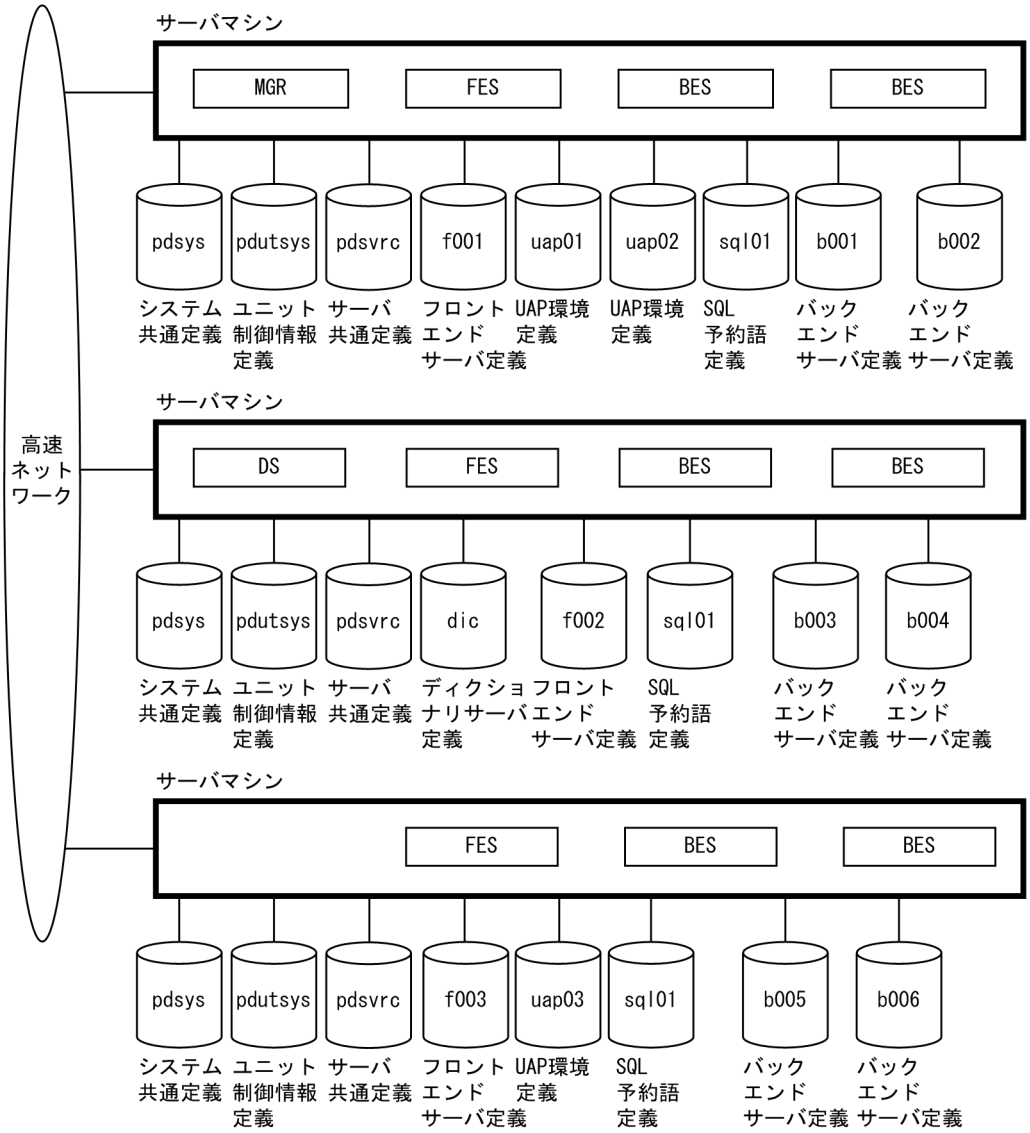
注※

ファイル名称は先頭がアルファベットの英数字列（最大８文字）にしてください。

(9) HiRDB システム定義ファイルの構成例

HiRDB システム定義ファイルの構成例を次の図に示します。

図 3-3　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）

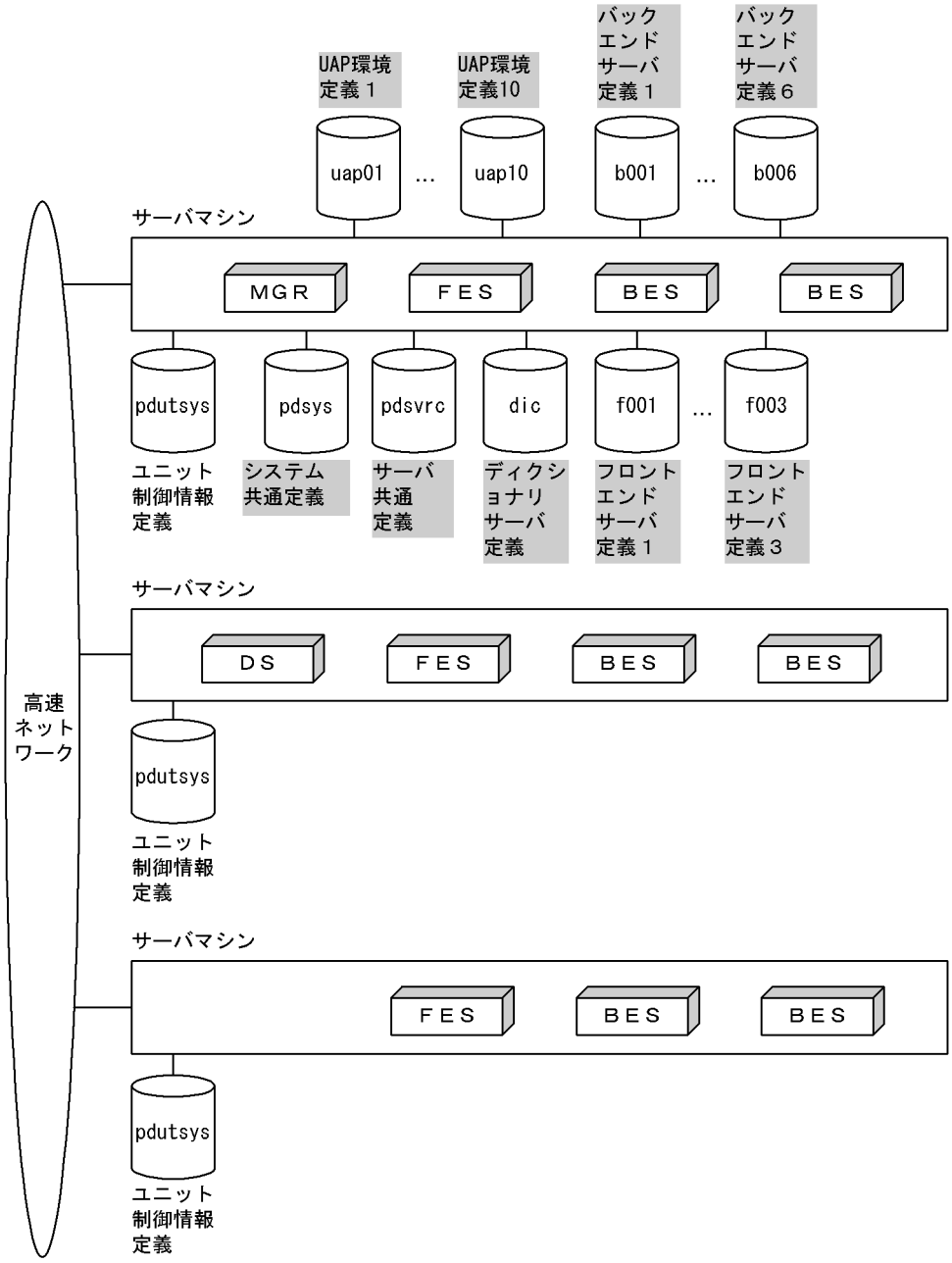


注１　各サーバマシンのシステム共通定義は同一内容にしてください。
注２　各サーバマシンのSQL予約語定義は同一内容にしてください。

3.2.3　HiRDB システム定義ファイルの共用化（HiRDB/パラレルサーバの場合）

HiRDB/パラレルサーバの場合，HiRDB システム定義ファイルをサーバマシンごとに作成して管理する必要があります。しかし，ファイル共用機能（NFS）を利用すると，ユニット制御情報定義ファイルを除いた HiRDB システム定義ファイルを一つのサーバマシンで管理できます。これを**HiRDB システム定義ファイルの共用化**といいます。HiRDB システム定義ファイルの共用化を次の図に示します。

図 3-4　HiRDB システム定義ファイルの共用化



注　網掛け部分が共用化したファイルです。

(1)　共用化しない場合とする場合の比較

共用化していない場合	共用化する場合
HiRDB システム定義をサーバマシンごとに管理する必要があります。	HiRDB システム定義を一つのサーバマシンで管理できます。ただし，ユニット制御情報定義を除きます。
システム共通定義を変更する場合，サーバマシンの数だけ修正が必要となります。	• システム共通定義は一つになるため，修正が 1 回で済みます。 • 修正回数が少なくなるため，修正誤りの可能性が少なくなります。

共用化していない場合	共用化する場合
例えば，四つのサーバマシンで構成されている場合，四つのシステム共通定義があります。システム共通定義の内容は同じである必要があるため，四つとも修正が必要となります。	• HiRDB が共用ディレクトリを参照できなくなると，HiRDB が異常終了することがあります。 また，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。

(2) 共用化するための手順

1. 共用化する HiRDB システム定義ファイルを格納する任意のディレクトリを作成します。このディレクトリは，HiRDB システム定義ファイルを共通管理するサーバマシンに作成してください。このディレクトリを**共用ディレクトリ**といいます。共通管理するサーバマシンは任意ですが，システムマネジャを定義するサーバマシンで管理することをお勧めします。
2. 共用ディレクトリ下にユニット制御情報定義ファイルを除いた HiRDB システム定義ファイルを作成します。
3. ユニット制御情報定義ファイルを各サーバマシンの次に示すディレクトリ下に作成します。
   - $PDDIR/conf/

   このとき，PDCONFPATH オペランドに共用ディレクトリ名称を指定します。

3.2.4 HiRDB システム定義（UAP 環境定義を除く）の変更方法

HiRDB システム定義を変更する手順を説明します。

留意事項

- HiRDB システム定義を変更した後に，$PDDIR/conf 下のファイルのバックアップを取得してください。HiRDB 運用ディレクトリがあるディスクの障害などに備えて，HiRDB 運用ディレクトリ下のファイル（$PDDIR/conf 下のファイル）のバックアップを取得します。HiRDB 運用ディレクトリを回復するには，$PDDIR/conf 下のファイルのバックアップが必要になります。また，$PDCONFPATH が HiRDB 運用ディレクトリ下にある場合は，同様にバックアップを取得してください。
- HiRDB/パラレルサーバの場合，ユニットごとに$PDDIR/conf 及び$PDCONFPATH 下にサブディレクトリを作成して，HiRDB システム定義の内容をチェックしてください。

(1) HiRDB システム定義の変更手順

HiRDB システム定義の変更手順を次に示します。なお，$PDDIR/conf はユニット制御情報定義ファイルを格納しているディレクトリを意味しています。$PDCONFPATH はそれ以外の HiRDB システム定義ファイルを格納しているディレクトリを意味しています。

〈手順〉

1. $PDDIR/conf 及び$PDCONFPATH 下にサブディレクトリを作成します。この例ではサブディレクトリとして work を作成します。
2. ユニット制御情報定義ファイルを$PDDIR/conf/work 下にコピーします。そのほかの HiRDB システム定義ファイルを$PDCONFPATH/work 下にコピーします。
3. $PDDIR/conf/work 及び$PDCONFPATH/work 下にコピーした HiRDB システム定義を変更します。
4. pdconfchk -d work コマンドで，$PDDIR/conf/work 及び$PDCONFPATH/work 下の HiRDB システム定義の内容をチェックします。エラーがある場合は HiRDB システム定義を修正して，再度 pdconfchk コマンドを実行してください。
5. pdstop コマンドで HiRDB を正常終了します。
6. 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します。
7. 3 で変更した HiRDB システム定義ファイルを$PDDIR/conf 又は$PDCONFPATH 下にコピーして，HiRDB システム定義ファイルを置き換えます。
8. 次に示すオペランドの指定値を変更した場合は，pdloginit コマンドでシステムログファイルを初期化します。
   - pd_log_dual
   - pdstart
9. pdstart コマンドで HiRDB を正常開始します。

(2) システム構成変更コマンドを使用した HiRDB システム定義の変更手順

システム構成変更コマンド（pdchgconf 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るため，HiRDB を終了する必要はありません。ただし，このコマンドを使用する場合は HiRDB Advanced High Availability が必要になります。システム構成変更コマンドを使用した HiRDB システム定義の変更手順を次に示します。

〈手順〉

1. $PDDIR/conf/chgconf ディレクトリを作成します。
2. 使用中の HiRDB システム定義ファイルを 1 で作成したディレクトリ下にコピーします。
3. $PDDIR/conf/chgconf 下の HiRDB システム定義を変更します。
4. pdconfchk コマンドで，$PDDIR/conf/chgconf 下の HiRDB システム定義のチェックを行います。エラーがある場合は HiRDB システム定義を修正して，再度 pdconfchk コマンドを実行してください。
5. pdchgconf コマンドで，HiRDB システム定義を変更後の HiRDB システム定義に置き換えます。
   pdchgconf コマンドを実行すると，使用中（変更前）の HiRDB システム定義ファイルが$PDDIR/conf/backconf 下に退避されます。そして，$PDDIR/conf/chgconf 下の変更後の HiRDB システム定義ファイルが$PDDIR/conf 下にコピーされます。

注意事項

- pdchgconf コマンドの入力後，15 分以上トランザクション又はユティリティが動き続けた場合，pdchgconf コマンドが異常終了します。
- システム構成変更コマンドを使用した HiRDB システム定義の変更には制限事項があります。制限事項については，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) 注意事項

- システム共通定義を修正する場合は，すべてのサーバマシンのシステム共通定義を同じように修正してください（HiRDB/パラレルサーバの場合）。
- 稼働中の HiRDB が使用している HiRDB システム定義は，変更又は削除しないでください。変更又は削除した場合，その HiRDB の動作は保証できません。
- HiRDB が計画停止，強制終了，又は異常終了した場合，HiRDB システム定義のオペランドで変更できるものと変更できないものがあります。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。

3.2.5 UAP 環境定義の追加又は変更方法

UAP 環境定義の追加又は変更手順を次に示します。

〈手順〉

1. UAP 環境定義を使用する UAP が実行中でないか確認します。UAP の実行中に UAP 環境定義を追加又は変更すると，実行中の UAP は変更前の UAP 環境定義が適用されます。ただし，タイミングによっては変更後の UAP 環境定義が適用されることがあります。
2. UAP 環境定義を追加又は変更します。
3. 追加又は変更した UAP 環境定義を使用して UAP を実行します。

3.3　HiRDB ファイルシステム領域の作成

実行者　HiRDB 管理者

HiRDB ファイルを作成する領域（**HiRDB ファイルシステム領域**）を pdfmkfs コマンドで作成します。HiRDB ファイルシステム領域を作成する場所は，通常ファイル又はキャラクタ型スペシャルファイル上になります。

なお，NFS などのネットワークを経由するファイルシステム上のファイルは，HiRDB ファイルシステム領域として使用できません。

3.3.1　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**用途ごとに**作成してください。用途は pdfmkfs コマンドの-k オプションで指定します。

表 3–1　HiRDB ファイルシステム領域の種類

項番	HiRDB ファイルシステム領域の種類	-k オプションの指定
1	RD エリア用	DB
2	共用 RD エリア用	SDB
3	システムファイル用	SYS
4	作業表用ファイル用	WORK
5	ユティリティ用	UTL
6	リスト用 RD エリア用	WORK

HiRDB を稼働するためには，1，3，及び 4 の HiRDB ファイルシステム領域が必要です。

HiRDB ファイルシステム領域の設計方法については，HiRDB/シングルサーバの場合は，「HiRDB ファイルシステム領域の設計」，HiRDB/パラレルサーバの場合は，「HiRDB ファイルシステム領域の設計」を参照してください。

注意事項

作成する HiRDB ファイルシステムの領域長は，パーティションの領域長と等しいか又は小さくしてください。パーティションの領域長より大きくすると，そのパーティションに物理的に続くパーティションを破壊する場合があります。

3.3.2　キャラクタ型スペシャルファイルを使用する場合

(1)　キャラクタ型スペシャルファイルの適用範囲

pdfmkfs コマンドの-k オプション（HiRDB ファイルシステム領域の使用目的）に UTL を指定した場合に，キャラクタ型スペシャルファイル上に作成できるユティリティ用ファイルを次に示します。

- バックアップファイル
- アンロードログファイル
- アンロードデータファイル
- 差分バックアップ管理ファイル
- インデクス情報ファイル

(2)　初期設定

キャラクタ型スペシャルファイルを使用するには，pdfmkfs コマンドでキャラクタ型スペシャルファイルを HiRDB ファイルシステム領域用に初期設定してください。キャラクタ型スペシャルファイルをシンボリックリンクした場合は，リンク名称を指定します。

なお，初期設定をする前に，ファイルシステム領域の所有者及びアクセス権を変更してください。変更方法については，「キャラクタ型スペシャルファイル又はブロック型スペシャルファイル上に HiRDB ファイルシステム領域を作成する場合」を参照してください。

pdfmkfs コマンドの例を次に示します。

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

3.3.3　ラージファイルを作成する場合

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は 1,048,575 メガバイトになります。

●ラージファイルの作成方法

ラージファイルを作成する方法は，通常の HiRDB ファイルシステム領域の作成方法と同じです。pdfmkfs コマンドの-n オプションで，ラージファイルとする HiRDB ファイルシステム領域のサイズ（2 ギガバイト以上）を指定します。

●ユティリティで使用するファイル

次に示すユティリティで使用するファイルのうち，ラージファイルを使用できるものとできないものがあります。

- 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）

これらのユティリティで使用するファイルが，ラージファイルを使用できるかどうかについては，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。

3.3.4 例題 1（RD エリア用の HiRDB ファイルシステム領域の作成）

RD エリア用の HiRDB ファイルシステム領域を作成する例を次に示します。

（例）

RD エリア用の HiRDB ファイルシステム領域を作成します。

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

［説明］

-n：HiRDB ファイルシステム領域の領域長をメガバイト単位で指定します。

-l：HiRDB ファイルシステム領域内に作成する HiRDB ファイル数の上限値を指定します。

-k：HiRDB ファイルシステム領域の用途を指定します。

RD エリア用の HiRDB ファイルシステム領域なので，DB を指定します。

-i：HiRDB ファイルシステム領域の全領域を初期化する場合に指定します。

-i オプションを指定すると，領域全体を確保します。-i オプションを省略すると，HiRDB ファイルシステム領域の管理情報だけを作成します。

/dbarea01：作成する HiRDB ファイルシステム領域の名称を指定します。

コマンドの実行後，実行結果が正しいかどうかを確認することをお勧めします。コマンドの実行結果の確認方法については，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。

3.3.5 例題 2（システムファイル用の HiRDB ファイルシステム領域の作成）

システムファイル用の HiRDB ファイルシステム領域を作成する例を次に示します。

（例）

システムファイル用の HiRDB ファイルシステム領域を作成します。

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

［説明］

-n：HiRDB ファイルシステム領域の領域長をメガバイト単位で指定します。

-l：HiRDB ファイルシステム領域内に作成する HiRDB ファイル数の上限値を指定します。

-k：HiRDB ファイルシステム領域の用途を指定します。

システムファイル用の HiRDB ファイルシステム領域なので，SYS を指定します。

-i：HiRDB ファイルシステム領域の全領域を初期化する場合に指定します。

-i オプションを指定すると，領域全体を確保します。-i オプションを省略すると，HiRDB ファイルシステム領域の管理情報だけを作成します。

/sysarea01：作成する HiRDB ファイルシステム領域の名称を指定します。

コマンドの実行後，実行結果が正しいかどうかを確認することをお勧めします。コマンドの実行結果の確認方法については，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。

3.3.6　例題 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）

作業表用ファイル用の HiRDB ファイルシステム領域を作成する例を次に示します。

（例）

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。

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

〔説明〕

-n：HiRDB ファイルシステム領域の領域長をメガバイト単位で指定します。

領域長の見積もり方法については，「作業表用ファイルの容量の見積もり」を参照してください。

-l：HiRDB ファイルシステム領域内に作成する HiRDB ファイル数の上限値を指定します。

-k：HiRDB ファイルシステム領域の用途を指定します。

作業表用ファイル用の HiRDB ファイルシステム領域なので，WORK を指定します。

-e：HiRDB ファイルシステム領域内の HiRDB ファイルの増分回数を指定します。

-i：HiRDB ファイルシステム領域の全領域を初期化する場合に指定します。

-i オプションを指定すると，領域全体を確保します。-i オプションを省略すると，HiRDB ファイルシステム領域の管理情報だけを作成します。

-a：自動的に HiRDB ファイルシステム領域を拡張するときに指定します。

RD エリアの自動増分や作業表を使用する SQL の実行などで，-n オプションで指定したサイズを超えても，自動的に必要な分だけ HiRDB ファイルシステム領域を拡張するときに指定します。

/workarea01：

作成する HiRDB ファイルシステム領域の名称を指定します。**HiRDB システム定義の pdwork オペランドで指定した名称を指定します。**

コマンドの実行後，実行結果が正しいかどうかを確認することをお勧めします。コマンドの実行結果の確認方法については，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。

3.3.7　例題４（ユティリティ用の HiRDB ファイルシステム領域の作成）

ユティリティ用の HiRDB ファイルシステム領域を作成する例を次に示します。ユティリティ用の HiRDB ファイルシステム領域には，次に示すファイルを作成します。

- 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
- 差分バックアップ管理ファイル
- インデクス情報ファイル

（例）

ユティリティ用の HiRDB ファイルシステム領域を作成します。

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

［説明］

-n：HiRDB ファイルシステム領域の領域長をメガバイト単位で指定します。

-l：HiRDB ファイルシステム領域内に作成する HiRDB ファイル数の上限値を指定します。

-k：HiRDB ファイルシステム領域の用途を指定します。

ユティリティ用の HiRDB ファイルシステム領域なので，UTL を指定します。

-i：HiRDB ファイルシステム領域の全領域を初期化する場合に指定します。

-i オプションを指定すると，領域全体を確保します。-i オプションを省略すると，HiRDB ファイルシステム領域の管理情報だけを作成します。

/utlarea01：作成する HiRDB ファイルシステム領域の名称を指定します。

コマンドの実行後，実行結果が正しいかどうかを確認することをお勧めします。コマンドの実行結果の確認方法については，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。

3.3.8　例題５（リスト用 RD エリア用の HiRDB ファイルシステム領域の作成）

リスト用 RD エリア用の HiRDB ファイルシステム領域を作成する例を次に示します。

（例）

リスト用 RD エリア用の HiRDB ファイルシステム領域を作成します。

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

［説明］

-n：HiRDB ファイルシステム領域の領域長をメガバイト単位で指定します。

-l：HiRDB ファイルシステム領域内に作成する HiRDB ファイル数の上限値を指定します。

-k：HiRDB ファイルシステム領域の用途を指定します。

リスト用 RD エリア用の HiRDB ファイルシステム領域なので，WORK を指定します。

-i：HiRDB ファイルシステム領域の全領域を初期化する場合に指定します。

-i オプションを指定すると，領域全体を確保します。-i オプションを省略すると，HiRDB ファイルシステム領域の管理情報だけを作成します。

/listarea01：作成する HiRDB ファイルシステム領域の名称を指定します。

コマンドの実行後，実行結果が正しいかどうかを確認することをお勧めします。コマンドの実行結果の確認方法については，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。

3.4　システムファイルの作成

実行者　HiRDB 管理者

「HiRDB ファイルシステム領域の作成」で作成したシステムファイル用の HiRDB ファイルシステム領域に，次に示すファイル（システムファイル）を作成します。

- 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

システムファイルの設計方法については，HiRDB/シングルサーバの場合は「システムファイルの設計」，HiRDB/パラレルサーバの場合は「システムファイルの設計」を参照してください。

3.4.1　システムログファイルの作成

pdloginit コマンドで，HiRDB ファイルシステム領域にシステムログファイルを作成します。

（例）

HiRDB ファイルシステム領域（/sysarea01）にシステムログファイル（log01）を作成します。

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

〔説明〕

-d sys：システムログファイルを作成する場合に指定します。

-s：システムログファイルを作成するサーバの名称を指定します。

HiRDB/シングルサーバの場合は指定は不要です。

-f：システムログファイルの名称を指定します。

HiRDB システム定義のサーバ定義の pdlogadpf -d sys オペランドで指定した名称を指定します。

-n：システムログファイルのレコード数を指定します。

1 システムログファイルの容量はレコード長×レコード数（バイト）になります。システムログファイルのレコード長は通常 1024 バイトですが，pd_log_rec_leng オペランドを指定している場合は pd_log_rec_leng オペランドの指定値になります。

コマンドの実行後，実行結果が正しいかどうか確認することをお勧めします。コマンドの実行結果の確認方法については，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。

HiRDB システム定義との関係

HiRDB システム定義のサーバ定義の次に示すオペランドと関係があります。

- pdlogadfg -d sys
- pdlogadpf -d sys

作成したシステムログファイルは，これらのオペランドで定義しておく必要があります。

3.4.2 シンクポイントダンプファイルの作成

pdloginit コマンドで，HiRDB ファイルシステム領域にシンクポイントダンプファイルを作成します。

（例）

HiRDB ファイルシステム領域（/sysarea01）にシンクポイントダンプファイル（sync01）を作成します。

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

〔説明〕

-d spd：シンクポイントダンプファイルを作成する場合に指定します。

-s：シンクポイントダンプファイルを作成するサーバの名称を指定します。

HiRDB/シングルサーバの場合は指定は不要です。

-f：シンクポイントダンプファイルの名称を指定します。

HiRDB システム定義のサーバ定義の pdlogadpf -d spd オペランドで指定した名称を指定します。

-n：シンクポイントダンプファイルのレコード数を指定します。

1 シンクポイントダンプファイルの容量は 4096×レコード数（バイト）になります。

コマンドの実行後，実行結果が正しいかどうか確認することをお勧めします。コマンドの実行結果の確認方法については，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。

HiRDB システム定義との関係

HiRDB システム定義のサーバ定義の次に示すオペランドと関係があります。

- pdlogadfg -d spd
- pdlogadpf -d spd

作成したシンクポイントダンプファイルは，これらのオペランドで定義しておく必要があります。

3.4.3 ステータスファイルの作成

pdstsinit コマンドで，HiRDB ファイルシステム領域にステータスファイルを作成します。ステータスファイルは，ユニット用ステータスファイルとサーバ用ステータスファイルの両方を作成します。

（例）

HiRDB ファイルシステム領域（/sysarea01）にサーバ用ステータスファイル（sts01）を作成します。

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

〔説明〕

-s：サーバ用ステータスファイルを作成するサーバの名称を指定します。

-f：サーバ用ステータスファイルの名称を指定します。

HiRDB システム定義のサーバ定義の pd_sts_file_name オペランドで指定した名称を指定します。

-l：ステータスファイルのレコード長を指定します。

-c：ステータスファイルのレコード数を指定します。

1 ステータスファイルの容量はレコード長×レコード数（バイト）になります。

コマンドの実行後，実行結果が正しいかどうか確認することをお勧めします。コマンドの実行結果の確認方法については，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。

HiRDB システム定義との関係

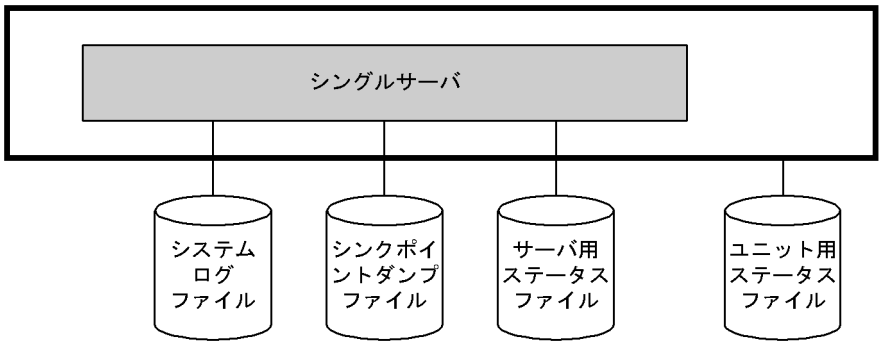
HiRDB システム定義の次に示すオペランドと関係があります。

- pd_syssts_file_name（ユニット用ステータスファイル）
- pd_sts_file_name（サーバ用ステータスファイル）

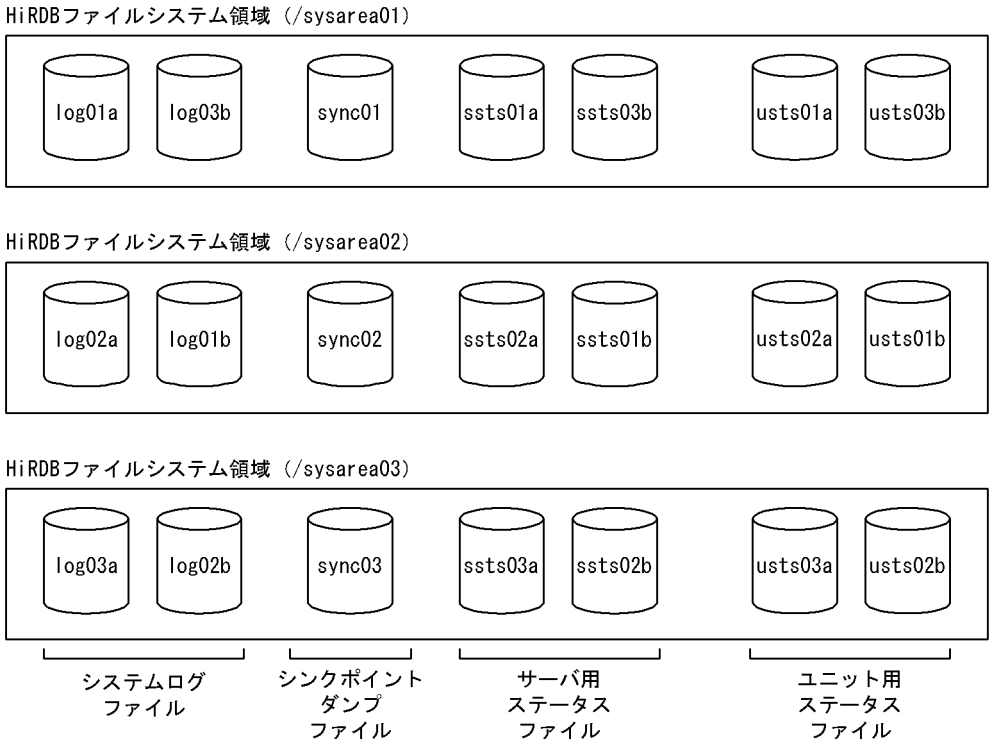
作成したステータスファイルは，これらのオペランドで定義しておく必要があります。

3.4.4 システムファイルの作成例（HiRDB/シングルサーバの場合）

次に示すシステム構成のシステムファイルの作成例を説明します。



HiRDB ファイルシステム領域の構成



(1) システムファイルの定義（HiRDB システム定義の指定）

HiRDB システム定義にシステムファイルを定義します。

(a) ユニット制御情報定義（ユニット用ステータスファイルの定義）

ユニット制御情報定義にユニット用ステータスファイルを定義します。

定義例

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

(b) シングルサーバ定義

シングルサーバ定義にシステムログファイル，シンクポイントダンプファイル，及びサーバ用ステータスファイルを定義します。

システムログファイルの定義例

```
pdlogadfg -d sys -g log1 ONL
pdlogadfg -d sys -g log2 ONL
pdlogadfg -d sys -g log3 ONL
pdlogadpf -d sys -g log1 -a "/sysarea01/log01a"¥
```

```
                          -b ”/sysarea02/log01b”
pdlogadpf -d sys -g log2 -a ”/sysarea02/log02a”¥
                          -b ”/sysarea03/log02b”
pdlogadpf -d sys -g log3 -a ”/sysarea03/log03a”¥
                          -b ”/sysarea01/log03b”
```

シンクポイントダンプファイルの定義例

```
pdlogadfg -d spd -g sync1 ONL
pdlogadfg -d spd -g sync2 ONL
pdlogadfg -d spd -g sync3 ONL
pdlogadpf -d spd -g sync1 -a ”/sysarea01/sync01”
pdlogadpf -d spd -g sync2 -a ”/sysarea02/sync02”
pdlogadpf -d spd -g sync3 -a ”/sysarea03/sync03”
```

サーバ用ステータスファイルの定義例

```
set pd_sts_file_name_1=”ssts1”,”/sysarea01/ssts01a”¥
                              ,”/sysarea02/ssts01b”
set pd_sts_file_name_2=”ssts2”,”/sysarea02/ssts02a”¥
                              ,”/sysarea03/ssts02b”
set pd_sts_file_name_3=”ssts3”,”/sysarea03/ssts03a”¥
                              ,”/sysarea01/ssts03b”
```

(2) HiRDB ファイルシステム領域の作成

pdfmkfs コマンドで HiRDB ファイルシステム領域を作成します。

コマンドの入力例

```
pdfmkfs -n 50 -l 20 -i -k SYS /sysarea01
pdfmkfs -n 50 -l 20 -i -k SYS /sysarea02
pdfmkfs -n 50 -l 20 -i -k SYS /sysarea03
```

(3) システムファイルの作成

(a) 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。

コマンドの入力例

```
pdloginit -d sys -f /sysarea01/log01a -n 1024
pdloginit -d sys -f /sysarea01/log03b -n 1024
pdloginit -d sys -f /sysarea02/log02a -n 1024
pdloginit -d sys -f /sysarea02/log01b -n 1024
pdloginit -d sys -f /sysarea03/log03a -n 1024
pdloginit -d sys -f /sysarea03/log02b -n 1024
```

(b) 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。

コマンドの入力例

```
pdloginit -d spd -f /sysarea01/sync01 -n 64
pdloginit -d spd -f /sysarea02/sync02 -n 64
pdloginit -d spd -f /sysarea03/sync03 -n 64
```

(c)　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。

コマンドの入力例

```
pdstsinit -s sds1 -f /sysarea01/ssts01a -l 4096 -c 256
pdstsinit -s sds1 -f /sysarea01/ssts03b -l 4096 -c 256
pdstsinit -s sds1 -f /sysarea02/ssts02a -l 4096 -c 256
pdstsinit -s sds1 -f /sysarea02/ssts01b -l 4096 -c 256
pdstsinit -s sds1 -f /sysarea03/ssts03a -l 4096 -c 256
pdstsinit -s sds1 -f /sysarea03/ssts02b -l 4096 -c 256
```

(d)　ユニット用ステータスファイルの作成

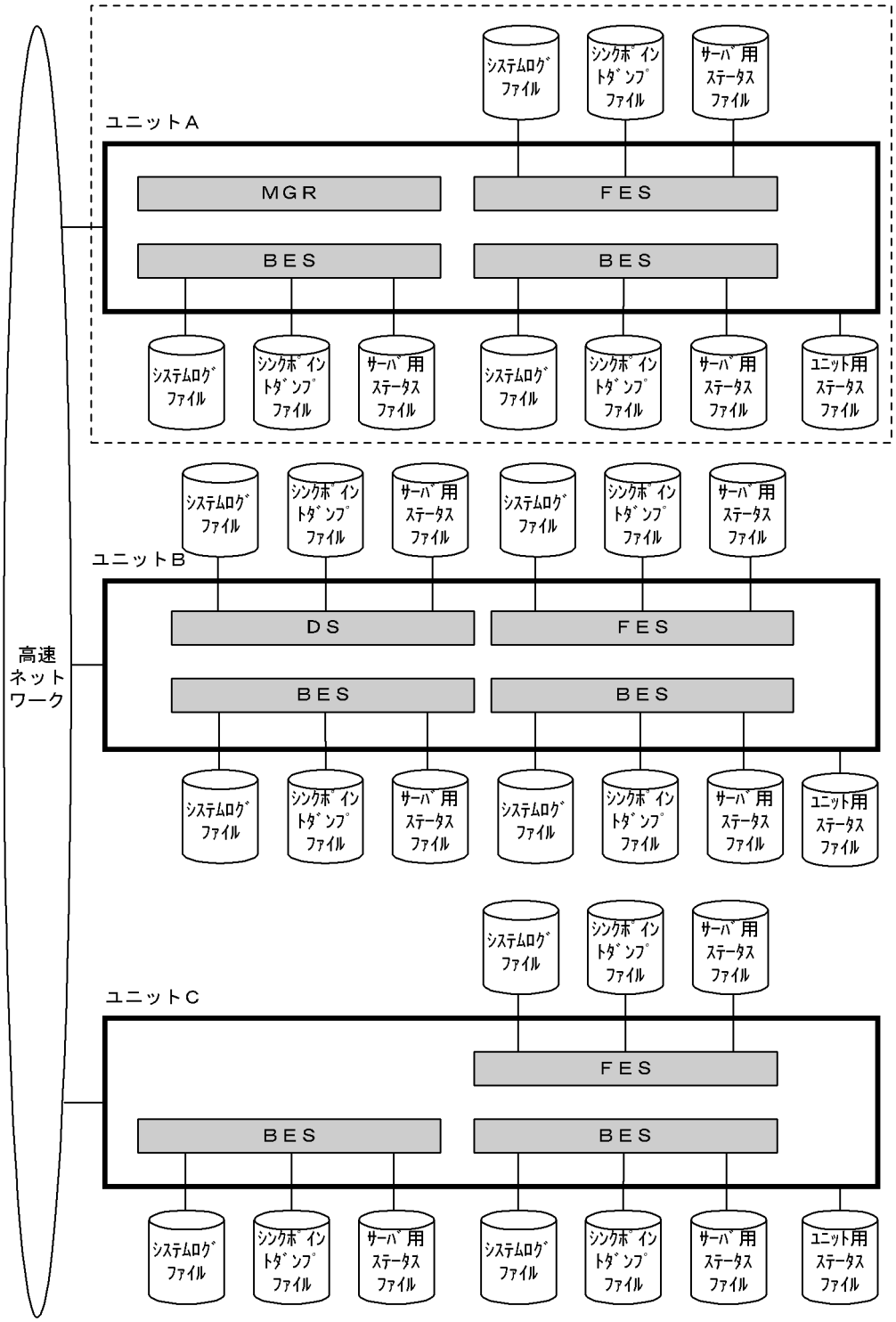
pdstsinit コマンドでユニット用ステータスファイルを作成します。

コマンドの入力例

```
pdstsinit -u unt1 -f /sysarea01/usts01a -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea01/usts03b -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea02/usts02a -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea02/usts01b -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea03/usts03a -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea03/usts02b -l 4096 -c 256
```

3.4.5　システムファイルの作成例（HiRDB/パラレルサーバの場合）

次に示すシステム構成のシステムファイルの作成例を説明します。



（凡例）

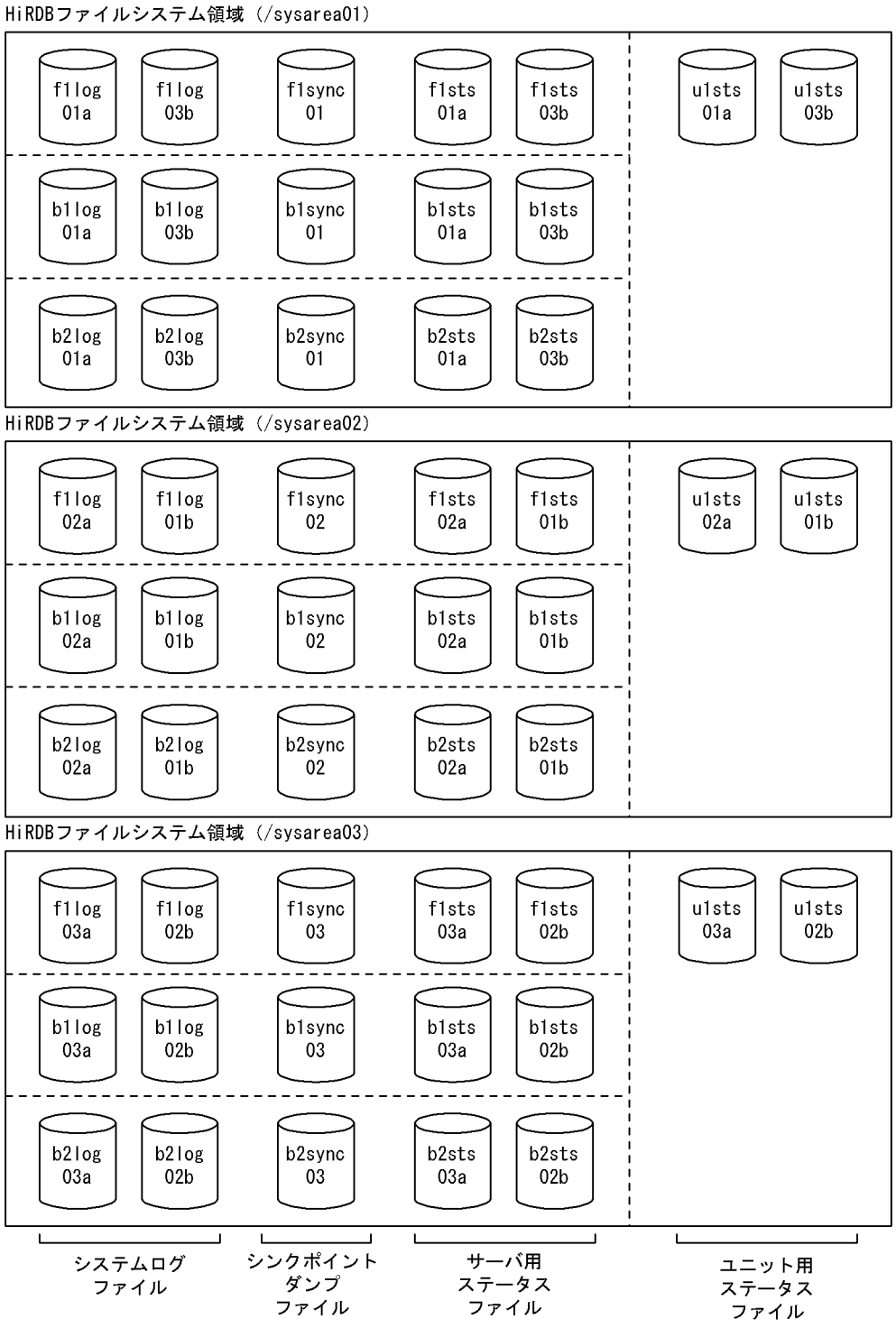
MGR：システムマネジャ

FES：フロントエンドサーバ

DS：ディクショナリサーバ

BES：バックエンドサーバ

HiRDB ファイルシステム領域の構成



［説明］

ユニット A の HiRDB ファイルシステム領域の構成例です。以降の例題では，ユニット A のシステムファイルの作成例についてだけ説明します。

(1) システムファイルの定義（HiRDB システム定義の指定）

HiRDB システム定義にシステムファイルを定義します。

(a) ユニット制御情報定義（ユニット用ステータスファイルの定義）

ユニット制御情報定義にユニット用ステータスファイルを定義します。

定義例

```
set pd_syssts_file_name_1="u1sts1","/sysarea01/u1sts01a"¥
                                  ,"/sysarea02/u1sts01b"
set pd_syssts_file_name_2="u1sts2","/sysarea02/u1sts02a"¥
                                  ,"/sysarea03/u1sts02b"
set pd_syssts_file_name_3="u1sts3","/sysarea03/u1sts03a"¥
                                  ,"/sysarea01/u1sts03b"
```

(b) FES1 のフロントエンドサーバ定義

FES1 のフロントエンドサーバ定義にシステムログファイル，シンクポイントダンプファイル，及びサーバ用ステータスファイルを定義します。

システムログファイルの定義例

```
pdlogadfg -d sys -g f1log1 ONL
pdlogadfg -d sys -g f1log2 ONL
pdlogadfg -d sys -g f1log3 ONL
pdlogadpf -d sys -g f1log1 -a "/sysarea01/f1log01a"¥
                           -b "/sysarea02/f1log01b"
pdlogadpf -d sys -g f1log2 -a "/sysarea02/f1log02a"¥
                           -b "/sysarea03/f1log02b"
pdlogadpf -d sys -g f1log3 -a "/sysarea03/f1log03a"¥
                           -b "/sysarea01/f1log03b"
```

シンクポイントダンプファイルの定義例

```
pdlogadfg -d spd -g f1sync1 ONL
pdlogadfg -d spd -g f1sync2 ONL
pdlogadfg -d spd -g f1sync3 ONL
pdlogadpf -d spd -g f1sync1 -a "/sysarea01/f1sync01"
pdlogadpf -d spd -g f1sync2 -a "/sysarea02/f1sync02"
pdlogadpf -d spd -g f1sync3 -a "/sysarea03/f1sync03"
```

サーバ用ステータスファイルの定義例

```
set pd_sts_file_name_1="f1sts1","/sysarea01/f1sts01a"¥
                               ,"/sysarea02/f1sts01b"
set pd_sts_file_name_2="f1sts2","/sysarea02/f1sts02a"¥
                               ,"/sysarea03/f1sts02b"
set pd_sts_file_name_3="f1sts3","/sysarea03/f1sts03a"¥
                               ,"/sysarea01/f1sts03b"
```

(c) BES1 のバックエンドサーバ定義

BES1 のバックエンドサーバ定義にシステムログファイル，シンクポイントダンプファイル，及びサーバ用ステータスファイルを定義します。

システムログファイルの定義例

```
pdlogadfg -d sys -g b1log1 ONL
pdlogadfg -d sys -g b1log2 ONL
pdlogadfg -d sys -g b1log3 ONL
pdlogadpf -d sys -g b1log1 -a "/sysarea01/b1log01a"¥
                           -b "/sysarea02/b1log01b"
pdlogadpf -d sys -g b1log2 -a "/sysarea02/b1log02a"¥
                           -b "/sysarea03/b1log02b"
pdlogadpf -d sys -g b1log3 -a "/sysarea03/b1log03a"¥
                           -b "/sysarea01/b1log03b"
```

シンクポイントダンプファイルの定義例

```
pdlogadfg -d spd -g b1sync1 ONL
pdlogadfg -d spd -g b1sync2 ONL
pdlogadfg -d spd -g b1sync3 ONL
pdlogadpf -d spd -g b1sync1 -a "/sysarea01/b1sync01"
pdlogadpf -d spd -g b1sync2 -a "/sysarea02/b1sync02"
pdlogadpf -d spd -g b1sync3 -a "/sysarea03/b1sync03"
```

サーバ用ステータスファイルの定義例

```
set pd_sts_file_name_1="b1sts1","/sysarea01/b1sts01a"¥
                              ,"/sysarea02/b1sts01b"
set pd_sts_file_name_2="b1sts2","/sysarea02/b1sts02a"¥
                              ,"/sysarea03/b1sts02b"
set pd_sts_file_name_3="b1sts3","/sysarea03/b1sts03a"¥
                              ,"/sysarea01/b1sts03b"
```

(d) BES2 のバックエンドサーバ定義

BES2 のバックエンドサーバ定義にシステムログファイル，シンクポイントダンプファイル，及びサーバ用ステータスファイルを定義します。

システムログファイルの定義例

```
pdlogadfg -d sys -g b2log1 ONL
pdlogadfg -d sys -g b2log2 ONL
pdlogadfg -d sys -g b2log3 ONL
pdlogadpf -d sys -g b2log1 -a "/sysarea01/b2log01a"¥
                           -b "/sysarea02/b2log01b"
pdlogadpf -d sys -g b2log2 -a "/sysarea02/b2log02a"¥
                           -b "/sysarea03/b2log02b"
pdlogadpf -d sys -g b2log3 -a "/sysarea03/b2log03a"¥
                           -b "/sysarea01/b2log03b"
```

シンクポイントダンプファイルの定義例

```
pdlogadfg -d spd -g b2sync1 ONL
pdlogadfg -d spd -g b2sync2 ONL
pdlogadfg -d spd -g b2sync3 ONL
pdlogadpf -d spd -g b2sync1 -a "/sysarea01/b2sync01"
pdlogadpf -d spd -g b2sync2 -a "/sysarea02/b2sync02"
pdlogadpf -d spd -g b2sync3 -a "/sysarea03/b2sync03"
```

サーバ用ステータスファイルの定義例

```
set pd_sts_file_name_1=”b2sts1”,”/sysarea01/b2sts01a”¥
                               ,”/sysarea02/b2sts01b”
set pd_sts_file_name_2=”b2sts2”,”/sysarea02/b2sts02a”¥
                               ,”/sysarea03/b2sts02b”
set pd_sts_file_name_3=”b2sts3”,”/sysarea03/b2sts03a”¥
                               ,”/sysarea01/b2sts03b”
```

(2) HiRDB ファイルシステム領域の作成

pdfmkfs コマンドで HiRDB ファイルシステム領域を作成します。

コマンドの入力例

```
pdfmkfs -n 50 -l 20 -i -k SYS /sysarea01
pdfmkfs -n 50 -l 20 -i -k SYS /sysarea02
pdfmkfs -n 50 -l 20 -i -k SYS /sysarea03
```

(3) システムファイルの作成

(a) 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。

コマンドの入力例（FES1 用）

```
pdloginit -d sys -s f001 -f /sysarea01/f1log01a -n 1024
pdloginit -d sys -s f001 -f /sysarea01/f1log03b -n 1024
pdloginit -d sys -s f001 -f /sysarea02/f1log02a -n 1024
pdloginit -d sys -s f001 -f /sysarea02/f1log01b -n 1024
pdloginit -d sys -s f001 -f /sysarea03/f1log03a -n 1024
pdloginit -d sys -s f001 -f /sysarea03/f1log02b -n 1024
```

コマンドの入力例（BES1 用）

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

コマンドの入力例（BES2 用）

```
pdloginit -d sys -s b002 -f /sysarea01/b2log01a -n 1024
pdloginit -d sys -s b002 -f /sysarea01/b2log03b -n 1024
pdloginit -d sys -s b002 -f /sysarea02/b2log02a -n 1024
pdloginit -d sys -s b002 -f /sysarea02/b2log01b -n 1024
pdloginit -d sys -s b002 -f /sysarea03/b2log03a -n 1024
pdloginit -d sys -s b002 -f /sysarea03/b2log02b -n 1024
```

(b) 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。

コマンドの入力例（FES1 用）

```
pdloginit -d spd -s f001 -f /sysarea01/f1sync01 -n 64
pdloginit -d spd -s f001 -f /sysarea02/f1sync02 -n 64
pdloginit -d spd -s f001 -f /sysarea03/f1sync03 -n 64
```

コマンドの入力例（BES1 用）

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

コマンドの入力例（BES2 用）

```
pdloginit -d spd -s b002 -f /sysarea01/b2sync01 -n 64
pdloginit -d spd -s b002 -f /sysarea02/b2sync02 -n 64
pdloginit -d spd -s b002 -f /sysarea03/b2sync03 -n 64
```

(c) 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。

コマンドの入力例（FES1 用）

```
pdstsinit -s f001 -f /sysarea01/f1sts01a -l 4096 -c 256
pdstsinit -s f001 -f /sysarea01/f1sts03b -l 4096 -c 256
pdstsinit -s f001 -f /sysarea02/f1sts02a -l 4096 -c 256
pdstsinit -s f001 -f /sysarea02/f1sts01b -l 4096 -c 256
pdstsinit -s f001 -f /sysarea03/f1sts03a -l 4096 -c 256
pdstsinit -s f001 -f /sysarea03/f1sts02b -l 4096 -c 256
```

コマンドの入力例（BES1 用）

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

コマンドの入力例（BES2 用）

```
pdstsinit -s b002 -f /sysarea01/b2sts01a -l 4096 -c 256
pdstsinit -s b002 -f /sysarea01/b2sts03b -l 4096 -c 256
pdstsinit -s b002 -f /sysarea02/b2sts02a -l 4096 -c 256
pdstsinit -s b002 -f /sysarea02/b2sts01b -l 4096 -c 256
pdstsinit -s b002 -f /sysarea03/b2sts03a -l 4096 -c 256
pdstsinit -s b002 -f /sysarea03/b2sts02b -l 4096 -c 256
```

(d) 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。

コマンドの入力例

```
pdstsinit -u unt1 -f /sysarea01/u1sts01a -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea01/u1sts03b -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea02/u1sts02a -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea02/u1sts01b -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea03/u1sts03a -l 4096 -c 256
pdstsinit -u unt1 -f /sysarea03/u1sts02b -l 4096 -c 256
```

3.5　システム用 RD エリアの作成

実行者　HiRDB 管理者

HiRDB を初期開始するときは，**データベース初期設定ユティリティ（pdinit）**でシステム用 RD エリアを作成する必要があります。

データベース初期設定ユティリティ（pdinit）は，HiRDB を初期開始する場合（インストールしてから最初に pdstart コマンドを実行するとき）に，コマンドの入力要求が来たときに実行します。それ以外のタイミングでデータベース初期設定ユティリティ（pdinit）は実行できません。

ここでは，データベース初期設定ユティリティ（pdinit）の引数として指定する**制御文ファイル**の内容とデータベース初期設定ユティリティ（pdinit）の実行例について説明します。システム用 RD エリアは，create rdarea 文で作成します。

システム用 RD エリアとは，次に示す RD エリアのことです。

- 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

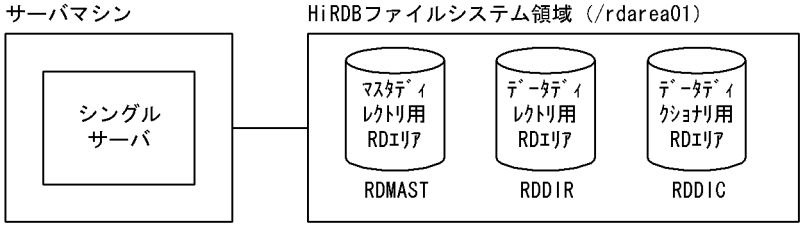
3.5.1　基本事項

1. システム用 RD エリアは，「HiRDB ファイルシステム領域の作成」で作成した RD エリア用の HiRDB ファイルシステム領域に作成します。
2. HiRDB/パラレルサーバの場合，ディクショナリサーバを定義したサーバマシンの HiRDB ファイルシステム領域にシステム用 RD エリアを作成します。
3. システム用 RD エリアの設計方法については，HiRDB/シングルサーバの場合は「RD エリアの配置」を，HiRDB/パラレルサーバの場合は「RD エリアの配置」を参照してください。
4. HiRDB/パラレルサーバの場合，システムマネジャを定義したサーバマシンでデータベース初期設定ユティリティ（pdinit）を実行してください。
5. ここの説明では，データベース初期設定ユティリティ（pdinit）の create rdarea 文でシステム用 RD エリアしか作成しません。これはシステム用 RD エリアが HiRDB の稼働に必要であるためです。ただし，次に示す RD エリアもデータベース初期設定ユティリティ（pdinit）の create rdarea 文で定義できます。
   - ユーザ用 RD エリア
   - ユーザ LOB 用 RD エリア
   - データディクショナリ LOB 用 RD エリア
   - リスト用 RD エリア

3.5.2 例題 1（HiRDB/シングルサーバの場合）

次に示す RD エリア用の HiRDB ファイルシステム領域に，システム用 RD エリアを作成します。

- /rdarea01



(1) 制御文ファイルの作成

データベース初期設定ユティリティ（pdinit）の引数に指定する制御文ファイルを作成します。制御文ファイルを作成する場所は任意ですが，ここでは次に示すファイル名で作成することにします。

- /usr/hirdb/pdinit01

制御文ファイルの内容

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

〔説明〕

1. マスタディレクトリ用 RD エリアの定義

   HiRDB ファイルシステム領域に rdmast01 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 10 とします。

2. データディレクトリ用 RD エリアの定義

   HiRDB ファイルシステム領域に rddir01 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 5 とします。

3. データディクショナリ用 RD エリアの定義

   HiRDB ファイルシステム領域に rddic01 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 20 とします。

   RD エリアの自動増分機能を使用します。増分量を 50 セグメントとします。

(2) データベース初期設定ユティリティ（pdinit）の実行

コマンドの入力例

```
pdinit -d /usr/hirdb/pdinit01
```

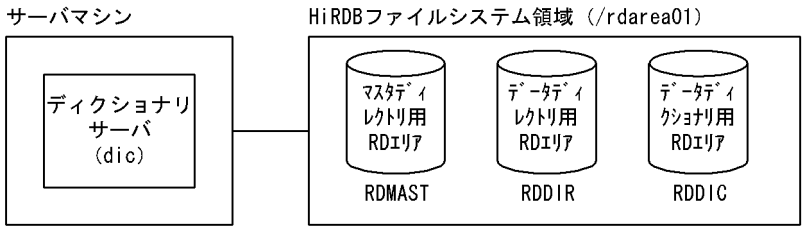
［説明］

-d：(1）で作成した制御文ファイルのファイル名を指定します。

3.5.3 例題 2（HiRDB/パラレルサーバの場合）

次に示す RD エリア用の HiRDB ファイルシステム領域に，システム用 RD エリアを作成します。システム用 RD エリアは，ディクショナリサーバを定義したサーバマシンに作成します。

- /rdarea01



(1) 制御文ファイルの作成

データベース初期設定ユティリティ（pdinit）の引数に指定する制御文ファイルを作成します。制御文ファイルを作成する場所は任意ですが，ここでは次に示すファイル名で作成することにします。

- /usr/hirdb/pdinit01

制御文ファイルの内容

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

［説明］

1. マスタディレクトリ用 RD エリアの定義

マスタディレクトリ用 RD エリアを管理するディクショナリサーバの名称（dic）を指定します。HiRDB ファイルシステム領域に rdmast01 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 10 とします。

2. データディレクトリ用 RD エリアの定義

   データディレクトリ用 RD エリアを管理するディクショナリサーバの名称（dic）を指定します。HiRDB ファイルシステム領域に rddir01 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 5 とします。

3. データディクショナリ用 RD エリアの定義

   データディクショナリ用 RD エリアを管理するディクショナリサーバの名称（dic）を指定します。HiRDB ファイルシステム領域に rddic01 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 20 とします。

   RD エリアの自動増分機能を使用します。増分量を 50 セグメントとします。

(2) データベース初期設定ユティリティ（pdinit）の実行

コマンドの入力例

```
pdinit -d /usr/hirdb/pdinit01
```

〔説明〕

-d：(1）で作成した制御文ファイルのファイル名を指定します。

3.6　HiRDB の初期開始

3.6.1　HiRDB の初期開始で行うこと

実行者　HiRDB 管理者

「システム用 RD エリアの作成」で説明したデータベース初期設定ユティリティ（pdinit）は，HiRDB の初期開始コマンド（pdstart コマンド）を実行してから，その実行途中でだけ実行できます。

(1)　HiRDB の初期開始の方法

HiRDB ファイルシステム領域を作成してから HiRDB を最初に開始（初期開始）するときは，pdstart コマンドを実行します。初期開始のために pdstart コマンドを実行すると，データベース初期設定ユティリティ（pdinit）の実行を要求するメッセージが表示されます。

- HiRDB/シングルサーバを開始する場合は，シングルサーバを定義したサーバマシンから pdstart コマンドを実行してください。
- HiRDB/パラレルサーバを開始する場合は，システムマネジャを定義したサーバマシンから pdstart コマンドを実行してください。

(2)　RD エリアの作成の前提条件

次に示す RD エリアを作成するときは，HiRDB が稼働中であることが前提です。事前に HiRDB を開始しておいてください。

- ユーザ用 RD エリア
- ユーザ LOB 用 RD エリア
- データディクショナリ LOB 用 RD エリア
- リスト用 RD エリア

3.7　ユーザ用 RD エリアの作成

実行者　HiRDB 管理者

表及びインデクスを格納するユーザ用 RD エリアを作成します。ユーザ用 RD エリアは，**データベース構成変更ユティリティ（pdmod）**の create rdarea 文で作成します。

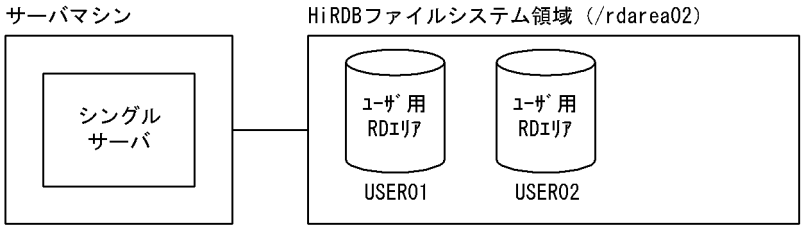
3.7.1　基本事項

1. 「HiRDB ファイルシステム領域の作成」で作成した RD エリア用の HiRDB ファイルシステム領域に，ユーザ用 RD エリアを作成します。
2. HiRDB/パラレルサーバの場合，バックエンドサーバを定義したサーバマシンの HiRDB ファイルシステム領域にユーザ用 RD エリアを作成します。
3. ユーザ用 RD エリアの設計方法については，HiRDB/シングルサーバの場合は「RD エリアの配置」を，HiRDB/パラレルサーバの場合は「RD エリアの配置」を参照してください。
4. HiRDB/パラレルサーバの場合，システムマネジャを定義したサーバマシンでデータベース構成変更ユティリティ（pdmod）を実行してください。
5. ユーザ用 RD エリアを作成する前に，HiRDB が稼働しているかどうかを pdls コマンドで確認してください。HiRDB/パラレルサーバの場合は，システムマネジャを定義したサーバマシンから pdls コマンドを入力してください。
6. HiRDB が稼働していない場合，pdstart コマンドで HiRDB を開始してください。HiRDB/パラレルサーバを開始する場合は，システムマネジャを定義したサーバマシンから pdstart コマンドを入力してください。

3.7.2　例題 1（HiRDB/シングルサーバの場合）

次に示す RD エリア用の HiRDB ファイルシステム領域に，ユーザ用 RD エリアを作成します。

- /rdarea02



(1) 制御文ファイルの作成

データベース構成変更ユティリティ（pdmod）の引数に指定する制御文ファイルを作成します。制御文ファイルを作成する場所は任意ですが，ここでは次に示すファイル名で作成することにします。

- /usr/hirdb/pdmod01

制御文ファイルの内容

```
create rdarea USER01 for user used by PUBLIC                    1
    extension use 50 segments
    file name ”/rdarea02/user01”
         initial 500 segments;
create rdarea USER02 for user used by PUBLIC                    2
    extension use 50 segments
    file name ”/rdarea02/user02”
         initial 500 segments;
```

〔説明〕

1. ユーザ用 RD エリア（USER01）の定義

   USER01 を公用 RD エリア（PUBLIC）とします。HiRDB ファイルシステム領域に user01 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 500 とします。

   RD エリアの自動増分機能を使用します。増分量を 50 セグメントとします。

2. ユーザ用 RD エリア（USER02）の定義

   USER02 を公用 RD エリア（PUBLIC）とします。HiRDB ファイルシステム領域に user02 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 500 とします。

   RD エリアの自動増分機能を使用します。増分量を 50 セグメントとします。

(2) データベース構成変更ユティリティ（pdmod）の実行

コマンドの入力例

```
pdmod -a /usr/hirdb/pdmod01
```

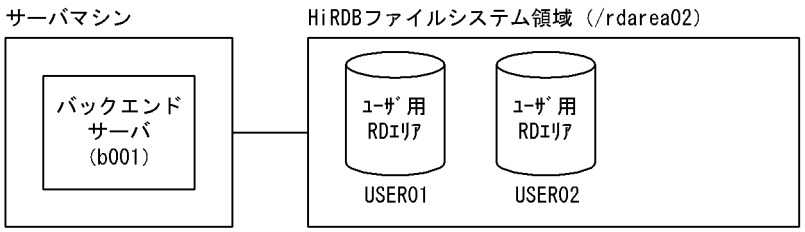
〔説明〕

-a：(1）で作成した制御文ファイルのファイル名を指定します。

3.7.3 例題 2（HiRDB/パラレルサーバの場合）

次に示す RD エリア用の HiRDB ファイルシステム領域に，ユーザ用 RD エリアを作成します。ユーザ用 RD エリアは，バックエンドサーバを定義したサーバマシンに作成します。

- /rdarea02



(1) 制御文ファイルの作成

データベース構成変更ユティリティ（pdmod）の引数に指定する制御文ファイルを作成します。制御文ファイルを作成する場所は任意ですが，ここでは次に示すファイル名で作成することにします。

- /usr/hirdb/pdmod01

制御文ファイルの内容

```
create rdarea USER01 for user used by PUBLIC                  1
    server name b001
    extension use 50 segments
    file name ”/rdarea02/user01”
         initial 500 segments;
create rdarea USER02 for user used by PUBLIC                  2
    server name b001
    extension use 50 segments
    file name ”/rdarea02/user02”
         initial 500 segments;
```

［説明］

1. ユーザ用 RD エリア（USER01）の定義

   USER01 を公用 RD エリア（PUBLIC）とします。USER01 を管理するバックエンドサーバの名称（b001）を指定します。HiRDB ファイルシステム領域に user01 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 500 とします。

   RD エリアの自動増分機能を使用します。増分量を 50 セグメントとします。

2. ユーザ用 RD エリア（USER02）の定義

   USER02 を公用 RD エリア（PUBLIC）とします。USER02 を管理するバックエンドサーバの名称（b001）を指定します。HiRDB ファイルシステム領域に user02 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 500 とします。

   RD エリアの自動増分機能を使用します。増分量を 50 セグメントとします。

(2) データベース構成変更ユティリティ（pdmod）の実行

コマンドの入力例

```
pdmod -a /usr/hirdb/pdmod01
```

［説明］

-a：（1）で作成した制御文ファイルのファイル名を指定します。

3.8　ユーザ LOB 用 RD エリアの作成

実行者　HiRDB 管理者

LOB 属性のデータを使用する場合，LOB 属性のデータを格納するユーザ LOB 用 RD エリアが必要になります。ユーザ LOB 用 RD エリアは，**データベース構成変更ユティリティ（pdmod）**の create rdarea 文で作成します。

3.8.1　基本事項

1. 「HiRDB ファイルシステム領域の作成」で作成した RD エリア用の HiRDB ファイルシステム領域に，ユーザ LOB 用 RD エリアを作成します。
2. HiRDB/パラレルサーバの場合，バックエンドサーバを定義したサーバマシンの HiRDB ファイルシステム領域にユーザ LOB 用 RD エリアを作成します。
3. ユーザ LOB 用 RD エリアの設計方法については，HiRDB/シングルサーバの場合は「RD エリアの配置」を，HiRDB/パラレルサーバの場合は「RD エリアの配置」を参照してください。
4. HiRDB/パラレルサーバの場合，システムマネジャを定義したサーバマシンでデータベース構成変更ユティリティ（pdmod）を実行してください。
5. ユーザ LOB 用 RD エリアを作成する前に，HiRDB が稼働しているかどうかを pdls コマンドで確認してください。HiRDB/パラレルサーバの場合は，システムマネジャを定義したサーバマシンから pdls コマンドを入力してください。
6. HiRDB が稼働していない場合，pdstart コマンドで HiRDB を開始してください。HiRDB/パラレルサーバを開始する場合は，システムマネジャを定義したサーバマシンから pdstart コマンドを入力してください。

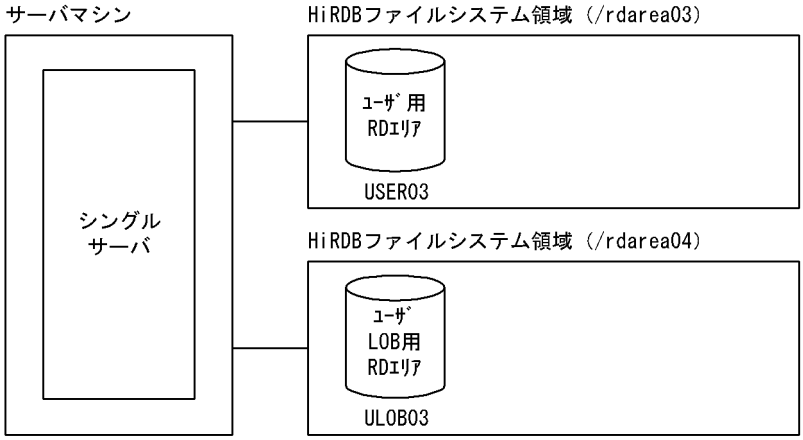
3.8.2　例題 1（HiRDB/シングルサーバの場合）

次に示す RD エリア用の HiRDB ファイルシステム領域に，ユーザ用 RD エリアを作成します。このユーザ用 RD エリアには，LOB 列構成基表を格納します。

- /rdarea03

さらに，次に示す RD エリア用の HiRDB ファイルシステム領域にユーザ LOB 用 RD エリアを作成します。このユーザ LOB 用 RD エリアには，LOB 属性のデータを格納します。

- /rdarea04



(1) 制御文ファイルの作成

データベース構成変更ユティリティ（pdmod）の引数に指定する制御文ファイルを作成します。制御文ファイルを作成する場所は任意ですが，ここでは次に示すファイル名で作成することにします。

- /usr/hirdb/pdmod02

制御文ファイルの内容

```
create rdarea USER03 for user used by PUBLIC                    1
    extension use 50 segments
    file name ”/rdarea03/user03”
         initial 500 segments;
create rdarea ULOB03 for LOB used by PUBLIC                     2
    extension use 50 segments
    file name ”/rdarea04/ulob03”
         initial 20000 segments;
```

〔説明〕

1. ユーザ用 RD エリア（USER03）の定義

   USER03 を公用 RD エリア（PUBLIC）とします。HiRDB ファイルシステム領域に user03 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 500 とします。

   RD エリアの自動増分機能を使用します。増分量を 50 セグメントとします。

2. ユーザ LOB 用 RD エリア（ULOB03）の定義

   ULOB03 を公用 RD エリア（PUBLIC）とします。HiRDB ファイルシステム領域に ulob03 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 20000 とします。

   RD エリアの自動増分機能を使用します。増分量を 50 セグメントとします。

(2) データベース構成変更ユティリティ（pdmod）の実行

コマンドの入力例

```
pdmod -a /usr/hirdb/pdmod02
```

〔説明〕

-a：(1) で作成した制御文ファイルのファイル名を指定します。

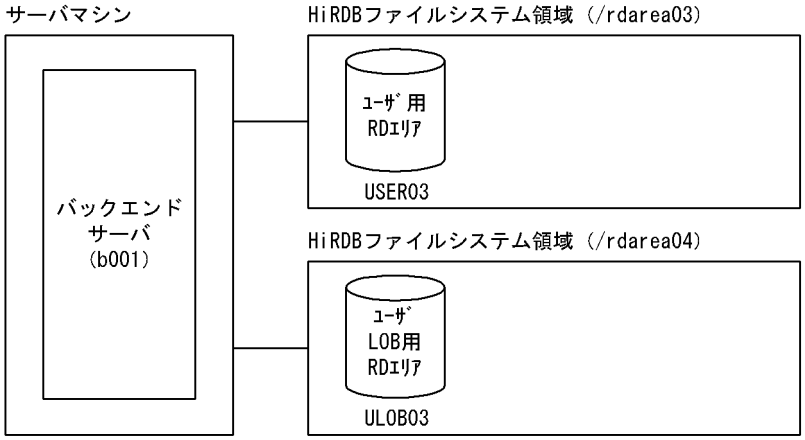
3.8.3 例題 2（HiRDB/パラレルサーバの場合）

次に示す RD エリア用の HiRDB ファイルシステム領域にユーザ用 RD エリアを作成します。このユーザ用 RD エリアには，LOB 列構成基表を格納します。

- /rdarea03

さらに，次に示す RD エリア用の HiRDB ファイルシステム領域にユーザ LOB 用 RD エリアを作成します。このユーザ LOB 用 RD エリアには，LOB 属性のデータを格納します。

- /rdarea04



(1) 制御文ファイルの作成

データベース構成変更ユティリティ（pdmod）の引数に指定する制御文ファイルを作成します。制御文ファイルを作成する場所は任意ですが，ここでは次に示すファイル名で作成することにします。

- /usr/hirdb/pdmod02

制御文ファイルの内容

```
create rdarea USER03 for user used by PUBLIC                1
    server name b001
    extension use 50 segments
    file name "/rdarea03/user03"
         initial 500 segments;
create rdarea ULOB03 for LOB used by PUBLIC                 2
    server name b001
    extension use 50 segments
    file name "/rdarea04/ulob03"
         initial 20000 segments;
```

〔説明〕

1. ユーザ用 RD エリア（USER03）の定義

   USER03 を公用 RD エリア（PUBLIC）とします。USER03 を管理するバックエンドサーバの名称（b001）を指定します。HiRDB ファイルシステム領域に user03 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 500 とします。

   RD エリアの自動増分機能を使用します。増分量を 50 セグメントとします。

2. ユーザ LOB 用 RD エリア（ULOB03）の定義です。

   ULOB03 を公用 RD エリア（PUBLIC）とします。ULOB03 を管理するバックエンドサーバの名称（b001）を指定します。HiRDB ファイルシステム領域に ulob03 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 20000 とします。

   RD エリアの自動増分機能を使用します。増分量を 50 セグメントとします。

(2) データベース構成変更ユティリティ（pdmod）の実行

コマンドの入力例

```
pdmod -a /usr/hirdb/pdmod02
```

〔説明〕

-a：（1）で作成した制御文ファイルのファイル名を指定します。

3.9　データディクショナリ LOB 用 RD エリアの作成

実行者　HiRDB 管理者

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，データディクショナリ LOB 用 RD エリアが必要になります。データディクショナリ LOB 用 RD エリアは，**データベース構成変更ユティリティ**（pdmod）の create rdarea 文で作成します。

データディクショナリ LOB 用 RD エリアは，次に示す用途ごとに用意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定義ソース格納用の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の SQL オブジェクト格納用のデータディクショナリ LOB 用 RD エリア

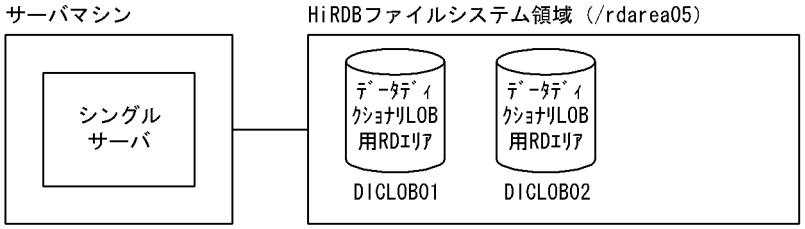
3.9.1　基本事項

1. 「HiRDB ファイルシステム領域の作成」で作成した RD エリア用の HiRDB ファイルシステム領域に，データディクショナリ LOB 用 RD エリアを作成します。
2. HiRDB/パラレルサーバの場合，ディクショナリサーバを定義したサーバマシンの HiRDB ファイルシステム領域にデータディクショナリ LOB 用 RD エリアを作成します。
3. データディクショナリ LOB 用 RD エリアの設計方法については，HiRDB/シングルサーバの場合は「RD エリアの配置」を，HiRDB/パラレルサーバの場合は「RD エリアの配置」を参照してください。
4. HiRDB/パラレルサーバの場合，システムマネジャを定義したサーバマシンでデータベース構成変更ユティリティ（pdmod）を実行してください。
5. データディクショナリ LOB 用 RD エリアを作成する前に，HiRDB が稼働しているかどうかを pdls コマンドで確認してください。HiRDB/パラレルサーバの場合は，システムマネジャを定義したサーバマシンから pdls コマンドを入力してください。
6. HiRDB が稼働していない場合，pdstart コマンドで HiRDB を開始してください。HiRDB/パラレルサーバを開始する場合は，システムマネジャを定義したサーバマシンから pdstart コマンドを入力してください。

3.9.2　例題 1（HiRDB/シングルサーバの場合）

次に示す RD エリア用の HiRDB ファイルシステム領域に，データディクショナリ LOB 用 RD エリアを作成します。

- /rdarea05



(1) 制御文ファイルの作成

データベース構成変更ユティリティ（pdmod）の引数に指定する制御文ファイルを作成します。制御文ファイルを作成する場所は任意ですが，ここでは次に示すファイル名で作成することにします。

- /usr/hirdb/pdmod03

制御文ファイルの内容

```
create rdarea DICLOB01 for LOB used by HiRDB(SQL_ROUTINES)    1
    extension use 1000 segments
    file name ”/rdarea05/diclob01”
        initial 10000 segments;
create rdarea DICLOB02 for LOB used by HiRDB(SQL_ROUTINES)    2
    extension use 1000 segments
    file name ”/rdarea05/diclob02”
        initial 10000 segments;
```

［説明］

1. データディクショナリ LOB 用 RD エリア（DICLOB01）の定義

   DICLOB01 は，定義ソース格納用のデータディクショナリ LOB 用 RD エリアになります。最初に定義したデータディクショナリ LOB 用 RD エリアが定義ソース格納用になります。HiRDB ファイルシステム領域に diclob01 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 10000 とします。

   RD エリアの自動増分機能を使用します。増分量を 1000 セグメントとします。

2. データディクショナリ LOB 用 RD エリア（DICLOB02）の定義

   DICLOB02 は，SQL オブジェクト格納用のデータディクショナリ LOB 用 RD エリアになります。2 番目に定義したデータディクショナリ LOB 用 RD エリアが SQL オブジェクト格納用になります。HiRDB ファイルシステム領域に diclob02 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 10000 とします。

   RD エリアの自動増分機能を使用します。増分量を 1000 セグメントとします。

(2) データベース構成変更ユティリティ（pdmod）の実行

コマンドの入力例

```
pdmod -a /usr/hirdb/pdmod03
```

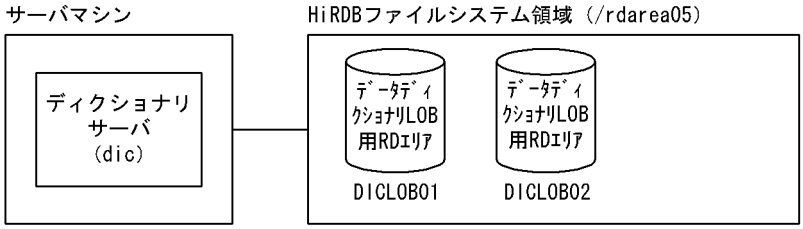
［説明］

-a：（1）で作成した制御文ファイルのファイル名を指定します。

3.9.3 例題 2（HiRDB/パラレルサーバの場合）

次に示す RD エリア用の HiRDB ファイルシステム領域に，データディクショナリ LOB 用 RD エリアを作成します。

- /rdarea05



(1) 制御文ファイルの作成

データベース構成変更ユティリティ（pdmod）の引数に指定する制御文ファイルを作成します。制御文ファイルを作成する場所は任意ですが，ここでは次に示すファイル名で作成することにします。

- /usr/hirdb/pdmod03

制御文ファイルの内容

```
create rdarea DICLOB01 for LOB used by HiRDB(SQL_ROUTINES)   1
    server name dic
    extension use 1000 segments
    file name ”/rdarea05/diclob01”
        initial 10000 segments;
create rdarea DICLOB02 for LOB used by HiRDB(SQL_ROUTINES)   2
    server name dic
    extension use 1000 segments
    file name ”/rdarea05/diclob02”
        initial 10000 segments;
```

［説明］

1. データディクショナリ LOB 用 RD エリア（DICLOB01）の定義

   DICLOB01 は，定義ソース格納用のデータディクショナリ LOB 用 RD エリアになります。最初に定義したデータディクショナリ LOB 用 RD エリアが定義ソース格納用になります。データディクショナリ LOB 用 RD エリアを管理するディクショナリサーバの名称（dic）を指定します。HiRDB ファイルシステム領域に diclob01 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 10000 とします。

   RD エリアの自動増分機能を使用します。増分量を 1000 セグメントとします。

2. データディクショナリ LOB 用 RD エリア（DICLOB02）の定義

DICLOB02 は，SQL オブジェクト格納用のデータディクショナリ LOB 用 RD エリアになります。2 番目に定義したデータディクショナリ LOB 用 RD エリアが SQL オブジェクト格納用になります。データディクショナリ LOB 用 RD エリアを管理するディクショナリサーバの名称（dic）を指定します。HiRDB ファイルシステム領域に diclob02 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 10000 とします。

RD エリアの自動増分機能を使用します。増分量を 1000 セグメントとします。

(2) データベース構成変更ユティリティ（pdmod）の実行

コマンドの入力例

```
pdmod -a /usr/hirdb/pdmod03
```

〔説明〕

-a：(1）で作成した制御文ファイルのファイル名を指定します。

3.10　リスト用 RD エリアの作成

実行者　HiRDB 管理者

絞込み検索をする場合は，リスト用 RD エリアが必要になります。リスト用 RD エリアは，**データベース構成変更ユティリティ（pdmod）**の create rdarea 文で作成します。

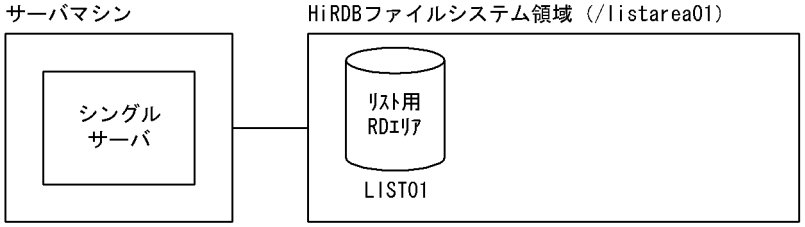
3.10.1　基本事項

1. 「HiRDB ファイルシステム領域の作成」で作成したリスト用 RD エリア用の HiRDB ファイルシステム領域に，リスト用 RD エリアを作成します。
2. HiRDB/パラレルサーバの場合，バックエンドサーバ（基表があるバックエンドサーバ）を定義したサーバマシンの HiRDB ファイルシステム領域にリスト用 RD エリアを作成します。
3. リスト用 RD エリアの設計方法については，HiRDB/シングルサーバの場合は「RD エリアの配置」を，HiRDB/パラレルサーバの場合は「RD エリアの配置」を参照してください。
4. HiRDB/パラレルサーバの場合，システムマネジャを定義したサーバマシンでデータベース構成変更ユティリティ（pdmod）を実行してください。
5. リスト用 RD エリアを作成する前に，HiRDB が稼働しているかどうかを pdls コマンドで確認してください。HiRDB/パラレルサーバの場合は，システムマネジャを定義したサーバマシンから pdls コマンドを入力してください。
6. HiRDB が稼働していない場合，pdstart コマンドで HiRDB を開始してください。HiRDB/パラレルサーバを開始する場合は，システムマネジャを定義したサーバマシンから pdstart コマンドを入力してください。

3.10.2　例題 1（HiRDB/シングルサーバの場合）

次に示すリスト用 RD エリア用の HiRDB ファイルシステム領域に，リスト用 RD エリアを作成します。

- /listarea01



(1) 制御文ファイルの作成

データベース構成変更ユティリティ（pdmod）の引数に指定する制御文ファイルを作成します。制御文ファイルを作成する場所は任意ですが，ここでは次に示すファイル名で作成することにします。

- /usr/hirdb/pdmod04

制御文ファイルの内容

```
create rdarea LIST01 for list                                    1
    page 4096 characters storage control segment 2 pages
    file name ”/listarea01/list01”
        initial 1000 segments;
```

〔説明〕

1. リスト用 RD エリア（LIST01）の定義

   RD エリアのページ長及びセグメントサイズを指定します。HiRDB ファイルシステム領域に list01 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 1000 とします。

(2) データベース構成変更ユティリティ（pdmod）の実行

コマンドの入力例

```
pdmod -a /usr/hirdb/pdmod04
```

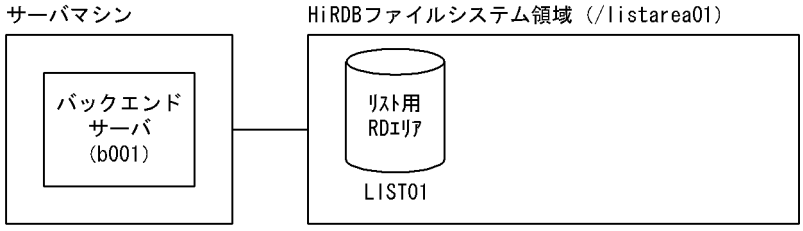
〔説明〕

-a：(1）で作成した制御文ファイルのファイル名を指定します。

3.10.3 例題 2（HiRDB/パラレルサーバの場合）

次に示すリスト用 RD エリア用の HiRDB ファイルシステム領域に，リスト用 RD エリアを作成します。

- /listarea01



(1) 制御文ファイルの作成

データベース構成変更ユティリティ（pdmod）の引数に指定する制御文ファイルを作成します。制御文ファイルを作成する場所は任意ですが，ここでは次に示すファイル名で作成することにします。

- /usr/hirdb/pdmod04

制御文ファイルの内容

```
create rdarea LIST01 for list                                          1
    server name b001
    page 4096 characters storage control segment 2 pages
    file name ”/listarea01/list01”
        initial 1000 segments;
```

〔説明〕

1. リスト用 RD エリア（LIST01）の定義

   LIST01 を管理するバックエンドサーバの名称（b001）を指定します。

   RD エリアのページ長及びセグメントサイズを指定します。HiRDB ファイルシステム領域に list01 という HiRDB ファイルを作成します。HiRDB ファイル内のセグメント数を 1000 とします。

(2) データベース構成変更ユティリティ（pdmod）の実行

コマンドの入力例

```
pdmod -a /usr/hirdb/pdmod04
```

〔説明〕

-a：(1) で作成した制御文ファイルのファイル名を指定します。

4 プラグインの環境設定

プラグインの環境設定は，HiRDB の環境設定の後で実施します。この章では，プラグインの環境設定方法，バージョンアップ方法及び削除（アンインストール）方法について説明します。

4.1　プラグインの環境設定の概要

HiRDB のプラグインの環境を設定する方法について説明します。

4.1.1　環境設定手順

実行者　HiRDB 管理者

ここでは，コマンドを使用したプラグインの環境設定方法について説明します。ここでの説明は，HiRDB の環境設定が終了している（HiRDB が既に稼働している）ことを前提として説明します。

プラグインの環境設定手順を次に示します。

〈手順〉

1. プラグインを組み込むのに必要なリソースの見積もり
2. 稼働中の HiRDB の終了
3. プラグインのインストール
4. プラグインのセットアップ
5. HiRDB の開始
6. データディクショナリ LOB 用 RD エリア，ユーザ用 RD エリア及びユーザ LOB 用 RD エリアの追加※1
7. プラグインの登録
8. レジストリ機能の初期設定※2
9. HiRDB の終了
10. pdplugin オペランドの追加
11. HiRDB の開始
12. レジストリ情報の登録

注※1

データディクショナリ LOB 用 RD エリアは，既にストアドファンクション，ストアドプロシジャ又はプラグインを使用している場合は不要です。ユーザ用 RD エリア（ユーザ LOB 用 RD エリア）は，新規追加したプラグイン用に表を作成した場合に必要です。

注※2

使用するプラグインによっては不要な場合があります。

(1)　リソースの見積もり

プラグインを HiRDB に組み込む前に次に示すリソースを見積もる必要があります。

- プラグインを実行するために必要なメモリ所要量
- プラグインをインストールするときのディスク容量

プラグインを組み込むときに必要なリソースの見積もり方法については，該当するプラグインのマニュアルを参照してください。

(2) HiRDB の終了

プラグインをセットアップする前に，pdstop コマンドで現在稼働している HiRDB を終了してください。

(3) プラグインのインストール

プラグインをインストールします。インストール方法については，該当するプラグインのマニュアルを参照してください。

(4) プラグインのセットアップ

pdplgset コマンドでプラグインをセットアップしてください。

HiRDB/パラレルサーバの場合，各サーバマシンでプラグインのセットアップをしてください。プラグインのセットアップ手順を次に示します。

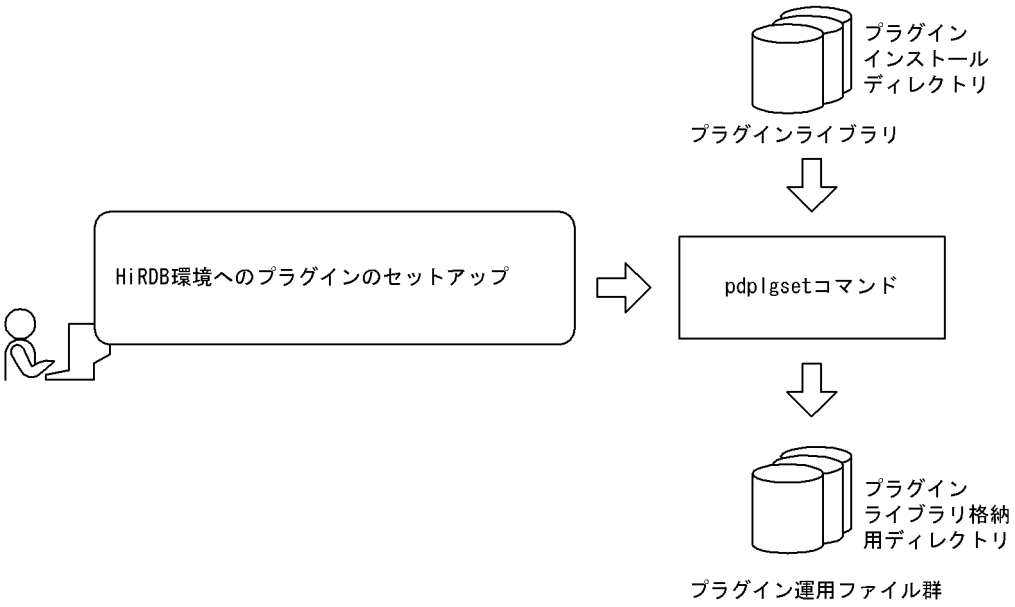
1. HiRDB が稼働中でないかどうかを pdls コマンドで確認してください。稼働中の場合，pdstop コマンドで HiRDB を正常終了させてください。
2. pdplgset コマンドを実行します。

pdplgset コマンドを実行すると，プラグインをインストールしたディレクトリからプラグインライブラリを下記の場所にコピーします。

- $PDDIR/plugin/

プラグインのセットアップの流れを次の図に示します。

図 4-1　プラグインのセットアップの流れ



〔説明〕

HiRDB 運用ディレクトリ（環境変数 PDDIR）下の plugin ディレクトリ及びプラグイン名のディレクトリは，pdplgset コマンドで自動的に作成されます。

(5)　HiRDB の開始

pdstart コマンドで HiRDB を開始します。

(6)　RD エリアの追加

HiRDB にプラグインを登録する前に，データベース構成変更ユティリティ（pdmod）の create rdarea 文で必要な RD エリアを追加してください。追加する必要がある RD エリアを次に示します。

- ユーザ用 RD エリア※1
- ユーザ LOB 用 RD エリア※1
- データディクショナリ LOB 用 RD エリア※2（既にストアドプロシジャ，ストアドファンクション，プラグインを使用している場合は不要です）

RD エリアの追加方法については，「ユーザ用 RD エリアの作成」，「ユーザ LOB 用 RD エリアの作成」又は「データディクショナリ LOB 用 RD エリアの作成」を参照してください。

なお，既にデータベース環境を構築している場合には，プラグインインストール後の RD エリアの追加は不要です。

注※1

プラグイン用に別の表を作成し，その表を格納するための RD エリアを新しく用意したい場合に必要です。

注※2

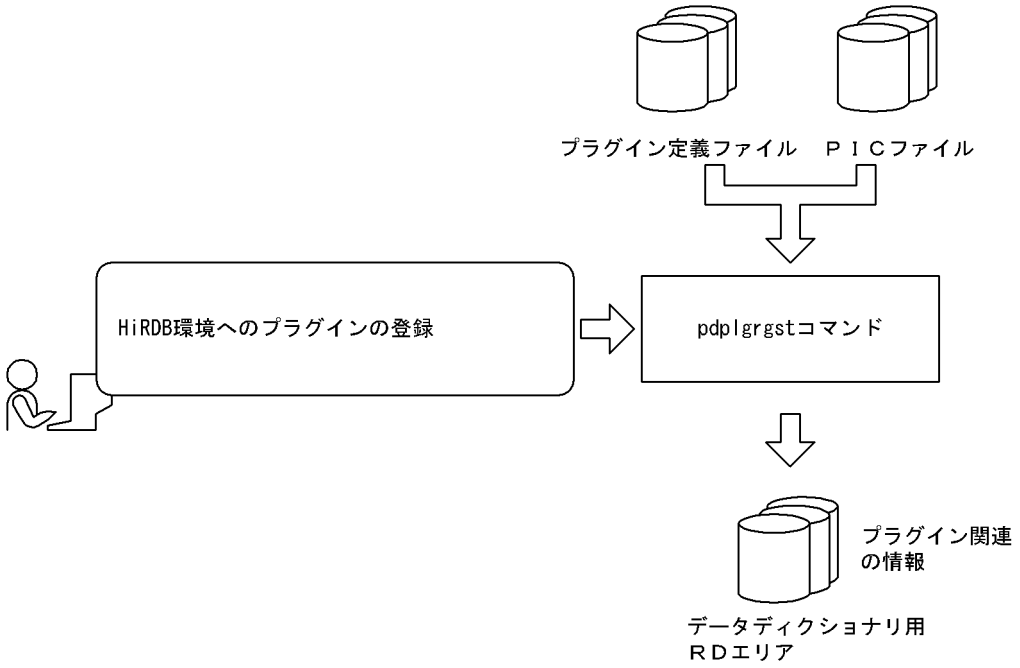
レジストリ機能初期設定ユティリティ（pdreginit）を実行する場合に，あらかじめ HiRDB がストアドプロシジャ機能を使用できるようにしておく必要があるため，この RD エリアが必要です。

(7) プラグインの登録

pdplgrgst コマンドでプラグインを HiRDB に登録してください。pdplgrgst コマンドは，任意のサーバマシンから入力してください。

プラグインの登録の流れを次の図に示します。

図 4-2 プラグインの登録の流れ



(a) pdplgrgst コマンドの入力形式

pdplgrgst コマンドの入力形式を次に示します。

pdplgrgst　プラグイン定義ファイル名　PIC ファイル名

HiRDB Text Search Plug-in の場合の指定例

- データ型プラグインの場合：
  pdplgrgst _phsgml.adt _phsgml.pic
  （カレントディレクトリが/TSPlugin/_phsgml/etc の場合）
- インデクス型プラグインの場合：
  pdplgrgst _phngram.idx _phngram.pic
  （カレントディレクトリが/TSPlugin/_phngram/etc の場合）

注意事項

- インデクス型プラグインを登録する場合，あらかじめ対応するデータ型プラグインが登録されていなければなりません。
- データ型プラグインとインデクス型プラグインは，必ず同じスキーマ内に登録してください。

(b) プラグインの所有者

プラグインの所有者（プラグインが提供する抽象データ型，インデクス型及び関数の所有者）は MASTER になります。したがって，プラグインが提供する関数を呼び出す処理を SQL 文に記述する場合，認可識別子を省略できます。

MASTER 以外にする場合

所有者を MASTER ではなく，pdplgrgst コマンドの実行者にできます。pdplgrgst コマンドに-u オプションを指定すると，プラグインの所有者は pdplgrgst コマンドの実行者（クライアント環境定義の PDUSER オペランドに指定した認可識別子）になります。ただし，この場合，次に示す注意があります。

注意事項

1. pdplgrgst コマンド実行者のスキーマが既に定義されている必要があります。
2. プラグインが抽象データ型及びインデクス型の両方を提供している場合は，必ず同じ所有者にしてください。
3. プラグインの削除又はバージョンアップは，プラグインの所有者しかできません。また，pdplgrgst コマンドに-u オプションを指定してプラグインを削除又はバージョンアップしてください。
4. プラグイン所有者のスキーマを削除すると，プラグインも同時に削除されます。この場合，次に示す作業が必要です。

   ・システム共通定義から pdplugin オペランドを削除します。

   ・すべてのサーバマシンで pdplgset -d コマンドを実行してプラグインをアンセットアップします。
5. 複数のプラグインで，関数名及びパラメタ数が同じである関数を提供している場合，一方のプラグインを登録し，そのプラグインが提供する関数を呼び出す関数をユーザが定義した後，もう一方のプラグインを登録します。このとき，ユーザが定義した関数のパラメタ，又は戻り値のデータ型に抽象データ型を使用し，かつ，その関数をビュー定義に使用している場合は，プラグインの登録時にエラーとなります。この場合，その関数を使用したビュー表を削除した後，再度プラグインの登録を行う必要があります。

(c) HiRDB Version 5.0（HiRDB Object Option 付）01-00 を使用していた場合

プラグインの所有者は変わりません

HiRDB Version 5.0（HiRDB Object Option 付）01-00 では，プラグインの所有者は pdplgrgst コマンドの実行者です。HiRDB Version 6 06-00 以降にバージョンアップしても，プラグインの所有者は変わりません（MASTER になりません）。

所有者を MASTER にする方法

所有者を MASTER にするには，いったんプラグインを削除して再登録する必要があります。

なお，プラグインを削除するには，プラグインが提供している抽象データ型，インデクス型及び関数を使用している表やビュー表，インデクスを事前に削除する必要があります。その後，pdplgrgst コマンドを実行します。

また，既存の SQL 文でプラグインが提供する関数を呼び出している箇所に認可識別子を記述している場合，それを削除するか MASTER に書き換える必要があります。

(8) レジストリ機能の初期設定

プラグインによっては，レジストリ機能が必要な場合があります。その場合，レジストリ機能初期設定ユティリティ（pdreginit）の create rdarea 文で次に示す RD エリアを作成してください。ただし，既にプラグインでレジストリ機能を使用している場合は不要です。

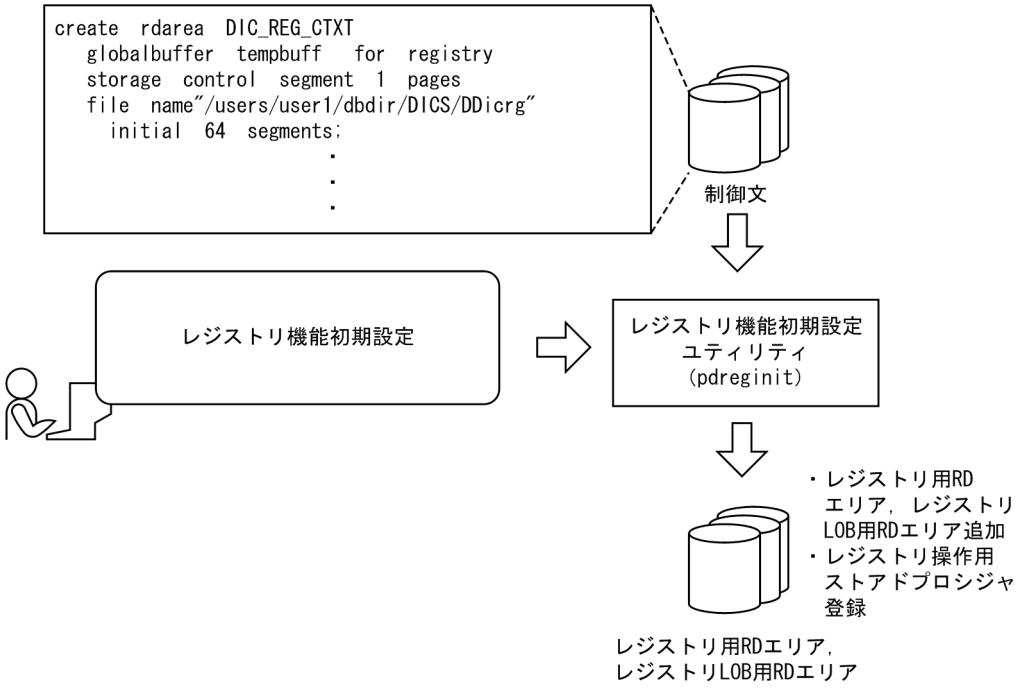
- レジストリ用 RD エリア
- レジストリ LOB 用 RD エリア

なお，レジストリ機能初期設定ユティリティ（pdreginit）は，すべてのプラグインを登録するまでの間で一度だけ実行します。

また，レジストリ用 RD エリア及びレジストリ LOB 用 RD エリアには，レジストリ情報が格納されます。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によって自動的に決定されます。

レジストリ用 RD エリア，レジストリ LOB 用 RD エリアの作成の手順を次の図に示します。

図 4-3 レジストリ用 RD エリア，レジストリ LOB 用 RD エリアの作成の手順



(9) HiRDB の終了

プラグインを使用できる状態にするため，いったん pdstop コマンドで HiRDB を正常終了させます。再度開始するまでは，登録したプラグインを使用した表の定義やインデクスの定義などは実行できません。

HiRDB 終了後，更新した RD エリアのバックアップを必ず取得してください。

(10) pdplugin オペランドの追加

HiRDB が正常終了したら，システム共通定義に pdplugin オペランドを指定してください。pdplugin オペランドには，使用するプラグインの名称を指定します。

HiRDB/パラレルサーバの場合，すべてのサーバマシン上のシステム共通定義に pdplugin オペランドを追加してください。追加漏れがあると HiRDB を開始できません。

(11) HiRDB の開始

pdstart コマンドで HiRDB を開始します。

(12) レジストリ情報の登録

レジストリ機能の初期設定の完了後，プラグインの必要に応じてレジストリ情報を登録します。登録すると，プラグイン及びレジストリ機能を使用できる状態になります。レジストリ情報の登録については，該当するプラグインのマニュアルを参照してください。

4.1.2 プラグイン使用時の注意

(1) HiRDB の設定／開始時の状態とプラグインの使用可否

HiRDB（ユニット）の設定／開始時の状態とプラグインの使用可否を次の表に示します。

表 4-1 HiRDB（ユニット）の設定／開始時の状態とプラグインの使用可否

プラグインの利用宣言 (pdplugin オペランド)	プラグインの登録 (pdplgrgst)	プラグインの初期化エラー	ユニット開始可否	プラグイン使用可否
なし	登録済み	なし	○	×
		あり	○	×
	未登録	なし	○	×
		あり	○	×
あり	登録済み	なし	○	○
		あり	○	×

プラグインの利用宣言(pdplugin オペランド)	プラグインの登録(pdplgrgst)	プラグインの初期化エラー	ユニット開始可否	プラグイン使用可否
	未登録	なし	○	×
		あり	○	×

(凡例)

○：ユニットを開始できます。及びプラグインを利用できます。

×：ユニットを開始できません。及びプラグインを利用できません。

(2) プラグイン初期化エラーが発生したとき

プラグインの初期化は，HiRDB 開始時に自動的に実行されます。システム共通定義に複数の pdplugin オペランドが記述され，一部のプラグインの初期化でエラーが発生した場合，ユニット内のすべてのプラグインが利用できません。

(3) ユニット間でプラグインの使用可否が異なる場合

次に示す場合，ユニット間で利用できるプラグインが異なります。

- ユニット間でシステム共通定義に記述されたプラグイン利用宣言（pdplugin オペランド）が異なる場合
- プラグイン初期化処理時にエラーが発生し，そのユニットでプラグインが利用できなくなった場合

プラグインの呼び出しを伴う SQL 文で利用できるプラグインだけ呼び出した場合は成功しますが，一つでも利用できないプラグインが呼び出された場合は，SQL 文の実行は失敗します。

4.2 プラグインのバージョンアップ

4.2.1 バージョンアップの手順

HiRDB に導入済みのプラグイン（データ型プラグイン，インデクス型プラグイン）をバージョンアップする手順について説明します。プラグインのバージョンアップとは，次に示す内容を削除しないでプラグインのバージョンを上げることです。

- プラグインの提供するデータ型を使用した表やビュー表
- プラグインの提供するインデクス型を使用したインデクス
- プラグインの提供する関数

なお，プラグインをバージョンアップする場合の注意事項については，マニュアル「HiRDB コマンドリファレンス」の「pdplgrgst（プラグインの登録・削除)」の「注意事項」を参照してください。

バージョンアップの手順を次に示します。

(1) バックアップの取得

障害時に備えて，**データベース複写ユティリティ（pdcopy）**で次に示す RD エリアのバックアップを取得します。なお，データベース複写ユティリティ（pdcopy）に-M x オプションを指定してください。

- 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

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。

(2) HiRDB の終了

pdstop コマンドで HiRDB を正常終了させます。

(3) 必要なファイルの退避

次に示す場所にあるファイルのうち，必要なものを退避します。

- $PDDIR/plugin/プラグイン名 のディレクトリ

どのファイルを退避するかについては，該当するプラグインのマニュアルを参照してください。HiRDB がセットアップされたすべてのサーバマシンでファイルを退避してください。

(4) 旧バージョンのプラグインのアンセットアップ

pdplgset -d コマンドで，プラグインをアンセットアップします。このコマンドを実行すると，下記の場所にある内容がすべて削除されます。

- $PDDIR/plugin/プラグイン名 のディレクトリ

ただし，conf ディレクトリ以下は削除されません。

HiRDB がセットアップされたすべてのサーバマシンでプラグインをアンセットアップしてください。プラグインのアンセットアップについては，「プラグインの環境設定の概要」を参照してください。

(5) 新バージョンのプラグインのインストール

HiRDB の各サーバマシンで，新しいバージョンのプラグインをインストールします。インストール方法については，各プラグイン提供のマニュアルを参照してください。

(6) 新バージョンのプラグインのセットアップ

pdplgset コマンドを実行して，新しいプラグインをセットアップします。HiRDB がセットアップされたすべてのサーバマシンでプラグインをセットアップしてください。プラグインのセットアップについては，「プラグインの環境設定の概要」を参照してください。

(7) システム共通定義の変更

システム共通定義から，バージョンアップ対象のプラグインの pdplugin オペランドを削除します。システム共通定義ファイルがあるすべてのサーバマシンから削除してください。

(8) 退避したファイルの復旧

HiRDB がセットアップされたすべてのサーバマシンで，(3) で退避したファイルを復旧してください。

(9) HiRDB の開始

pdstart コマンドで HiRDB を開始します。

(10) 新バージョンのプラグインの登録

pdplgrgst コマンドに-a オプションを指定して実行し，プラグインを更新登録します。

(11) HiRDB の終了

pdstop コマンドで HiRDB を正常終了させます。

(12) システム共通定義の変更

HiRDB が正常終了したら，システム共通定義に pdplugin オペランドを指定してください。pdplugin オペランドには，バージョンアップしたプラグインの名称を指定します。

HiRDB/パラレルサーバの場合，すべてのサーバマシン上のシステム共通定義に pdplugin オペランドを追加してください。追加漏れがあると HiRDB を開始できません。

(13) HiRDB の開始

pdstart コマンドで HiRDB を開始します。開始すると，バージョンアップ前に定義していた表，インデクスを使用できるようになります。さらに，バージョンアップ後のプラグインが新しい機能を提供する場合，その機能を利用できます。

4.3　プラグインの削除

4.3.1　プラグインを削除する手順

ここでは，HiRDB に登録されたプラグインを削除する方法と手順について説明します。なお，プラグインの削除とは，次に示す内容を削除することです。

- ディクショナリに登録されたプラグインの定義情報
- プラグインが提供する関数，抽象データ型及びインデクス型

ただし，プラグインファイルセットを含むプラグイン提供のファイルは削除しません。プラグインを削除する手順を次に示します。

(1)　プラグインが提供する機能を使用したデータベース資源の削除

プラグインを削除する前に，次に示すデータベース資源を削除する必要があります。これらを削除するために発行する SQL 文を次の表に示します。

- 削除するプラグインの提供する抽象データ型を利用した表，ビュー表，関数，手続き，抽象データ型（ユーザが定義した抽象データ型の中で，プラグインが提供する抽象データ型を一つの属性として指定した場合）
- 削除するプラグインの提供するインデクス型を利用したインデクス
- 削除するプラグインの提供する関数を利用した関数，手続き

表 4-2　データベース資源を削除するための SQL

削除する対象となるデータベース資源	削除するための SQL
表	DROP TABLE
インデクス	DROP INDEX
ビュー表	DROP VIEW
関数	DROP FUNCTION
手続き	DROP PROCEDURE
抽象データ型	DROP DATA TYPE ※

注※　プラグインが提供するデータ型を削除してはいけません。

(2)　登録したプラグインの削除

削除する個々のプラグインについて次に示すコマンドを実行します。

pdplgrgst -d　プラグイン定義ファイル名　PIC ファイル名

注意事項

1. データ型プラグインを削除する場合で，そのデータ型のインデクス機能を提供するインデクス型プラグインも登録されているときは，インデクス型プラグインを先に削除する必要があります。
2. プラグインの所有者が MASTER 以外の場合，プラグインを削除するときに次に示すことに注意してください。

- プラグインの削除はプラグインの所有者だけができます。クライアント環境定義の PDUSER オペランドにプラグインの所有者の認可識別子とパスワードを指定してください。そして，pdplgrgst コマンドを実行するときに-u オプションを指定してください。
- HiRDB Version 5.0（HiRDB Object Option 付）01-00 で登録したプラグインの所有者は MASTER ではありません。したがって，前記の作業が必要になります。
- プラグイン所有者のスキーマを削除した場合，プラグインも同時に削除されます。プラグインが削除された後の処理については，(3) 以降を参照してください。

(3) レジストリの削除

レジストリに登録した情報を削除する方法については，各プラグインで提供しているマニュアルを参照してください。

(4) HiRDB の終了

pdstop コマンドで HiRDB を正常終了させます。

(5) システム共通定義の変更

HiRDB が正常終了したら，システム共通定義から pdplugin オペランドを削除します。

(6) プラグインのアンセットアップ

セットアップ済みのプラグインを pdplgset -d コマンドで，アンセットアップします。必要なファイルがある場合，このコマンドを実行する前に退避しておく必要があります。ただし，conf ディレクトリ以下は削除されません。HiRDB がセットアップされたすべてのユニットでプラグインをアンセットアップしてください。

(7) プラグインのアンインストール

プラグインをサーバマシンからアンインストールします。アンインストール方法については，該当するプラグインでの手順に従ってください。

5 データベースの作成

この章では，スキーマ，表，インデクスを作成し，データを格納するまでの方法について説明します。

5.1　データベース作成の概要

ここでは，データベース（表及びインデクス）を作成する前に理解しておくことについて説明します。ここで説明する項目を次に示します。

- データベースの作成前に必要な作業
- データベースの作成手順
- データベースの更新ログ取得方式
- ユニークインデクスを定義した表にデータロードする場合の注意
- 大量のデータをロードする場合（同期点指定のデータロード）
- 横分割表にデータをロードする場合（パラレルローディング機能）
- 横分割表にデータをロードする場合（分割入力データファイルの作成）
- 自動採番機能を使用したデータロード
- 入力データファイル UOC
- 不要な RD エリアの確認

5.1.1　データベースを作成する前に必要な作業

実行者　HiRDB 管理者

データベースを作成する前に必要な作業について説明します。

(1)　クライアント環境定義の設定

次に示すクライアント環境定義を設定してください。クライアント環境定義の設定方法については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

- PDHOST
- PDUSER
- PDNAMEPORT

(2)　パスワードの変更

HiRDB 管理者用の認可識別子のパスワードが，認可識別子と同じ文字列になっている場合，定義系 SQL の GRANT でパスワードを変更してください。データベース定義ユティリティ（pddef）又は HiRDB SQL Executer で次に示す SQL を実行します。

```
GRANT DBA TO HiRDB管理者用の認可識別子 IDENTIFIED BY 新しいパスワード;
```

(3) スキーマの定義

定義系 SQL の CREATE SCHEMA でスキーマを定義します。CREATE SCHEMA は，データベース定義ユティリティ（pddef）又は HiRDB SQL Executer で実行してください。なお，1 ユーザに対して 1 個のスキーマを定義します。

(4) HiRDB 管理者以外がデータベースを作成する場合

HiRDB 管理者以外がデータベースを作成する場合，ユーザ権限を与える作業が必要になります。定義系 SQL の GRANT で，データベースを作成するユーザに必要なユーザ権限を与えてください。必要なユーザ権限を次に示します。

- CONNECT 権限
- スキーマ定義権限
- RD エリア利用権限

ユーザ権限については，マニュアル「HiRDB システム運用ガイド」を参照してください。

（例）

表を作成するユーザ（認可識別子：USER002，パスワード：HIRDB002）に，CONNECT 権限，スキーマ定義権限，RD エリア利用権限（RD エリア名：RDAREA01）を与えます。

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

(5) データの変換方式の設定

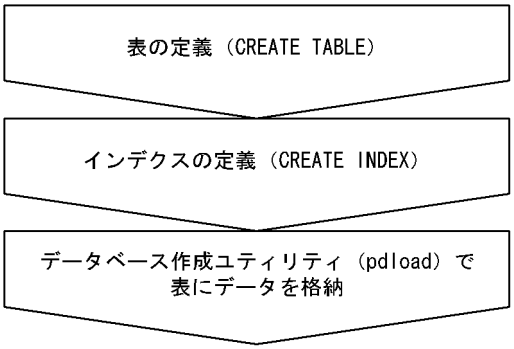
データをデータベースに格納するとき，データの形式を変換する機能があります。次に示す機能を適用するかどうかを検討してください。これらの機能については，マニュアル「HiRDB システム運用ガイド」を参照してください。

- 空白変換機能
- DECIMAL 型の符号正規化機能

5.1.2 データベースの作成手順

データベースの作成手順を次の図に示します。

図 5-1　データベースの作成手順



注

CREATE TABLE 及び CREATE INDEX は次に示すどちらかの方法で実行します。

- データベース定義ユティリティ（pddef）
- HiRDB SQL Executer

5.1.3　データベースの更新ログ取得方式

データベース作成ユティリティ（pdload）で表にデータを格納するときに，データベースの更新ログ取得方式を指定します。データベースの更新ログ取得方式は，データベース作成ユティリティ（pdload）の-l オプションで指定します。

(1)　データベースの更新ログ取得方式の種類

データベースの更新ログ取得方式には，次に示す 3 種類のモードがあります。

- **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。データ件数が少ない場合に使用します。

- **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。データ件数が多いときに使用します。

- **ログレスモード**

  データベース更新ログを取得しません。そのため，データロードの処理時間が最も掛かりません。一つの RD エリアに一つの表だけ（分割格納している場合は一つの横分割表）を格納している場合で，関連するインデクスも一つの RD エリア内にあるときに使用します。

これらのモードの機能詳細については，マニュアル「HiRDB システム運用ガイド」を参照してください。

(2) データの格納先がユーザ LOB 用 RD エリアの場合

データの格納先がユーザ LOB 用 RD エリアの場合，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。

ユーザ LOB 用 RD エリアのデータベースの更新ログ取得方式（CREATE　TABLE の RECOVERY オペランド指定値）は，pdload の-l オプションの指定値によって変わることがあります。pdload の-l オプションの指定値によって変わるユーザ LOB 用 RD エリアのデータベースの更新ログ取得方式を次の表に示します。

表 5-1　pdload の-l オプションの指定値によって変わるユーザ LOB 用 RD エリアのデータベースの更新ログ取得方式

pdload の -l オプションの指定値	CREATE TABLE の RECOVERY オペランドの指定値		
	ALL	PARTIAL	NO
a（ALL 指定相当）	ALL	PARTIAL	NO
p（PARTIAL 指定相当）	PARTIAL※	PARTIAL※	NO
n（NO 指定相当）	NO	NO	NO

（凡例）

ALL：ログ取得モード

PARTIAL：更新前ログ取得モード

NO：ログレスモード

例えば，CREATE TABLE の RECOVERY オペランドに PARTIAL を指定し，pdload の-l オプションに n を指定した場合，NO（ログレスモード）がユーザ LOB 用 RD エリアに設定されます。

注※

プラグインが出力するログの場合は，ALL（ログ取得モード）が仮定されます。

(3) モードの選択基準

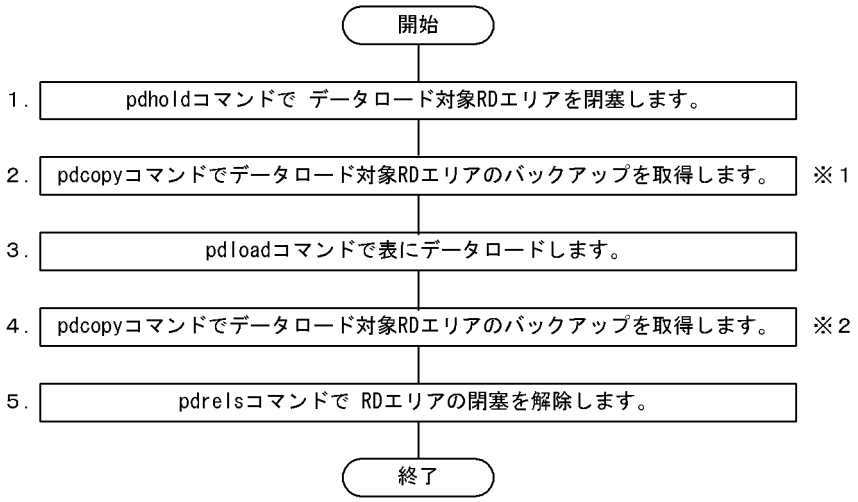
基本的には省略値である更新前ログ取得モードを選択してください。ただし，次に示す条件を満たすような場合はほかのモードの選択を検討してください。

条件	選択するモード
RD エリアにデータロード対象表（インデクス）だけを格納していて，かつ初期ロードである	ログレスモード
入力データ量が多く，データロードに時間が掛かる	
入力データ量が少ない	ログ取得モード

(4) 運用方法の違い

選択したモードによってデータロードするときの運用が異なります。運用方法の違いを次の図に示します。

図 5-2 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（データロード）



注※ 1

ログレスモードを選択したときに必要な操作です。ログレスモードの pdload コマンドが異常終了した場合，このバックアップを使用して RD エリアを回復します。ただし，「データロードの前にバックアップを取得しないでよい場合」で説明している条件を満たすときはバックアップを取得する必要はありません。

ただし，更新ログ取得方式に関係なく，プラグインインデクスを定義した表にインデクス一括作成モードで追加データロードをする場合はバックアップを取得してください。理由は，pdload コマンドが異常終了した場合のデータベース回復には既存データ部分を含め，全プラグインインデクスの再作成が必要となり，データベース回復に長時間必要となるためです。

注※ 2

更新前ログ取得モード又はログレスモードを選択したときに必要な操作です。ここでバックアップを取得しないと，pdrstr コマンドで RD エリアを回復する必要が生じた場合，RD エリアを最新の状態に回復できません（データロード実行後の反映処理を回復できません）。RD エリアはデータロード実行前の状態にしか回復できません。

補足事項

更新前ログ取得モード又はログレスモードを選択した場合，前記の手順 1～4 の間はデータロード対象 RD エリアを閉塞したままにしてください。手順 4 でバックアップを取得する前に RD エリアの内容が更新された場合，pdrstr コマンドで RD エリアを回復する必要が生じたときにその更新内容を回復できません。RD エリアはデータロード実行前の状態にしか回復できません。pdrstr コマンドで RD エリアを回復するとき，入力情報のシステムログ中に更新前ログ取得モード又はログレスモードで取得したログが入っていると pdrstr コマンドがエラーになります。

(5) データロードの前にバックアップを取得しないでよい場合

ログレスモードでデータロードを実行する場合は，データロードの実行前にバックアップを取得する必要があります。ただし，次表に示す 1，2 の条件のうちどちらかを満たす場合は，pdload コマンドが異常終了したときに RD エリアの状態をデータロード実行前の状態に戻すことができるため，バックアップの取得を省略できます。

項番	条件		障害発生時の RD エリア回復方法
1	初期ロードの場合 作成モードでデータロードを実行する場合	データロード対象 RD エリアにはデータロード対象の表，及びその表のインデクスだけを格納している場合	データロード対象 RD エリアをデータベース構成変更ユティリティ（pdmod コマンド）で再初期化した後に，再度データロードすると回復できます。
2	データロード対象 RD エリアにデータロード対象表（インデクス）以外の表（インデクス）がある場合 追加モードでデータロードを実行する場合	バックアップとシステムログを使用してデータロード実行前の状態に RD エリアを回復できる場合	pdclose コマンドでデータロード対象 RD エリアをクローズした後に，pdlogswap コマンドでシステムログファイルをスワップして，データベース回復ユティリティ（pdrstr コマンド）にここまでのシステムログを入力すれば回復できます。

注

項番 2 の条件の場合は，バックアップを取得した方が RD エリアを回復するときの運用が簡単なため，基本的にはバックアップを取得することをお勧めします。特に，インデクス一括作成モードの pdload が異常終了した場合は，ログ取得モードであっても更新前ログ取得モードであっても，ロールバックではインデクスは回復されません。pdload が異常終了して，すぐにデータロード前の状態に回復する必要がある場合は必ずバックアップを取得してください。

5.1.4 ユニーク属性のインデクスを定義した表にデータロードする場合の注意

主キーインデクス（PRIMARY インデクス），又は**ユニークインデクス（UNIQUE 指定のインデクス）**を定義した表にデータロードをする場合は，次に示す注意が必要です。

- **入力データファイル中にキー値が重複するデータがある場合，インデクス一括作成モードでデータロードをしないでください。**

インデクス一括作成モードでデータロードをすると，データを表に格納し，インデクスのキー情報をインデクス情報ファイルに出力します。この段階ではキー値の重複チェックはされません。キー値の重複チェックは，その後のインデクスデータの格納段階でチェックされます。キー値が重複していると，インデクスの作成処理はロールバックされますが，データは既に格納されています（コミットされて元に戻りません）。この場合，バックアップを使用して RD エリアを回復する必要があります。

したがって，**キー値の重複データがある入力データファイルを使用してデータロードをする場合は，必ずインデクス更新モードを指定してください。**このモードはデータを格納するごとにインデクスを更新するのでキー値の重複をすぐに検知し，該当するデータはデータベースに格納しません。

インデクス一括作成モード及びインデクス更新モードは，データベース作成ユティリティ（pdload）の-i オプションで指定します。なお，**省略値がインデクス一括作成モードになっているので注意してください。**

5.1.5　大量のデータをロードする場合（同期点指定のデータロード）

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。

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。

注意事項

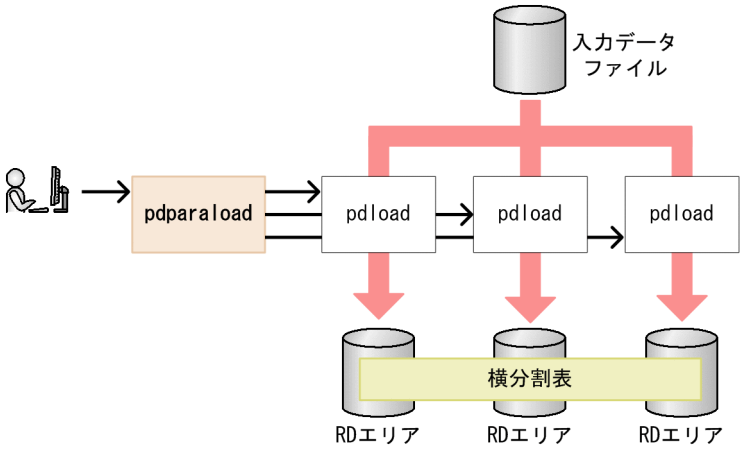
1. この機能を適用しない場合よりも同期点処理が実行される分，処理性能が低下します。
2. ユティリティが異常終了したとき，そのタイミングによって対処方法が異なります。異常終了時の対処方法については，「同期点指定のデータロード実行中にユティリティが異常終了したときの対処方法」を参照してください。特に，インデクス一括作成モードでデータロードをしたときにユティリティが異常終了すると，対処方法が複雑になるため注意してください。
3. 同期点のたびに新しいページからデータの格納を開始するので，この機能を適用しない場合よりも格納ページ数が多く必要になります。

5.1.6　横分割表にデータをロードする場合（パラレルローディング機能）

パラレルローディング機能とは，一つの入力データファイルから横分割表を構成する複数の RD エリアに対してデータロードを並列実行する機能です。pdparaload コマンドを実行することで，複数の RD エリアに対するデータロードを一度に実行できます。

パラレルローディング機能の概要を次の図に示します。

図 5-3　パラレルローディング機能の概要



〔説明〕

ユーザが pdparaload コマンドを実行すると，横分割表を構成する RD エリアの数だけ pdload コマンドが自動的に実行されます。このとき，pdparaload コマンドが pdload コマンドの制御文ファイルを生成します。

各 pdload コマンドが入力データファイルを参照し，該当するデータを抽出して RD エリア内の横分割表に格納します。

(1)　効果

パラレルローディング機能を使うことで得られる効果は，次の二つです。

- **データロードに掛かる処理時間を短縮できます。**

  pdparaload を実行すると複数の pdload コマンドが並列で実行されます。そのため，一つの pdload コマンドで実行する表単位データロードと比べ，データロードに掛かる処理時間が短くなります。

- **一つの入力データファイルと 1 回のコマンド入力で済むため，運用が容易です。**

  分割入力データファイルを使用した RD エリア単位のデータロードの場合でも pdload コマンドの並列実行はできますが，その際，入力データファイルを分割したり，pdload コマンドを複数回入力したりと，運用が煩雑です。パラレルローディング機能を使う場合，入力データファイルを RD エリア単位に分割する必要はありません。また，コマンドの入力も 1 回で済みます。

このように，パラレルローディング機能は，RD エリア単位データロードの高速処理性能と，表単位データロードの運用容易性という両方の長所を併せ持った機能です。

(2)　コマンド実行前に準備しておくこと

パラレルローディングを実行する前に，次の準備をしてください。

(a)　リソース使用量の見積もり

パラレルローディングでは，複数の pdload コマンドを並列実行します。例えば，表を三つの RD エリアに分割して格納する場合，三つの pdload コマンドが同時に実行されるため，リソースも三つ分必要になります。これを考慮して必要なリソース使用量を見積もってください。

(b)　入力データの準備

表に入力するデータを入力データファイルとして準備します。pdparaload コマンドの入力データファイルは一つです。

(3)　運用手順

パラレルローディングの運用手順を次の図に示します。

図 5-4　パラレルローディングの運用手順

注※1

この手順は，データベースの更新ログ取得方式（-l オプション）の指定によって，必要かどうかが異なります。

注※2

この手順は，インデクスの作成方法（-i オプション）に n（インデクス情報出力モード），又は x（インデクス情報出力抑止モード）を指定した場合に必要です。また，c（インデクス一括作成モード）を指定している場合でも，非分割キーインデクスを定義しているときには必要になります。

(4)　制限事項

パラレルローディング機能を使用してデータロードをする場合の制限事項を次に示します。

- フレキシブルハッシュ分割を定義している表はデータロードできません。
- 同期点指定のデータロードはできません。
- パラレルローディングを実行中の表に対して，NOWAIT 検索はできません。
- LOB 列構成基表と LOB データを別々にロードすることはできません。
- EasyMT などテープ装置を媒体とする入力データファイルは使用できません。

pdparaload コマンドのオプションや制御文には，pdload コマンドのオプションや制御文と同様の指定ができますが，一部指定できないものもあります。指定可否の詳細については，マニュアル「HiRDB コマンドリファレンス」の「pdparaload」を参照してください。

(5) 性能に影響を及ぼす条件

パラレルローディング機能の性能は，入力データのデータ配置など，データロードの実行環境に左右されます。そのため，実行環境によっては，パラレルローディング機能を使用してもデータロードに掛かる時間が短縮されないことがあります。したがって，実際に運用を開始する前に，一度パラレルローディング機能を使用して，データロードに掛かる時間を計測してください。その結果，パラレルローディング機能を使用することでデータロードに掛かる時間が短縮された場合に，パラレルローディング機能を使用してください。期待する処理性能が得られなかった場合は，表単位のデータロード，又は入力データファイルを分割して RD エリア単位のデータロードを行ってください。分割入力データファイルの作成については，「横分割表にデータをロードする場合（分割入力データファイルの作成）」を参照してください。

パラレルローディング機能の性能に影響を及ぼす条件について以降で説明します。

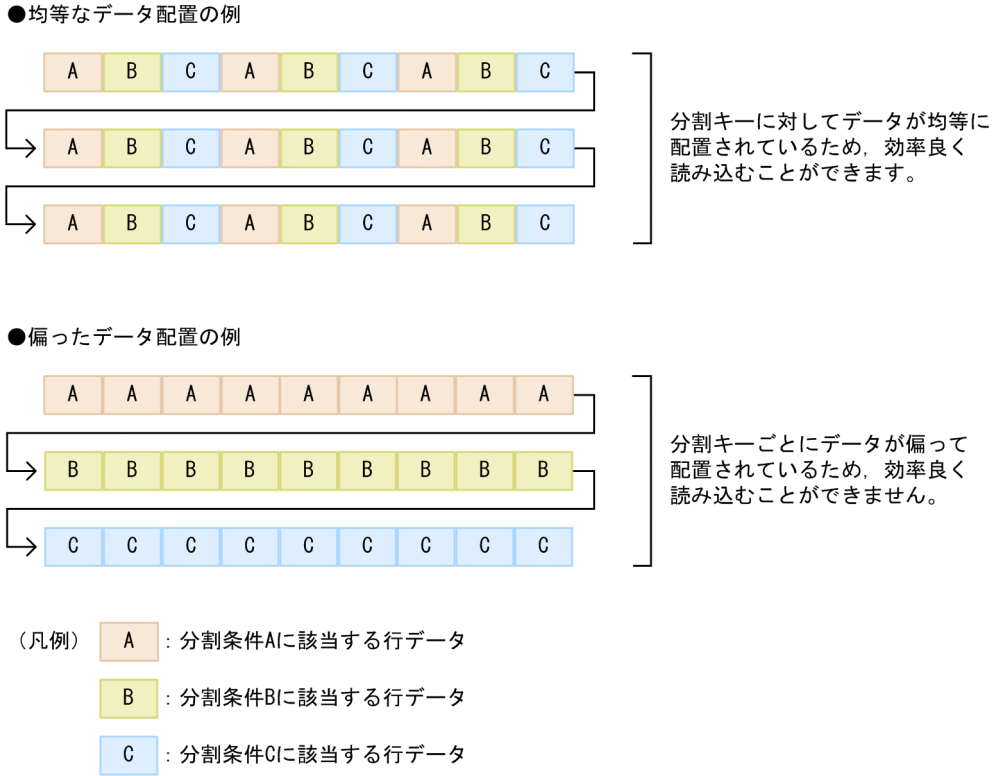
(a) サーバ間横分割

パラレルローディング機能は，サーバ間横分割をしている場合に効果が得られます。サーバ内横分割をしている表にもパラレルローディングは実行できます。しかし，サーバが分かれている方が，データを格納するときに一つの pdload コマンドの処理がサーバを占有できるため，その分早くデータロードできます。

(b) 入力データのデータ配置

パラレルローディングを実行すると，複数の pdload コマンドが一つの入力データファイルからデータを読み込みます。このとき，読み込み処理が効率良くできるようなデータ配置にしておくと，処理時間が短くなります。次の図に示すように，入力データが分割キーに対して均等に配置されていると，効率良く読み込み処理ができます。

図 5-5　入力データのデータ配置と読み込み効率の関係



なお，サーバ内横分割をしている表のデータを pdrorg コマンドで一つのファイルにアンロードした場合，アンロードデータファイルは図の「偏ったデータ配置の例」のようになります。そのため，これを入力データファイルとしてパラレルローディングを実行する場合も性能が低下するおそれがあります。

(c)　入力データのデータ形式

パラレルローディング機能では，pdload コマンドでデータロードできるすべてのデータ形式でデータロードできます。ただし，大量のデータを扱う場合には，入力データのデータ形式をバイナリ形式にすることをお勧めします。DAT 形式や固定長データ形式の場合，データベースにデータを格納する際にデータ形式の変換処理が必要になります。これによって，CPU 使用率が上がり，データロードの並列処理の性能が低下するおそれがあるためです。

(d)　LOB 作成種別

処理対象の表に LOB 列や LOB パラメタを持つ抽象データ型の列を定義している場合は，データロードの LOB 作成種別（-k オプション）に f を指定することをお勧めします。f を指定したデータロードの場合，LOB 入力ファイルとして LOB データごとにファイルを用意します。そのため，データロードを並列実行する際に LOB データを重複して読み込むことがなく，入出力を削減できます。

LOB 作成種別に d を指定する場合，入力データファイル中にすべての LOB データを格納します。そうすると，各 RD エリア単位のデータロードで，処理対象外の LOB データを読み飛ばすという処理が発生するため，性能低下の原因になります。

(6) 運用例

ここでは，パラレルローディングの運用例について説明します。

- 運用例 1 は LOB 列の定義がある場合について説明します。
- 運用例 2 は非分割キーインデクスの定義がある場合について説明します。

運用例のデータロードの条件を次の表に示します。

表 5-2 パラレルローディングの運用例の条件

運用例	表定義		インデクス定義		入力データの形式	インデクス作成方法	ログ取得方式
	横分割	LOB 列	分割キーインデクス	非分割キーインデクス			
運用例 1	○	○	○	×	バイナリ形式	インデクス一括作成モード	更新前ログ取得モード
運用例 2	○	×	○	○	バイナリ形式	インデクス一括作成モード	更新前ログ取得モード

（凡例）

○：定義します。

×：定義しません。

(a) 運用例 1（LOB 列の定義がある場合）

LOB 列を定義した表 2 にパラレルローディング機能を使用してデータロードします。LOB 作成種別（-k オプション）に f を指定して，LOB データごとにファイルを作成します。

●表定義とインデクス定義

表定義

```
create table "表2"(
  col001    int not null,
  col002    varchar(20),
  col003    blob(1M) in ((LOB01),(LOB02),(LOB03)),
  col004    decimal(10,3)
) fix hash hashf by col001
in (RDUSER11,RDUSER21,RDUSER31);
```

インデクス定義

```
create index "インデクス2" on "表2"(col001,col002)
in ((RDUSER12),(RDUSER22),(RDUSER32));
```

●データロードの手順

1. pdhold コマンドでデータロード対象の RD エリアを閉塞します。

```
pdhold -r LOB01,LOB02,LOB03,RDUSER11,RDUSER21,RDUSER31,RDUSER12,RDUSER22,RDUSER32
```

2. pdparaload コマンドの制御文ファイルを作成します。

source：サーバ名，入力データファイル，エラー情報ファイルを指定します。HiRDB/パラレルサーバの場合は，サーバ名を必ず指定してください。

lobdata：LOB 入力ファイルを指定します。

idxwork：インデクス情報ファイルの格納ディレクトリを指定します。

sort：ソート用ワークファイルの格納ディレクトリを指定します。

report：処理結果ファイル名を指定します。

```
source fes01:/users/data/input_file error=/users/rep/error_file
lobdata /users/data/lob
idxwork bes01 /users/work
idxwork bes02 /users/work
idxwork bes03 /users/work
sort bes01 /users/work
sort bes02 /users/work
sort bes03 /users/work
report file=/users/rep/result_file
```

3. pdparaload コマンドでデータロードを行います。

-d：作成モードでデータロードします。

-b：バイナリ形式の入力データファイルを使用します。

-k：LOB 作成種別を指定します。LOB データごとにファイルを作成します（f）。

-i：インデクス一括作成モード（c）を指定します。

-l：更新前ログ取得モード（p）を指定します。

```
pdparaload -d -b -k f -i c -l p ”表2” /users/cntl/control_file
```

4. pdcopy コマンドで RD エリアのバックアップを取得します。

```
pdcopy -m /users/rdarea/mast/mast01 -M r -p /users/pdcopy/list/list01
-b /users/pdcopy/backup/backup01
-r LOB01,LOB02,LOB03,RDUSER11,RDUSER21,RDUSER31,RDUSER12,RDUSER22,RDUSER32
```

5. pdrels コマンドで RD エリアの閉塞を解除します。

```
pdrels -r LOB01,LOB02,LOB03,RDUSER11,RDUSER21,RDUSER31,RDUSER12,RDUSER22,RDUSER32
```

(b) 運用例 2（非分割キーインデクスの定義がある場合）

非分割キーインデクスを定義した表 3 にパラレルローディング機能を使用してデータロードします。非分割キーインデクスを定義しているため，pdparaload コマンド実行後に，pdrorg コマンドでインデクスの一括作成をする必要があります。インデクスの一括作成時には，pdparaload コマンドで出力されたインデクス情報ファイルを指定します。

●表定義とインデクス定義

表定義

```
create table ”表3”(
  col001    int not null,
  col002    varchar(20),
  col003    char(32),
  col004    decimal(10,3)
) partitioned by col001
in ((RDUSER11) 10000000, (RDUSER21) 20000000, (RDUSER31));
```

インデクス定義（分割キーインデクス）

```
create index ”インデクス3” on ”表3”(col001,col002)
in ((RDUSER12),(RDUSER22),(RDUSER32));
```

インデクス定義（非分割キーインデクス）

```
create index ”インデクス4” on ”表3”(col003)
in ((RDUSER13));
```

●データロードの手順

1. pdhold コマンドでデータロード対象の RD エリアを閉塞します。

```
pdhold -r RDUSER11,RDUSER21,RDUSER31,RDUSER12,RDUSER22,RDUSER32,RDUSER13
```

2. pdparaload コマンドの制御文ファイルを作成します。

source：入力データファイル，エラー情報ファイルを指定します。HiRDB/パラレルサーバの場合は，サーバ名を必ず指定してください。

idxwork：インデクス情報ファイルの格納ディレクトリを指定します。データロード後は，ここに作成されたインデクス情報ファイルを使ってインデクスの一括作成をします。

sort：ソート用ワークファイルの格納ディレクトリを指定します。

report：処理結果ファイル名を指定します。

```
source /users/data/input_file error=/users/rep/error_file
idxwork /users/work
sort /users/work
report file=/users/rep/result_file
```

3. pdparaload コマンドでデータロードを行います。

-d：作成モードでデータロードします。

-b：バイナリ形式の入力データファイルを使用します。

-i：インデクス一括作成モード（c）を指定します。

-l：更新前ログ取得モード（p）を指定します。

```
pdparaload -d -b -i c -l p ”表3” /users/cntl/control_file
```

4. pdrorg コマンドの制御文ファイルを作成します。

index 文に，pdparaload コマンドで出力されたインデクス情報ファイルを指定します。

```
index ”インデクス4” RDUSER13 ”/users/work/INDEX-インデクス4-RDUSER13-faaG4MnMf”
index ”インデクス4” RDUSER13 ”/users/work/INDEX-インデクス4-RDUSER13-caa34EnEc”
index ”インデクス4” RDUSER13 ”/users/work/INDEX-インデクス4-RDUSER13-caa06MnMc”
sort /users/work
report file=/users/rep/result_file2
```

5. pdrorg コマンドでインデクスの一括作成をします。

```
pdrorg -k ixmk -l p -t ”表3” /users/cntl/control_rorg
```

6. pdcopy コマンドで RD エリアのバックアップを取得します。

```
pdcopy -m /users/rdarea/mast/mast01 -M r -p /users/pdcopy/list/list01
-b /users/pdcopy/backup/backup01
-r RDUSER11,RDUSER21,RDUSER31,RDUSER12,RDUSER22,RDUSER32,RDUSER13
```

7. pdrels コマンドで RD エリアの閉塞を解除します。

```
pdrels -r RDUSER11,RDUSER21,RDUSER31,RDUSER12,RDUSER22,RDUSER32,RDUSER13
```

(7) コマンドエラーが発生した場合の運用

pdparaload コマンドの実行時に発生するエラーと対処には，大きく分けて次の二つがあります。

- RD エリア単位データロード（pdload コマンド）実行中のエラー

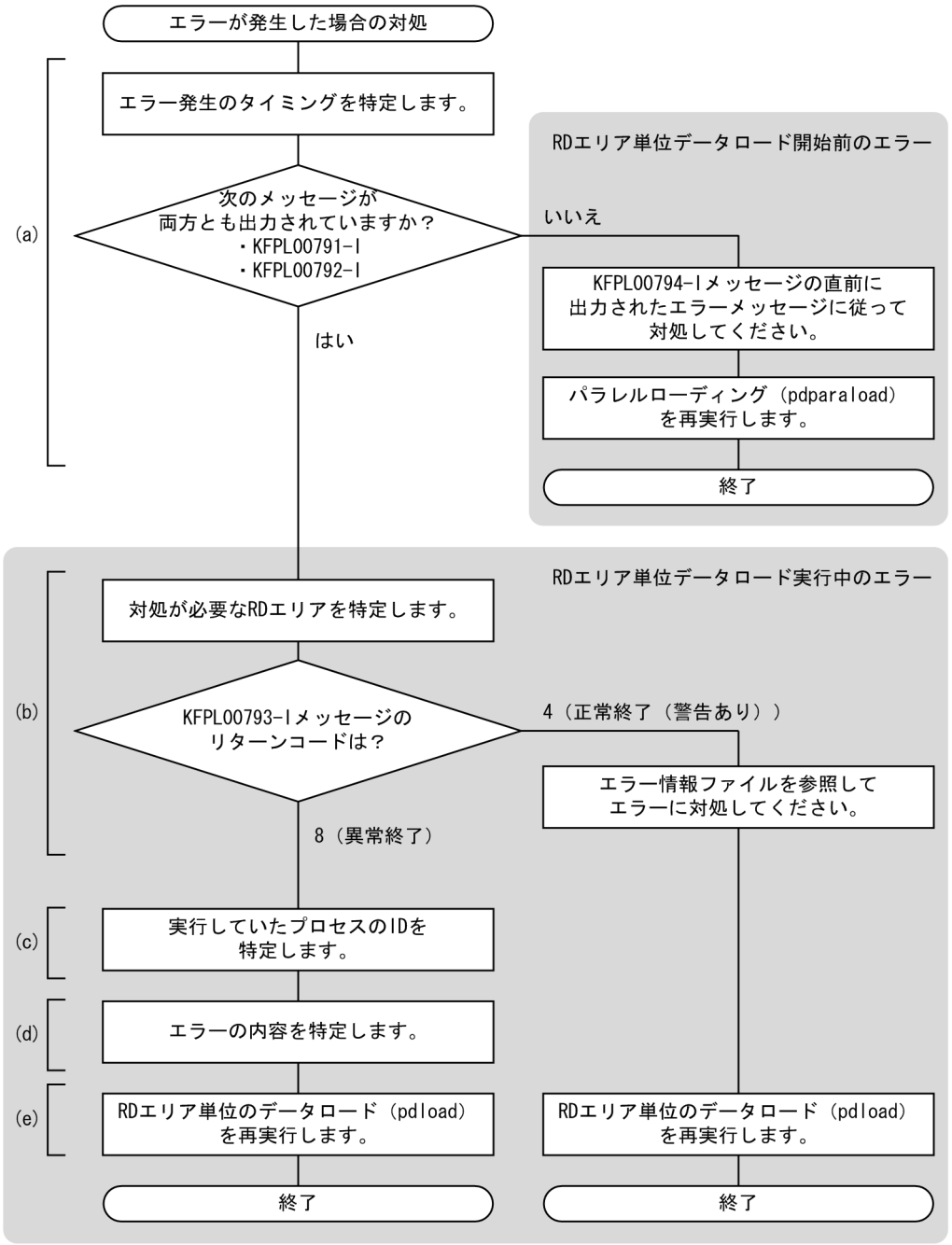
  この場合，一部の RD エリアへのデータロードに失敗しています。エラーになったそれぞれの RD エリアについてエラーの対処をした後，pdload コマンドで RD エリア単位のデータロードを再実行します。

- RD エリア単位データロード（pdload コマンド）開始前のエラー

  この場合，すべての RD エリアへのデータロードに失敗しています。エラーに対処した後，pdparaload コマンドを再実行します。

pdparaload コマンドの実行時に発生するエラーと対処の詳細について説明します。まず最初に，pdparaload コマンドの実行でエラーが発生した場合の対処の流れを次に示します。

図 5-6　pdparaload コマンドの実行でエラーが発生した場合の対処の流れ



注

図中の(a)から(e)は，以降の(a)から(e)と対応しています。

(a)　エラー発生のタイミングを特定します

エラー発生のタイミングを特定します。次の両方のメッセージが出力されているかどうかを確認してください。

- KFPL00791-I メッセージ
- KFPL00792-I メッセージ

≪両方のメッセージが出力されている場合≫

RD エリア単位データロードの実行中にエラーが発生しています。これによって，一部の RD エリアへのデータロードに失敗しています。この場合は，(b)以降の手順に従ってエラーに対処してください。その後，pdload コマンドで RD エリア単位のデータロードを再実行してください。

≪どちらかのメッセージが出力されていない場合≫

RD エリア単位データロードの開始前にエラーが発生しています。つまり，すべての RD エリアへのデータロードに失敗しています。この場合は，KFPL00794-I メッセージの直前に出力されたエラーメッセージに従って対処してください。その後，pdparaload コマンドを再実行してください。

(b) 対処が必要な RD エリアを特定します

pdparaload コマンドを実行した OS コンソール又は syslog ファイルに出力されたメッセージから KFPL00793-I メッセージを抽出し，対処が必要な RD エリアを特定します。メッセージを抽出する際には，次の項目を検索キーにしてください。

- メッセージ ID（KFPL00793-I）
- 認可識別子
- 表識別子

メッセージ抽出例

```
KFPL00793-I Pdload execution abnormal terminated, table=user1."T1", RDAREA=RDAREA1, retur
n code=8, pdload process id=1385412 (1212478)
   :
KFPL00793-I Pdload execution abnormal terminated, table=user1."T1", RDAREA=RDAREA2, retur
n code=8, pdload process id=1385433 (1212478)
```

[説明]

認可識別子（user1）と表識別子（T1）を基に，KFPL00793-I メッセージを抽出します。これによって，データロードに失敗した RD エリア（RDAREA1，RDAREA2）を特定できます。

特定した RD エリアに対して，KFPL00793-I メッセージのリターンコードによって，次の対処をしてください。

≪リターンコードが 4 の場合≫

pdload コマンドが出力したエラー情報ファイルを参照して，エラーに対処してください。エラー情報ファイルは，pdparaload コマンドの source 文の error オプションで指定しています（ファイル名には自動的に RD エリア名称が付加されます）。指定を省略した場合は，次のディレクトリとファイル名称で作成されます。

- エラー情報ファイルの格納ディレクトリ
  1. pd_tmp_directory オペランドで指定したディレクトリ
  2. 1.の指定がない場合，環境変数 TMPDIR で指定したディレクトリ
  3. 2.の指定がない場合，/tmp ディレクトリ
- エラー情報ファイルの名称

(c)で特定するプロセス ID とメッセージ ID（KFPL00793-I）が含まれるファイル名

エラーに対処したら，(e)の手順に従って，RD エリア単位のデータロードを再実行してください。

≪リターンコードが 8 の場合≫

(c)，(d)の手順に従って，エラーの内容を特定し，対処してください。

その後，(e)の手順に従って，RD エリア単位のデータロードを再実行してください。

(c) 実行していたプロセスの ID を特定します

メッセージログファイルから，pdparaload コマンドが実行した pdload 制御プロセスとサーバプロセスの ID を特定します。次の手順でプロセス ID を抽出してください。

1. pdcat コマンドで，pdparaload コマンドの実行開始（KFPL00791-I メッセージ）から終了（KFPL00794-I メッセージ）までの間のメッセージログをファイルに出力します。
2. 1.で出力した内容からプロセスの開始を示す KFPL00711-I メッセージを抽出します。メッセージを抽出する際には，次の項目を検索キーにしてください。
   - メッセージ ID（KFPL00711-I）
   - 認可識別子
   - 表識別子
   - (b)で特定した RD エリア名

メッセージ抽出例

```
1572976 2010/11/04 15:55:48 0mload1 lod KFPL00711-I pdloadm started, table=user1."T1", RD
AREA=RDAREA1
1728690 2010/11/04 15:55:48 bes1    lod KFPL00711-I pdbes started, table=user1."T1", RDAR
EA=RDAREA1
```

[説明]

認可識別子（user1）と表識別子（T1）と RD エリア名（RDAREA1）を基に，KFPL00711-I メッセージを抽出します。これによって，エラーが発生したプロセスの ID（1572976，1728690）を特定できます。

この場合，それぞれの ID は次のプロセスを示しています。

- 1572976 は，プロセス名が pdloadm であるため，pdload 制御プロセスです。
- 1728690 は，プロセス名が pdbes であるため，バックエンドサーバプロセスです。

(d) エラーの内容を特定します

メッセージログファイルから，エラーの内容を特定します。(c)で特定したプロセス ID を検索キーにして，(c)の手順 1.で取得したメッセージログファイルから対処が必要なエラーメッセージを抽出してください。

メッセージ抽出例（pdload 制御プロセス）

```
1572976 2010/11/04 15:55:48 0mload1 lod KFPL00711-I pdloadm started, table=user1.”T1”, RD
AREA=RDAREA1
1572976 2010/11/04 15:55:48 0mload1 lod KFPL00704-I Pdload terminated, return code=8
```

メッセージ抽出例（バックエンドサーバプロセス）

```
1728690 2010/11/04 15:55:48 bes1 lod KFPL00711-I pdbes started, table=user1.”T1”, RDAREA=
RDAREA1
1728690 2010/11/04 15:55:48 bes1 lod KFPL00709-I Error information file was created, file
=/tmp/ERROR-4cd258f41728690
1728690 2010/11/04 15:55:48 bes1 lod KFPL00709-I Lobmid file was created, file=/tmp/LOBMI
D-T1-4cd258f41728690
1728690 2010/11/04 15:55:48 bes1 lod KFPL00702-I Pdload started, table=user1.”T1”, genera
tion=0
1728690 2010/11/04 15:55:48 bes1 lod KFPL00710-I Index information file assigned, index=u
ser1.”T1NX”, RDAREA=”USER01”, file=/tmp/INDEX-T1NX-USER01-GN0-daan_ylid
1728690 2010/11/04 15:55:48 bes1 lod KFPL00723-I 0 rows loaded, table=user1.”T1”, RDAREA=
”USER01”
1728690 2010/11/04 15:55:48 bes1 lod KFPLxxxxx-E YYYYYY
```

［説明］

プロセス ID（1572976，1728690）を基に，メッセージを抽出します。この場合，バックエンドサーバプロセスでエラーメッセージ（KFPLxxxxx-E）が出力されていることが分かります。このエラーメッセージに従って，対処します。

エラーに対処したら，(e)の手順に従って，RD エリア単位のデータロードを再実行してください。

(e)　RD エリア単位のデータロード（pdload コマンド）を再実行します

pdload コマンドで RD エリア単位のデータロードを再実行します。この際，pdparaload コマンドが生成した pdload コマンドの制御文を再利用することをお勧めします。pdparaload コマンドが生成した pdload コマンドの制御文ファイルの格納ディレクトリとファイル名を次に示します。これを基に pdload の制御文ファイルを作成してください。

- pdload 制御文の格納ディレクトリ
  1. pd_tmp_directory オペランドで指定したディレクトリ
  2. 1.の指定がない場合，環境変数 TMPDIR で指定したディレクトリ
  3. 2.の指定がない場合，/tmp ディレクトリ
- pdload 制御文のファイル名

  LOD_CTL_認可識別子_表識別子_RD エリア名称

注意事項

RD エリア単位のデータロードを再実行する際には，pdload の option 文で divermsg=off を指定してください。

RD エリア単位にデータロードする場合，入力データ中に RD エリアの格納条件と一致しない行データがあると，エラーデータ情報を出力し，pdload コマンドはリターンコード 4 で終了します。divermsg=off を指定すると，このエラーデータ情報の出力が抑止され，pdload コマンドをリターンコード 0 で終了できます。

RD エリア単位データロードを再実行する場合の，pdload コマンドの制御文とコマンドラインの例を次に示します。

制御文の例

```
option divermsg=off
source ”RDUSER02” fes01:/users/data/input_file error=/users/rep/error_file_RDUSER02
lobdata /users/data/lob
idxwork bes02 /users/work
sort bes02 /users/work
report file=/users/rep/result_file_RDUSER02
```

[説明]

- option 文に divermsg=off を指定することによって，エラーデータ情報の出力を抑止します。
- RD エリア単位データロードのため，source 文には対象とする RD エリア名称を指定します。

コマンドラインの例

```
pdload -d -b -k f -i c -l p ”表2” ”/users/tmp/LOD_CTL_USER02_表2_RDUSER02”
```

5.1.7 横分割表にデータをロードする場合（分割入力データファイルの作成）

横分割表にデータをロードする場合は，パラレルローディング機能を使う方法もあります。パラレルローディング機能については，「横分割表にデータをロードする場合（パラレルローディング機能）」を参照してください。パラレルローディング機能で期待する効果が得られない場合には，ここで説明する分割入力データファイルを使用したデータロードを適用してください。

横分割表にデータをロードする場合，格納する RD エリア単位に入力データファイルを分割してデータロードを並列実行すると，データロードに掛かる時間を短縮でき，表の占有時間が短縮できます。制御情報ファイルに src_work 文を指定してデータベース作成ユティリティ（pdload）を実行すると，ユーザが作成した入力データファイルから，RD エリア単位の入力データファイルが作成できます。これによって，RD エリア単位にデータロードを実行できます。このとき，データベース作成ユティリティ（pdload）が作成するファイルを**分割入力データファイル**といいます。分割入力データファイルを作成する場合のオプション及び制御文については，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。

5.1.8　自動採番機能を使用したデータロード

順序数生成子を使用すると，自動的に採番ができます。これを自動採番機能といいます。データロード時に，順序数生成子が生成した順序番号を，表の列に格納できます。ここでは，順序番号の取得方式と格納方式の選択基準について説明します。

なお，自動採番機能についてはマニュアル「HiRDB UAP 開発ガイド」を，自動採番機能を使用したデータロードの詳細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。

(1)　順序番号の取得方式の選択基準

順序番号の取得方式には，次の 3 種類があります。

全数一括取得方式：

データロード完了後に一括して順序数生成子の値を使用した順序番号にします。

指定単位取得方式：

指定した単位ごとに順序番号を取得しながらデータロードします。

バッファ単位取得方式：

入力バッファに読み込める行数分の順序番号を取得しながらデータロードします。

順序番号の取得方式は，次の表に示す特徴を考慮して選択してください。

表 5-3　順序番号の取得方式ごとの特徴

検討項目	特徴		
	全数一括取得方式	指定単位取得方式	バッファ単位取得方式
正常時の欠番発生	発生しません。	データロードした行数が指定した単位の倍数でなければ，欠番が発生します。	発生しません。※2
ロールバック時の大量の欠番発生	発生しません。	現在値はロールバックが発生しても回復されないため，大量の欠番が発生します。	現在値はロールバックが発生しても回復されないため，大量の欠番が発生します。
順序数生成子への採番要求時の通信オーバヘッド※1	データロードのコミットの回数分しか採番要求をしないため，採番処理による性能への影響は小さくなります。	取得する単位を大きくすることで，採番要求の回数を抑え，性能への影響を小さくできます。	入力バッファ長を大きくすることで，採番要求の回数を抑え，性能への影響を小さくできます。ただし，1 回に取得する単位は入力バッファ長と列の定義長から算出した行長で決まるため，行長が大きいと入力バッファに多くのメモリを必要とします。
同じ順序数生成子を使用する UAP との同時実行	同時実行はできません。データロード中は順序数生成子に排他が掛かかります。	同時実行できます。	同時実行できます。

検討項目	特徴		
	全数一括取得方式	指定単位取得方式	バッファ単位取得方式
RD エリア単位のデータロードの並列実行	同時実行はできません。データロード中は順序数生成子に排他が掛かかります。	並列実行できます。	並列実行できます。

注※1

HiRDB/パラレルサーバの場合，データベース作成ユティリティが入力データを読み込むサーバと順序数生成子が定義されたサーバが異なるとき，順序番号の取得時に通信が発生します。そのため，採番要求が頻繁に行われると，通信回数が多くなり，データロードの性能に影響します。

注※2

順序番号の格納方式が列データ全置換以外の場合は，欠番が発生する可能性があります。

(2) 順序番号の格納方式の選択基準

順序番号の格納方式には，次の 3 種類があります。それぞれの選択基準について説明します。

列データ全置換：

順序番号を格納する列に対して，入力データファイル中の該当する列データをすべて順序番号に置き換えます。該当する列の値に，すべて新しく番号を振り直す場合に選択します。

列データ一部置換：

順序番号を格納する列に対して，入力データファイル中の該当する列データのうち，指定した置換条件に一致するデータだけ順序番号に置き換えます。例えば，入力データファイルが DAT 形式，又は拡張 DAT 形式で，NULL 値の部分だけ順序番号に置き換える場合などに選択します。

列データ追加：

順序番号を格納する列に対応するデータが入力データファイル中にない場合，順序番号を入力データとして追加します。番号を格納する列を新しく追加する場合に選択します。なお，入力データファイルがバイナリ形式の場合，この方式は指定できません。

5.1.9 入力データファイル UOC

任意に作成したプログラムで，データロードするデータを編集できます。編集されたデータは，直接 pdload に渡されます。そのため，入力ファイルを編集するプログラムで，いったんワークファイルを作成しない形でデータロードができます。

データを編集するためにユーザが作成したプログラムを **UOC（ユーザオウンコーディング）**といいます。UOC は，データベースに格納したいデータファイルが pdload の入力データファイルのフォーマットと異なる場合や，データベースに格納したいデータが HiRDB の文字コードと異なる場合など，入力データの編集が必要なときに使用できます。

5.1.10　不要な RD エリアの削除

データベースを作成後，ディクショナリ表の SQL_RDAREAS 表を検索して，表やインデクスが定義されなかったユーザ用 RD エリア又は LOB 列が定義されなかったユーザ LOB 用 RD エリアがないかどうか確認することをお勧めします。もし不要な RD エリアがあれば，削除できます。不要な RD エリアを削除することで，ディスク所要量を削減できます。

ディクショナリ表の検索方法と SQL_RDAREAS 表についてはマニュアル「HiRDB UAP 開発ガイド」を，RD エリアの削除方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

5.2　横分割表の作成

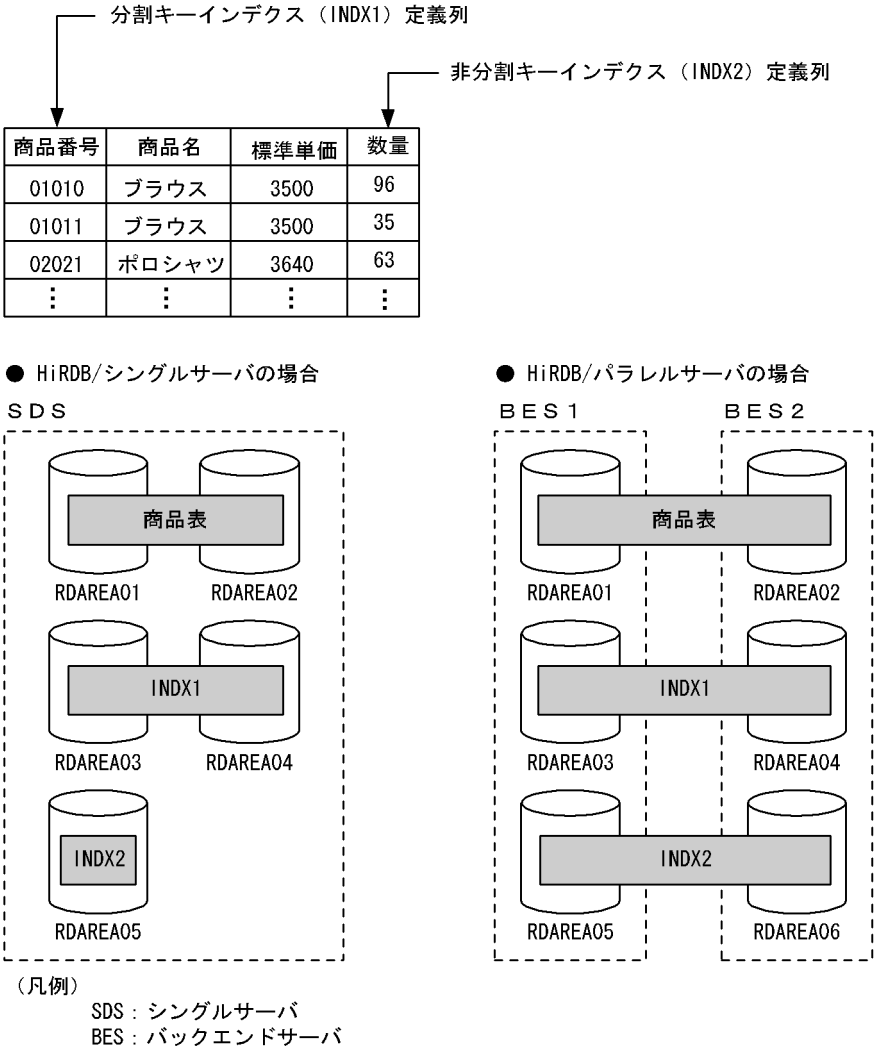
5.2.1　横分割表の作成例

商品表を作成します。商品表の作成条件を次に示します。

- 商品表を横分割します。ユーザ用 RD エリア RDAREA01～RDAREA02 に商品表を格納します。
- 商品表に分割キーインデクス（INDX1）を定義します。ユーザ用 RD エリア RDAREA03～RDAREA04 に INDX1 を格納します。
- 商品表に非分割キーインデクス（INDX2）を定義します。ユーザ用 RD エリア RDAREA05 に INDX2 を格納します。HiRDB/パラレルサーバの場合は，ユーザ用 RD エリア RDAREA05～RDAREA06 に INDX2 を格納します。
- RDAREA01～RDAREA06 にはデータロード対象表（インデクス）だけを格納していて，初期ロードとします。
- データロードをするときにインデクスを一括作成（省略値）します。
- ログレスモードでデータロードをします。

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，「インデクスの横分割」を参照してください。

また，横分割表にデータをロードする場合には，パラレルローディング機能を使う方法もあります。パラレルローディング機能については，「横分割表にデータをロードする場合（パラレルローディング機能）」を参照してください。



(1) 商品表の定義

CREATE TABLE で商品表を定義します。定義例を次に示します。

(a) キーレンジ分割の場合

格納条件指定

```
CREATE TABLE 商品表
(商品番号 CHAR(5) NOT NULL,
 商品名 NCHAR(15),
 標準単価 INTEGER,
 数量 INTEGER
)IN ((RDAREA01)商品番号<='10000',(RDAREA02));
```

境界値指定の場合

```
CREATE TABLE 商品表
(商品番号 CHAR(5) NOT NULL,
 商品名 NCHAR(15),
 標準単価 INTEGER,
 数量 INTEGER
```

```
    )PARTITIONED BY 商品番号
    IN ((RDAREA01)'10000',(RDAREA02));
```

(b) フレキシブルハッシュ分割，FIX ハッシュ分割の場合

```
   CREATE TABLE 商品表
   (商品番号 CHAR(5) NOT NULL,
    商品名 NCHAR(15),
    標準単価 INTEGER,
    数量 INTEGER
    )[FIX]※ HASH HASH6 BY 商品番号
    IN (RDAREA01,RDAREA02);
```

注※　FIX ハッシュ分割の場合に指定します。

(2) インデクスの定義

CREATE INDEX で商品表にインデクスを定義します。定義例を次に示します。

(a) HiRDB/シングルサーバの場合

```
CREATE INDEX INDX1 ON 商品表（商品番号）
    IN ((RDAREA03),(RDAREA04));
CREATE INDEX INDX2 ON 商品表（数量）
    IN (RDAREA05);
```

(b) HiRDB/パラレルサーバの場合

```
CREATE INDEX INDX1 ON 商品表（商品番号）
    IN ((RDAREA03),(RDAREA04));
CREATE INDEX INDX2 ON 商品表（数量）
    IN ((RDAREA05),(RDAREA06));
```

(3) 表へのデータの格納

データベース作成ユティリティ（pdload）で表にデータを格納します。格納手順を次に示します。

〈手順〉

1. pdhold コマンドで，データロード対象 RD エリア（RDAREA01～RDAREA05）を閉塞します。HiRDB/パラレルサーバの場合は RDAREA01～RDAREA06 を閉塞します。
2. pdload コマンドで，入力データファイルを表にデータロードします。RD エリアにはデータロード対象表（インデクス）だけを格納していて，かつ初期ロードのため，データベースの更新ログ取得方式にログレスモードを選択します。また，インデクスの作成方法にインデクス一括作成モード（省略値）を選択します。pdload コマンドに指定するオプションについては，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。

3. ログレスモードで pdload コマンドを実行しているため，データロード対象 RD エリアのバックアップを取得します。RD エリア単位の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。
4. pdrels コマンドで，データロード対象 RD エリアの閉塞を解除します。

上記のコマンドとユティリティの詳細及びこれらのコマンドとユティリティの実行結果の確認方法については，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。

補足事項

- ログレスモードで pdload コマンドを実行するため，前記の手順 1～3 の間はデータロード対象 RD エリアを閉塞したままにしてください。
- 改竄防止表に対して pdload コマンドでデータロードするとき，-d オプションは使用できません。
- インデクス一括作成中にエラーが発生した場合の対処方法については，「インデクス一括作成中に発生したエラーの対処方法」を参照してください。

(4) データの格納状態の確認

データロードをした場合は，運用を開始する前にデータベース状態解析ユティリティ（pddbst）を実行して，データの格納状態を確認することをお勧めします。設計どおりにデータベースを作成できたかどうかを確認できます。データベース状態解析ユティリティ（pddbst）を実行すると，次に示す情報を取得できます。

- ユーザ用 RD エリア単位のデータの格納状態
- 表又はインデクス単位のデータの格納状態

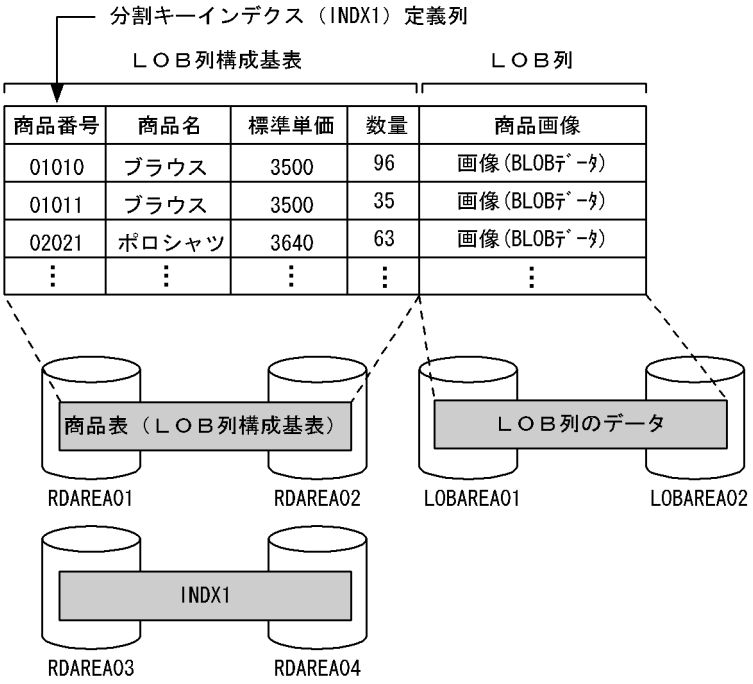
5.3 LOB 列を定義した表の作成

5.3.1 LOB 列を定義した表の作成例

商品表を作成します。商品表の作成条件を次に示します。

- 商品表を横分割します。ユーザ用 RD エリア RDAREA01～RDAREA02 に商品表の LOB 列構成基表を格納します。
- ユーザ LOB 用 RD エリア LOBAREA01～LOBAREA02 に LOB 列のデータを格納します。
- 商品表に分割キーインデクス（INDX1）を定義します。ユーザ用 RD エリア RDAREA03～RDAREA04 に INDX1 を格納します。
- RDAREA01～RDAREA04，LOBAREA01～LOBAREA02 にはデータロード対象表（インデクス）だけを格納していて，初期ロードとします。
- データロードをするときにインデクスを一括作成（省略値）します。
- ログレスモードでデータロードをします。

なお，横分割表にデータをロードする場合には，パラレルローディング機能を使う方法もあります。パラレルローディング機能については，「横分割表にデータをロードする場合（パラレルローディング機能）」を参照してください。



補足事項

- 一つのユーザ LOB 用 RD エリアには，表中の一つの LOB 列だけを格納します。また，一つの表に複数の LOB 列がある場合には，それぞれ別のユーザ LOB 用 RD エリアに格納する必要があります。

- LOB 列が横分割表の場合には，LOB 列ごとにユーザ LOB 用 RD エリアと，表を格納しているユーザ用 RD エリアの数を 1 対 1 に対応させる必要があります。

(1) 商品表の定義

CREATE TABLE で商品表を定義します。定義例を次に示します。

(a) キーレンジ分割の場合

格納条件指定

```
CREATE TABLE 商品表
(商品番号 CHAR(5),
 商品名 NCHAR(15),
 標準単価 INTEGER,
 数量 INTEGER,
 商品画像 BLOB(64K) IN ((LOBAREA01),(LOBAREA02))
)IN ((RDAREA01) 商品番号<='10000',(RDAREA02));
```

境界値指定

```
CREATE TABLE 商品表
(商品番号 CHAR(5),
 商品名 NCHAR(15),
 標準単価 INTEGER,
 数量 INTEGER,
 商品画像 BLOB(64K) IN ((LOBAREA01),(LOBAREA02))
)PARTITIONED BY 商品番号
IN ((RDAREA01)'10000',(RDAREA02));
```

(b) フレキシブルハッシュ分割，FIX ハッシュ分割の場合

```
CREATE TABLE 商品表
(商品番号 CHAR(5),
 商品名 NCHAR(15),
 標準単価 INTEGER,
 数量 INTEGER,
 商品画像 BLOB(6000) IN ((LOBAREA01),(LOBAREA02))
)[FIX]※ HASH HASH6 BY 商品番号
 IN (RDAREA01,RDAREA02);
```

注※　FIX ハッシュ分割の場合に指定します。

(2) インデクスの定義

CREATE INDEX で商品表にインデクスを定義します。定義例を次に示します。

```
CREATE INDEX INDX1 ON 商品表 (商品番号)
    IN ((RDAREA03),(RDAREA04));
```

(3) 表へのデータの格納

データベース作成ユティリティ（pdload）で表にデータを格納します。格納手順を次に示します。

〈手順〉

1. pdhold コマンドで，データロード対象 RD エリア（RDAREA01～RDAREA04，LOBAREA01～LOBAREA02）を閉塞します。
2. pdload コマンドで，入力データファイルを表にデータロードします。RD エリアにはデータロード対象表（インデクス）だけを格納していて，かつ初期ロードのため，データベースの更新ログ取得方式にログレスモードを選択します。また，インデクスの作成方法にインデクス一括作成モード（省略値）を選択します。pdload コマンドに指定するオプションについては，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。
3. ログレスモードで pdload コマンドを実行しているため，データロード対象 RD エリアのバックアップを取得します。RD エリア単位の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。
4. pdrels コマンドで，データロード対象 RD エリアの閉塞を解除します。

上記のコマンドとユティリティの詳細及びこれらのコマンドとユティリティの実行結果の確認方法については，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。

補足事項

- ログレスモードで pdload コマンドを実行するため，前記の手順 1～3 の間はデータロード対象 RD エリアを閉塞したままにしてください。
- 改竄防止表に対して pdload コマンドでデータロードするとき，-d オプションは使用できません。
- インデクス一括作成中にエラーが発生した場合の対処方法については，「インデクス一括作成中に発生したエラーの対処方法」を参照してください。

(4) データの格納状態の確認

データロードをした場合は，運用を開始する前にデータベース状態解析ユティリティ（pddbst）を実行して，データの格納状態を確認することをお勧めします。設計どおりにデータベースを作成できたかどうかを確認できます。データベース状態解析ユティリティ（pddbst）を実行すると，次に示す情報を取得できます。

- ユーザ用 RD エリア又はユーザ LOB 用 RD エリア単位のデータの格納状態
- 表又はインデクス単位のデータの格納状態

(5) 補足事項

LOB 列を定義した表にデータロードする場合，LOB 列構成基表と LOB データを別々にデータロードすることもできます。このときの手順を次に示します。

なお，データベースの更新ログ取得方式にログレスモードを，インデクスの作成方法にインデクス一括作成モード（省略値）を選択したとします。

〈手順〉

1. pdhold コマンドで，データロード対象 RD エリア（RDAREA01～RDAREA04，LOBAREA01～LOBAREA02）を閉塞します。
2. pdload コマンドで，入力データファイルを表（LOB 列構成基表及びインデクス）にデータロードします。このときのデータロード対象 RD エリアは RDAREA01～RDAREA04 になります。このとき，LOB 中間ファイルに LOB 列のデータロード時に必要な情報を出力するようにしてください。pdload コマンドに指定するオプションについては，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。
3. pdload コマンドで，ユーザ LOB 用 RD エリア LOBAREA01～LOBAREA02 にデータロードします。このとき，LOB 入力ファイルと 2.で作成した LOB 中間ファイルを指定します。
4. ログレスモードで pdload コマンドを実行しているため，データロード対象 RD エリアのバックアップを取得します。RD エリア単位の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。
5. pdrels コマンドで，データロード対象 RD エリアの閉塞を解除します。

上記のコマンドとユティリティの詳細及びこれらのコマンドとユティリティの実行結果の確認方法については，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。

5.4　プラグインが提供する抽象データ型を定義した表の作成

ここでは，プラグインが提供する抽象データ型（SGMLTEXT 型，XML 型）を定義した表の作成について説明します。

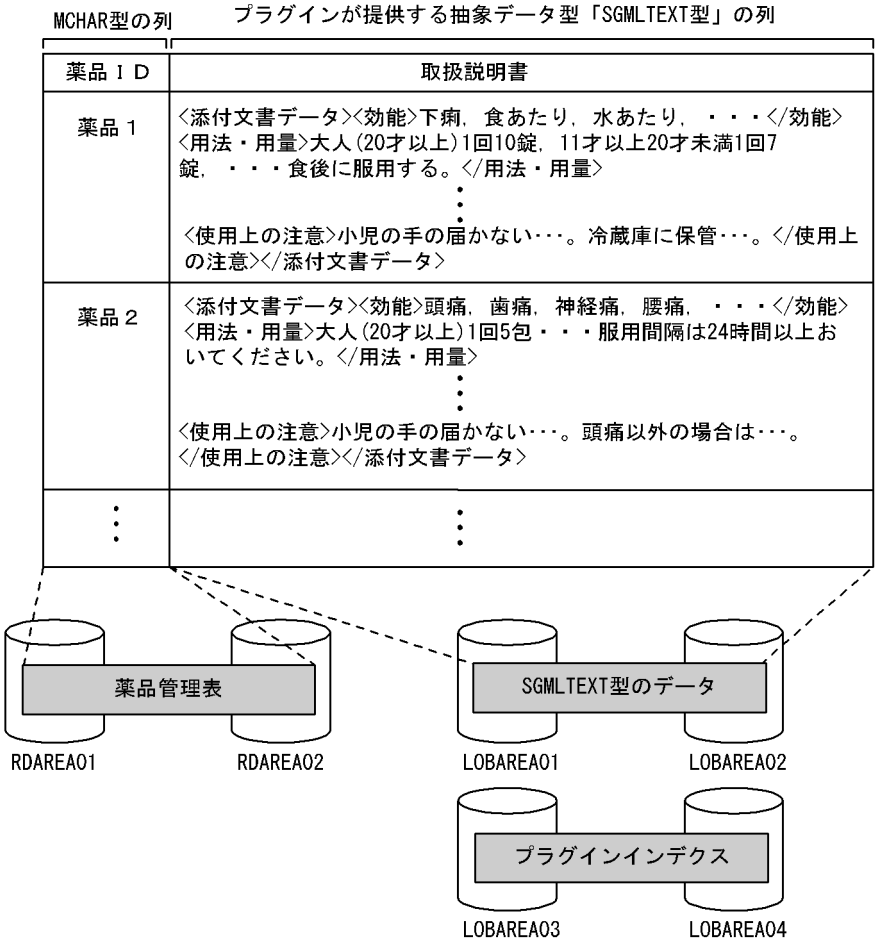
SGMLTEXT 型を使用する場合には HiRDB Text Search Plug-in，XML 型を使用する場合には HiRDB XML Extension が必要です。プラグインの環境設定については，「プラグインの環境設定」を参照してください。なお，プラグインの所有者は MASTER にしてください。

5.4.1　SGMLTEXT 型

ここでは，HiRDB Text Search Plug-in が提供する抽象データ型（SGMLTEXT 型）を定義した表の作成方法について説明します。

ここでは，薬品管理表を作成します。薬品管理表の作成条件を次に示します。

- 薬品管理表を横分割します。ユーザ用 RD エリア RDAREA01～RDAREA02 に薬品管理表の LOB 列構成基表を格納します。
- ユーザ LOB 用 RD エリア LOBAREA01～LOBAREA02 に SGMLTEXT 型の列のデータを格納します。
- ユーザ LOB 用 RD エリア LOBAREA03～LOBAREA04 にプラグインインデクスを格納します。
- RDAREA01～RDAREA02，LOBAREA01～LOBAREA04 にはデータロード対象表（インデクス）だけを格納していて，初期ロードとします。
- データロードをするときにインデクスを一括作成（省略値）します。
- ログレスモードでデータロードをします。



〔説明〕

薬品 ID（MCHAR 型）をユーザ用 RD エリアに格納し，取扱説明書（SGMLTEXT 型）をユーザ LOB 用 RD エリアに格納します。

(1)　薬品管理表の定義

CREATE TABLE で薬品管理表を定義します。定義例を次に示します。

(a)　キーレンジ分割の場合

格納条件指定

```
CREATE TABLE 薬品管理表(
薬品ID MCHAR(15),
取扱説明書 SGMLTEXT          …1
  ALLOCATE(SGMLTEXT IN((LOBAREA01),(LOBAREA02)))  …2
  PLUGIN'<DTD>medicine.dtd</DTD>'        …3
  )IN((RDAREA01)薬品ID<='薬品10',(RDAREA02));     …4
```

境界値指定

```
CREATE TABLE 薬品管理表(
薬品ID MCHAR(15),
```

```
取扱説明書 SGMLTEXT          …1
  ALLOCATE(SGMLTEXT IN((LOBAREA01),(LOBAREA02)))  …2
  PLUGIN'<DTD>medicine.dtd</DTD>'        …3
  )PARTITIONED BY 薬品ID
  IN((RDAREA01)'薬品10',(RDAREA02));              …4
```

〔説明〕

1. プラグインモジュールで提供されたデータ型を指定します。
2. 薬品管理表中の LOB 列（SGMLTEXT）をユーザ LOB 用 RD エリア LOBAREA01 及び LOBAREA02 に分割して格納します。
3. プラグインオプションを指定します。指定方法については，プラグインのマニュアルを参照してください。
4. 薬品管理表の LOB 列構成基表をユーザ用 RD エリア RDAREA01，RDAREA02 に分割して格納します。

(b) フレキシブルハッシュ分割，FIX ハッシュ分割の場合

```
CREATE TABLE 薬品管理表(
薬品ID MCHAR(15),
取扱説明書 SGMLTEXT                …1
  ALLOCATE(SGMLTEXT IN((LOBAREA01),(LOBAREA02)))  …2
  PLUGIN'<DTD>medicine.dtd</DTD>'        …3
  )[FIX]※ HASH HASH6 BY 薬品ID
  IN(RDAREA01,RDAREA02)                  …4
```

注※　FIX ハッシュ分割の場合に指定します。

〔説明〕

1. プラグインモジュールで提供されたデータ型を指定します。
2. 薬品管理表中の LOB 列 SGMLTEXT をユーザ LOB 用 RD エリア LOBAREA01 及び LOBAREA02 に分割して格納します。
3. プラグインオプションを指定します。指定方法については，プラグインのマニュアルを参照してください。
4. 薬品管理表の LOB 列構成基表をユーザ用 RD エリア RDAREA01，RDAREA02 に分割して格納します。

(2) プラグインインデクスの定義

プラグインによって提供されたデータ検索用のインデクス型をインデクスに定義するとデータを高速に検索できます。プラグインから提供されたインデクス型を定義したインデクスを**プラグインインデクス**といいます。ここでは，HiRDB Text Search Plug-in が提供するインデクス型（NGRAM）を使用したプラグインインデクスの定義方法について説明します。

CREATE INDEX で薬品管理表にプラグインインデクスを定義します。定義例を次に示します。

```
CREATE INDEX PLGINDX1
        USING TYPE NGRAM
        ON 薬品管理表(取扱説明書)
        IN ((LOBAREA03),(LOBAREA04));
```

［説明］

プラグインインデクス PLGINDX1 を横分割した薬品管理表に対応させて，ユーザ LOB 用 RD エリア LOBAREA03 及び LOBAREA04 に分割して格納します。なお，プラグインインデクス PLGINDX1 を構成する列に取扱説明書列を指定しています。

(3) 表へのデータの格納

データベース作成ユティリティ（pdload）で表にデータを格納します。格納手順を次に示します。

〈手順〉

1. pdhold コマンドで，データロード対象 RD エリア（RDAREA01～RDAREA02，LOBAREA01～LOBAREA04）を閉塞します。
2. pdload コマンドで，入力データファイルを表にデータロードします。RD エリアにはデータロード対象表（インデクス）だけを格納していて，かつ初期ロードのため，データベースの更新ログ取得方式にログレスモードを選択します。また，インデクスの作成方法にインデクス一括作成モード（省略値）を選択します。また，コンストラクタ関数やコンストラクタ関数に渡すデータ型の情報を列構成情報ファイルに指定します。pdload コマンドに指定するオプションについては，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。
3. ログレスモードで pdload コマンドを実行しているため，データロード対象 RD エリアのバックアップを取得します。RD エリア単位の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。
4. pdrels コマンドで，データロード対象 RD エリアの閉塞を解除します。

上記のコマンドとユティリティの詳細及びこれらのコマンドとユティリティの実行結果の確認方法については，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。

補足事項

- ログレスモードで pdload コマンドを実行するため，前記の手順 1～3 の間はデータロード対象 RD エリアを閉塞したままにしてください。
- 改竄防止表に対して pdload コマンドでデータロードするとき，-d オプションは使用できません。
- インデクス一括作成中にエラーが発生した場合の対処方法については，「インデクス一括作成中に発生したエラーの対処方法」を参照してください。

(4) データの格納状態の確認

データロードをした場合は，運用を開始する前にデータベース状態解析ユティリティ（pddbst）を実行して，データの格納状態を確認することをお勧めします。設計どおりにデータベースを作成できたかどうか

を確認できます。データベース状態解析ユティリティ（pddbst）を実行すると，次に示す情報を取得できます。

- ユーザ用 RD エリア及びユーザ LOB 用 RD エリア単位（物理解析だけ）のデータの格納状態
- レジストリ用 RD エリア及びレジストリ LOB 用 RD エリア単位（物理解析，論理解析）のデータの格納状態

(5) ハッシュ関数で分割条件を指定している表に RD エリア単位でデータロードする場合

表分割ハッシュ関数を使用した UAP を作成し，RD エリア単位に入力データファイルを作成できます。これによって，各 RD エリアに格納されるデータ量が確認できるため，均等に分割できるハッシュ関数を選択できます。表分割ハッシュ関数を使用した 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。

5.4.2 XML 型

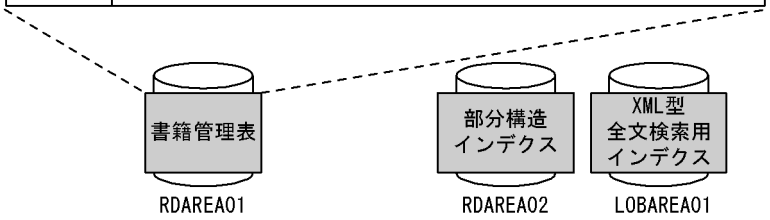
ここでは，HiRDB XML Extension が提供する抽象データ型（**XML 型**）を定義した表の作成方法について説明します。

ここでは，書籍管理表を作成します。書籍管理表の作成条件を次に示します。

- ユーザ用 RD エリア RDAREA01 に書籍管理表を格納します。
- ユーザ用 RD エリア RDAREA02 に部分構造インデクスを格納します。
- ユーザ LOB 用 RD エリア LOBAREA01 に XML 型全文検索用インデクスを格納します。
- RDAREA01，RDAREA02，LOBAREA01 にはデータロード対象表（インデクス）だけを格納していて，初期ロードとします。
- データロードをするときにインデクスを一括作成（省略値）します。
- ログレスモードでデータロードをします。

書籍管理表

書籍ID	書籍情報
452469630	<書籍情報 書籍ID="452469630"> 　<カテゴリ>データベース</カテゴリ> 　<タイトル>リレーショナルデータベース解説</タイトル> 　<説明>リレーショナルモデルの概念から，それに基づくRDBMSの仕組みを解説している。</説明> </書籍情報>
126513592	<書籍情報 書籍ID="126513592"> 　<カテゴリ>データベース</カテゴリ> 　<タイトル>SQL徹底解説</タイトル> 　<説明>最新の標準SQL言語について詳細に解説している。</説明> </書籍情報>
940123531	<書籍情報 書籍ID="940123531"> 　<カテゴリ>ネットワーク</カテゴリ> 　<タイトル>図解TCP/IP</タイトル> 　<説明>TCP/IPプロトコルについて図を用いながら分かりやすく説明している。</説明> </書籍情報>
⋮	⋮



(1)　書籍管理表の定義

CREATE TABLE で書籍管理表を定義します。定義例を次に示します。

XML 型の列を含む表の定義例

```
CREATE TABLE 書籍管理表（書籍ID INTEGER, 書籍情報 XML）IN RDAREA01
```

(2)　インデクスの定義

(a)　部分構造インデクス（B-tree）

XML 型の列には，特定の部分構造をキーとし，その値をキー値としたインデクスを定義できます。このインデクスを利用すると，XMLEXISTS 述語や XMLQUERY 関数の XQuery 式中に，部分構造インデクスを定義した構造に対する述語を指定した場合，行の絞り込みの処理時間を削減できることがあります。

部分構造インデクスを利用できる XQuery 式中の述語を次に示します。

- キーである部分構造に対する比較式（=，!=，>，>=，<，<=，<>，eq，ne，lt，le，gt，ge）
- キーである部分構造に対する fn:contains 関数，fn:starts-with 関数，fn:ends-with 関数

インデクスの使用条件については，「インデクスの使用条件」で説明します。

部分構造インデクスを使用した検索については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(b) XML 型全文検索用インデクス（n-gram）

XML 型の列には，XML 型の値に対する全文検索用の n-gram インデクス（IXXML）を定義できます。XML 型全文検索用インデクスを定義することで，XMLEXISTS 述語の XQuery 式中で文字列一致などの全文検索条件を含む述語を記述した場合に，行の絞り込みの処理時間を削減できることがあります。

XML 型全文検索の条件となる XQuery 式中の述語を次に示します。

- 文字列（xs:string 型）同士の完全一致（=）
- fn:contains 関数
- fn:starts-with 関数
- fn:ends-with 関数
- hi-fn:contains 関数

インデクスの使用条件については，「インデクスの使用条件」で説明します。

XML 型全文検索用インデクスを使用した検索については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(3) 表へのデータの格納

表へデータを格納する場合の入力データには，次の二つがあります。入力データの種類によって，データの格納方法が異なります。

1. XML 文書を XML 挿入データに変換した ESIS-B 形式

   この場合は，XML 変換コマンド（phdxmlcnv）又は XML 変換ライブラリ（Java ライブラリ）を使用して XML 文書を解析し，XML 挿入データ（ESIS-B 形式といいます）を生成します。この ESIS-B 形式のデータをバイナリ形式にして出力し，pdload 又は INSERT 文を使用して表へ格納します。

   XML 変換コマンド及び XML 変換ライブラリについては，マニュアル「HiRDB XML Extension」を参照してください。

2. XML 文書

   この場合は，データベース作成ユティリティ（pdload）又は XMLPARSE 関数を使用して XML 文書を XML 挿入データ（ESIS-B 形式）に変換し，表へ格納します。XML 文書から ESIS-B 形式への変換は pdload 又は XMLPARSE 関数の中で行われます。pdload 内で ESIS-B 形式への変換を行う場合には，-G オプションを指定します。

(a) データロード

データベース作成ユティリティ（pdload）で表にデータを格納する場合の手順を次に示します。

〈手順〉

1. pdhold コマンドで，データロード対象 RD エリア（RDAREA01，RDAREA02，LOBAREA01）を閉塞します。
2. pdload コマンドで，入力データファイルを表にデータロードします。
   - XML 文書をそのまま入力データとする場合は-G オプションを指定します。
   - RD エリアにはデータロード対象表（インデクス）だけを格納していて，かつ初期ロードのため，データベースの更新ログ取得方式にログレスモードを選択します。
   - インデクスの作成方法にインデクス一括作成モード（省略値）を選択します。
   - コンストラクタ関数やコンストラクタ関数に渡すデータ型の情報を列構成情報ファイルに指定します。
   - 入力データファイルの形式をバイナリ形式に設定します。

   pdload コマンドに指定するオプションについては，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。
3. ログレスモードで pdload コマンドを実行しているため，データロード対象 RD エリアのバックアップを取得します。RD エリア単位の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。
4. pdrels コマンドで，データロード対象 RD エリアの閉塞を解除します。

上記のコマンドとユティリティの詳細及びこれらのコマンドとユティリティの実行結果の確認方法については，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。

補足事項

- ログレスモードで pdload コマンドを実行するため，前記の手順 1～3 の間はデータロード対象 RD エリアを閉塞したままにしてください。
- 改竄防止表に対して pdload コマンドでデータロードするとき，-d オプションは使用できません。
- インデクス一括作成中にエラーが発生した場合の対処方法については，「インデクス一括作成中に発生したエラーの対処方法」を参照してください。

(b) XML 文書の挿入

ESIS-B 形式のデータを表へ挿入又は更新する場合

INSERT 文の挿入値，又は UPDATE 文の更新値に XML コンストラクタ関数を指定し，その引数に生成した ESIS-B 形式のデータを設定します。

書籍管理表に埋込み変数 bookinfo に格納された XML 文書（ESIS-B 形式）の値を挿入する例を次に示します。

XML 文書（ESIS-B 形式）の挿入例

```
INSERT INTO 書籍管理表
  VALUES ( 310494321, XML(:bookinfo AS BINARY(102400)))
```

XML 文書を表へ挿入又は更新する場合

INSERT 文の挿入値，又は UPDATE 文の更新値に XMLPARSE 関数を指定し，その引数に XML 文書を設定します。

書籍管理表に埋込み変数 bookdoc に格納された XML 文書の値を挿入する例を次に示します。

XML 文書の挿入例

```
INSERT INTO 書籍管理表
  VALUES ( 310494321, XMLPARSE(DOCUMENT :bookdoc AS BINARY(32000)))
```

(4) インデクスの使用条件

ここでは，(2)で説明した二つのインデクスの使用条件について説明します。

部分構造インデクスの使用条件

部分構造インデクスを定義した場合，次の表に示す部分構造インデクスの使用条件を満たすと，インデクスが使用されます。

表 5-4 部分構造インデクスの使用条件

USING UNIQUE TAG の指定	XQuery 指定箇所	XQuery 中の演算子又は関数	部分構造インデクスの使用条件※1
あり	XMLEXISTS 述語	=	(a)1，2，3※2，3 (b)1，2，5，6，7※2，8，9
		!=，>，>=，<，<=，<>， eq，ne，gt，ge，lt，le， fn:contains， fn:starts-with， fn:ends-with	(a)1，2，3※2，3 (b)1，2，4，6，7※2，8，9
	XMLQUERY 関数	=	(c)1，2，3
なし	XMLEXISTS 述語	=	(a)1，2，3※2，3 (b)1，3，5，6，7※2，8，9，11※3
		!=，>，>=，<，<=，<>	(a)1，2，3※2，3 (b)1，3，4，6，7※2，8，9，11※3
		fn:contains fn:starts-with fn:ends-with	(a)1，2，3※2，3 (b)1，3，4，6，7※2，8，10，11※3
	XMLQUERY 関数	=	(c)1，2，3

注※1

(a)は，(a)で説明する部分構造インデクス及び XML 型全文検索用インデクス共通の使用条件を示します。

(b)は，(b)で説明する部分構造インデクスの使用条件を示します。

(c)は，(c)で説明する XMLQUERY 関数の XQuery に関する部分構造インデクスの使用条件を示します。

また，数字は(a)，(b)，及び(c)での項番に対応しています。

注※2

XMLEXISTS 述語の XQuery 問合せに XQuery 論理式（OR）を指定する場合，インデクスが使用されます。

注※3

XMLEXISTS 述語の XQuery 問合せに XQuery 論理式（AND）を指定する場合，インデクスが使用されます。

XML 型全文検索用インデクスの使用条件

XML 型全文検索用インデクスを定義した場合，次の表に示す XML 型全文検索用インデクスの使用条件を満たすと，インデクスが使用されます。

表 5-5　XML 型全文検索用インデクスの使用条件

XQuery 指定箇所	XQuery 中の演算子又は関数	XML 型全文検索用インデクスの使用条件※1
XMLEXISTS 述語	fn:contains fn:starts-with fn:ends-with =	(a)1，2，3※2，3 (d)1，2，4，5，6，7，8
	hi-fn:contains	(a)1，2，3※2，3 (d)1，3，4，5，6，7，8

注※1

(a)は，(a)で説明する部分構造インデクス及び XML 型全文検索用インデクス共通の使用条件を示します。

(d)は，(d)で説明する XML 型全文検索用インデクスの使用条件を示します。

また，数字は(a)及び(d)での項番に対応しています。

注※2

XMLEXISTS 述語の XQuery 問合せに XQuery 論理式（OR）を指定する場合，インデクスが使用されます。

注※3

XMLEXISTS 述語の XQuery 問合せに XQuery 論理式（AND）を指定する場合，インデクスが使用されます。

複数のインデクスを使用できる演算子又は関数の場合

部分構造インデクスと XML 型全文検索用インデクスの両方を使用できる XQuery 中の演算子又は関数を使用して検索を行う場合，評価に使用するインデクスは演算子又は関数によって決まります。演算子又は関数ごとに，どのインデクスが使用されるかを次の表に示します。

表 5-6　複数のインデクスが使用できる演算子又は関数の評価に使用するインデクス

項番	演算子又は関数	評価に使用するインデクス
1	=	部分構造インデクス
2	fn:contains	XML 型全文検索用インデクス
3	fn:starts-with	部分構造インデクス
4	fn:ends-with	XML 型全文検索用インデクス

使用するインデクスを指定したい場合は，使用インデクスの SQL 最適化指定をします。詳細については，マニュアル「HiRDB SQL リファレンス」の「使用インデクスの SQL 最適化指定」を参照してください。

なお，HiRDB が見積もるアクセスコストによっては，これらのインデクスを使用しないことがあります。インデクスを使用した検索が行われたかどうかについては，アクセスパス表示ユティリティ（pdvwopt）で確認してください。

また，XMLEXISTS 述語の XQuery 問合せ中に指定した，部分構造インデクス，又は XML 型全文検索インデクスを使用できる述語については，最大で 255 個しかインデクスを用いて評価しません。

(a) 部分構造インデクス及び XML 型全文検索用インデクス共通の使用条件

部分構造インデクス及び XML 型全文検索用インデクス共通の使用条件を次に示します。

なお，次のように定義したインデクスを例文中で使用します。

```
create index idx1 on t1(c1) key using unique tag from '/root/elm1/@attr1' as varchar(10)
```

1. XMLEXISTS 述語の XML 問合せ引数に XML 問合せ文脈項目を指定している。

   （例）

   ```
   select c2 from t1
     where xmlexists('/root/elm1[@attr1 eq "ABC"]'
           passing by value c1,'DEF' as A)
   ```

   注　アンダーライン部分が XML 問合せ文脈項目です。

2. XMLEXISTS 述語の XQuery 問合せ中に指定した文脈項目式（ピリオド）をすべて XQuery 述語中に指定している。

   （例）

   ```
   select c2 from t1
     where xmlexists('/root/elm1[./@attr1 eq "ABC"]' passing by value c1)
   ```

   注　アンダーライン部分が XQuery 述語中に指定した文脈項目式です。

3. XMLEXISTS 述語の XQuery 問合せ中の XQuery 論理式（AND，OR）をすべて XQuery 述語中に指定している。

   （例）

   ```
   select c2 from t1
     where xmlexists('/root[elm1/@attr1 = "ABC" or elm1/@attr1 = "DEF"]'
           passing by value c1)
   ```

   注　アンダーライン部分が XQuery 述語中に指定した XQuery 論理式です（OR）。

(b) XMLEXISTS 述語の XQuery に関する部分構造インデクスの使用条件

XMLEXISTS 述語の XQuery に関する部分構造インデクスの使用条件を次に示します。

なお，1.，2.，4.～9.では，次のように定義したインデクスを例文中で使用します。

```
create index idx1 on t1(c1) key using unique tag from '/root/elm1/@attr1' as varchar(10)
create index idx4 on t1(c1) key using unique tag from '/root/elm1/elm2' as varchar(10)
```

3.，10.，11.では，次のように定義したインデクスを例文中で使用します。

```
create index idx2 on t1(c1) key from '/root/elm1/@attr1' as varchar(10)
create index idx5 on t1(c1) key from '/root/elm1/elm2' as varchar(10)
```

1. 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する。

   （例）

   ```
   select c2 from t1
     where xmlexists('/root/elm1[@attr1 eq "ABC"]' passing by value c1)
   ```

   注　アンダーライン部分が一致する部分構造パスです。

2. USING UNIQUE TAG の指定がある部分構造インデクスの場合，汎用比較，値比較，fn:contains 関数，fn:starts-with 関数，又は fn:ends-with 関数を使用して，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を比較する。

   （例）

   ```
   select c2 from t1
     where xmlexists('/root/elm1[@attr1 eq "ABC"]' passing by value c1)
   ```

   注　アンダーライン部分が XQuery 比較式です（値比較）。

3. USING UNIQUE TAG の指定がない部分構造インデクスの場合，汎用比較，fn:contains 関数，fn:starts-with 関数，又は fn:ends-with 関数を使用して，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を比較する。

   （例）

   ```
   select c2 from t1
     where xmlexists('/root/elm1[@attr1 = "ABC"]' passing by value c1)
   ```

   注　アンダーライン部分が XQuery 比較式です（汎用比較）。

4. 汎用比較又は値比較と，fn:contains 関数，fn:starts-with 関数，又は fn:ends-with 関数の場合に分けて説明します。

   ＜汎用比較，又は値比較の場合＞

   汎用比較，又は値比較で比較する対象は，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と，一つの XQuery 定数又は XQuery 変数である。

（例）

```
select c2 from t1
  where xmlexists('/root/elm1[@attr1 >= "ABC"]' passing by value c1)
```

注　アンダーライン部分が部分構造パスと XQuery 定数の比較です。

＜ fn:contains 関数，fn:starts-with 関数，又は fn:ends-with 関数の場合＞

fn:contains 関数，fn:starts-with 関数，又は fn:ends-with 関数の第 1 引数が 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で，かつ第 2 引数が一つの XQuery 定数又は XQuery 変数である。

（例）

```
select c2 from t1
  where xmlexists('/root/elm1[fn:starts-with(@attr1,"ABC")]' passing by value c1)
```

注　アンダーライン部分が部分構造パスと XQuery 定数の比較です。

5. =で比較する対象は，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と，システム共通定義 pd_apply_search_ats_num オペランドの指定値以下の XQuery 定数又は XQuery 変数から成る XQuery シーケンス連結式である。

（例）

```
select c2 from t1
  where xmlexists('/root/elm1[@attr1 = ("ABC","DEF","GHI")]'
        passing by value c1)
```

注　アンダーライン部分がシステム共通定義 pd_apply_search_ats_num オペランドの指定値以下の XQuery 定数から成る XQuery シーケンス連結式です。

6. 部分構造インデクス定義時に指定したキー値のデータ型が，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と比較する XQuery 定数若しくは XQuery 問合せ中の XQuery 変数に渡す値式のデータ型と同じ，又は変換可能である。

（例）

```
select c2 from t1
  where xmlexists('/root/elm1[@attr1 = "ABC"]' passing by value c1)
```

注　アンダーライン部分が，キー値のデータ型である VARCHAR 型と同じ string 型データです。

7. XMLEXISTS 述語の XQuery 問合せに指定した XQuery 論理式（OR）オペランドに部分構造インデクスを使用できる条件だけを含む。

（例）

```
select c2 from t1
  where xmlexists('/root[elm1/@attr1 = "ABC" or elm1/@attr1 = "DEF"]'
        passing by value c1)
```

注　アンダーライン部分が，すべて部分構造インデクスを使用できる条件です。

8. 4.又は 5.で，条件として指定した部分構造パスと比較する値として XQuery 変数を指定した場合，その XQuery 変数に渡す値式が次のどれかである。

- 定数
- USER 値関数
- 値式が?パラメタ，SQL パラメタ，又は SQL 変数の CAST 指定
- 外への参照なしスカラ副問合せ

（例 1）

```
select c2 from t1
  where xmlexists('/root/elm1[@attr1 eq $A]'
        passing by value c1,'ABC' as A)
```

注　アンダーライン部分が XQuery 問合せ中の XQuery 変数に渡す値式です（定数）。

（例 2）

```
select c2 from t1
  where xmlexists('/root/elm1[@attr1 eq $A]'
        passing by value c1,cast(? as varchar(256)) as A)
```

注　アンダーライン部分が，XQuery 問合せ中の XQuery 変数に渡す値式（値式が?パラメタ，SQL パラメタ，又は SQL 変数の CAST 指定）です。

9. 値比較，又は汎用比較で比較する対象の部分構造パスが次の形式で指定されている。又は，USING UNIQUE TAG の指定がある部分構造インデクスの場合で，かつ fn:contains 関数，fn:starts-with 関数，又は fn:ends-with 関数の第 1 引数を含む部分構造パスが次の形式で指定されている。

```
部分構造パス:: = 〔XML名前空間宣言〕…部分構造パス式
XML名前空間宣言::= {declare namespace 接頭辞 = XML名前空間URI;
                  | declare default element namespace XML名前空間URI;}
部分構造パス式:: = 〔/ステップ式…〕/ステップ式
ステップ式:: = { 〔{child:: | attribute:: | @ }〕 修飾名 | 文脈項目式}
文脈項目式:: = ピリオド
ピリオド::= .
修飾名:: = 〔接頭辞:〕局所名
```

＜値比較又は汎用比較を使用した例＞

次の（例 1）～（例 3）は，値比較又は汎用比較を使用した例です。これらの例で値比較演算子 eq にほかの値比較又は汎用比較を指定した場合も，インデクスを使用します。なお，USING UNIQUE TAG の指定有無に関するインデクスの使用条件については，2.及び 3.を参照してください。

（例 1）ステップ式に@又は attribute::と，修飾名を指定した場合（@と attribute::は同じ意味）

```
select c2 from t1
  where xmlexists('/root/child::elm1[@attr1 eq "ABC"]'
        passing by value c1)
```

注　アンダーライン部分が上記の形式と一致する部分構造パスです。

（例 2）ステップ式に child::を指定，又は child::を省略し，修飾名を指定した場合

```
select c2 from t1
  where xmlexists('/root/child::elm1[child::elm2 eq "ABC"]'
        passing by value c1)
```

注　アンダーライン部分が上記の形式と一致する部分構造パスです。

（例 3）ステップ式に文脈項目式を指定した場合（値比較は使用条件 2.に該当する場合だけ）

```
select c2 from t1
  where xmlexists('/root/child::elm1/elm2[ . eq "ABC"]'
        passing by value c1)
```

注　アンダーライン部分が上記の形式と一致する部分構造パスです。

＜ XQuery 関数を使用した例＞

次の（例 4）～（例 6）は，XQuery 関数を使用した例です。これらの例で fn:starts-with 関数にほかの XQuery 関数を指定した場合も，インデクスを使用します。

（例 4）ステップ式に@又は attribute::と，修飾名を指定した場合（@と attribute::は同じ意味）

```
select c2 from t1
  where xmlexists('/root/elm1[fn:starts-with(@attr1 ,"ABC")]'
        passing by value c1)
```

注　アンダーライン部分が上記の形式と一致する部分構造パスです。

（例 5）ステップ式に child::を指定，又は child::を省略し，修飾名を指定した場合

```
select c2 from t1
  where xmlexists('/root/elm1[fn:starts-with(elm2 ,"ABC")]'
        passing by value c1)
```

注　アンダーライン部分が上記の形式と一致する部分構造パスです。

（例 6）ステップ式に文脈項目式を指定した場合

```
select c2 from t1
  where xmlexists('/root/elm1/@attr1[fn:starts-with( . ,"ABC")]'
        passing by value c1)
```

注　アンダーライン部分が上記の形式と一致する部分構造パスです。

10. USING UNIQUE TAG の指定がない部分構造インデクスの場合で，かつ fn:contains 関数，fn:starts-with 関数，又は fn:ends-with 関数の第 1 引数を含む部分構造パスが次の形式で指定されており，さらに第 1 引数が次のステップ式終端の形式で指定されている。

```
部分構造パス:: = 〔XML名前空間宣言〕…部分構造パス式
XML名前空間宣言::= {declare namespace 接頭辞 = XML名前空間URI;
                  | declare default element namespace XML名前空間URI;}
部分構造パス式:: = 〔/ステップ式…〕/ステップ式終端
ステップ式:: = { 〔{child:: | attribute:: | @ }〕 修飾名 | 文脈項目式}
ステップ式終端:: = {{attribute:: | @ } 修飾名 | 文脈項目式}
文脈項目式:: = ピリオド
ピリオド::= .
修飾名:: = 〔接頭辞:〕局所名
```

次に示す例で fn:starts-with 関数にほかの XQuery 関数を指定した場合も，インデクスを使用します。

（例 1）ステップ式終端に@又は attribute::と，修飾名を指定した場合（@と attribute::は同じ意味）

```
select c2 from t1
  where xmlexists('/root/elm1[fn:starts-with(@attr1 ,"ABC")]'
        passing by value c1)
```

注　アンダーライン部分が上記の形式と一致する部分構造パスです。

（例 2）ステップ式終端に文脈項目式を指定した場合

```
select c2 from t1
  where xmlexists('/root/elm1/@attr1[fn:starts-with( . ,"ABC")]'
        passing by value c1)
```

注　アンダーライン部分が上記の形式と一致する部分構造パスです。

11. USING UNIQUE TAG の指定がない部分構造インデクスで，次のどちらかの条件を満たす。

- XMLEXISTS 述語の XQuery 問合せの XQuery 論理式（AND）オペランドに同じノードを複数回指定していない。
- XMLEXISTS 述語の XQuery 問合せの XQuery 論理式（AND）オペランドに同じノードを複数回指定しているが，汎用比較，又は fn:contains 関数，fn:starts-with 関数，若しくは fn:ends-with 関数の第 1 引数で指定する同じノードが，次のステップ式終端の形式で指定されている。

```
ステップ式終端 ：：＝ { {attribute:: | @}  修飾名 | 文脈項目式}
修飾名 ：：＝ 〔接頭辞:〕局所名
文脈項目式 ：：＝ ピリオド
ピリオド ：：＝ .
```

＜部分構造インデクスを使用する例＞

- XMLEXISTS 述語の XQuery 問合せの XQuery 論理式（AND）オペランドに同じノードを複数回指定していない場合の例

（例 1）

```
select c2 from t1
  where xmlexists('/root/elm1[@attr1 >= "A01" and elm2 <= "A99"]'
        passing by value c1);
```

注　アンダーライン部分が同じノードを複数回指定していない部分構造パスです。

（例 2）

```
select c2 from t1
  where xmlexists('/root/elm1[fn:starts-with(@attr1 ,"A01") and elm2 = "A01"]'
        passing by value c1);
```

注　アンダーライン部分が同じノードを複数回指定していない部分構造パスです。

- XMLEXISTS 述語の XQuery 問合せの XQuery 論理式（AND）オペランドに同じノードを複数回指定した場合の例

（例 3）汎用比較の比較項に@又は attribute::と，修飾名を指定した場合（@と attribute::は同じ意味）

```
select c2 from t1
  where xmlexists('/root/elm1[@attr1 >= "A01" and @attr1 <= "A99"]'
        passing by value c1);
```

注　アンダーライン部分が形式と一致する部分構造パスです。

（例 4）fn:contains 関数，fn:starts-with 関数，又は fn:ends-with 関数の第 1 引数に@又は attribute::と，修飾名を指定した場合（@と attribute::は同じ意味）

```
select c2 from t1
  where xmlexists('/root/elm1[fn:starts-with(@attr1 ,"A") and fn:ends-with(@attr1 ,"01")]
'
        passing by value c1);
```

注　アンダーライン部分が形式と一致する部分構造パスです。

（例 5）汎用比較の比較項に文脈項目式を指定した場合

```
select c2 from t1
  where xmlexists('/root/elm1/elm2[. >= "A01" and . <= "A99"]'
        passing by value c1);
```

注　アンダーライン部分が形式と一致する部分構造パスです。

（例 6）fn:contains 関数，fn:starts-with 関数，又は fn:ends-with 関数の第 1 引数に文脈項目式を指定した場合

```
select c2 from t1
  where xmlexists('/root/elm1/elm2 [fn:contains(. ,"A") and fn:contains(. ,"01")]'
        passing by value c1);
```

注　アンダーライン部分が形式と一致する部分構造パスです。

＜部分構造インデクスを使用しない例＞

（例 1）汎用比較の比較項の@attr1 の前に elm1 が指定されているため，形式と一致しない場合

```
select c2 from t1
  where xmlexists('/root[elm1/@attr1 >= "A01" and elm1/@attr1 <= "A99"]'
        passing by value c1);
```

注　アンダーライン部分が形式と一致しない部分構造パスです。

（例 2）fn:contains 関数，fn:starts-with 関数，又は fn:ends-with 関数の第 1 引数の@attr1 の前に elm1 が指定されているため，形式と一致しない場合

```
select c2 from t1
  where xmlexists('/root[fn:starts-with(elm1/@attr1 ,"A")
                 and fn:ends-with(elm1/@attr1 ,"01")]'
        passing by value c1);
```

注　アンダーライン部分が形式と一致しない部分構造パスです。

（例 3）汎用比較の比較項に，属性又は文脈項目以外の同じノードが複数回指定されているため，形式と一致しない場合

```
select c2 from t1
  where xmlexists('/root/elm1[elm2 >= "A01" and elm2 <= "A99"]'
        passing by value c1);
```

注　アンダーライン部分が形式と一致しない部分構造パスです。

（例 4）fn:contains 関数，fn:starts-with 関数，又は fn:ends-with 関数の第 1 引数に，属性又は文脈項目以外の同じノードが複数回指定されているため，形式と一致しない場合

```
select c2 from t1
  where xmlexists('/root/elm1[fn:starts-with(elm2 ,"A") and fn:ends-with(elm2 ,"01")]'
        passing by value c1);
```

注　アンダーライン部分が形式と一致しない部分構造パスです。

(c)　XMLQUERY 関数の XQuery に関する部分構造インデクスの使用条件

XMLQUERY 関数の XQuery に関する部分構造インデクスの使用条件を次に示します。

なお，1.及び 2.では，次のように定義したインデクスを例文中で使用します。

```
create index idx1 on t1(c1) key using unique tag from '/root/elm1' as varchar(10)
```

1. SQL が次のすべての条件を満たす。

- SELECT 文，又は INSERT～SELECT 文である
- 主問合せの選択式が一つである
- 上記の主問合せの選択式が XMLQUERY である（ただし，XMLSERIALIZE の引数が XMLQUERY でもよい）
- 上記の XMLQUERY の XML 問合せ引数が XML 問合せ変数一つで，かつ変数に渡す値式は XMLAGG である
- 上記の XML 問合せ変数に指定した XMLAGG の引数は列指定単独である
- 表の結合を指定していない
- 集合演算を指定していない
- 副問合せを指定していない
- 集合関数を指定していない
- GROUP BY 句を指定していない
- HAVING 句を指定していない
- WHERE 句が指定されている場合，AND の指定数が 255 以下である

SQL の例については，2.の例を参照してください。

2. XMLQUERY 関数に指定した XQuery が次のすべての条件を満たす。

a. XML 問合せ変数をルートとするパス式である

b. 最も外側の XQuery 述語の指定が一つだけ存在する

c. b.の述語で汎用比較'='による比較を行っている

d. c.の比較が，XML 列の特定の部分構造と XQuery 変数をルートとするパス式の比較である

（例）

```
select xmlserialize(xmlquery('$VAR1/root[elm1 = $VAR1/root[elm2 = "ABC"]/elm1]/elm1'
  passing by value xmlagg(c1) as VAR1 empty on empty) as varchar(32000)) from t1
```

注　アンダーライン部分が上記の条件と一致する箇所です。

3. 2.の d.の XML 列の特定の部分構造，及び XQuery 変数をルートとするパス式に，同じデータ型の部分構造インデクスを定義している（これらのインデクスは同一のものでもかまいません）。

(d)　XML 型全文検索用インデクスの使用条件

XML 型全文検索用インデクスの使用条件を次に示します。

1. 検索対象の XML 型の列に全文検索インデクスを定義している。

（例）

```
create index idx3 using type ixxml on t2(c1) in (LOB1)
```

2. fn:contains 関数，fn:starts-with 関数，又は fn:ends-with 関数の第 1 引数を含む部分構造パスが次の形式で指定されており，さらに第 1 引数が次のテキストステップ式終端，又は属性ステップ式終端の形式で指定されている。又は=で比較する対象の部分構造パスが次の形式で指定されており，XQuery 述語中に指定した部分構造パスが次のテキストステップ式終端又は属性ステップ式終端の形式で指定されている。

```
部分構造パス:: = 〔XML名前空間宣言〕… 部分構造パス式
XML名前空間宣言:: =  {declare default element namespace
                       "http://www.w3.org/XML/1998/namespace";
                     |declare namespace 接頭辞 = XML名前空間URI;※
                     |declare default element namespace XML名前空間URI;※}
部分構造パス式:: = 〔 {/ | //※} ステップ式 …〕
                       {/ | //※}  {テキストステップ式 | 属性ステップ式終端}
ステップ式:: =  { 〔child::〕  名前テスト | 文脈項目式}
テキストステップ式:: = 〔{child:: | descendant::}〕テキストテスト
                          /テキストステップ式終端
テキストステップ式終端:: = 文脈項目式
属性ステップ式終端:: = {{attribute:: | @} 名前テスト
                      | 〔{attribute:: | @}〕  属性テスト}
文脈項目式:: = ピリオド
ピリオド:: = .
名前テスト:: =  {修飾名 | *※ | 接頭辞:*※ | *:局所名※}
修飾名:: = 〔接頭辞:〕局所名
```

注※　HiRDB XML Extension のバージョンが 08-04 以降の場合に指定できます。

次に示す例で fn:contains 関数にほかの XQuery 関数を指定した場合も，インデクスを使用します。

（例 1）第 1 引数に@又は attribute::と，名前テストを指定した場合（@と attribute::は同じ意味）

```
select c2 from t2
  where xmlexists('/root/child::elm1[fn:contains(@attr1,"ABC")]'
        passing by value c1)
```

注　アンダーライン部分が上記の形式と一致する部分構造パスです。

（例 2）第 1 引数に@又は attribute::と，属性テストを指定した場合（@と attribute::は同じ意味）

```
select c2 from t2
  where xmlexists('/root/child::elm1[fn:contains(@attribute(),"ABC")]'
        passing by value c1)
```

注　アンダーライン部分が上記の形式と一致する部分構造パスです。

（例 3）第 1 引数に属性テストだけを指定した場合

```
select c2 from t2
  where xmlexists('/root/child::elm1[fn:contains(attribute(),"ABC")]'
        passing by value c1)
```

注　アンダーライン部分が上記の形式と一致する部分構造パスです。

（例 4）第 1 引数に文脈項目式を指定した場合

```
select c2 from t2
  where xmlexists('/root/child::elm1/text()[fn:contains( . ,"ABC")]'
        passing by value c1)
```

注　アンダーライン部分が上記の形式と一致する部分構造パスです。

3. hi-fn:contains 関数の第 1 引数を含む部分構造パスが次の形式で指定されており，さらに第 1 引数が次のテキストステップ式，テキストステップ式終端，又は属性ステップ式終端の形式で指定されている。また，HiRDB XML Extension のバージョンが 08-04 以降である。

```
部分構造パス:: = 〔XML名前空間宣言〕… 部分構造パス式
XML名前空間宣言:: = {declare namespace 接頭辞 = XML名前空間URI;
                    |declare default element namespace XML名前空間URI;}
部分構造パス式:: = 〔 {/ | //} ステップ式 …〕
                     {/ | //}  {テキストステップ式 | 属性ステップ式終端}
ステップ式:: = { 〔child::〕 名前テスト | 文脈項目式}
テキストステップ式:: = 〔{child:: | descendant::}〕テキストテスト
                       〔/テキストステップ式終端〕
テキストステップ式終端:: = 文脈項目式
属性ステップ式終端:: = {{attribute:: | @} 名前テスト
                      | 〔{attribute:: | @}〕 属性テスト}
文脈項目式:: = ピリオド
ピリオド:: = .
名前テスト:: = {修飾名 | * | 接頭辞:* | *:局所名}
修飾名:: = 〔接頭辞:〕局所名
```

（例 1）第 1 引数にテキストテストを指定した場合

```
select c2 from t2
  where xmlexists('/root/elm1[hi-fn:contains(text()," " " ABC AND DEF" " " )]'
        passing by value c1)

select c2 from t2
```

```
  where xmlexists('/root/elm1[hi-fn:contains(descendant::text(),""" ABC AND DEF""" )]
'
        passing by value c1)
```

注　アンダーライン部分が上記の形式と一致する部分構造パスです。

（例 2）第 1 引数にテキストテスト及び文脈項目式を指定した場合

```
select c2 from t2
  where xmlexists('/root/elm1[hi-fn:contains(text()/.,""" ABC AND DEF""" )]'
        passing by value c1)

select c2 from t2
  where xmlexists('/root/elm1[hi-fn:contains(descendant::text()/.,""" ABC AND DEF""" 
)]'
        passing by value c1)
```

注　アンダーライン部分が上記の形式と一致する部分構造パスです。

（例 3）第 1 引数に文脈項目式を指定した場合

```
select c2 from t2
  where xmlexists('/root/elm1/text()[hi-fn:contains( . ,""" ABC AND DEF""" )]'
        passing by value c1)

select c2 from t2
  where xmlexists('/root/elm1/ descendant::text()[hi-fn:contains( . ,""" ABC AND DE
F""" )]'
        passing by value c1)
```

注　アンダーライン部分が上記の形式と一致する部分構造パスです。

（例 4）第 1 引数に@又は attribute::と，名前テストを指定した場合（@と attribute::は同じ意味）

```
select c2 from t2
  where xmlexists('/root/elm1[hi-fn:contains(@attr1,""" ABC AND DEF""" )]'
        passing by value c1)
```

注　アンダーライン部分が上記の形式と一致する部分構造パスです。

（例 5）第 1 引数に@又は attribute::と，属性テストを指定した場合（@と attribute::は同じ意味）

```
select c2 from t2
  where xmlexists('/root/elm1[hi-fn:contains(@attribute(),""" ABC AND DEF""" )]'
        passing by value c1)
```

注　アンダーライン部分が上記の形式と一致する部分構造パスです。

（例 6）第 1 引数に属性テストだけを指定した場合

```
select c2 from t2
  where xmlexists('/root/elm1[hi-fn:contains(attribute(),""" ABC AND DEF""" )]'
        passing by value c1)
```

注　アンダーライン部分が上記の形式と一致する部分構造パスです。

4. XMLEXISTS 述語の XQuery 問合せ中に指定した文字列の長さが 32,000 バイト以下である。

（例）

```
select c2 from t2
  where xmlexists('/root/elm1[fn:contains(@attr1,"ABCDEF")]'
        passing by value c1)
```

注　アンダーライン部分が 32,000 バイト以下の文字列です。

5. 2.又は 3.の形式で指定した部分構造パスと比較する値が文字列の XQuery 定数である。

（例）

```
select c2 from t2
  where xmlexists('/root/child::elm1[fn:contains(@attr1,"ABC")]'
        passing by value c1)
```

注　アンダーライン部分が文字列の XQuery 定数です。

6. XMLEXISTS 述語の XQuery 問合せ中の部分構造パス式に指定した，/，//，@，及び局所名の長さの和（部分構造パス式の 1 文字目の/は除く）が 1,024 バイト以下である。

（例）

```
select c2 from t2
  where xmlexists('/root/elm1[fn:contains(@attr1,"ABCDEF")]'
        passing by value c1)
```

注　アンダーライン部分が 1,024 バイト以下の部分構造パス式です。

7. XMLEXISTS 述語の XQuery 問合せ中に指定した部分構造パス式中での//の指定が一つ以内である。

（例）

```
select c2 from t2
  where xmlexists('/root/elm1[fn:contains(@attr1,"ABCDEF")]'
        passing by value c1)
```

注　アンダーライン部分が//の指定が一つ以内の部分構造パス式です。

8. 検索対象の XML 型の列に定義した全文検索インデクスに，次のどのプラグインオプションも指定していない。

- DELcode=ファイル名
- NOindex=ファイル名
- ENGLISH
- ENGLISH_STANDARD

プラグインオプションの詳細については，マニュアル「HiRDB XML Extension」を参照してください。

（例）

```
create index idx6 using type ixxml on t2(c1) in (LOB1)
PLUGIN'SAMECASE=ON,SAMEWIDE=ON,SAMEY=ON,SAMED=ON,DELcode=ON'
```

注　アンダーライン部分がインデクスを使用できるオプションだけを指定した全文検索インデクスです。

(e) インデクスを使用しない場合

次に示す場合は，部分構造インデクス又は XML 型全文検索用インデクスを使用しません。

なお，1.及び 3.では，次のように定義したインデクスを例文中で使用します。

```
create index idx1 on t1(c1) key using unique tag from '/root/elm1/@attr1' as varchar(10)
```

2.，及び 4.では，次のように定義したインデクスを例文中で使用します。

```
create index idx3 using type ixxml on t2(c1) in (LOB1)
```

1. '/root[elm1/@attr1'が"ABC"又は"DEF"であるかを評価する XQuery を XMLEXISTS 述語の XQuery 問合せに指定する場合

   次の例では，XMLEXISTS 述語の XQuery 問合せ中の XQuery 論理式（AND，OR）すべてを XQuery 述語中に指定していないため，インデクスを使用しません。この指定方法では，XMLEXISTS 述語の引数の XQuery 問合せ（XQuery 述語中を除く）に直接 XQuery 論理式を指定しているため，XQuery 問合せの結果は必ず真か偽どちらかのブーリアン値になります。このため，XQuery 問合せの結果は空のシーケンスではないので XMLEXISTS 述語の結果は常に真となり，意図した結果になりません（XMLEXISTS 述語は XQuery 問合せの結果が空のシーケンスの場合だけ偽になり，それ以外は真になる）。

   （例：変更前）

   ```
   select c2 from t1
     where xmlexists('/root[elm1/@attr1 = "ABC"] or /root[elm1/@attr1 = "DEF"]'
           passing by value c1)
   ```

   注　アンダーライン部分が XQuery 述語中に指定していない XQuery 論理式（OR）です。

   次のように変更すると，インデクスを使用するようになります。

   （例：変更後）

   ```
   select c2 from t1
     where xmlexists('/root[elm1/@attr1 = "ABC" or elm1/@attr1 = "DEF"]'
           passing by value c1)
   ```

   注　アンダーライン部分が XQuery 述語中に指定した XQuery 論理式（OR）です。

2. '/root/elm1'以下のテキストノードを検索する XQuery を XMLEXISTS 述語の XQuery 問合せに指定する場合

   次の例では，contains 関数の第 1 引数を含む部分構造パスのテキストステップ式終端が，(4)(c)の 2.で示した形式と一致しないため，インデクスを使用しません。

   （例：変更前）

   ```
   select c2 from t2
     where xmlexists('/root[fn:contains(elm1/text(),"ABC")]' passing by value c1)
   ```

   注　アンダーライン部分が，(4)(c)の 2.で示した形式と一致しない部分構造パスです。

次のように変更すると，インデクスを使用するようになります。

（例：変更後）

```
select c2 from t2
  where xmlexists('/root/elm1/text()[fn:contains( .,"ABC")]'
        passing by value c1)
```

注　アンダーライン部分が，(4)(c)の 2.で示した形式と一致する部分構造パスです。

3. '/root/elm1/@attr1/' が" ABC" であるかを評価する XQuery を XMLEXISTS 述語の XQuery 問合せに指定する場合

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の部分構造パス式が，(4)(b)の 9.で示した形式と一致しないため，インデクスを使用しません。

（例：変更前）

```
select c2 from t1
  where xmlexists('$A/root/elm1[@attr1 eq "ABC"]'
        passing by value c1,c1 as A)
```

注　アンダーライン部分が，(4)(b)の 9.で示した形式と一致しない部分構造パスです。

次のように変更すると，インデクスを使用するようになります。

（例：変更後）

```
select c2 from t1
  where xmlexists('/root/elm1[@attr1 eq "ABC"]'
        passing by value c1,c1 as A)
```

注　アンダーライン部分が，(4)(b)の 9.で示した形式と一致する部分構造パスです。

4. 論理演算子（OR，AND）で XMLEXISTS 述語を多数連結した場合

HiRDB が見積もるアクセスコストによって，インデクスを使用しない方が最適なアクセスパスとなると判断し，XMLEXISTS 述語の評価にインデクスを使用しません。hi-fn:contains 関数を指定した場合はインデクスだけで評価できないため，SQL エラーとなります。

（例：変更前）

```
select c2 from t2
  where xmlexists('/root/elm1[hi-fn:contains(text()," " "01ABC" " ")]'
        passing by value c1)
     or xmlexists('/root/elm1[hi-fn:contains(text()," " "02ABC" " ")]'
        passing by value c1)
        … （省略） …
     or xmlexists('/root/elm1[hi-fn:contains(text()"" " 30ABC"" " )]'
        passing by value c1)
```

注　アンダーライン部分が，単独の指定ではインデクスを使用する XMLEXISTS 述語を 30 個指定した条件です。

次のように使用インデクスの SQL 最適化指定をすると，インデクスを使用するようになります。

（例：変更後）

```
select c2 from t2 with index(idx3,idx3)
  where xmlexists('/root/elm1[hi-fn:contains(text()," " "01ABC" " ")]'
        passing by value c1)
     or xmlexists('/root/elm1[hi-fn:contains(text()," " "02ABC" " ")]'
        passing by value c1)
        … (省略) …
     or xmlexists('/root/elm1[hi-fn:contains(text()"" " 30ABC"" " )]'
        passing by value c1)
```

注　アンダーライン部分が，複数インデクス利用の実行に必要な種類のインデクスを指定した，インデクスの SQL 最適化指定です。

5.5　ユーザが定義した抽象データ型を定義した表の作成

5.5.1　抽象データ型の定義

抽象データ型とルーチンを使用して，複雑な構造を持つデータ型とその操作を独自に定義し，利用できます。

(1)　定義方法

任意の構造を持つデータ型（抽象データ型）を定義するには，定義系 SQL の CREATE TYPE を実行します。CREATE TYPE では，データ構造とそのデータに対する操作を定義します。ここでは，次に示すデータ構造を持つ抽象データ型「t_従業員」を定義し，それに対する操作を関数として定義します。

データ構造

- 氏名，性別，役職，入社年月日，顔写真，基本給というデータ構造を定義します。

データ操作

- 現在の日付と入社年月日から勤続年数を算出します。
- 勤続年数に応じた報酬率を算出します。
- 基本給に報酬率を乗じて従業員の報酬を算出します。

（例）

```
CREATE TYPE t_従業員 (              …1
   PUBLIC    氏名    NCHAR(16),
             性別    CHAR(1),
             役職    NCHAR(10),
   PRIVATE   入社年月日 date,       …2
   PUBLIC    顔写真  BLOB(64K),
   PROTECTED 基本給  INTEGER,       …3

   PUBLIC FUNCTION t_従業員(p_氏名 NCHAR(16),   …4
                p_性別 CHAR(1),
                p_役職 NCHAR(10),
                p_入社年月日 date,
                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;    …5
           SET d_従業員=t_従業員();      …6
           SET d_従業員..氏名=p_氏名;    …7
           SET d_従業員..性別=p_性別;    …7
           SET d_従業員..役職=p_役職;    …7
           SET d_従業員..入社年月日=p_入社年月日;   …7
           SET d_従業員..顔写真=p_顔写真;    …7
           SET d_従業員..基本給=p_基本給;    …7
           RETURN d_従業員;                  …8
       END,
```

```
    PUBLIC FUNCTION 勤続年数(p t_従業員) RETURNS INTEGER   …9
        BEGIN
             DECLARE working_years INTERVAL YEAR TO DAY;
             SET working_years=CURRENT_DATE - p..入社年月日;
             RETURN YEAR(working_years);
        END,

    PROTECTED FUNCTION 報酬率(p t_従業員) RETURNS FLOAT    …10
        BEGIN
             DECLARE rate FLOAT;
             SET rate=勤続年数(p)*0.2/30;
             RETURN rate;
        END,

    PUBLIC FUNCTION 報酬(p t_従業員) RETURNS INTEGER   …11
        BEGIN
             DECLARE bonus INTEGER;
             SET bonus=p..基本給*報酬率(p);
             RETURN bonus;
        END
    )
```

〔説明〕

1. データ構造の定義です。抽象データ型「t_従業員」を定義します。
2. 「t_従業員」型の属性「入社年月日」は，報酬査定の算出に使用します。この属性は，外部から直接参照又は変更する必要がないため，隠蔽レベル「PRIVATE」を指定しています。隠蔽レベルについては，「抽象データ型を含む表」を参照してください。
3. 「t_従業員」型の属性「基本給」は，報酬査定の算出に使用します。この属性も，外部から直接参照又は変更する必要がありません。ただし，この属性はサブタイプでも共通に参照するため，隠蔽レベル「PROTECTED」を指定しています。隠蔽レベルについては，「抽象データ型を含む表」を参照してください。
4. ユーザ定義のコンストラクタ関数を定義します。
5. 値（インスタンス）を生成し，関数の戻り値とするための SQL 変数を t_従業員型で宣言します。
6. システムが提供するデフォルトコンストラクタ関数によって，すべての属性の値がナル値である値（インスタンス）を生成します。デフォルトコンストラクタ関数は，引数なしの t_従業員型と同じ名称の関数です。
7. 6.で指定した値に対し，コンポネント指定による代入文で，各属性の値を代入します。コンストラクタ関数の引数から得た値を設定したり，その値を使ってデータを加工したものを代入したりできます。
8. RETURN 文によって，新しく生成した値（インスタンス）を戻り値とします。戻り値のデータ型は，t_従業員型でなければなりません。これは，抽象データ型名と同じ名前のコンストラクタ関数であること及び RETURNS 句で型を決めているためです。
9. データ操作のための関数です。従業員の勤続年数を返します。現在の日付と入社年月日から算出します。隠蔽レベル「PRIVATE」が指定されている属性「入社年月日」にアクセスする関数です。
10. データ操作のための関数です。従業員の報酬率を返します。勤続年数に応じて算出します。

11. データ操作のための関数です。従業員の報酬を返します。基本給に報酬率を乗じて，勤続年数に応じた従業員の報酬を算出します。

(2) 継承を利用する場合の定義方法

抽象データ型「t_従業員」をスーパタイプとするサブタイプ「t_営業部員」の定義例を次に示します。

（例）

```
CREATE TYPE t_営業部員 UNDER t_従業員
(       PUBLIC 担当顧客       NCHAR(15),
        PUBLIC FUNCTION 報酬(p t_営業部員) RETURNS INTEGER
          BEGIN
              DECLARE salebonus INTEGER;
              SET salebonus = 顧客総数(・・・)*1000+p..基本給*報酬率(p);
              RETURN salebonus;
          END
)
```

(3) 抽象データ型のナル値

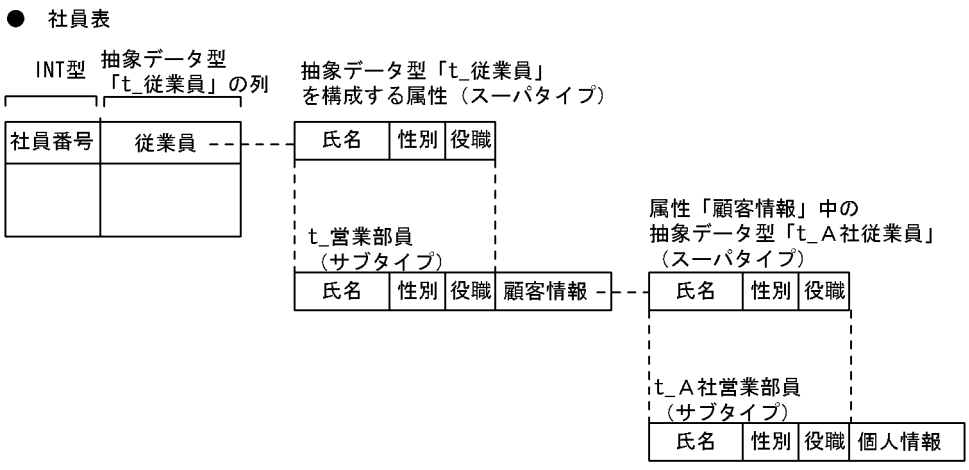
操作系 SQL の INSERT 文で値が明示的に指定されない場合，抽象データ型全体の値はナル値になります。

(4) 抽象データ型のサブタイプを削除する方法

表定義で，ある抽象データ型を直接指定していない場合でも，その抽象データ型の上位の抽象データ型（スーパタイプ）が列の型に指定されている場合，代替可能性によってその抽象データ型（サブタイプ）の値は表に格納されている場合があります。このため，抽象データ型（サブタイプ）を削除する場合には注意が必要です。

次の図に示す**代替可能性**を利用した抽象データ型を含む表の場合にサブタイプを削除する方法について説明します。

図 5-7 代替可能性を利用した抽象データ型を含む表の例



1. 社員表を削除します。

2. t_従業員のサブタイプ「t_営業部員」を削除します。
3. t_従業員を削除します。
4. t_A 社従業員のサブタイプ「t_A 社営業部員」を削除します。
5. t_A 社従業員を削除します。

抽象データ型のサブタイプを削除できないケースについては，「注意」を参照してください。

(5) 注意

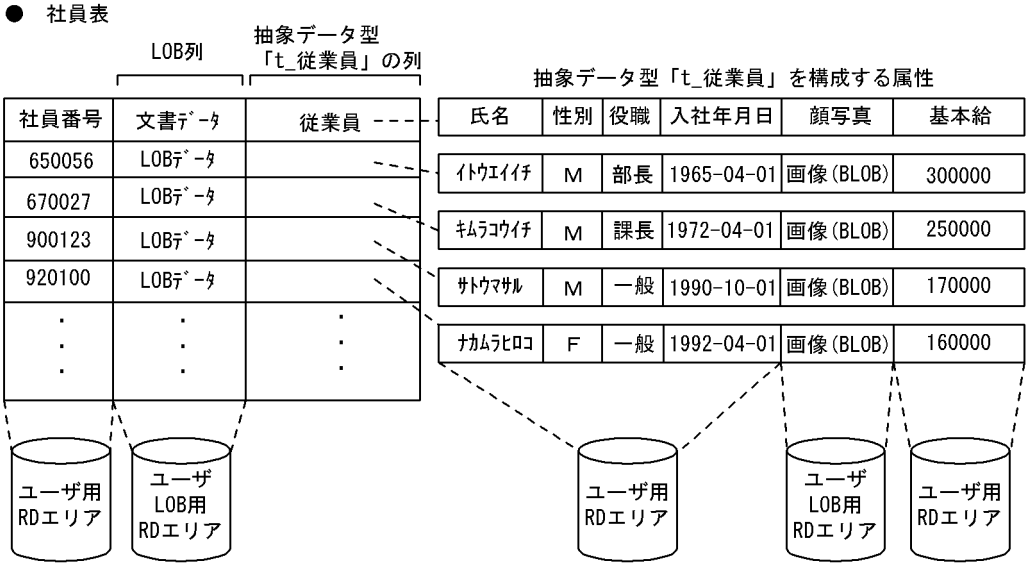
1. コンストラクタ関数を指定して値を生成している場合，抽象データ型を構成する各属性の値がナル値であっても，抽象データ型全体としての値はナル値にはなりません。
2. ある抽象データ型及びそのスーパタイプが表で定義されている場合，その抽象データ型のサブタイプは削除できません。
3. ある抽象データ型及びそのスーパタイプが，ほかの抽象データ型の属性に指定されている場合，その抽象データ型のサブタイプは削除できません。
4. サブタイプを定義する場合，作成するデータ型の上位のデータ型に次に示す条件を満足するストアドプロシジャ及びストアドファンクションがあると，それらは無効状態になります。
   - ストアドプロシジャ及びストアドファンクションの SQL パラメタに指定したデータ型
   - 関数の戻り値のデータ型
   - ストアドプロシジャ及びストアドファンクションから呼び出している関数の引数及び戻り値のデータ型
   - ストアドプロシジャ及びストアドファンクション中で指定しているデータ型（コンポネント指定で抽象データ型をアクセスする場合，その途中のデータ型を含みます）

5.5.2 表の定義

表を構成する列のデータ型の違いによって，RD エリアに格納する単位が異なります。ここでは，社員No，文書データ（LOB データ）及び抽象データ型「t_従業員」から構成される社員表の例について説明します。

なお，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を**抽象データ型列構成基表**といいます。

● 社員表



注
・一つのユーザＬＯＢ用ＲＤエリアには，表中の一つのＬＯＢ列だけを格納します。また，一つの表に複数のＬＯＢ列がある場合には，それぞれ別のユーザＬＯＢ用ＲＤエリアに格納する必要があります。ＬＯＢ列以外の列は複数のユーザ用ＲＤエリアに分割して格納する必要はありません。
・ＬＯＢ列が横分割表の場合には，ＬＯＢ列ごとにユーザＬＯＢ用ＲＤエリアと，表を格納しているユーザ用ＲＤエリアの数を１対１に対応させる必要があります。

〔説明〕

社員表をディスク A,B のユーザ用 RD エリア RDAREA01 と RDAREA02 に，社員表中の文書データ（LOB 列）をユーザ LOB 用 RD エリア LOBAREA01 と LOBAREA02 に，社員表中の抽象データ型（LOB 属性）顔写真の列をユーザ LOB 用 RD エリア LOBAREA03 と LOBAREA04 に分割して格納します。

(1) キーレンジ分割の場合

格納条件指定

```
CREATE TABLE 社員表
(社員番号 CHAR(6),
 文書データ BLOB(64K) IN ((LOBAREA01),(LOBAREA02)),
 従業員 t_従業員 ALLOCATE(顔写真
     IN ((LOBAREA03),(LOBAREA04)))
    )IN ((RDAREA01)社員番号<=700000,(RDAREA02));
```

境界値指定

```
CREATE TABLE 社員表
(社員番号 CHAR(6),
 文書データ BLOB(64K) IN ((LOBAREA01),(LOBAREA02)),
 従業員 t_従業員 ALLOCATE(顔写真
     IN ((LOBAREA03),(LOBAREA04)))
    )PARTITIONED BY 社員番号
     IN ((RDAREA01)800000,(RDAREA02));
```

(2) フレキシブルハッシュ分割，FIX ハッシュ分割の場合

```
CREATE TABLE 社員表
(社員番号 CHAR(6),
 文書データ BLOB(64K) IN ((LOBAREA01),(LOBAREA02)),
 従業員 t_従業員 ALLOCATE(顔写真
     IN ((LOBAREA03),(LOBAREA04)))
    )[FIX]※ HASH HASH6 BY 社員番号
     IN (RDAREA01,RDAREA02);
```

注※　FIX ハッシュ分割の場合に指定します。

5.5.3　インデクスの定義

「社員番号」列にインデクスを定義します。なお，抽象データ型の列にはインデクスを定義できません。

（例）

```
CREATE INDEX INDX1 ON 社員表（社員番号）
 IN ((RDAREA03),(RDAREA04));
```

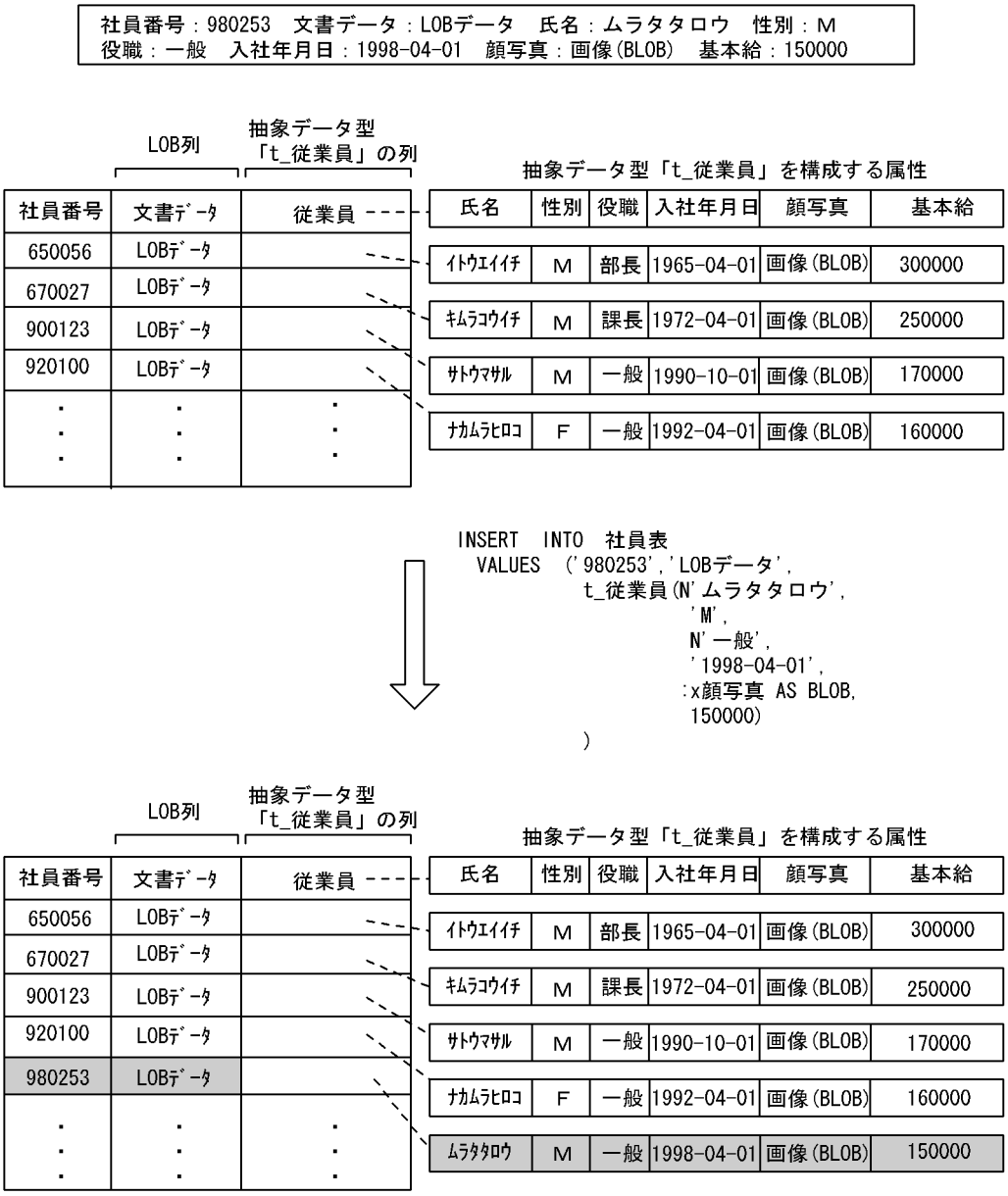
〔説明〕

分割キーインデクス INDX1 を横分割した社員表に対応させて，ユーザ用 RD エリア RDAREA03 と RDAREA04 に分割して格納します。なお，インデクス INDX1 を構成する列に社員番号を指定しています。

5.5.4　表へのデータの格納

ユーザが定義した抽象データ型を定義した表にデータを格納するには，操作系 SQL の INSERT 文を実行します。データベース作成ユティリティ（pdload）でデータロードはできません。データを挿入するには，関数を定義して値を生成したものを INSERT 文で挿入します。抽象データ型の列を含む表にデータを挿入するときの手順を次の図に示します。

図 5-8　抽象データ型の列を含む表にデータを挿入する手順



注　: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。

5.5.5　データベースの更新ログ取得方式

(1)　データベースの更新ログ取得方式の種類

データベースの更新ログ取得方式には，次に示す 3 種類のモードがあります。

1. 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。データ件数が少ない場合の追加データ作成，再編成の場合に使用します。

2. 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。データ件数が多いときの初期作成，追加データ作成及び再編成の場合に使用します。

3. ログレスモード

データベース更新ログを取得しません。一つの RD エリアに一つの表だけ（分割格納している場合は一つの横分割表）格納している場合で，関連するインデクスも一つの RD エリア内にあるときの初期作成，再編成の場合に使用します。

(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は，次に示すものがあります。

- クライアント環境定義の PDDBLOG に指定する方法※1
- CREATE TABLE の RECOVERY オペランドに指定する方法※2

注※1

ユーザ用 RD エリアを更新する UAP の，データベースの更新ログ取得方式を指定するオペランドです。

注※2

ユーザ LOB 用 RD エリアを更新する UAP の，データベースの更新ログ取得方式を指定するオペランドです。

注意事項

ユーザ LOB 用 RD エリアのデータベースの更新ログ取得方式（CREATE　TABLE の RECOVERY オペランド指定値）は，クライアント環境定義の指定値によって変わることがあります。クライアント環境定義によって変わるユーザ LOB 用 RD エリアのデータベースの更新ログ取得方式を次の表に示します。

表 5-7　クライアント環境定義によって変わるユーザ LOB 用 RD エリアのデータベースの更新ログ取得方式

クライアント環境定義 PDDBLOG	CREATE TABLE の RECOVERY オペランドの指定値		
	ALL	PARTIAL	NO
ALL	ALL	PARTIAL	NO
NO	NO	NO	NO

（凡例）

ALL：ログ取得モード

PARTIAL：更新前ログ取得モード

NO：ログレスモード

例えば，CREATE TABLE の RECOVERY オペランドに「PARTIAL」を指定し，クライアント環境定義のログ取得モードが「NO」の場合，NO（ログレスモード）がユーザ LOB 用 RD エリアに設定されます。

5.5.6　データの格納状態の確認

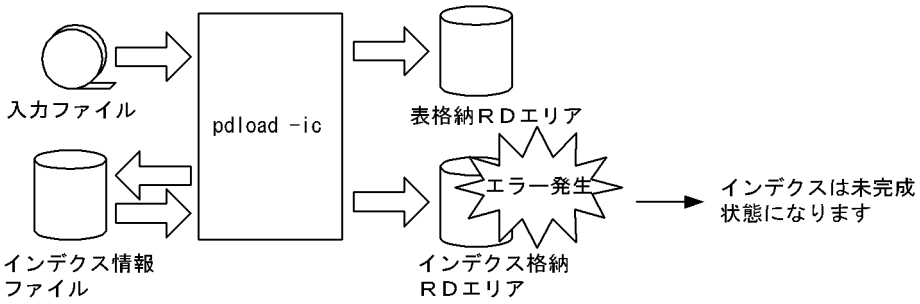
抽象データ型の列を含む表にデータを挿入した場合は，運用を開始する前にデータベース状態解析ユティリティ（pddbst）を実行して，データの格納状態を確認することをお勧めします。設計どおりにデータベースを作成できたかどうかを確認できます。

データベース状態解析ユティリティ（pddbst）を実行すると，RD エリア単位のデータの格納状態（物理解析だけ）の情報を取得できます。

5.6　インデクス一括作成中に発生したエラーの対処方法

インデクス一括作成中にエラーが発生した場合，表へのデータ格納は完了しても，インデクスは未完成という不整合な状態になります。ここでは，この状態を回復する方法について説明します。データベース作成ユティリティ（pdload）でインデクス一括作成中にエラーが発生した場合のインデクスの状態を次の図に示します。

図 5-9　データベース作成ユティリティ（pdload）でインデクス一括作成中にエラーが発生した場合のインデクスの状態



注　ログレスモードの場合は，表及びインデクス格納RDエリアはログレス閉塞状態になります。ただし，KFPL00703-Iメッセージが出力されている場合は，表へのデータ格納は完了しています。このため，インデクス格納RDエリアと表格納RDエリアが別の場合は，表格納RDエリアの閉塞は解除してかまいません。

5.6.1　ログ取得モード又は更新前ログ取得モードでデータロードをしていた場合

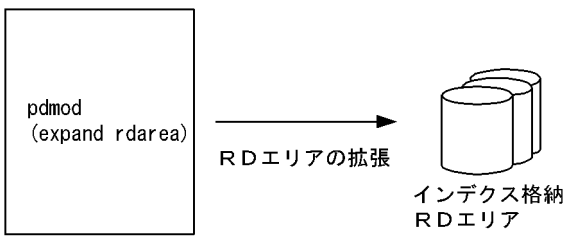
更新前ログ取得モード又はログ取得モードでデータロードをしていた場合の対処方法を説明します。

なお，プラグインインデクスを定義している場合，そのプラグインにプラグインインデクス一括作成部分回復機能があることを前提とします。プラグインにプラグインインデクス一括作成部分回復機能がない場合は，「ログレスモードでデータロードをしていた場合」を参照してください。

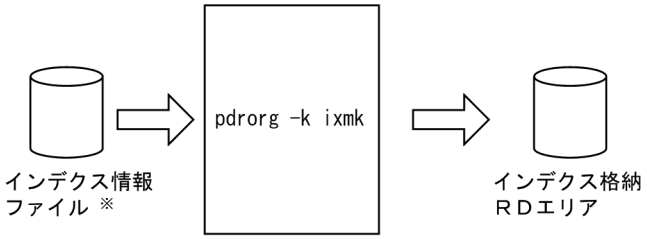
発生したエラー原因に応じて，インデクス格納 RD エリアを回復します。次に示す手順で回復してください。

(1)　インデクス格納 RD エリアの容量不足がエラー原因の場合

1. インデクス格納 RD エリアを拡張します。

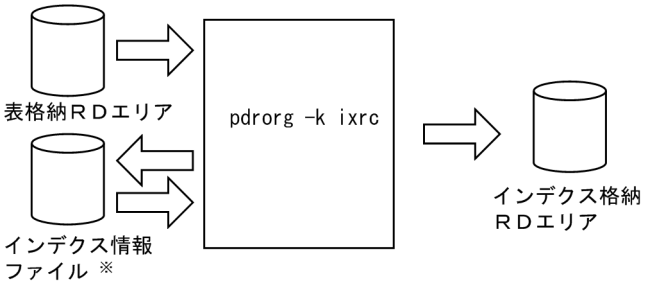


2. インデクスを作成します。データベース作成ユティリティ（pdload）が作成したインデクス情報ファイルを基にインデクスを一括作成します。



注※　このファイルには，追加ロードしたキーと既存データのキー情報が出力されています。
プラグインインデクスの場合は，追加ロードしたデータのキー情報だけが出力されています。

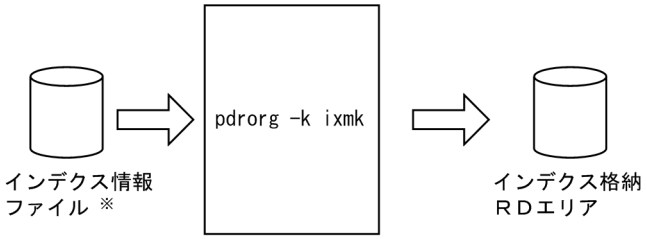
なお，データベース作成ユティリティ（pdload）が作成したインデクス情報ファイルが残っていない場合は，データベース再編成ユティリティ（pdrorg）でインデクスを再作成します。



注※　このファイルは，データベース再編成ユティリティ（pdrorg）が新たに作成します。
追加ロードしたキーと既存データのキー情報を出力します。

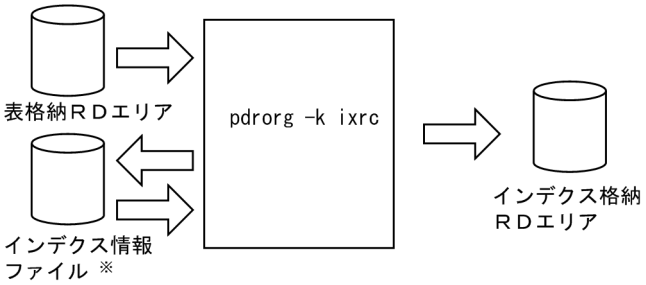
(2)　ソート処理エラー（KFPL15062-E メッセージが出力）又は pdcancel コマンドによるユティリティの強制終了が原因の場合

1. インデクスを作成します。データベース作成ユティリティ（pdload）が作成したインデクス情報ファイルを基にインデクスを一括作成します。



注※　このファイルには，追加ロードしたキーと既存データのキー情報が出力されています。
プラグインインデクスの場合は，追加ロードしたデータのキー情報だけが出力されています。

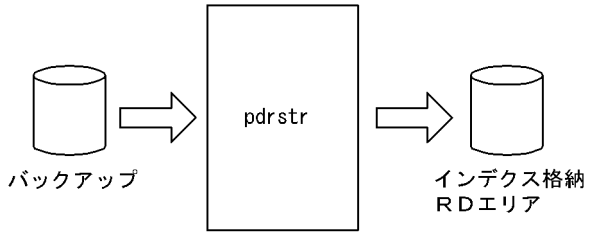
なお，データベース作成ユティリティ（pdload）が作成したインデクス情報ファイルが残っていない場合は，データベース再編成ユティリティ（pdrorg）でインデクスを再作成します。



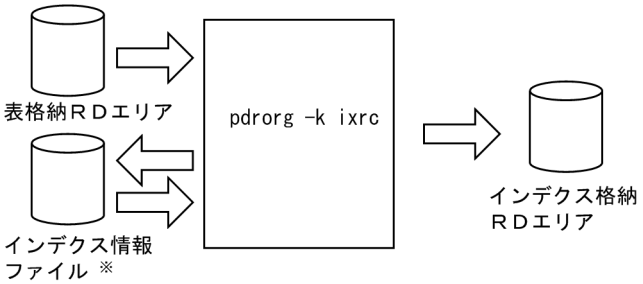
注※　このファイルは，データベース再編成ユティリティ（pdrorg）が新たに作成します。
追加ロードしたキーと既存データのキー情報を出力します。

(3)　ディスク障害が原因の場合

1. 障害となったディスクを交換し，バックアップからユティリティ実行前の状態に戻します。ただし，手順 2 で実行するインデクスロードが完了するまでアクセスされないように，RD エリアを閉塞状態にしてください。



2. データベース再編成ユティリティ（pdrorg）でインデクスを再作成します。



注※　このファイルは，データベース再編成ユティリティ（pdrorg）が新たに作成します。
追加ロードしたキーと既存データのキー情報を出力します。

5.6.2　ログレスモードでデータロードをしていた場合

ログレスモードでデータロードをしていた場合又はプラグインにプラグインインデクス一括作成部分回復機能がない場合の対処方法を説明します。

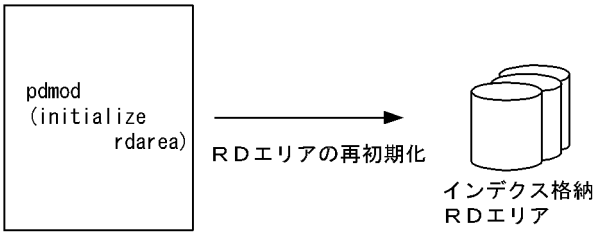
発生したエラー原因に応じて，インデクス格納 RD エリアを回復します。次に示す手順で回復してください。

(1) インデクス格納 RD エリアの容量不足がエラー原因の場合

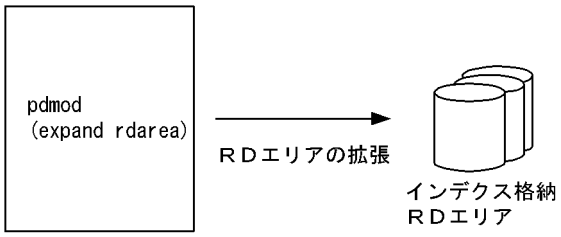
1. インデクス格納 RD エリアを再初期化します。

   なお，バックアップからも回復できますが，その場合はインデクスロードが完了するまでアクセスされないよう，RD エリアを閉塞状態にしてください。また，次の場合はバックアップから回復する必要があります。

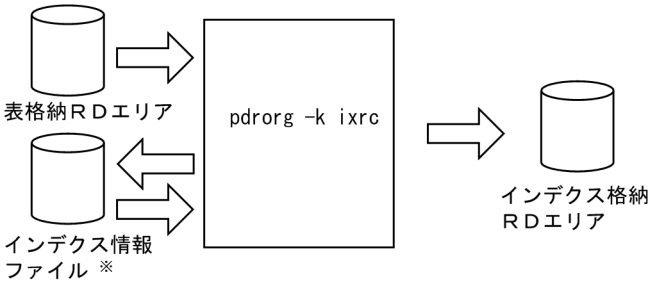
   - インデクス格納 RD エリアに該当インデクスとは別の表やインデクス，又は改竄防止表が格納されている



2. インデクス格納 RD エリアを拡張します。



3. データベース再編成ユティリティ（pdrorg）でインデクスを再作成します。



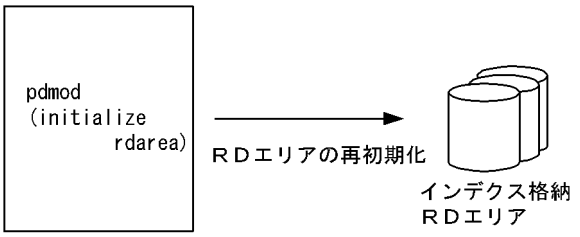
注※　このファイルは，データベース再編成ユティリティ（pdrorg）が新たに作成します。
追加ロードしたキーと既存データのキー情報を出力します。

(2) ソート処理エラー（KFPL15062-E メッセージが出力）又は pdcancel コマンドによるユティリティの強制終了が原因の場合

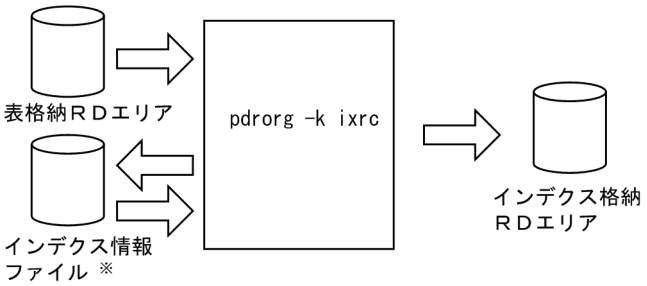
1. インデクス格納 RD エリアを再初期化します。

   なお，バックアップからも回復できますが，その場合はインデクスロードが完了するまでアクセスされないよう，RD エリアを閉塞状態にしてください。また，次の場合はバックアップから回復する必要があります。

- インデクス格納 RD エリアに該当インデクスとは別の表やインデクス，又は改竄防止表が格納されている



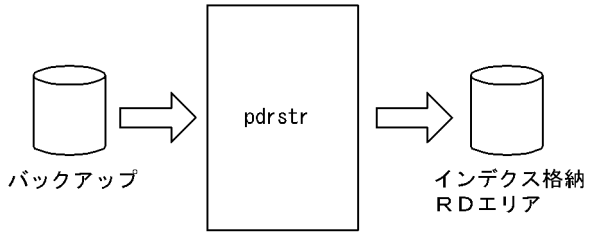
2. データベース再編成ユティリティ（pdrorg）でインデクスを再作成します。



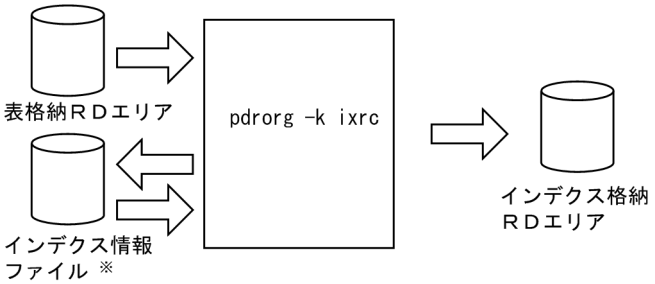
注※　このファイルは，データベース再編成ユティリティ(pdrorg)が新たに作成します。
追加ロードしたキーと既存データのキー情報を出力します。

(3)　ディスク障害が原因の場合

1. 障害となったディスクを交換し，バックアップからユティリティ実行前の状態に戻します。



2. データベース再編成ユティリティ（pdrorg）でインデクスを再作成します。



注※　このファイルは，データベース再編成ユティリティ(pdrorg)が新たに作成します。
追加ロードしたキーと既存データのキー情報を出力します。

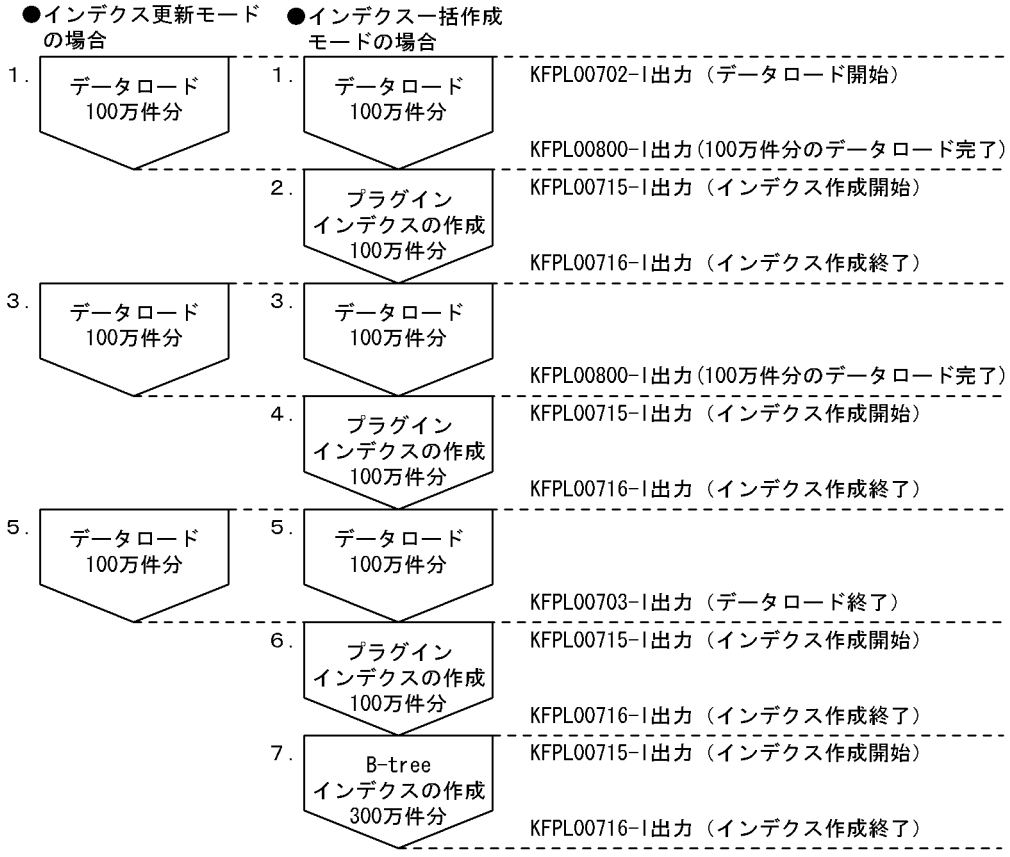
5.7　同期点指定のデータロード実行中にユティリティが異常終了したときの対処方法

ここでは，**同期点指定のデータロード**の実行中に，データベース作成ユティリティが異常終了したときの対処方法を説明します。

5.7.1　対処方法の概要

異常終了したタイミングによって対処方法が異なります。同期点指定のデータロード実行中にユティリティが異常終了したときの対処方法を次の図に示します。

図 5-10　同期点指定のデータロード実行中にユティリティが異常終了したときの対処方法



〔説明〕

- 総データロード件数を 300 万件，同期点行数を 100 万件としています。
- 1，3，5 の時点でユティリティが異常終了した場合は，データロードを再実行してください。
- 2，4 の時点でユティリティが異常終了した場合は，作成に失敗したプラグインインデクスをデータベース再編成ユティリティのインデクス一括作成機能（-k ixmk）で作成してください。その後，データロードを再実行してください。

- 6 の時点でユティリティが異常終了した場合は，作成に失敗したプラグインインデクスをデータベース再編成ユティリティのインデクス一括作成機能（-k ixmk）で作成してください。その後，データベース再編成ユティリティのインデクス再作成機能（-k ixrc）で B-tree インデクスを作成してください。
- 7 の時点でユティリティが異常終了した場合，インデクス情報ファイルが作成されていれば（KFPL00710-I メッセージが出力されていれば），データベース再編成ユティリティのインデクス一括作成機能（-k ixmk）で B-tree インデクスを作成してください。インデクス情報ファイルが作成されていなければ，データベース再編成ユティリティのインデクス再作成機能（-k ixrc）で B-tree インデクスを作成してください。

5.7.2 例題

300 万件のデータロード中にデータベース作成ユティリティが異常終了しました。インデクス一括作成モードで，同期点行数は 100 万件とします。

(1) メッセージを確認します

次に示すメッセージが出力されています。

```
KFPL00800-I Loading until 2000000th row committed

KFPL00710-I Index information file assigned, index=k87m271.”INDX01”,
RDAREA=”LOB02”, file=/pdrorg/INDX01_2

KFPL00715-I Index load started at bes2, index=k87m271.”INDX01”, RDAREA=”LOB02”
```

〔説明〕

- KFPL00800-I メッセージから，200 万件までデータロード処理が完了していることが分かります。
- KFPL00715-I メッセージから，プラグインインデクスの作成処理が開始されていることが分かります。それに対応する完了メッセージ（KFPL00716-I）が出力されていません。

これによって，100～200 万件のプラグインインデクス作成中に，ユティリティが異常終了したことが分かります。

(2) pdrorg コマンドでプラグインインデクスを一括作成します

データベース再編成ユティリティで，100 万件（100～200 万件）分のプラグインインデクスを一括作成します。

```
pdrorg -k ixmk -t TABLE1 /pdrorg/rorg01
```

〔説明〕

-k：プラグインインデクスを一括作成するため ixmk を指定します。

-t：表の名称を指定します。

/pdrorg/rorg01：

pdrorg コマンドの制御文ファイル名を指定します。制御文ファイルの内容を次に示します。制御文ファイル中に指定するインデクス情報ファイルは(1)で出力される KFPL00710-I メッセージから分かります。

```
index INDX01 LOB02 /pdrorg/INDX01_2
```

(3) データロードを再実行します

```
pdload TABLE1 /pdload/load01
```

［説明］

オプションの指定を変更する必要はありません。

(4) データロード対象 RD エリアのバックアップを取得します

データロード対象 RD エリアのバックアップを取得します。RD エリア単位の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。

(5) pdrels コマンドでデータロード対象 RD エリアの閉塞を解除します

```
pdrels -r DATA01,DATA02,DATA03,INX01,INX02,INX03,LOB01,LOB02,LOB03
```

コマンドの実行後，実行結果が正しいかどうか確認することをお勧めします。コマンドの実行結果の確認方法については，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。

6 ほかの製品との連携

この章では，HiRDB とほかの製品とを連携する方法について説明します。

6.1　レプリケーション機能との連携

HiRDB のレプリケーション機能（HiRDB Datareplicator，HiRDB Dataextractor）を使用するときに指定する項目について説明します。

6.1.1　HiRDB Datareplicator との連携

HiRDB Datareplicator を使用すると，HiRDB のデータベースの更新に連動して自動的にデータを抽出し，そのデータベース更新内容をほかの HiRDB のデータベースに反映できます。HiRDB Datareplicator を使用するには，HiRDB システム定義で次に示すオペランドを指定します。

- pd_rpl_init_start オペランド
  HiRDB Datareplicator 連携機能を HiRDB 開始時から使用するかどうかを指定します。
- pd_rpl_hdepath オペランド
  抽出側 HiRDB Datareplicator 運用ディレクトリ名を指定します。ここで指定するディレクトリ名は，抽出側 HiRDB Datareplicator の環境変数 HDEPATH で指定した名称にしてください。
- pd_log_rpl_no_standby_file_opr オペランド
  HiRDB Datareplicator 連携機能の使用時に，HiRDB Datareplicator でのシステムログの抽出が完了していないため，すべてのシステムログファイルがスワップ先にできない状態でスワップ要求が発生した場合の運用方法を指定します。

HiRDB Datareplicator を使用してデータ連動する場合のシステムの環境設定及び運用方法については，マニュアル「HiRDB データ連動機能 HiRDB Datareplicator」を参照してください。

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。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。
- 回復不要 FES を使用する場合
  回復不要 FES では，反映側 HiRDB Datareplicator の同期点処理方式に二相コミット方式を利用（反映システム定義 commitment_method オペランドに fxa_sqle を指定）した反映処理を実行できないため，回復不要 FES 以外のフロントエンドサーバを使用して反映処理を実行する必要があります。詳細は，「回復不要 FES」を参照してください。

6.1.2　HiRDB Dataextractor との連携

HiRDB Dataextractor を使用すると，メインフレーム又は HiRDB のデータベースのデータをまとめて抽出し，HiRDB のデータベースへ順次格納できます。HiRDB Dataextractor 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できます。これによって，部門データベースの表の初期作成又は全データのリフレッシュができます。
- 基幹データベースから部分的にデータを抽出し，各業務に適した部門データベースを作成できます。

HiRDB Dataextractor 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。

6.2 OLTP との連携

ここでは，HiRDB が X/Open XA インタフェースを使用して，OLTP と連携する方法について説明します。ここで説明する項目は次のとおりです。

1. OLTP と連携できる製品
2. HiRDB XA ライブラリ
3. OLTP と連携した HiRDB システムの構成例
4. トランザクションの移行
5. トランザクションマネジャへの登録
6. トランザクションマネジャに登録する情報
7. トランザクションマネジャへの登録例
8. トランザクションマネジャへの登録の変更
9. トランザクションマネジャと HiRDB 間のコネクションが切断されたときの再接続方法
10. TP1/Resource Manager Monitor の機能を使用した HiRDB の監視
11. 注意事項

6.2.1 OLTP と連携できる製品

HiRDB では次に示す 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，HiRDB の種類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。HiRDB の種類による OLTP 連携の適用可否を次の表に示します。

表 6-1 HiRDB の種類による 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	○	×	×	○

（凡例）

○：適用できます。

×：適用できません。

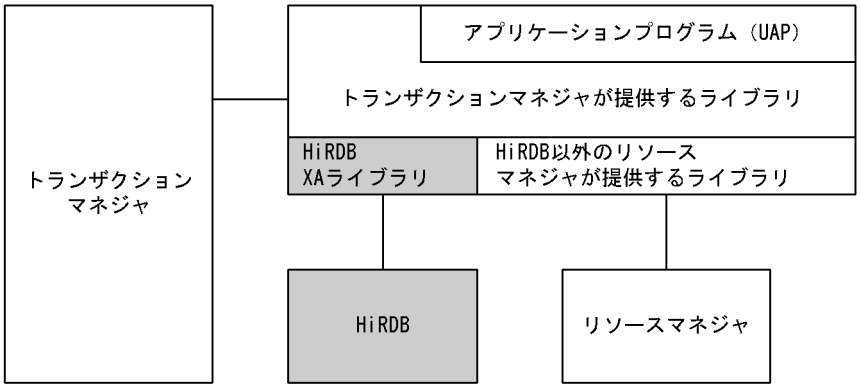
6.2.2 HiRDB XA ライブラリ

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 6-1　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。

(1) 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 6-2　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，「トランザクションの移行」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。 一相最適化を使用する場合，トランザクションの完了種別がトランザクションマネジャと HiRDB で一致しないことがあります。詳細については，「一相最適化に関する注意事項」を参照してください。
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。
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)　マルチスレッド対応の XA インタフェース

マルチスレッド対応の XA インタフェースを利用すると，TPBroker for C++と HiRDB で OTS（Object Transaction Service）連携ができます。

なお，マルチスレッド用のライブラリは C 言語又は C++言語でだけ使用できます。COBOL 言語では使用できません。

マルチスレッド対応の XA インタフェースを使用するには専用の HiRDB クライアントライブラリをリンクしてください。バージョン 05-06 より前の HiRDB クライアントライブラリはマルチスレッドに対応していません。マルチスレッド用のライブラリは，既存の HiRDB クライアントが接続できるすべての HiRDB サーバに対して使用できます。マルチスレッド用のライブラリを使用すると，スレッドの間で接続を共有できます。

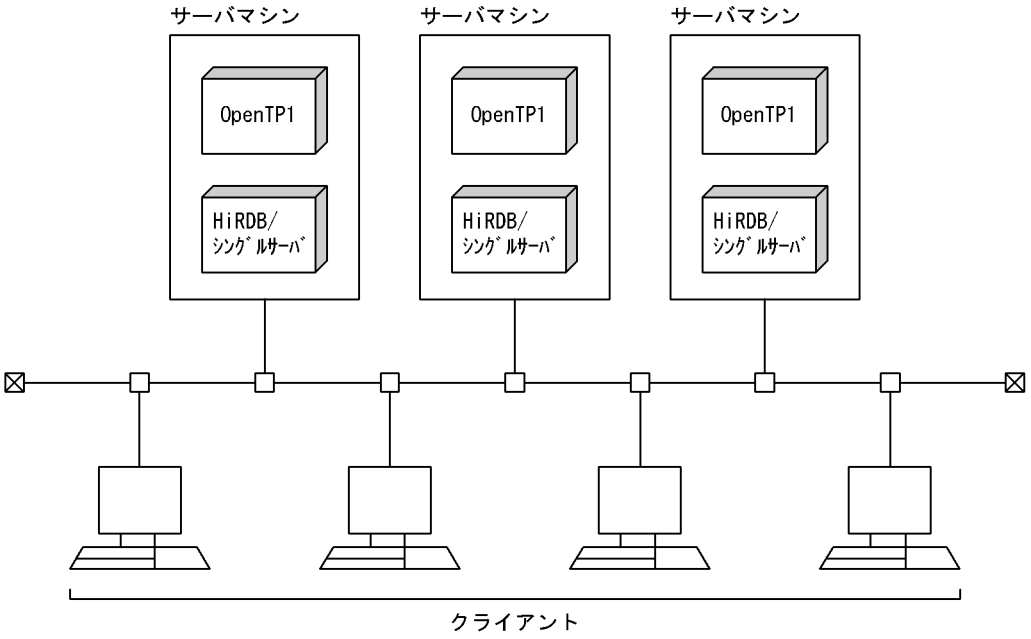
6.2.3　OLTP と連携した HiRDB システムの構成例

OLTP と連携した HiRDB システムについて，OpenTP1 を使用した例で説明します。

(1) HiRDB/シングルサーバとの連携

HiRDB/シングルサーバと OLTP（OpenTP1）を連携すると，複数の HiRDB/シングルサーバの更新処理を一つのトランザクションとして実行できます。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 OLTP（OpenTP1）が，各 HiRDB/シングルサーバへ処理を振り分けます。処理を振り分けることで，高速にトランザクション処理を実行できるようにしています。複数の HiRDB/シングルサーバを統合化する場合は，OLTP との連携を検討してください。HiRDB/シングルサーバと OLTP（OpenTP1）との連携を次の図に示します。

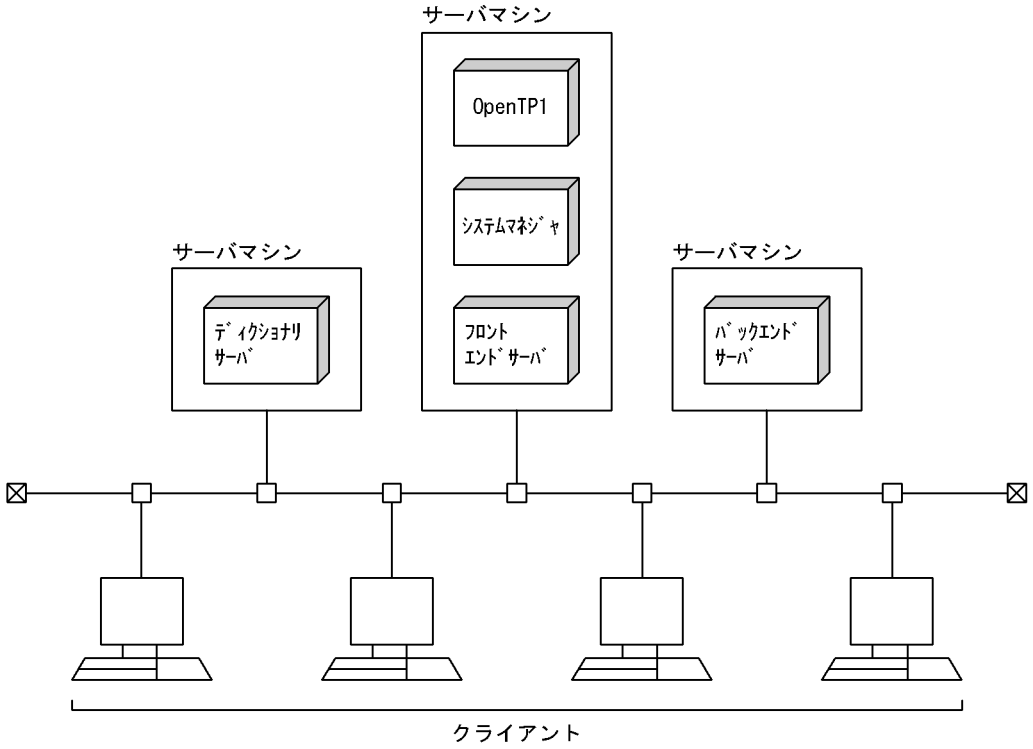
図 6-2 HiRDB/シングルサーバと OLTP（OpenTP1）との連携



(2) HiRDB/パラレルサーバとの連携

HiRDB/パラレルサーバと OLTP（OpenTP1）を連携すると，HiRDB/パラレルサーバで負荷分散したデータベースの更新処理をトランザクション処理として実行できます。HiRDB/パラレルサーバと OLTP（OpenTP1）との連携を次の図に示します。

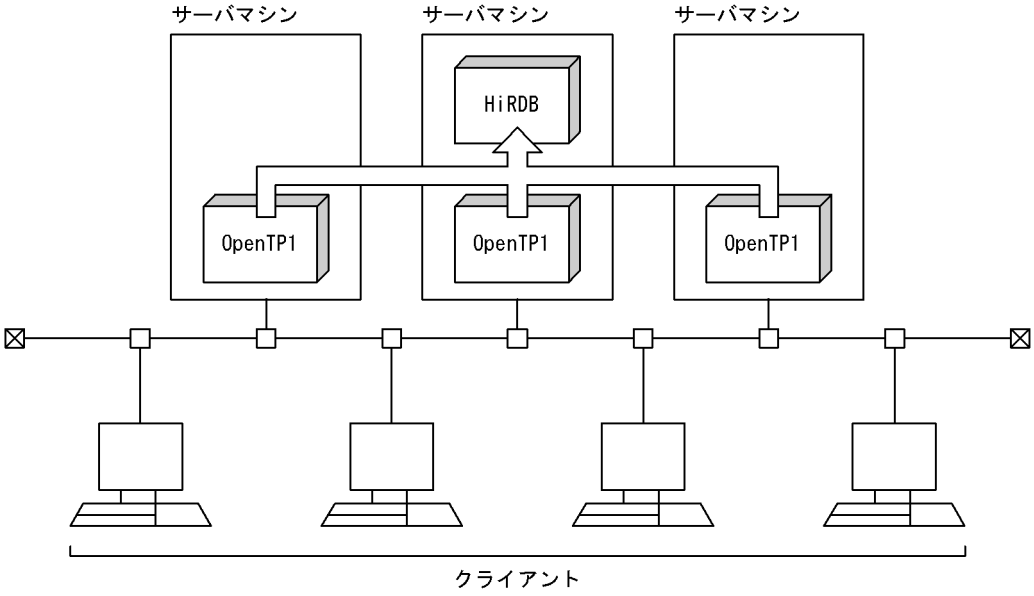
図 6-3　HiRDB/パラレルサーバと OLTP（OpenTP1）との連携



(3)　複数の OLTP（OpenTP1）と HiRDB との連携

複数の OLTP（OpenTP1）と一つの HiRDB 間でクライアント／サーバ型で通信して連携する形態です。異なる OLTP（OpenTP1）から一つの HiRDB に，同時に接続できます。この場合，各 OLTP（OpenTP1）の OLTP 識別子（クライアント環境定義の PDTMID）が異なるようにしてください。複数の OLTP（OpenTP1）と HiRDB との連携を次の図に示します。

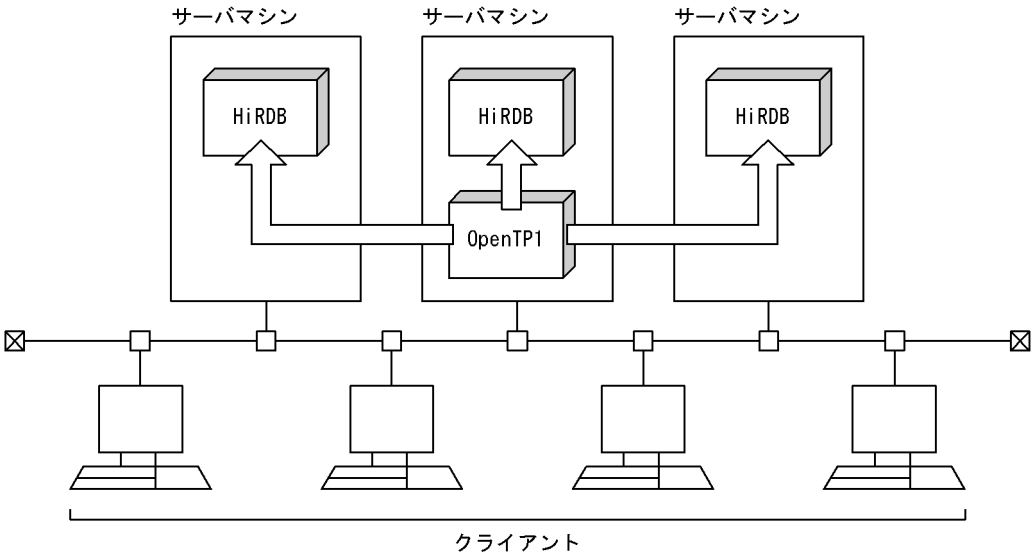
図 6-4　複数の OLTP（OpenTP1）と HiRDB との連携



(4) 一つの OLTP（OpenTP1）と複数の HiRDB との連携

一つの OLTP（OpenTP1）と複数の HiRDB で連携する形態です。異なるサーバマシンの HiRDB に同時に接続して SQL 文を実行できます。この場合，複数接続機能を使用する必要があります。一つの OLTP（OpenTP1）と複数の HiRDB との連携を次の図に示します。

図 6-5　一つの OLTP（OpenTP1）と複数の HiRDB との連携

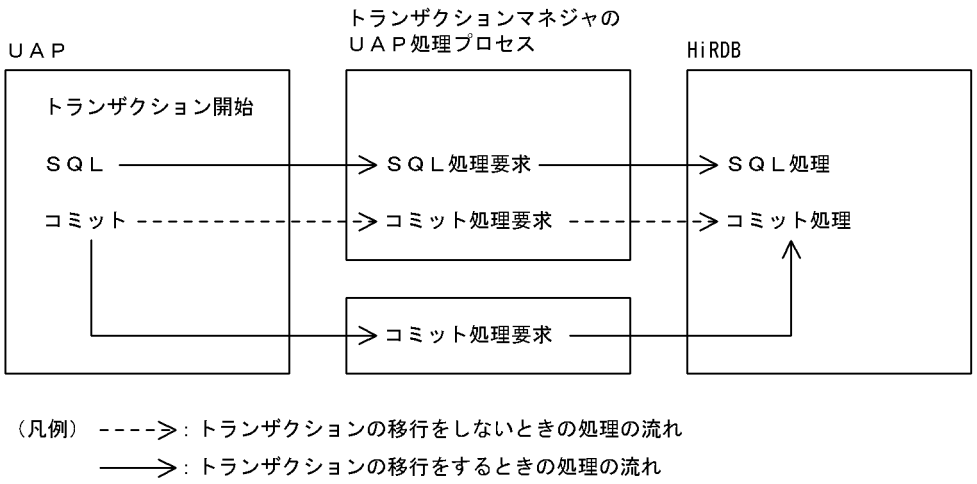


複数接続機能については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

6.2.4　トランザクションの移行

HiRDB XA ライブラリを使用して HiRDB に接続する UAP では，HiRDB に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できます。このことを**トランザクションの移行**といいます。トランザクションの移行の概要を次の図に示します。

図 6-6　トランザクションの移行の概要



メリット

トランザクションの移行を使用すると，トランザクションマネジャの UAP 処理はトランザクションの完了を待たなくても次回のサービス要求を受け付けられます。このため，この機能を使用しないときに比べて，少ないプロセス数で UAP を実行できます。ただし，HiRDB が使用するサーバプロセス数及び排他待ち※回数が増える場合があります。

注※

トランザクションが完了するまでの間，次回のサービス要求による HiRDB へのアクセスは，排他待ちになる場合が増えます。

適用基準

トランザクションマネジャがトランザクションの移行を使用する場合は，HiRDB もトランザクションの移行を使用してください。

トランザクションマネジャがトランザクションの移行を使用しない場合は，HiRDB もトランザクションの移行を使用しないでください。

トランザクションマネジャがトランザクションの移行の使用可否を設定できる場合は，次に示す点に考慮してトランザクションの移行の使用可否を設定してください。

- **HiRDB へのアクセス負荷より，UAP 自身の処理の負荷が大きい場合にトランザクションの移行を使用します。**

運用方法

トランザクションの移行を使用する場合は，クライアント環境定義の PDXAMODE オペランドに 1 を指定してください。この機能を使用しない場合は，このオペランドに 0 を指定するか，このオペランドを省略してください。

PDXAMODE オペランドについては，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

注意事項

1. トランザクションマネジャと HiRDB の間で，トランザクションの移行を使用するかどうかの設定が合っていないと，トランザクションが決着できなかったり，HiRDB が異常終了したり，トランザクションマネジャにエラーリターンしたりすることがあります。
2. この機能を使用すると，LOCK 文の LOCK TABLE UNTIL DISCONNECT の有効範囲が変わります。LOCK TABLE UNTIL DISCONNECT の有効範囲については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(1) トランザクションマネジャが OpenTP1 の場合

トランザクションの移行を使用すると，OpenTP1 のコミット最適化及びプリペア最適化に HiRDB が対応します。したがって，OpenTP1 の trnstring オペランドの-d オプションを省略した場合は，この機能を使用してください。-d オプションを指定した場合は，この機能を使用しないでください。

OpenTP1 システム定義のトランザクションサービス定義の trnstring オペランドと HiRDB の PDXAMODE オペランドの関係を次の表に示します。

表 6–3　OpenTP1 の trnstring オペランドと HiRDB の PDXAMODE オペランドの関係

trnstring オペランドの指定	PDXAMODE オペランドの値
-d オプションを省略	1
-d オプションを指定	0

注

- trnstring オペランドと PDXAMODE オペランドの指定が合っていないと，HiRDB がトランザクションを決着できません。このとき，HiRDB は OpenTP1 に対して XA 関数エラーリターンコード（−6）を返します。
- -d オプションは，TP1/Server Base のバージョンが 03-03 以降のときに指定できます。

trnstring オペランドについては，マニュアル「OpenTP1 システム定義」を参照してください。コミット最適化又はプリペア最適化については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

6.2.5　トランザクションマネジャへの登録

OLTP と連携するには，HiRDB をトランザクションマネジャに登録する必要があります。HiRDB をトランザクションマネジャに登録するには，各トランザクションマネジャのコマンド又は機能を使用します。

- OpenTP1 の場合：trnlnkrm コマンドで HiRDB を登録します。
- TPBroker for C++の場合：tslnkrm コマンドで HiRDB を登録します。
- TUXEDO の場合：$TUXDIR/udataobj/RM に HiRDB を登録します。$TUXDIR は，TUXEDO システム・ソフトウェアがあるディレクトリの絶対パス名を示しています。
- TP1/EE の場合：eetrnmkobj コマンドで HiRDB を登録します。

(1)　動的登録と静的登録

HiRDB をトランザクションマネジャに登録するときに，次に示すどちらかの方法を選択してください。

- 動的登録
- 静的登録

なお，一つのトランザクションマネジャに対して，動的登録と静的登録を混在して使用できません。

(a)　動的登録とは

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，又は 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。

(b) 静的登録とは

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，又は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。

(2) 動的登録と静的登録の違い

動的登録と静的登録の違いを次の表に示します。

表 6-4 動的登録と静的登録の違い

差異のポイント	動的登録	静的登録
トランザクション開始時	管理しない	• コネクション確立中かの確認 • トランザクションマネジャ制御下でのトランザクションの管理を開始 • コネクション確立※1
トランザクション内で最初の SQL 発行時	• トランザクションマネジャの制御下での管理を開始 • HiRDB のトランザクション開始 • SQL 処理 • コネクション確立※1	• HiRDB のトランザクション開始 • SQL 処理
トランザクション処理中のトランザクションマネジャと HiRDB 間の通信回数	SQL 文数＋コミット処理通信回数＋ 1（コネクション確立用通信分）※1	SQL 文数＋コミット処理通信回数＋ 1（トランザクション開始処理用の通信分）＋ 1（コネクション確立用通信分）※1
トランザクションマネジャと HiRDB 間のコネクションが，途中で切断したときの再接続方法※2	次回のトランザクション開始時に自動的に再接続※3	次回のトランザクション開始時に自動的に再接続※4

注※1

マルチスレッド対応の XA インタフェース使用時にする処理です。

注※2

ネットワーク障害による切断は検知できません。ただし，トランザクションマネジャが TPBroker for C++の場合はトランザクション開始時に接続を確立するため，再接続できます。

注※3

トランザクションマネジャが OpenTP1/Server Base の場合は，OpenTP1/Server Base の trn_rm_open_close_scope オペランドに transaction を指定することで，ネットワーク障害による切断の場合でも再接続できます。

注※4

トランザクションマネジャが OpenTP1/Server Base の場合は，ネットワーク障害による切断の場合でも再接続できます。また，トランザクションマネジャが TP1/EE の場合は，HiRDB のクライアント環境定義 PDXAAUTORECONNECT に YES を指定することで，ネットワーク障害による切断の場合でも再接続できます。

6.2.6 トランザクションマネジャに登録する情報

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方法については，トランザクションマネジャのマニュアルを参照してください。このとき，次に示す情報をトランザクションマネジャに指定します。

(1) RM スイッチ名

動的登録にするか又は静的登録にするかは，RM スイッチ名の指定で決まります。HiRDB の RM スイッチ名（xa_switch_t 構造体名）を次に示します。

- 動的登録の場合：pdtxa_switch
- 静的登録の場合：pdtxa_switch_y

(2) RM 名

RM スイッチ（xa_switch_t 構造体）で定義されている RM 名（リソースマネジャ名）は，HiRDB_DB_SERVER です。

(3) オープン文字列

トランザクションマネジャが xa_open でリソースマネジャをオープンするときに使用するオープン文字列は，**複数接続機能**を使用する場合に指定してください。複数接続機能を使用しない場合はオープン文字列を指定する必要はありません。ただし，トランザクションマネジャが TP1/EE の場合は，複数接続機能を使用しないときでも，登録している一つの HiRDB に対してオープン文字列を指定してください。トランザクションマネジャが TUXEDO の場合は，複数接続機能を使用できません。

複数接続機能を使用する場合は，トランザクションマネジャに複数の HiRDB を登録し，各 HiRDB に対してオープン文字列を指定します。オープン文字列には次に示す項目を指定します。

- 接続先で有効にする環境変数を設定したファイルの絶対パス名
- 環境変数グループ識別子

次のどちらかの書式で記述します。

- "環境変数グループ識別子＋環境変数設定ファイル名"
- "環境変数グループ識別子＊環境変数設定ファイル名"

これ以外の形式で指定した場合は，オープン文字列が無視されます。また，環境変数グループ識別子は 4 バイト固定，オープン文字列は全体で 257 バイト以上にできません。

トランザクションマネジャが OpenTP1，TPBroker for C++，又は TP1/EE の場合のオープン文字列の登録例を次に示します。

(a) OpenTP1 の場合

OpenTP1 のトランザクションサービス定義の trnstring オペランドでオープン文字列を登録します。ここでは二つの HiRDB を OpenTP1 に登録します。登録条件は次のとおりとします。

リソースマネジャ	環境変数グループ識別子	環境変数設定ファイル名
HiRDB1	HDB1	/usr/conf/HiRDB/HiRDB11.ini /usr/conf/HiRDB/HiRDB12.ini
HiRDB2	HDB2	/usr/conf/HiRDB/HiRDB21.ini /usr/conf/HiRDB/HiRDB22.ini

オープン文字列の登録例を次に示します。

```
trnstring -n HiRDB_DB_SERVER -i H1 -o ”HDB1*/usr/conf/HiRDB/HiRDB11.ini”
-O ”HDB1+/usr/conf/HiRDB/HiRDB12.ini”
trnstring -n HiRDB_DB_SERVER -i H2 -o ”HDB2*/usr/conf/HiRDB/HiRDB21.ini”
-O ”HDB2+/usr/conf/HiRDB/HiRDB22.ini”
```

〔説明〕

-n：リソースマネジャ名を指定します。

-i：リソースマネジャ拡張子を指定します。

-o：トランザクションサービス用 xa_open 関数用文字列を指定します。

OpenTP1 のトランザクションサービスプロセスが使用するオープン文字列を指定します。

"環境変数グループ識別子＊環境変数設定ファイル名"の形式で指定します。

-O：ユーザサーバ用 xa_open 関数用文字列を指定します。

ユーザサーバプロセスが使用するオープン文字列を指定します。

"環境変数グループ識別子＋環境変数設定ファイル名"の形式で指定します。

- -o と-O には同じ環境変数グループ識別子を指定してください。
- -o と-O に指定するファイルで設定する環境変数は同じ内容にしてください。

備考

OpenTP1 のユーザサービス定義の trnrmid オペランドで，ユーザサービスから接続する HiRDB を選択します。HiRDB1 と HiRDB2 に接続する例を次に示します。

```
trnrmid -n HiRDB_DB_SERVER -i H1,H2
```

(b) TPBroker for C++の場合

TPBroker for C++のリソースマネジャ定義の xa_open_string_info オペランドでオープン文字列を登録します。ここでは二つの HiRDB を TPBroker for C++に登録します。登録条件は次のとおりとします。

リソースマネジャ	環境変数グループ識別子	環境変数設定ファイル名
HiRDB1	HDB1	/usr/conf/HiRDB/HiRDB11.ini /usr/conf/HiRDB/HiRDB12.ini
HiRDB2	HDB2	/usr/conf/HiRDB/HiRDB21.ini /usr/conf/HiRDB/HiRDB22.ini

オープン文字列の登録例を次に示します。

```
tsdefvalue /OTS/RM/HiRDB_DB_SERVER_1/DMN/xa_open_string_info     1
-s "HDB1*/usr/conf/HiRDB/HiRDB11.ini"
tsdefvalue /OTS/RM/HiRDB_DB_SERVER_1/xa_open_string_info         2
-s "HDB1+/usr/conf/HiRDB/HiRDB12.ini"

tsdefvalue /OTS/RM/HiRDB_DB_SERVER_2/DMN/xa_open_string_info     1
-s "HDB2*/usr/conf/HiRDB/HiRDB21.ini"
tsdefvalue /OTS/RM/HiRDB_DB_SERVER_2/xa_open_string_info         2
-s "HDB2+/usr/conf/HiRDB/HiRDB22.ini"
```

〔説明〕

1. /OTS/RM/RM 名/DMN/xa_open_string_info には，TPBroker for C++の回復プロセスが使用するオープン文字列を指定します。環境変数グループ識別子と環境変数設定ファイル名の間の文字には＊を指定してください。
2. /OTS/RM/RM 名/xa_open_string_info には，アプリケーションプログラムプロセス及び決着プロセスが使用するオープン文字列を指定します。環境変数グループ識別子と環境変数設定ファイル名の間の文字には＋を指定してください。

- RM 名が同じ場合は，同じ環境変数グループ識別子を指定してください。
- RM 名が同じ場合は，各環境変数設定ファイルに設定する環境変数を同じ内容にしてください。
- 決着プロセスに対して環境変数 TPRMINFO を設定している場合，/OTS/RM/RM 名/（TPRMINFO 設定値）/xa_open_string_info に指定するオープン文字列には/OTS/RM/RM 名/xa_open_string_info と同じ文字列を指定してください。また，決着プロセスに対して TPRMINFO を設定しない場合でも，複数接続機能を使用するときは/OTS/completion_process_env にデフォルトとして'TPRMINFO='を指定してください。指定例を次に示します。

（例）tsdefvalue /OTS completion_process_env -a 'TPRMINFO='

(c) TP1/EE の場合

TP1/EE のトランザクション関連定義の trnstring オペランドでオープン文字列を登録します。ここでは二つの HiRDB を TP1/EE に登録します。登録条件は次のとおりとします。

リソースマネジャ	環境変数グループ識別子	環境変数設定ファイル名
HiRDB1	HDB1	/usr/conf/HiRDB/HiRDB11.ini /usr/conf/HiRDB/HiRDB12.ini
HiRDB2	HDB2	/usr/conf/HiRDB/HiRDB21.ini /usr/conf/HiRDB/HiRDB22.ini

オープン文字列の登録例を次に示します。

```
trnstring -n HiRDB_DB_SERVER -i H1 -o ”HDB1*/usr/conf/HiRDB/HiRDB11.ini” ¥
    -O ”HDB1+/usr/conf/HiRDB/HiRDB12.ini”
trnstring -n HiRDB_DB_SERVER -i H2 -o ”HDB2*/usr/conf/HiRDB/HiRDB21.ini” ¥
    -O ”HDB2+/usr/conf/HiRDB/HiRDB22.ini”
```

〔説明〕

-n：リソースマネジャ名を指定します。

-i：リソースマネジャ拡張子を指定します。

-o：

TP1/EE の回復スレッド及び監視スレッドが使用するオープン文字列を指定します。

"環境変数グループ識別子＊環境変数設定ファイル名"の形式で指定します。

-O：

処理スレッドが使用するオープン文字列を指定します。

"環境変数グループ識別子＋環境変数設定ファイル名"の形式で指定します。

- -o と-O には同じ環境変数グループ識別子を指定してください。
- -o と-O に指定するファイルで設定する環境変数は同じ内容にしてください。

(4) クローズ文字列

トランザクションマネジャが xa_close でリソースマネジャをクローズするときに使用するクローズ文字列は指定不要です。

(5) RM 関連オブジェクト名

RM 関連オブジェクト名には，コンパイル，及びリンケージをするときに指定するライブラリのファイル名を指定します。指定するファイル名については，次の表を参照してください。

- マニュアル「HiRDB UAP 開発ガイド」の「コンパイル，リンケージ時に指定するライブラリ」の表「コンパイル，及びリンケージをするときに指定するライブラリ（OLTP 下の場合（UNIX 環境））」
- マニュアル「HiRDB UAP 開発ガイド」の「UNIX クライアントのディレクトリ及びファイル構成」の表「各トランザクションマネジャが使用するライブラリ一覧（UNIX クライアント）」

(6) クライアント環境定義

トランザクションマネジャに HiRDB のトランザクション処理を制御させるためには，HiRDB のクライアント環境定義をトランザクションマネジャの定義に設定する必要があります。OLTP 環境下でのクライアント環境定義の設定方法については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(a) OpenTP1 の場合

トランザクションマネジャが OpenTP1 の場合，クライアント環境定義を次に示す OpenTP1 のシステム定義に指定する必要があります。

- システム環境定義
- ユーザサービスデフォルト定義
- ユーザサービス定義
- トランザクションサービス定義

これらの定義については，マニュアル「OpenTP1 システム定義」を参照してください。

なお，複数の OpenTP1 と接続する場合は，次に示すクライアント環境定義を必ず指定してください。

- HiRDB_PDTMID 又は PDTMID

(b) TPBroker for C++の場合

クライアント環境定義は TPBroker for C++のシステム定義に指定してください。

(c) TUXEDO の場合

TUXEDO コンフィギュレーション・ファイル（UBBCONFIG ファイル）の ENVFILE パラメタで指定したファイルに，クライアント環境定義を指定してください。TUXEDO コンフィギュレーション・ファイルについては，TUXEDO のマニュアルを参照してください。

(d) TP1/EE の場合

トランザクションマネジャが TP1/EE の場合，クライアント環境定義を，次に示す TP1/EE を実行する環境の OpenTP1 のシステム定義に指定する必要があります。

- ユーザサービスデフォルト定義
- ユーザサービス定義

これらの定義については，マニュアル「OpenTP1 システム定義」及び「分散トランザクション処理機能 TP1/Server Base Enterprise Option 使用の手引」を参照してください。

なお，複数の TP1/EE と接続する場合は，次に示すクライアント環境定義を必ず指定してください。

- PDTMID

6.2.7　トランザクションマネジャへの登録例

(1)　OpenTP1 の場合

HiRDB を OpenTP1 に登録するには，OpenTP1 の trnlnkrm コマンドを使用します。trnlnkrm コマンドの指定例を次に示します。

(a)　動的登録の場合

```
trnlnkrm -a HiRDB_DB_SERVER -s pdtxa_switch -o /HiRDB/client/lib/libzclty.sl
```

［説明］

-a：RM 名を指定します。

-s：RM スイッチ名（XA スイッチ構造体の名称）を指定します。RM スイッチ名は，登録方法（動的登録又は静的登録）によって異なります。

-o：RM 関連オブジェクト名（共用ライブラリのファイル名）を指定します。

(b)　静的登録の場合

```
trnlnkrm -a HiRDB_DB_SERVER -s pdtxa_switch_y -o /HiRDB/client/lib/libzclty.sl
```

［説明］

-a：RM 名を指定します。

-s：RM スイッチ名（XA スイッチ構造体の名称）を指定します。RM スイッチ名は，登録方法（動的登録又は静的登録）によって異なります。

-o：RM 関連オブジェクト名（共用ライブラリのファイル名）を指定します。

(2)　TPBroker for C++の場合

HiRDB を TPBroker for C++に登録するには，TPBroker for C++の tslnkrm コマンドを使用します。tslnkrm コマンドの指定例を次に示します。

(a)　動的登録の場合

```
tslnkrm -a HiRDB_DB_SERVER_1 -s pdtxa_switch -o '/HiRDB/client/lib/libzcltyk.sl'
-r -m
```

```
tslnkrm -a HiRDB_DB_SERVER_2 -s pdtxa_switch -o '/HiRDB/client/lib/libzcltyk.sl'
-r -m
```

〔説明〕

-a：RM 名を指定します。

-s：RM スイッチ名（XA スイッチ構造体の名称）を指定します。RM スイッチ名は，登録方法（動的登録又は静的登録）によって異なります。

-o：RM 関連オブジェクト名（共用ライブラリのファイル名）を指定します。

-r：動的登録する場合に指定します。

-m：OTS のデーモンがマルチスレッドで動作するようになります。

(b) 静的登録の場合

```
tslnkrm -a HiRDB_DB_SERVER_1 -s pdtxa_switch_y -o '/HiRDB/client/lib/libzcltyk.sl'
-r -m
tslnkrm -a HiRDB_DB_SERVER_2 -s pdtxa_switch_y -o '/HiRDB/client/lib/libzcltyk.sl'
-r -m
```

〔説明〕

-a：RM 名を指定します。

-s：RM スイッチ名（XA スイッチ構造体の名称）を指定します。RM スイッチ名は，登録方法（動的登録又は静的登録）によって異なります。

-o：RM 関連オブジェクト名（共用ライブラリのファイル名）を指定します。

-r：静的登録する場合に指定します。

-m：OTS のデーモンがマルチスレッドで動作するようになります。

(3) TUXEDO の場合

$TUXDIR/udataobj/RM ファイルで HiRDB を TUXEDO に登録します。$TUXDIR は，TUXEDO システム・ソフトウェアがあるディレクトリの絶対パス名を示しています。RM ファイルの指定例を次に示します。

(a) 動的登録の場合

```
HiRDB_DB_SERVER:pdtxa_switch:-L/HiRDB/client/lib -lzcltys
```

(b) 静的登録の場合

```
HiRDB_DB_SERVER:pdtxa_switch_y:-L/HiRDB/client/lib -lzcltys
```

(4) TP1/EE の場合

HiRDB を TP1/EE に登録するには，TP1/EE の eetrnmkobj コマンドを使用します。eetrnmkobj コマンドの指定例を次に示します。

(a) 動的登録の場合

```
eetrnmkobj -r HiRDB_DB_SERVER -o seigyo -s pdtxa_switch     ¥
-O /HiRDB/client/lib/libzcltyk.sl -i /HiRDB/include
```

〔説明〕

-r：RM 名を指定します。

-o：リソースマネジャ連携オブジェクト名を指定します。

-s：RM スイッチ名（XA スイッチ構造体の名称）を指定します。RM スイッチ名は，登録方法（動的登録又は静的登録）によって異なります。

-O：RM 関連オブジェクト名（共用ライブラリのファイル名）を指定します。

-i：HiRDB 提供ヘッダのパスを指定します。

(b) 静的登録の場合

```
eetrnmkobj -r HiRDB_DB_SERVER -o seigyo -s pdtxa_switch_y   ¥
-O /HiRDB/client/lib/libzcltyk.sl -i /HiRDB/include
```

〔説明〕

-r：RM 名を指定します。

-o：リソースマネジャ連携オブジェクト名を指定します。

-s：RM スイッチ名（XA スイッチ構造体の名称）を指定します。RM スイッチ名は，登録方法（動的登録又は静的登録）によって異なります。

-O：RM 関連オブジェクト名（共用ライブラリのファイル名）を指定します。

-i：HiRDB 提供ヘッダのパスを指定します。

6.2.8 トランザクションマネジャへの登録の変更

トランザクションマネジャへの登録を変更する場合（静的登録から動的登録，若しくは動的登録から静的登録），又は RM 関連オブジェクト名に指定するライブラリを変更する場合には，次に示す手順に従ってトランザクションマネジャに HiRDB を登録し直してください。

(1) OpenTP1 の場合

〈手順〉

1. OpenTP1 の trnlnkrm コマンドで，トランザクションマネジャに HiRDB を登録し直します。

2. OpenTP1 の trnmkobj コマンドで，トランザクション制御用オブジェクトファイルを再作成します。
3. 2 で再作成したトランザクション制御用オブジェクトファイル，及び「トランザクションマネジャに登録する情報」に示した情報を基に，HiRDB の XA ライブラリとリンクしていたすべての UAP を再リンケージしてください。再リンケージをしないと，UAP の動作を保証できません。

(2) TPBroker for C++の場合

〈手順〉

1. TPBroker for C++の tslnkrm コマンドで，トランザクションマネジャに HiRDB を登録し直します。
2. TPBroker for C++の tsmkobj コマンドで，トランザクション制御用オブジェクトファイルを再作成します。
3. 2 で再作成したトランザクション制御用オブジェクトファイル，及び「トランザクションマネジャに登録する情報」に示した情報を基に，HiRDB の XA ライブラリとリンクしていたすべての UAP を再リンケージしてください。再リンケージをしないと，UAP の動作を保証できません。

(3) TUXEDO の場合

〈手順〉

1. $TUXDIR/udataobj/RM でトランザクションマネジャに HiRDB を登録し直します。
2. TUXEDO の buildtms コマンドで，「トランザクションマネジャに登録する情報」に示した情報を基にトランザクションマネジャサーバのロードモジュールを再作成します。
3. TUXEDO の buildserver コマンドで，「トランザクションマネジャに登録する情報」に示した情報を基にサーバのロードモジュールを再作成します。
4. TUXEDO の buildclient コマンドで，「トランザクションマネジャに登録する情報」に示した情報を基にクライアントモジュールを再作成します。

(4) TP1/EE の場合

〈手順〉

1. TP1/EE の eetrnmkobj コマンドで，トランザクションマネジャに HiRDB を登録し直します。
2. TP1/EE の eetrnmkobj コマンドで，リソースマネジャ連携オブジェクトファイルを再作成します。
3. 2 で再作成したリソースマネジャ連携オブジェクトファイル，及び「トランザクションマネジャに登録する情報」に示した情報を基に，HiRDB の XA ライブラリとリンクしていたすべての UAP を再リンケージしてください。再リンケージをしないと，UAP の動作を保証できません。

6.2.9 トランザクションマネジャと HiRDB 間のコネクションが切断されたときの再接続方法

(1) UAP で対処する方法

コネクションが切断された場合，実行中の UAP を終了後，再起動してください。再起動すると，自動的にコネクションが再接続されます。

UAP を再起動したくない場合は，コネクションが切断されたことを示すエラーが UAP に返ったときに，tx_open 関数を再発行してください。そうすれば，UAP を終了しなくてもサービスを続行できます。tx_open 関数を再発行するときのコーディング例を次に示します。

コーディング例

```
int connection = 1;
void service(char *in_data,long *in_len,char *out_data,long *out_len) {
    if (connection == 0) {
       tx_close();
       tx_open();           ……コネクション切断時のtx_open再発行処理
    }
    tx_begin();
    EXEC SQL INSERT INTO ......;                ……… SQL文発行
    if (SQLCODE == 0) {
        tx_commit();
        *out_data = “OK” ;
    } else {
        tx_rollback();
        *out_data = “NG” ;
        if (SQLCODE == -563 || SQLCODE == -722) {
            connection = 0;              ……コネクション切断を記憶
        }
    }
}
```

(2) 連携する OLTP 製品が TPBroker for C++の場合

トランザクションの開始又は終了時に HiRDB とのコネクションを確立又は切断するため，途中で切断した場合も次回のトランザクション開始時にコネクションが再接続されます。

(3) OpenTP1 の機能を使用する

動的登録の場合は，OpenTP1/Server Base の trn_rm_open_close_scope オペランドに transaction を指定してください。そうすれば，OpenTP1/Server Base はトランザクションの開始又は終了で HiRDB とのコネクションを確立又は切断します。したがって，途中でコネクションが切断されても，次回のトランザクションの開始時にコネクションが再接続されます。

静的登録の場合は，トランザクションの開始時に HiRDB とのコネクションが確立されているかどうかをトランザクションマネジャが確認します。コネクションが切断されている場合は，自動的に再接続されて，

トランザクションを開始します。なお，TP1/EE の場合は，HiRDB のクライアント環境定義 PDXAAUTORECONNECT に YES を指定する必要があります。

(4) HiRDB の XA インタフェースに対応したクライアントライブラリの再接続

トランザクションマネジャでトランザクションを開始して，最初に HiRDB にアクセスする SQL 文を実行するまでに，HiRDB とのコネクションが切断されていた場合，SQL 文の実行時に HiRDB クライアントライブラリでコネクションが再接続されます。ただし，ネットワーク障害による切断は検知できないため，再接続されません。

6.2.10 TP1/Resource Manager Monitor の機能を使用した HiRDB の監視

OLTP に OpenTP1 を使う場合に，TP1/Resource Manager Monitor（リソースマネジャモニタ：RMM）を使用した運用するときの HiRDB に関する注意について説明します。リソースマネジャモニタの運用方法については，マニュアル「OpenTP1 運用と操作」を参照してください。

(1) 監視対象プロセス ID 取得コマンド作成時の注意

監視対象プロセス ID 取得コマンド作成時，監視対象とするプロセスを指定する必要があります。HiRDB の場合は，監視対象とするプロセスに「_scd」と指定してください。次に RMM サービスで用意している HiRDB 用のコマンド（シェルスクリプト）での指定例を示します。

```
#Watched Processes
PROCESSES=”_scd”
```

また，このコマンドの確実性を高めるために，次に示す記述を追加することをお勧めします。この記述を追加すると，HiRDB 開始完了前の不確かな監視対象プロセス ID を取得しなくなります。この記述はシェルスクリプト中のプロセス ID 取得部分の前（コメント行「#These Lines Are The Description Of Get Process_ID Process」の前）に追加してください。

```
#System status check
get_STATUS=’$PDDIR/bin/pdls 2>/dev/null | ¥
        /usr/bin/awk’{print$4}’ | /bin/grep -v STATUS’
for i in $get_STATUS
do
  if[$i!=”ACTIVE”]
  then
    exit 2
  fi
done
```

(2) RMM での運用を HiRDB/パラレルサーバに適用するときの注意

HiRDB/パラレルサーバの場合，システムマネジャがあるユニットと同一ホストにある OpenTP1 だけに，監視対象 RM 定義で一連の監視対象リソースマネジャ用コマンドの指定をしてください。

6.2.11　注意事項

(1)　SQL 関連の注意事項

1. リソースマネジャへの接続や切断を行う権限は，トランザクションマネジャにあります。UAP にリソースマネジャへの接続又は切断を行う SQL 文を記述しないでください。また，トランザクションの進行を調整し監視する権限も，トランザクションマネジャにあります。UAP にトランザクションをロールバック又はコミットする SQL 文を記述しないでください。したがって，EXEC SQL COMMIT WORK 文，EXEC COMMIT WORK RELEASE 文などもエラーになります。
2. 定義系 SQL 文はエラーとなります。CREATE TABLE などの定義系 SQL 文は自動的にコミットを指示するため，UAP に定義系 SQL 文を記述しないでください。

(2)　マルチスレッド用のライブラリに関する注意事項

一つのトランザクションから HiRDB サーバに対して複数のスレッドを使用して別々に接続できません。マルチスレッド環境であってもトランザクションから接続する HiRDB サーバのサーバプロセスは一つです。したがって，一つのトランザクションから同時に実行できるスレッドは一つであり，同一トランザクション内で複数のスレッドを使用して SQL 文を同時に実行できません。

(3)　一相最適化に関する注意事項

HiRDB はトランザクションマネジャがサポートしている一相最適化に対応しています。トランザクションマネジャはトランザクションブランチによって変更した共有リソースが HiRDB だけの場合，トランザクションブランチに対して一相コミットを要求できます。トランザクションマネジャが一相最適化を使用して一相コミットを要求してきた場合，HiRDB はトランザクションブランチの結果を決めた後，トランザクションブランチの情報を削除してトランザクションマネジャへ応答を返します。

一相最適化を使用しているトランザクションマネジャでは，グローバルトランザクションに関して安定ストレージに記憶する必要もなく，何らかの障害が発生してもその結果を知る必要もありません。したがって，次に示す条件をすべて満たす場合，トランザクションの完了種別がトランザクションマネジャと HiRDB で一致しないことがあります。

- 一相最適化を使用するトランザクションマネジャと XA インタフェースを使用して接続している
- 更新系トランザクションのコミットメント制御をトランザクションマネジャが一相最適化している
- コミットメント処理中にトランザクションマネジャの UAP が異常終了する

これらの条件下では，HiRDB のトランザクションブランチの結果を，トランザクションマネジャが発生した障害の結果として知ることができません。そのため，トランザクションの完了種別がトランザクションマネジャと HiRDB で一致しないことがあります。

これを防ぐには，更新系トランザクションのコミットメント制御をする場合，トランザクションマネジャの一相最適化を使用しないでください。

(4) 高速系切り替え機能を使用している場合の注意事項

次に示す条件をすべて満たす場合は注意が必要です。

- HiRDB/パラレルサーバの場合はシステムマネジャがあるユニットを高速系切り替え機能の対象にしている
- X/Open に従った API を使用した OLTP 製品（OpenTP1 又は TPBroker for C++など）と連携している
- HiRDB クライアントのバージョンが 06-02-/A 以前である
- OLTP 製品のクライアント環境変数 PDHOST に指定している現用系が待機系として待機完了状態になっている

この場合，OLTP 製品が未決着トランザクションの回復処理をすると，X/Open に従った API がエラーリターンしてトランザクションが回復されないことがあります。この現象が発生する場合は，HiRDB クライアントのバージョンを 06-02-/B 以降にバージョンアップしてください。業務を停止させたくないなどの理由で HiRDB クライアントのバージョンアップがすぐにできない場合は，現用系の HiRDB（ユニット）を待機系から実行系に系を切り替えてください。ただし，これは一時的な対応策です。HiRDB クライアントのバージョンアップで対応してください。

6.3　インナレプリカ機能との連携

インナレプリカ機能を使用すると，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。インナレプリカ機能を使用するには，HiRDB Staticizer Option が必要になります。

6.3.1　HiRDB システム定義の指定

HiRDB システム定義で次に示すオペランドを指定します。

- pd_inner_replica_control オペランド
  インナレプリカグループの最大数を指定します。
- pd_inner_replica_lock_shift オペランド
  インナレプリカ機能使用時に，UAP とコマンドを同時実行させるかどうかを指定します。
- pd_lv_mirror_use オペランド
  レプリカ RD エリアのオープン属性を SCHEDULE にするかどうかを指定します。

インナレプリカ機能を使用して更新可能なオンライン再編成をする場合，次のオペランドを指定します。

- pd_max_reflect_process_count オペランド
  追い付き反映処理時に確保する pdorend 反映プロセス数を指定します。このオペランドを省略すると更新可能なオンライン再編成はできません。
- pd_log_org_reflected_logpoint オペランド
  すべての更新ログの追い付き反映処理が完了したシステムログファイルの状態を変更するかどうかを指定します。
- pd_log_org_no_standby_file_opr オペランド
  全システムログファイルがオンライン再編成上書き禁止状態の場合に，システムログファイルのスワップが発生したときの HiRDB の処理を指定します。

前記のオペランドのほかに，次に示すオペランドの指定値を見直してください。

- pd_max_rdarea_no
- pd_max_file_no
- pd_assurance_index_no

6.3.2　環境設定方法

インナレプリカ機能を使用する場合のシステムの環境設定及び運用方法については，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。

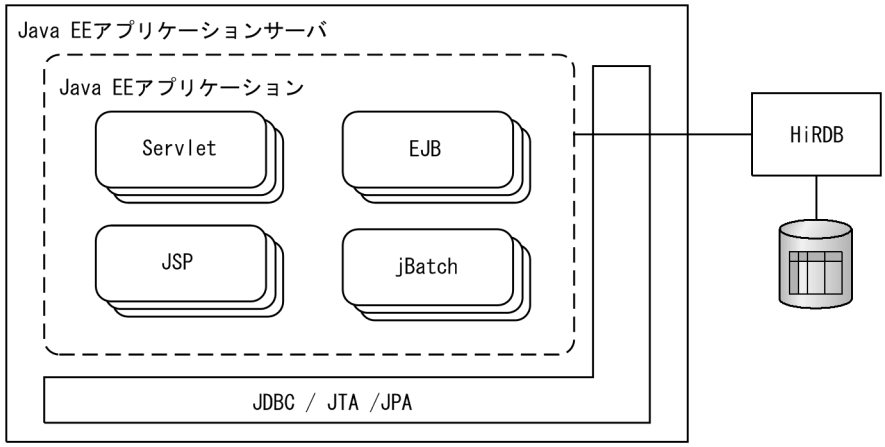
6.4 Java EE アプリケーションサーバとの連携

6.4.1 Java EE アプリケーションサーバとは

Java EE とは，Java で大規模な企業システムを構築する場合に使用される規格であり，その実行環境を提供する製品を Java EE アプリケーションサーバと呼びます。

Java EE は幾つかの標準仕様の集合として構成されているため，ユーザアプリケーションは Servlet/JSP/EJB/jBatch などを実装した Java EE アプリケーションとして作成します。JDBC/JTA/JPA などを介してデータベースと連携します。

図 6-7 Java EE アプリケーションサーバでの構成イメージ



6.4.2 連携できる Java EE アプリケーションサーバ

HiRDB では，次に示す Java EE アプリケーションサーバと連携できます。

- Cosminexus
- JBoss Enterprise Application Platform

(1) Cosminexus

Cosminexus をサポートしている JDBC ドライバの種別を次の表に示します。

表 6-5 Cosminexus をサポートしている JDBC ドライバの種別

JDBC ドライバ種別		サポート
Type2 JDBC ドライバ		×
Type4 JDBC ドライバ	JDBC2.0 版	○
	JDBC4.0 版	○

（凡例）

○：サポート

×：未サポート

(a) 設定方法

設定方法については Cosminexus のマニュアルを参照してください。

(b) Cosminexus 連携時に考慮が必要になる機能

●各種トレースファイル，監査証跡ファイル

・トレースファイル名

システムプロパティ ejbserver.serverName によって指定される J2EE サーバのサーバ名からトレースファイル名が決まります。トレースファイル名を次の表に示します。

表 6-6　トレースファイルの種類とトレースファイル名

トレースファイルの種類	トレースファイル名
Exception トレースログ	<サーバ名>exc{1|2}.trc
不正電文トレース	<サーバ名>dataerr{1|2}.trc

・トレースファイルの出力先

JDBC ドライバが出力する各種トレースファイルは，クライアント環境定義 PDCLTPATH などで出力先を明示的に指定していない場合，J2EE サーバのカレントディレクトリに出力します。具体的な出力先を次に示します。

- J2EE サーバの作業ディレクトリがデフォルトの場合
  <インストール先ディレクトリ>/CC/Server/public/ejb/<サーバ名>
- J2EE サーバの作業ディレクトリを変更している場合
  <作業ディレクトリ>/ejb/<サーバ名>

J2EE サーバのカレントディレクトリについて，詳細は Cosminexus のマニュアルを参照してください。

・ルートアプリケーション情報

ルートアプリケーション情報とは Cosminexus の PRF トレースに出力される情報であり，Cosminexus を呼び出すクライアントからの個々のリクエストを識別するために使用できます。HiRDB では次のファイルで出力しているため，ルートアプリケーション情報によって Cosminexus の PRF トレースと関連づけることで，どのリクエストの延長で出力された情報であるかを判別できます。

- 監査証跡ファイル
- PRF トレース
- SQL トレース

- Exception トレースログ

・再接続トレース機能

自動再接続機能で再接続された場合，再接続トレース機能を使用すると，Cosminexus の PRF トレース機能で出力されるトレース中の接続情報を追跡できます。

詳細はマニュアル「HiRDB UAP 開発ガイド」の「再接続トレース機能」を参照してください。

●クライアント環境定義

・UAP 名称

クライアント環境定義 PDCLTAPNAME を省略した場合，システムプロパティ ejbserver.serverName によって J2EE サーバのサーバ名から UAP 名称が決まります。UAP 名称を次の表に示します。なお，PDCLTAPNAME 以外の UAP 名称の指定方法については，マニュアル「HiRDB UAP 開発ガイド」の「接続情報の優先順位」を参照してください。

表 6-7　JDBC ドライバと UAP 名称

JDBC ドライバ	UAP 名称
JDBC2.0 版	JDBC20_<サーバ名>
JDBC4.0 版	JDBC40_<サーバ名>

(c)　注意事項

- クライアント環境定義 PDUSER で指定したユーザ名／パスワードは無効になります。ユーザ名／パスワードは，Cosminexus の Connector 属性ファイルのプロパティ値として設定してください。
- 使用する JDBC ドライバを DABroker for Java から Type4 JDBC ドライバへ移行する場合，UAP の修正が必要になることがあります。詳細はマニュアル「HiRDB UAP 開発ガイド」の「DABroker for Java からの移行」を参照してください。

(2)　JBoss Enterprise Application Platform

次に示すバージョンの JBoss Enterprise Application Platform（以降，JBoss と表記）と連携できます。

- JBoss Enterprise Application Platform 7.0.1～7.1

JBoss をサポートしている JDBC ドライバの種別を次の表に示します。

表 6-8　JBoss をサポートしている JDBC ドライバの種別

JDBC ドライバ種別		サポート
Type2 JDBC ドライバ		×
Type4 JDBC ドライバ	JDBC2.0 版	×
	JDBC4.0 版	○

（凡例）

○：サポート

×：未サポート

(a) 設定方法

JDBC ドライバを使用するまでの手順を次に示します。

1. JDBC ドライバをコアモジュールとして追加
2. JDBC ドライバの登録
3. データソースの追加

2,3 については，管理 CLI 又は管理コンソールを使用します。管理 CLI を使用した場合の手順を次に示します。

1. JDBC ドライバをコアモジュールとして追加

モジュールとは，JBoss でのクラスローディング及び依存関係管理に使用されるクラスの論理グループのことを指します。JDBC ドライバにクラスパスを通すために，JDBC ドライバ（pdjdbc4.jar）をコアモジュールとして追加します。

モジュール名を com.hirdb とした場合の設定例を次に示します。

次のディレクトリを作成してください。

<JBoss インストールディレクトリ>/modules/com/hirdb/main

作成したディレクトリに，次に示す module.xml ファイルと JDBC ドライバを格納してください。

```
<?xml version="1.0" ?>
<module xmlns="urn:jboss:module:1.1" name="com.hirdb">
    <resources>
        <resource-root path="pdjdbc4.jar"/>
    </resources>
    <dependencies>
        <module name="javax.api"/>
        <module name="javax.transaction.api"/>
    </dependencies>
</module>
```

2. JDBC ドライバの登録

管理 CLI コマンドで次のコマンドを実行してください。

```
/subsystem=datasources/jdbc-driver=<JDBCドライバ名>:
add(driver-name=<JDBCドライバ名>,driver-module-name=<モジュール名>,
driver-xa-datasource-class-name=JP.co.Hitachi.soft.HiRDB.JDBC.PrdbXADataSource,
driver-class-name=JP.co.Hitachi.soft.HiRDB.JDBC.HiRDBDriver)
```

JDBC ドライバ登録時の設定項目を次の表に示します。

表 6-9　JDBC ドライバ登録時の設定項目

項目	概要
JDBC ドライバ名	JDBC ドライバの識別子
モジュール名	1.で追加したモジュールのモジュール名

JDBC ドライバの登録例を次に示します。

```
/subsystem=datasources/jdbc-driver=hirdb:
add(driver-name=hirdb,driver-module-name=com.hirdb,
driver-xa-datasource-class-name=JP.co.Hitachi.soft.HiRDB.JDBC.PrdbXADataSource,
driver-class-name=JP.co.Hitachi.soft.HiRDB.JDBC.HiRDBDriver)
```

3. データソースの追加

管理 CLI コマンドで次のコマンドを実行してください。

- 非 XA データソース

```
data-source add --name=<データソース名> --jndi-name=<JNDI名>
--driver-name=<JDBCドライバ名> --connection-url=<接続URL>
--user-name=<ユーザ名> --password=<パスワード>
--valid-connection-checker-class-name=
org.jboss.jca.adapters.jdbc.extensions.novendor.JDBC4ValidConnectionChecker
--exception-sorter-class-name=
org.jboss.jca.adapters.jdbc.extensions.novendor.ListExceptionSorter
--exception-sorter-properties=FatalExceptions=
"563,720,722,723,728,732,735,932,1700"※1
--validate-on-match=true※2
```

- XA データソース

```
xa-data-source add --name=<データソース名> --jndi-name=<JNDI名>
--driver-name=<JDBCドライバ名> --user-name=<ユーザ名> --password=<パスワード>
--valid-connection-checker-class-name=
org.jboss.jca.adapters.jdbc.extensions.novendor.JDBC4ValidConnectionChecker
--xa-datasource-properties={"description"=>"<環境変数グループ識別子>",
"XAOpenString"=>"<XAオープン文字列>"}
--validate-on-match=true※2
```

注※1

--exception-sorter-class-name に指定している ListExceptionSorter は，発生した例外の SQLCODE の絶対値が--exception-sorter-properties に指定されている場合，致命的な例外と判定しコネクションを破棄します。

注※2

コネクションプールからのコネクションオブジェクト取得時に，取得したコネクションオブジェクトが有効であるかを検証します。

データソース追加時の設定項目を次の表に示します。

表 6-10　データソース追加時の設定項目

項目	概要
データソース名	データソースの識別子
JNDI 名	データソースの lookup に使用する JNDI 名
JDBC ドライバ名	表「JDBC ドライバ登録時の設定項目」の JDBC ドライバ名
ユーザ名	ユーザ名／パスワード（他で設定している場合は省略できます）
パスワード	
接続 URL	DriverManager クラスの getConnection メソッドに指定する URL
環境変数グループ識別子	XADataSource クラスの setDescription メソッドに指定する HiRDB の環境変数グループ識別子
XA オープン文字列	XADataSource クラスの setXAOpenString メソッドに指定する XA オープン文字列

ユーザ名，パスワード以外は必須項目になります。また，表に記載していない項目については，必要に応じて追加してください。

データソースの追加例を次に示します。

- 非 XA データソース

```
data-source add --name=HiRDBDS --jndi-name=java:jboss/HiRDBDS --driver-name=hirdb
--connection-url=jdbc:hitachi:hirdb://DBID=22200,DBHOST=localhost
--user-name=admin --password=admin
--valid-connection-checker-class-name=
org.jboss.jca.adapters.jdbc.extensions.novendor.JDBC4ValidConnectionChecker
--exception-sorter-class-name=
org.jboss.jca.adapters.jdbc.extensions.novendor.ListExceptionSorter
--exception-sorter-properties=FatalExceptions=
"563,720,722,723,728,732,735,932,1700"
--validate-on-match=true
```

- XA データソース

```
xa-data-source add --name=HiRDBXADS --jndi-name=java:jboss/HiRDBXADS
--driver-name=hirdb --user-name=admin --password=admin
--valid-connection-checker-class-name=
org.jboss.jca.adapters.jdbc.extensions.novendor.JDBC4ValidConnectionChecker
--xa-datasource-properties=
{"description"=>"HDB1","XAOpenString"=>"HDB1+/home/hirdb/HiRDB.ini"}
--validate-on-match=true
```

(b)　注意事項

JBoss で使用できない機能を次の表に示します。

表 6-11　JBoss で使用できない機能

使用できない機能		概要	対策
HiRDB を格納先とするトランザクションオブジェクトストア		JBoss がトランザクションログを格納するための場所	トランザクションオブジェクトストアとして次を使用してください。 • ファイルストア（デフォルト） • ほかのデータベースの JDBC データソース
データソースのパラメタ	connectable=true (未設定とするか，false を設定してください)	Commit Markable Resource（非 XA データソースを XA トランザクションに参加させるための機能）	XADataSource を使用してください。
	share-prepared-statements=true (未設定とするか，false を設定してください)	前処理済みのステートメントをクローズしないで，再度同一 SQL 文を指定して前処理済みのステートメントを取得した場合に，キャッシュされた同一のステートメントを取得する	なし (2 つ目のステートメントは新規に作成されます)
	interleaving=true (未設定とするか，false を設定してください)	同一コネクション上で複数の XA トランザクションを切り替えて使用する	なし (XA トランザクションが完了するまで，コネクションを占有します)

6.4.3　Java EE アプリケーションサーバの機能

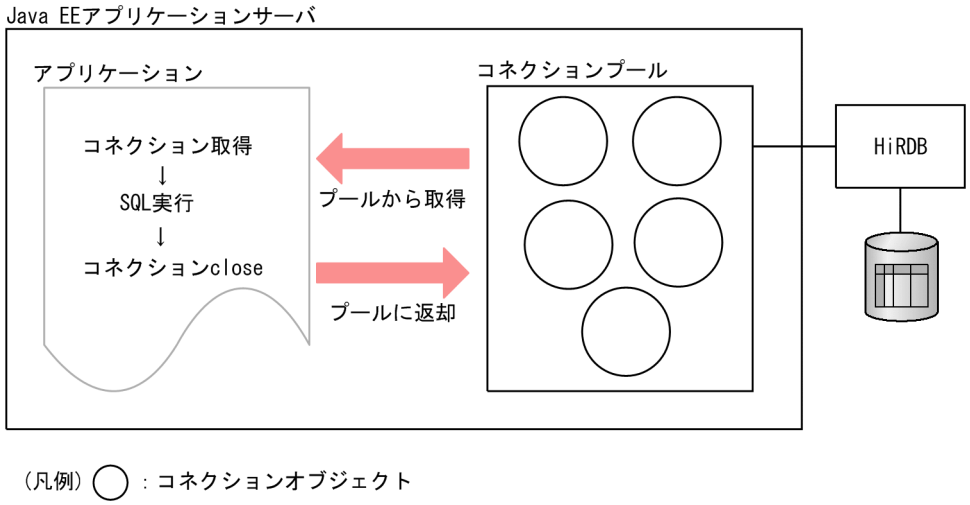
Java EE アプリケーションサーバが提供する一般的な機能について説明します。

(1)　コネクションプーリング

(a)　コネクションプーリングとは

ユーザアプリケーションを呼び出す度にデータベースと接続/切断するのではなく，接続済みのコネクションオブジェクトをプールして使い回す仕組みのことを，コネクションプーリングと呼びます。

図 6-8　コネクションプーリング



（凡例）◯：コネクションオブジェクト

(b)　コネクションプーリング使用時の注意事項

- クライアント環境定義などの JDBC ドライバの設定を変更しても無効となります。設定変更を反映させるには，Java EE アプリケーションサーバの再起動やコネクションプール内のコネクションオブジェクトを破棄するなどして，再接続する必要があります。具体的な方法については，各 Java EE アプリケーションサーバのマニュアルを参照してください。
- HiRDB サーバと常時接続している UAP では，クライアント環境定義 PDSWATCHTIME でのタイムアウトによって，接続が切断されることがあります。クライアント環境定義 PDSWATCHTIME を設定する場合の注意事項については，マニュアル「HiRDB UAP 開発ガイド」の「クライアント環境定義の設定内容」の PDSWATCHTIME を参照してください。
- ステートメントの close メソッド実行時に SQL 前処理を破棄するように設定している場合，ステートメントの close 後にトランザクションを決着する必要があります。詳細はマニュアル「HiRDB UAP 開発ガイド」の次の個所を参照してください。

  ステートメントの close メソッド実行時に SQL 前処理を破棄する方法

  「接続情報の優先順位」の「ステートメントの close メソッド実行時の SQL 前処理の破棄」

  ステートメントの close 後にトランザクションを決着する方法

  「ユーザプロパティ」の「HiRDB_for_Java_STATEMENT_CLOSE_BEHAVIOR」

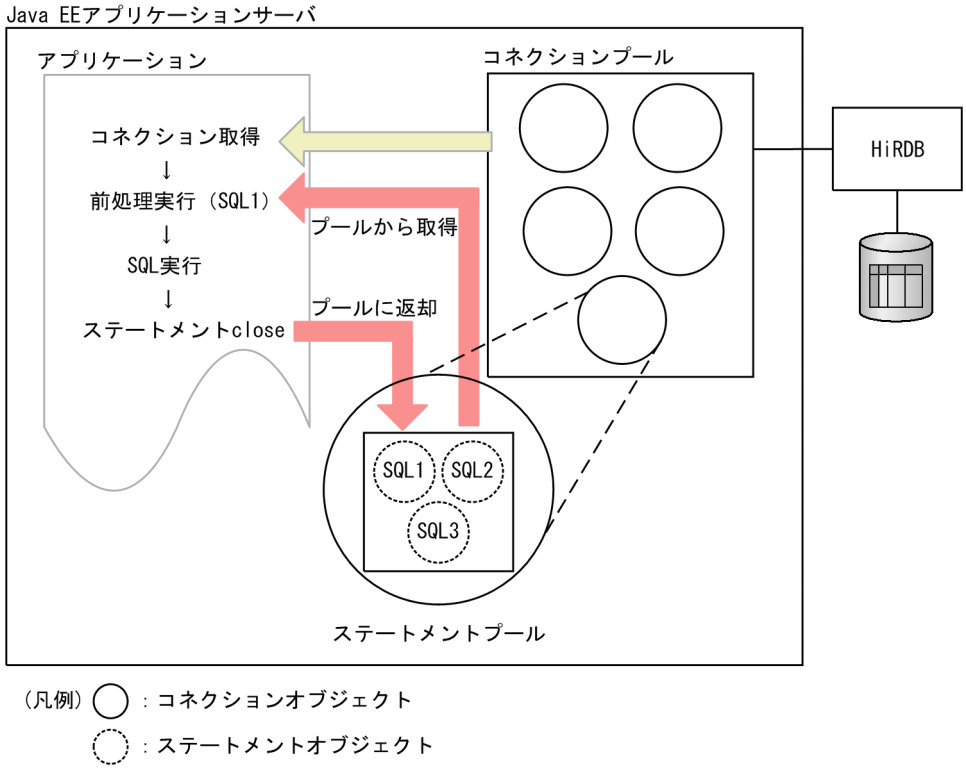
(2)　ステートメントプーリング／ステートメントキャッシュ

(a)　ステートメントプーリング／ステートメントキャッシュとは

同一の SQL 文を指定して前処理済みのステートメントオブジェクト（PreparedStatement,CallableStatement）を作成するユーザアプリケーションを呼び出す場合などで，ステートメントオブジェクトをプールして使い回す仕組みのことをステートメントプーリング／ステートメントキャッシュと呼びます。

ステートメントオブジェクトはコネクションをまたがって使い回すことはできないため，コネクションプーリングを使用していることが前提の機能となります。

図 6-9　ステートメントプーリング／ステートメントキャッシュ



(b)　ステートメントプーリング／ステートメントキャッシュ使用時の注意事項

- 前処理済みのステートメントオブジェクトがプーリングされている間，関連するスキーマ資源（表やインデクスなど）に対して排他を取得します。これらの資源に対して定義系 SQL を実行する場合は，クライアント環境定義 PDDDLDEAPRPEXE を指定してください（ホールダブルカーソルを使用している場合は PDDDLDEAPRP も指定してください）。

  ただし，定義系 SQL を実行した場合，前処理結果が無効になるため，Java EE アプリケーションサーバの再起動やコネクションプール内のコネクションオブジェクトを破棄する必要があります。

  詳細は，マニュアル「HiRDB UAP 開発ガイド」の「クライアント環境定義の設定内容」の PDDDLDEAPRPEXE を参照してください。

- 自動再接続機能を使用して再接続した場合，再接続前からプーリングされているステートメントオブジェクトが使い回されると，実行時に例外が発生します。ステートメントプーリング／ステートメントキャッシュ機能を使用する場合には，自動再接続機能は使用しないでください。

  ステートメントの close メソッド実行時に SQL 前処理を破棄しないよう，システムプロパティ HiRDB_for_Java_STATEMENT_CLOSE_BEHAVIOR に FALSE を設定するか，又は指定を省略してください。詳細はマニュアル「HiRDB UAP 開発ガイド」の次の個所を参照してください。

  システムプロパティ HiRDB_for_Java_STATEMENT_CLOSE_BEHAVIOR の設定方法

  「ユーザプロパティ」の「HiRDB_for_Java_STATEMENT_CLOSE_BEHAVIOR」

上記以外の設定方法

「接続情報の優先順位」の「ステートメントの close メソッド実行時の SQL 前処理の破棄」

- SELECT 文，INSERT 文，DELETE 文，UPDATE 文，PURGE TABLE 文，及び CALL 文以外の SQL 文では，コミット時に前処理済みの SQL 文が無効になります。

  前処理が無効となる SQL 文については PreparedStatement／CallableStatement オブジェクトを使用せず，Statement オブジェクトを使用して実行してください。SQL 文による切り分けができない場合は，ステートメントプーリング／ステートメントキャッシュを使用しないでください。

- 1 つの Connection オブジェクトに対し，使用中のステートメントオブジェクト（Statement，PreparedStatement，及び CallableStatement など）の数の合計が 4096 以上となった場合，SQLException を投入します。詳細はマニュアル「HiRDB UAP 開発ガイド」の「Statement インタフェース」の「注意事項」を参照してください。

- プーリングされているステートメントオブジェクトに対応する SQL オブジェクトは，SQL オブジェクト用バッファに保存されているため，SQL オブジェクト用バッファ長の見積もり時は，コネクションプーリング数及びステートメントプーリング数を考慮して必要なバッファ長を算出してください。

  詳細は，マニュアル「HiRDB システム定義」の「バッファに関するオペランド」の pd_sql_object_cache_size を参照してください。

7 HiRDB/シングルサーバの設計

この章では，HiRDB/シングルサーバのシステム構成，HiRDB ファイルシステム領域の設計方法，システムファイルの設計方法及び RD エリアの配置方法の考慮点について説明します。

7.1　HiRDB/シングルサーバのシステム設計

ここでは，HiRDB/シングルサーバのシステム設計とシステム構成について説明します。

7.1.1　システム設計

(1)　HiRDB/シングルサーバが使用するメモリ

HiRDB/シングルサーバが使用するメモリについて説明します。

HiRDB/シングルサーバは次に示すメモリを使用します。

- 共用メモリ
- プロセス固有メモリ

(a)　メモリ所要量

HiRDB/シングルサーバが必要とするメモリ所要量を見積もってください。HiRDB/シングルサーバのメモリ所要量については，「HiRDB/シングルサーバのメモリ所要量の見積もり」を参照してください。

(b)　共用メモリのページ固定

HiRDB では，次に示す共用メモリを実メモリ上に固定できます。

- ユニットコントローラ用共用メモリ
- グローバルバッファ用共用メモリ
- 動的変更したグローバルバッファが使用する共用メモリ
- インメモリデータバッファ用共用メモリ

共用メモリを実メモリ上に固定すると，ページの入出力が少なくなるため，性能が安定します。

前提条件

共用メモリのページ固定をするための，OS ごとの前提条件を次に示します。

OS	前提条件
AIX	64 ビットモードであること。
Linux	なし。

動作環境の設定

AIX の場合，オペレーティングシステムパラメタの設定が必要となります。オペレーティングシステムパラメタの設定については，「AIX 固有のパラメタ指定」を参照してください。

ページ固定の方法

共用メモリのページ固定の方法について，共用メモリの種別ごとに説明します。

- ユニットコントローラ用共用メモリ

  システム共通定義，又はユニット制御情報定義の pd_shmpool_attribute オペランドに fixed を指定します。

- グローバルバッファ用共用メモリ

  システム共通定義，又はユニット制御情報定義の pd_dbbuff_attribute オペランドに fixed を指定します。

- 動的変更したグローバルバッファが使用する共用メモリ

  システム共通定義，又はユニット制御情報定義の pd_dbbuff_attribute オペランドに fixed を指定します。これによって，pdbufmod コマンドを実行して動的変更したグローバルバッファが使用する共用メモリも実メモリ上に固定されます。

- インメモリデータバッファ用共用メモリ

  pdmemdb コマンドの-p オプションに fixed を指定します。

注意事項

実メモリ上に連続した領域が確保できなかった場合，共用メモリのページ固定ができません。ページ固定に失敗した場合の HiRDB の動作を次に示します。

OS	ページ固定に失敗した場合の HiRDB の動作			
	ユニットコントローラ用	グローバルバッファ用	動的変更したグローバルバッファ用	インメモリデータバッファ用
AIX	○※	○※	○※	○※
Linux	○	×	×	×

（凡例）

○：ページ固定をしないで共用メモリを確保し，処理を続行します。

×：HiRDB，又はコマンドが異常終了します。

注※

AIX の場合，ページ固定に失敗したときでもシステムコールは正常終了します。そのため，HiRDB からはページ固定に失敗したことが分かりません。ページ固定ができているかどうかは，次の手順で確認してください。

1. HiRDB 稼働中に pdls -d mem コマンドを実行し，次の共用メモリセグメントの識別子を確認します。

・ユニットコントローラ用共用メモリの場合は，SHM-OWNER に MANAGER と表示されている共用メモリセグメント

・上記以外の共用メモリの場合は，SHM-OWNER にユニット名を（）で囲んだ文字列，又は HiRDB サーバ名が表示されている共用メモリセグメント

2. OS の ipcs -s コマンドを実行して，1.で確認した共用メモリセグメントの識別子を持つ共用メモリの SID 値を確認します。

3. 2.で確認した SID 値に対して OS の svmon コマンドを実行して，該当する共用メモリの実メモリページ数と固定されているページ数が一致するかどうかを確認します。

(c) Hugepage 機能を用いた共用メモリの固定（Linux 限定）

Linux 版の HiRDB では，Linux の Hugepage 機能を用いて，次に示す共用メモリを実メモリ上に固定できます。

- ユニットコントローラ用共用メモリ
- グローバルバッファ用共用メモリ
- 動的変更したグローバルバッファが使用する共用メモリ
- インメモリデータバッファ用共用メモリ

Linux の Hugepage 機能を用いて共用メモリを固定すると，共用メモリのページサイズが通常のページサイズ（4KB）から hugepages のページサイズ（2MB）に拡大されます。ページサイズが拡大すると，仮想アドレスを実アドレスに変換するための管理領域（PTE：Page Table Entry）を通常よりも小さくできるため，メモリ不足を防止できます。また，仮想アドレスと物理アドレスの変換回数が少なくなり，ページフォルトの発生を抑えることができます。これによって，トランザクション性能が向上することがあります。

前提条件

Linux の Hugepage 機能を有効にする必要があります。Hugepage 機能については，OS のマニュアルを参照してください。

動作環境の設定

OS のオペレーティングシステムパラメタで，Hugepage 機能を有効にする設定をしてください。OS のオペレーティングシステムパラメタの設定をしないと，共用メモリの確保と固定化に失敗し，HiRDB は起動しません。OS のオペレーティングシステムパラメタに指定する値の詳細については，「Linux の Hugepage 機能の指定」を参照してください。

ページ固定の方法

共用メモリのページ固定の方法について，共用メモリの種別ごとに説明します。

- ユニットコントローラ用共用メモリ

  システム共通定義，又はユニット制御情報定義の pd_shmpool_attribute オペランドに hugepage を指定します。

- グローバルバッファ用共用メモリ

  システム共通定義，又はユニット制御情報定義の pd_dbbuff_attribute オペランドに hugepage を指定します。

- 動的変更したグローバルバッファが使用する共用メモリ

システム共通定義，又はユニット制御情報定義の pd_dbbuff_attribute オペランドに hugepage を指定します。これによって，pdbufmod コマンドを実行して動的変更したグローバルバッファが使用する共用メモリも実メモリ上に固定されます。

- インメモリデータバッファ用共用メモリ

  pdmemdb コマンドの-p オプションに hugepage を指定します。

注意事項

サーバマシン上に十分な hugepages が確保されていないと，共用メモリの確保に失敗します。各共用メモリの確保に失敗した場合，次に示すメッセージを出力して HiRDB の開始処理を中止するか，又はコマンドが異常終了します。

- **ユニットコントローラ用共用メモリの確保に失敗した場合**
  KFPO00113-E メッセージ
- **グローバルバッファ用共用メモリの確保に失敗した場合**
  KFPH23015-E 及び KFPH23005-E メッセージ
- **動的変更したグローバルバッファ用共用メモリの確保に失敗した場合**
  KFPH27031-E メッセージ
- **インメモリデータバッファ用共用メモリの確保に失敗した場合**
  KFPH23208-E メッセージ

上記の理由で HiRDB の開始処理が中止した場合は，マニュアル「HiRDB メッセージ」を参照して，エラーとなった原因を調査し，対策してください。

(2) ユティリティ専用ユニットの設置

入出力装置（MT ドライブなど）の配置の制約などによって，HiRDB/シングルサーバが稼働するサーバマシンに，ユティリティで使用する入出力装置を設置できない場合があります。この場合，別のサーバマシンにユティリティで使用する入出力装置を設置し，HiRDB/シングルサーバが稼働するサーバマシンと LAN を接続すると，その入出力装置を利用できます。このように，ユティリティで使用する入出力装置だけを設置するサーバマシンを**ユティリティ専用ユニット**といいます。

ユティリティ専用ユニットは，次に示すユティリティを実行するときに使用します。

- 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）

ユティリティを実行するときのユティリティ専用ユニットの使用方法については，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。

HiRDB/シングルサーバが稼働するサーバマシンに入出力装置を設置できない場合は，ユティリティ専用ユニットの設置を検討してください。

なお，ユティリティ専用ユニットは，複数の HiRDB/シングルサーバで共用できます。

7.1.2 HiRDB/シングルサーバのシステム構成

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。ユティリティ専用ユニットを設置する場合のシステム構成を図「HiRDB/シングルサーバのシステム構成（ユティリティ専用ユニットを設置する場合）」に示します。

HiRDB/シングルサーバのシステム構成は，HiRDB システム定義で定義します。HiRDB システム定義の定義例については，マニュアル「HiRDB システム定義」を参照してください。

図 7-1 HiRDB/シングルサーバのシステム構成

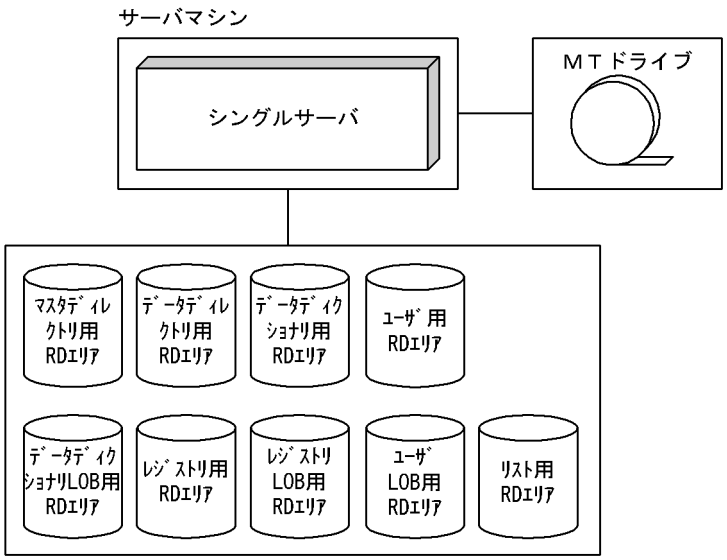
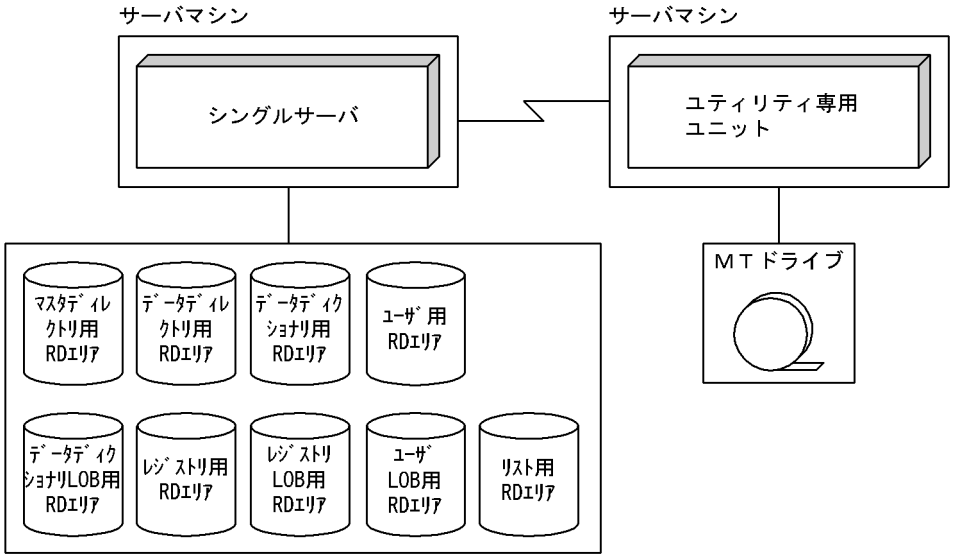


図 7-2　HiRDB/シングルサーバのシステム構成（ユティリティ専用ユニットを設置する場合）



7.2 HiRDB ファイルシステム領域の設計

HiRDB のシステム構築時に，HiRDB ファイルを作成する HiRDB ファイルシステム領域を作成します。ここでは，HiRDB ファイルシステム領域を作成するときの設計方針について説明します。

HiRDB ファイルシステム領域は，次に示す用途ごとに作成することをお勧めします。これによって，用途やアクセス特性の異なるファイルへの入出力が競合することを回避できます。また，通常ファイルを使用する場合，用途を明示することによる書き込み性能の向上のほか，用途に応じたデバイスを割り当てられます。

- RD エリア用
- システムファイル用
- 作業表用ファイル用
- ユティリティ用
- リスト用 RD エリア用

7.2.1 RD エリア用の HiRDB ファイルシステム領域の設計

RD エリア用の HiRDB ファイルシステム領域の設計方針について説明します。

(1) 信頼性向上のための方針

1. 更新系処理に対する信頼性は，通常ファイルよりキャラクタ型スペシャルファイルの方が高いです。また，通常ファイルは，OS が異常終了すると使用できなくなる場合があります。したがって，次に示す条件を満たすユーザ用 RD エリア用の 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に作成することをお勧めします。
   - 更新処理が主体となる表を格納するユーザ用 RD エリア
   - 重要度の高いデータを格納するユーザ用 RD エリア
2. 必要な HiRDB ファイルシステム領域の大きさを見積もって，その大きさ以上の HiRDB ファイルシステム領域を作成してください。
3. RD エリア用の HiRDB ファイルシステム領域は，シングルサーバを定義するサーバマシンに作成します。
4. 次に示す RD エリアを作成する HiRDB ファイルシステム領域は，シングルサーバを定義するサーバマシンに作成します。
   - システム用 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

5. 次に示す RD エリアを作成する HiRDB ファイルシステム領域は，シングルサーバを定義するサーバマシンに作成します。
   - ユーザ用 RD エリア
   - ユーザ LOB 用 RD エリア
6. 系切り替え機能を使用するときは，RD エリア用の HiRDB ファイルシステム領域をキャラクタ型スペシャルファイルに作成してください。

(2) 性能向上のための方針

1. RD エリアを作成する HiRDB ファイルシステム領域は，次に示す RD エリア用ごとに作成することをお勧めします。
   - システム用 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
2. システムファイル用の HiRDB ファイルシステム領域と，RD エリア用の HiRDB ファイルシステム領域は，別々のハードディスクに作成することをお勧めします。これによって，シンクポイントダンプを取得するときに入出力の分散ができ，シンクポイントダンプの取得処理時間を短縮できます。
3. プリフェッチ機能を使用しないときは，シーケンシャルリードに関してはキャラクタ型スペシャルファイルより通常ファイルの方が速いです。
4. ランダムな 1 ページリードは，通常ファイルよりキャラクタ型スペシャルファイルの方が速いです。
5. ライト処理は，通常ファイルよりキャラクタ型スペシャルファイルの方が速いです。
6. 通常ファイルは階層構造で構成されるため，ファイルが大きくなると階層が増えます。階層が増えたファイルにアクセスするときは入出力回数が増えるため，アクセス効率が下がります。
7. 次の表に示すとおりに HiRDB ファイルシステム領域を割り当てることをお勧めします。このようにすると，入出力時間を削減できます。

表 7-1 性能を向上するための HiRDB ファイルシステム領域の割り当て方法

HiRDB ファイルシステム領域の条件	割り当てるファイル
システム用 RD エリアを作成する 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
データディクショナリ LOB 用 RD エリアを作成する HiRDB ファイルシステム領域	
ユーザ LOB 用 RD エリアを作成する HiRDB ファイルシステム領域	
更新の多い表又は少量検索が主体となる表を格納するユーザ用 RD エリアを作成する HiRDB ファイルシステム領域	

HiRDB ファイルシステム領域の条件	割り当てるファイル
データ量が多くて，全件検索又はクラスタキーによる大量キー順検索が主体となる表を格納するユーザ用 RD エリアを作成する HiRDB ファイルシステム領域（ただし，更新をほとんどしない場合）	• 通常ファイル（プリフェッチ機能未使用時） • キャラクタ型スペシャルファイル（プリフェッチ機能使用時）

7.2.2　システムファイル用の HiRDB ファイルシステム領域の設計

システムファイル用の HiRDB ファイルシステム領域の設計方針について説明します。

(1)　信頼性向上のための方針

1. 更新系処理に対する信頼性は，通常ファイルよりキャラクタ型スペシャルファイルの方が高いです。通常ファイルは，システムダウン後にファイルシステム自体が使えなくなることがあります。このため，システムファイル用の 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に作成します。
2. システムファイル用の HiRDB ファイルシステム領域は二つ以上作成してください。一つしか作成しないと，システムファイルがあるハードディスクに障害が発生した場合，HiRDB が稼働できなくなります。
3. システムファイル用の HiRDB ファイルシステム領域は別々のハードディスクに作成してください。そうすれば，どちらかのハードディスクに障害が発生しても，HiRDB を再開始できます。
4. 必要な HiRDB ファイルシステム領域の大きさを見積もって，その大きさ以上の HiRDB ファイルシステム領域を作成してください。

(2)　性能向上のための方針

システムファイル用の HiRDB ファイルシステム領域と，RD エリア用の HiRDB ファイルシステム領域は，別々のハードディスクに作成することをお勧めします。これによって，シンクポイントダンプを取得するときに入出力の分散ができ，シンクポイントダンプの取得処理時間を短縮できます。

7.2.3　作業表用ファイル用の HiRDB ファイルシステム領域の設計

作業表用ファイル用の HiRDB ファイルシステム領域の設計方針について説明します。

(1)　設計方針

1. 作業表用ファイルは通常ファイルに割り当ててもかまいませんが，系切り替え機能を使用する場合は，キャラクタ型スペシャルファイルに割り当てると共用できるため，ディスク量を削減できます。
2. 作業表用ファイル用の HiRDB ファイルシステム領域長は，その HiRDB ファイルシステム領域に作成する作業表用ファイルの総容量より大きくしてください。なお，pdfmkfs コマンドで-a オプションを指定すると，HiRDB ファイルシステム領域を自動的に拡張できます。作業表用ファイルの総容量が

HiRDB ファイルシステム領域長に達した場合，自動で HiRDB ファイルシステム領域を拡張できるため，-a オプションを指定することをお勧めします。※

作業表用ファイルの容量については，「作業表用ファイルの容量の見積もり」を参照してください。

注※

HiRDB 再開始時に作業表用ファイル用の HiRDB ファイルシステム領域のディスク占有サイズを削減したい場合は，HiRDB 再開始前に pdfmkfs コマンドを実行して作業表用ファイル用の HiRDB ファイルシステム領域を再初期化してください。

(2) 最大使用量を調べる方法

作業表用ファイル用の HiRDB ファイルシステム領域の最大使用量を次に示す方法で調べられます。

pdfstatfs -d 作業表用の HiRDB ファイルシステム領域名

-d：

HiRDB ファイルシステム領域の最大使用量を表示するオプションです。出力情報の peak capacity が最大使用量です。なお，上記の最大使用量は pdfstatfs コマンドでクリアできます。

pdfstatfs -c 作業表用の HiRDB ファイルシステム領域名

-c：

HiRDB ファイルシステム領域の最大使用量をクリアするオプションです。

7.2.4 ユティリティ用の HiRDB ファイルシステム領域の設計

ユティリティ用（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作成用）の HiRDB ファイルシステム領域の設計方針について説明します。ユティリティ用の HiRDB ファイルシステム領域には，次に示すファイルを作成します。

- 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
- 差分バックアップ管理ファイル

(1) 設計方針

1. バックアップファイル作成用にする場合は，キャラクタ型スペシャルファイルに HiRDB ファイルシステム領域を作成してください。
2. バックアップファイル作成用にする場合は，HiRDB ファイルシステム領域長をバックアップ対象 RD エリアの総容量より大きくしてください。RD エリアの容量については，「RD エリアの容量の見積もり」を参照してください。

3. 系切り替え機能使用時は，アンロードログファイルを共有ディスク（キャラクタ型スペシャルファイル）上に作成してください。
4. アンロードログファイル作成用にする場合は，pdfmkfs コマンドのオプションに次に示す値を指定してください。
   - -k オプション：使用目的には UTL（ユティリティ用の HiRDB ファイルシステム領域）を指定します。
   - -n オプション：HiRDB ファイルシステム領域長には次に示す計算式の値を指定します。

     （アンロードするシステムログファイルの総レコード数※1×システムログファイルのレコード長）※2×作成するアンロードログファイル数×1.2÷1048576
   - -l オプション：最大ファイル数には，作成するアンロードログファイル数を指定します。
   - -e オプション：最大増分回数には，作成するアンロードログファイル数×24 を指定します。

注※1

システムログファイルの自動拡張機能を適用している場合，pd_log_auto_expand_size オペランドの拡張上限サイズに指定した値で計算してください。

注※2

システムログファイルの概算値です。

(2) 最大使用量を調べる方法

ユティリティ用の HiRDB ファイルシステム領域の最大使用量を次に示す方法で調べられます。

pdfstatfs -d ユティリティ用の HiRDB ファイルシステム領域名

-d：

HiRDB ファイルシステム領域の最大使用量を表示するオプションです。出力情報の peak capacity が最大使用量です。なお，上記の最大使用量は pdfstatfs コマンドでクリアできます。

pdfstatfs -c ユティリティ用の HiRDB ファイルシステム領域名

-c：

HiRDB ファイルシステム領域の最大使用量をクリアするオプションです。

7.2.5 リスト用 RD エリア用の HiRDB ファイルシステム領域の設計

リスト用 RD エリア用の HiRDB ファイルシステム領域の設計方針について説明します。

(1) 設計方針

1. リストは検索の一時的な中間結果を保存するものなので，ほかの RD エリアほど信頼性は要求されません。したがって，リスト用 RD エリア用の HiRDB ファイルシステム領域は通常ファイルに作成してもかまいません。

2. 系切り替え機能を使用するときは，キャラクタ型スペシャルファイルに作成すると HiRDB ファイルシステム領域を共用できるので，ディスク容量を削減できます。

(2) 性能向上のための方針

1. リスト用 RD エリア用の HiRDB ファイルシステム領域を RAID に作成する場合は，キャラクタ型スペシャルファイルに作成した方が処理時間が短縮されます。RAID 以外のディスクに作成する場合は，通常ファイルに作成した方が処理時間が短縮されます。
2. リスト用 RD エリア用の HiRDB ファイルシステム領域は，次に示す HiRDB ファイルシステム領域とは別々のハードディスクに作成することをお勧めします。別々のハードディスクに作成すると，リストを検索するときに入出力を分散できるため，処理時間を短縮できます。
   - ユーザ用 RD エリア用の HiRDB ファイルシステム領域
   - ユーザ LOB 用 RD エリア用の HiRDB ファイルシステム領域
   - 作業表用ファイル用の HiRDB ファイルシステム領域

7.2.6 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 7-2 HiRDB ファイルシステム領域の最大長

HiRDB の種類	条件		HiRDB ファイルシステム領域の最大長（単位：メガバイト）
AIX 版	ラージファイル未使用	通常ファイル	2,047
		キャラクタ型スペシャルファイル	
	ラージファイル使用	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	ラージファイル未使用	通常ファイル	2,047
		キャラクタ型スペシャルファイル	
	ラージファイル使用	通常ファイル	1,048,575
		キャラクタ型スペシャルファイル	

7.3　システムファイルの設計

ここでは，システムファイルの設計方針について説明します。

7.3.1　システムログファイルの設計

システムログファイルの設計方針について説明します。

(1)　設計方針

1. ユティリティ専用ユニットにシステムログファイルは必要ありません。
2. すべてのシステムログファイルのレコード長及びレコード数を同じにしてください。
3. 作成できるシステムログファイルは，2～200 グループです（ただし，6 グループ以上作成することを推奨します）。
4. 一つのサーバに対して，次の式を満たすようにシステムログファイルを作成します。

   ```
   100（単位：ギガバイト）×（200-サーバに割り当てるシステムログファイルグループ数）≧サーバに割り当てるシステムログファイルの総容量×3
   ```

   これは，システムログファイルの容量不足によって異常終了した HiRDB を再開始する場合に必要になります。システムログファイルが容量不足になった状況や運用によっては，サーバに割り当てられている 3 倍の容量のシステムログファイルを作成し，サーバに追加して対処する必要があるためです。
5. アンロードする回数を少なくしたい場合は，システムログファイルの 1 ファイル容量を大きくします。
6. HiRDB 運用ディレクトリがあるディスクに大量に入出力が発生するファイル（システムログのアンロードログファイルなど）を作成しないでください。
7. 一つのシステムログファイルの容量は，次に示す条件を満たす必要があります。

   ```
   一つのシステムログファイルの容量（バイト）≧↑（a＋368）÷c↑×c×b×d
   ```

   a：pd_log_max_data_size オペランドの値

   b：pd_log_sdinterval オペランドの値

   c：pd_log_rec_leng オペランドの値

   d：pd_spd_assurance_count オペランドの値
8. 全システムログファイルの容量（二重化している場合は片系の容量だけを対象とする）は，次に示す二つの条件を満たす必要があります。

条件 1

「システムログファイルの総容量の求め方」で見積もった容量以上の値にしてください。

条件 2

長大トランザクションが終了するまでは，長大トランザクション開始以降のシステムログファイルの上書きができません。また，シンクポイントダンプファイルの有効保証世代とカレント世代のシステムログファイルは上書きできません。そのため，これらの分のシステムログファイル容量を確保してください。次の計算式で求めます。

システムログファイルの総容量（バイト）≧3×a×（b＋1）

a：

データベースを更新するトランザクションのうち，実行時間が最大のトランザクション実行中に該当するサーバで出力されることのあるシステムログ量

システムログ量の見積もり式については，「システムログファイルの容量の見積もり」を参照してください。

b：pd_spd_assurance_count オペランドの値

シンクポイントダンプファイルの有効保証世代数です。

(a) システムログファイルの世代数と運用への影響

システムログファイルの総容量が同じ場合，システムログファイルの世代数によって，1 世代当たりの容量が変化します。システムログファイルの世代数と運用への影響を次の表に示します。システムログファイルの総容量は同じとします。

表 7-3 システムログファイルの世代数と運用への影響

比較項目	システムログファイルの構成	
	世代数を少なくした場合	世代数を多くした場合
システムログファイル 1 世代当たりの容量	大きくなります。	小さくなります。
スワップ間隔	システムログファイル 1 世代当たりの容量が大きくなるため，スワップ間隔は長くなります。	システムログファイル 1 世代当たりの容量が小さくなるため，スワップ間隔は短くなります。
アンロード回数	スワップ間隔が長くなるため，アンロード回数は少なくなります。	スワップ間隔が短くなるため，アンロード回数は多くなります。
ディスク障害などでシステムログファイルの数世代が使用できなくなった場合のシステムログ容量への影響	• システムログファイル 1 世代当たりの容量が大きくなるため，ディスク障害時にデータベースの回復に用いるログ量は多くなり，データベース回復に掛かる時間は長くなります。 • システムログ容量の減少幅が大きいため，システムログの容量が減少したことによる HiRDB の稼働に及ぼす影響は大きくなります。	• システムログファイル 1 世代当たりの容量が小さくなるため，ディスク障害時にデータベースの回復に用いるログ量は少なくなり，データベース回復に掛かる時間は短くなります。 • システムログ容量の減少幅が小さいため，システムログの容量が減少したことによる HiRDB の稼働に及ぼす影響は小さくなります。

通常の運用では，システムログファイルの世代数を少なくした場合の方がスワップ間隔及びアンロード回数の点で利点があります。ただし，障害発生時には，世代数を多くした場合の方が障害の影響が小さくなります。

(2) 信頼性向上のための方針

(a) システムログファイルの二重化

システムログファイルを二重化すると，HiRDB は両方の系に同じシステムログを取得します。取得したシステムログを読み込むとき，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からシステムログを読み込めるため，システムの信頼性を向上できます。なお，二重化する場合，ディスクをミラー化して二重化するより，HiRDB 管理下で二重化することをお勧めします。システムログファイルを二重化する場合は，それぞれの系のファイルを別々のハードディスクに作成してください。

システムログファイルを二重化する場合は，サーバ定義で次に示すオペランドを指定してください。

- pd_log_dual = Y
- pdlogadpf オペランドの-b オプション（B 系のシステムログファイル名を指定します）

(b) システムログファイルの片系運転

システムログファイルの片系運転は，システムログファイルを二重化する場合に適用されます。

システムログファイルに障害が発生して，両系とも使用できるシステムログファイルがない場合でも，HiRDB のユニットを異常終了しないで正常な系だけで処理を続行できます。これを**システムログファイルの片系運転**といいます。システムログファイルの片系運転をする場合は，サーバ定義で pd_log_singleoperation = Y を指定してください。

システムログファイルの片系運転に対し，両方のシステムログファイルで処理を続行すること（通常の処理形態）を**システムログファイルの両系運転**といいます。

(c) システムログファイルの自動オープン

HiRDB を再開始するときに，上書きできる状態のシステムログファイルがない場合，予約のファイルがあれば HiRDB が予約のファイルをオープンして上書きできる状態にし，処理を続行します。これを**システムログファイルの自動オープン**といいます。

システムログファイルの自動オープンをする場合は，サーバ定義で pd_log_rerun_reserved_file_open = Y を指定してください。

(3) 可用性向上のための方針

(a) 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スワップ時に，スワップ先にできる状態のシステムログファイルがないと HiRDB は異常終了します。これを予防するため，HiRDB にはシステムログファイルの空き容量を監視する機能

（**システムログファイルの空き容量監視機能**）があります。この機能は，システムログファイルの空き率が**警告値**未満になったときに作動します。次に示す二つのレベルのどちらかを選択できます。

レベル 1：

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，警告メッセージ KFPS01162-W を出力します。

レベル 2：

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，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します。このとき，KFPS01160-E メッセージを出力します。これによって，システムログの出力量を抑えます。

レベル 2 を選択した場合，システムログファイルの空き容量が不足したときにサーバ内の全トランザクションが強制終了されます。このため，システムログファイルの設計をより正確に行う必要があります。

システムログファイルの空き容量監視機能については，マニュアル「HiRDB システム運用ガイド」を参照してください。

(b) 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。

(c) シンクポイントダンプ有効化のスキップ回数監視機能

UAP が無限ループしてデータベースを更新し続けるとシンクポイントが有効化できないため，上書きできない状態のシステムログファイルが増えてしまいます。上書きできない状態のシステムログファイルが増えて全システムログファイルが上書きできない状態になると，HiRDB が異常終了します。

また，上書きできない状態のシステムログファイルが，全システムログファイルの半分以上になったときに HiRDB が異常終了又は強制終了すると，HiRDB を再開始するときのロールバック処理でシステムログファイルが不足します。この場合，システムログファイルを新規追加しないと，HiRDB を再開始できません。そして，この再開始処理に要する時間も長くなります。

このようなことを防ぐために，HiRDB では**シンクポイントダンプ有効化のスキップ回数監視機能**を設けています。

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) システムログの並列出力機能（AIX 版及び Linux 版限定）

システムログファイルを二重化している場合に，**システムログの並列出力機能**を使用すると，両系へのログの出力処理を並列して実行できるため，ログ出力に掛かる時間を短縮できます。システムログの並列出力機能を使用するには，**aio ライブラリ**（AIX の場合は Asynchronous I/O Subsystem，Linux の場合は libaio）が必要になります。Asynchronous I/O Subsystem 又は libaio については，OS のマニュアルを参照してください。

(a) お勧めの使い方

ログ出力に掛かる時間をより短縮するため，A 系と B 系をそれぞれ別デバイスに配置することをお勧めします。

(b) 環境設定（システム定義の設定）

サーバ定義で pd_log_dual_write_method=parallel を指定してください。ただし，次の場合は指定値に関係なく，システムログの並列出力機能は適用されません。

- システムログファイルが二重化されていない（pd_log_dual オペランドに Y が指定されていない）
- システムログファイルがキャラクタ型スペシャルファイルに配置されていない

(c) 環境設定（aio ライブラリの設定）

サーバマシンに aio ライブラリを導入し，必要な設定を行ってください。aio ライブラリの導入と設定方法については，OS のマニュアルを参照してください。

(b)で説明した環境設定を行っている状態で，aio ライブラリの導入と設定が正しく行われていない場合，HiRDB を開始できません。

(d) 注意事項

1. システムログの並列出力機能は，通常ファイルに配置されたシステムログファイルには適用されません。そのため，システムログファイルを追加する場合，キャラクタ型スペシャルファイルに配置してください。
2. システムログの並列出力機能が適用されるのは，両系の現用ファイルがキャラクタ型スペシャルファイルに配置されており，かつ両系の現用ファイルにシステムログが出力できる状態の場合（クローズ状態，予約状態，又は障害発生時ではない）だけです。現用ファイルが次の条件に該当する場合，システム定義に関係なく，システムログの並列出力を行いません。
   - A 系又は B 系のどちらかの現用ファイルが通常ファイルに配置されている
   - A 系又は B 系のどちらかの現用ファイルにログが出力できない状態である
3. 系切り替え機能使用時など，システムログの並列出力機能を適用するサーバが複数のサーバマシンで稼働する場合，aio ライブラリを有効化していないサーバマシンがあると，待機系ユニットの開始，又は系切り替えに失敗します。全サーバマシンで aio ライブラリを有効化してください。

(5) システムログファイルのレコード長（既に HiRDB を稼働している場合）

レコード長が 1024 以外の場合，システムログの格納効率を良くするために 1024 に変更することをお勧めします。

システムログファイルのレコード長の変更方法については，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) システムログファイルの定義

作成したシステムログファイルをどのファイルグループに対応させるかをサーバ定義の pdlogadfg 及び pdlogadpf オペランドで定義します。

7.3.2 シンクポイントダンプファイルの設計

シンクポイントダンプファイルの設計方針について説明します。

(1) 設計方針

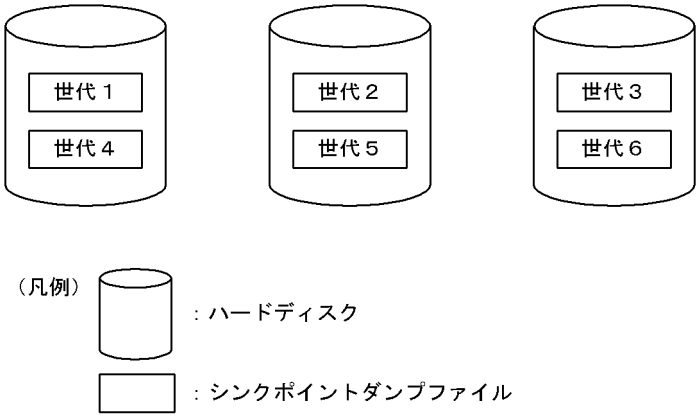
1. ユティリティ専用ユニットにシンクポイントダンプファイルは必要ありません。
2. 作成できるシンクポイントダンプファイルは，2～60 グループです。pdlogadfg -d spd オペランドに ONL を指定した場合は，2～30 グループです。
3. HiRDB は pdlogadfg -d spd オペランドの指定順にシンクポイントダンプファイルを使用します。
4. 4 個以上のシンクポイントダンプファイルを作成することをお勧めします。
5. シンクポイントダンプファイルの容量が不足すると，HiRDB を開始できません。このため，シンクポイントダンプファイルのレコード数は，システム共通定義の最大同時接続数（pd_max_users オペランド）の指定値より大きな値にしてください。詳細なシンクポイントダンプファイルの容量計算式については，「シンクポイントダンプファイルの容量の見積もり」を参照してください。

(2) 信頼性向上のための方針

(a) ファイル構成例

ハードディスク障害に備えて，シンクポイントダンプファイルをそれぞれ別々のハードディスクに作成してください。そのような構成を取れない場合は，隣り合わせの世代を別々のハードディスクに作成する構成にしてください。例を次の図に示します。

図 7-3　隣り合わせの世代を別々のハードディスクに作成する構成例



(b)　シンクポイントダンプファイルの二重化

シンクポイントダンプファイルを二重化すると，HiRDB は A 系及び B 系の両方に同じシンクポイントダンプを取得します。取得したシンクポイントダンプを読み込むとき，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からシンクポイントダンプを読み込めるため，システムの信頼性を向上できます。また，二重化している場合，有効保証世代数を 1 世代にすると，信頼性を損ねることなく上書きできない状態のシンクポイントダンプファイル数を削減できます。

シンクポイントダンプファイルを二重化する場合は，サーバ定義で次に示すオペランドを指定してください。

- pd_spd_dual = Y
- pdlogadpf オペランドの-b オプション（B 系のシステムログファイル名を指定します）

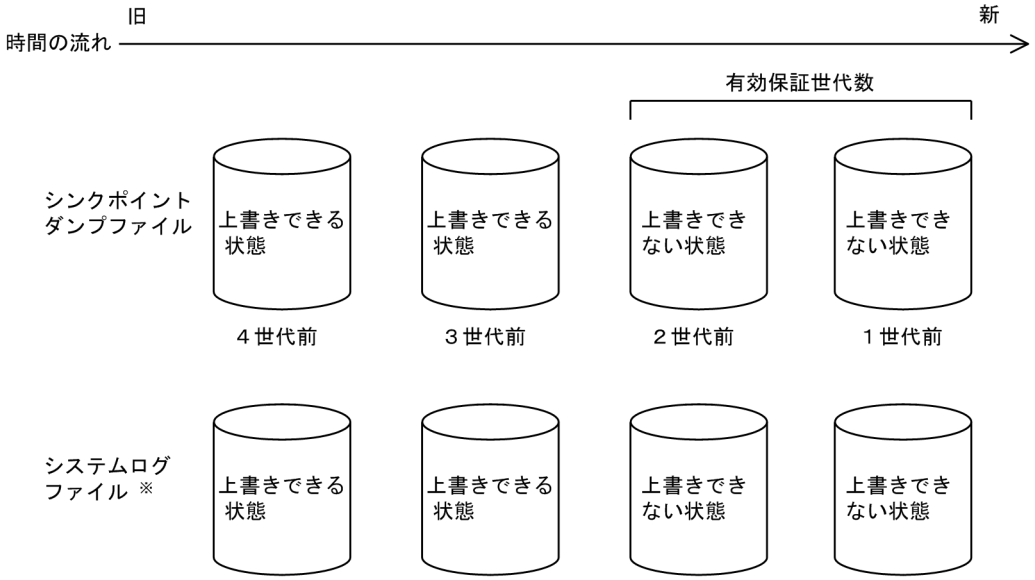
(c)　シンクポイントダンプファイルの片系運転

シンクポイントダンプファイルの片系運転は，シンクポイントダンプファイルを二重化する場合に適用されます。シンクポイントダンプファイルは，片方のファイルに異常が発生しても，正常な系だけで処理を続行します。これをシンクポイントダンプファイルの片系運転といいます。

(d)　シンクポイントダンプファイルの有効保証世代数

一つのシンクポイントダンプファイルには，HiRDB が 1 回に取得するシンクポイントダンプが格納されます。HiRDB は，シンクポイントダンプファイルを世代という概念で管理しています。HiRDB 管理者は，何世代前までのシンクポイントダンプファイルに対応するシステムログファイルを上書きできない状態にするかを指定できます。これを**シンクポイントダンプファイルの有効保証世代数**といいます。シンクポイントダンプファイルの有効保証世代数を次の図に示します。

図 7-4　シンクポイントダンプファイルの有効保証世代数



注※　シンクポイントダンプファイルに対応するシステムログファイルを示します。

〔説明〕

有効保証世代を 2 とすると，2 世代前までのシンクポイントダンプファイル及びそのシンクポイントダンプファイルに対応するシステムログファイルが，上書きできない状態になります。3 世代前より前の世代のシンクポイントダンプファイル及びそのシンクポイントダンプファイルに対応するシステムログファイルは，上書きできる状態になります。

シンクポイントダンプファイルの運用に必要なファイル数は，**有効保証世代数＋ 1** となります。シンクポイントダンプファイルの有効保証世代数は，サーバ定義の pd_spd_assurance_count オペランドで指定してください。

なお，シンクポイントダンプファイルを二重化している場合，必要な有効保証世代数は 1 世代をお勧めします。二重化していない場合，2 世代をお勧めします。

(e)　シンクポイントダンプファイルの縮退運転

シンクポイントダンプファイルに障害が発生して，運用に必要なファイル数（有効保証世代数＋ 1）を下回った場合でも，最低二つのファイルで処理を続行できます。これを**シンクポイントダンプファイルの縮退運転**といいます。

シンクポイントダンプファイルの縮退運転をする場合は，サーバ定義の pd_spd_reduced_mode オペランドを指定してください。

(f)　シンクポイントダンプファイルの自動オープン

シンクポイントダンプファイルに障害が発生して，運用に必要なファイル数（有効保証世代数＋ 1）を下回った場合，予約のファイルがあれば HiRDB が予約のファイルをオープンして上書きできる状態にし，処理を続行します。これを**シンクポイントダンプファイルの自動オープン**といいます。

シンクポイントダンプの自動オープンをする場合は，サーバ定義で pd_spd_reserved_file_auto_open = Y を指定してください。

(3) シンクポイントダンプファイルの定義

作成したシンクポイントダンプファイルをどのファイルグループに対応させるかを pdlogadfg 及び pdlogadpf オペランドで定義します。

なお，pdlogadfg オペランドだけを指定しておくと，HiRDB 稼働中にシンクポイントダンプファイルを追加できます。

7.3.3 ステータスファイルの設計

ステータスファイルの設計方針について説明します。

(1) 設計方針

1. 両系のファイルに障害が起きないように，A 系と B 系のファイルは別々のディスクに作成します。
2. ステータスファイルの容量不足による HiRDB の異常終了を防ぐため，見積もったファイル容量よりも大きい容量の予備ファイルを幾つか作成してください。ステータスファイルは，容量が満杯になると予備ファイルとスワップしますが，満杯になったステータスファイルと予備ファイルの容量が同じ場合，スワップ先のファイルでも容量不足になり，HiRDB は異常終了します。そのため，例えばステータスファイルを 6 組作成する場合，そのうちの 2 組以上のファイル容量をほかのファイル容量より大きくすることをお勧めします。
3. ユティリティ専用ユニットにサーバ用ステータスファイルは必要ありません。ユニット用ステータスファイルだけを作成してください。
4. A 系と B 系のファイルのレコード長及びレコード数を同じにしてください。
5. 作成できるユニット用ステータスファイルは 1～7 組です。
6. 作成できるサーバ用ステータスファイルは 1～7 組です。

(2) 信頼性向上のための方針

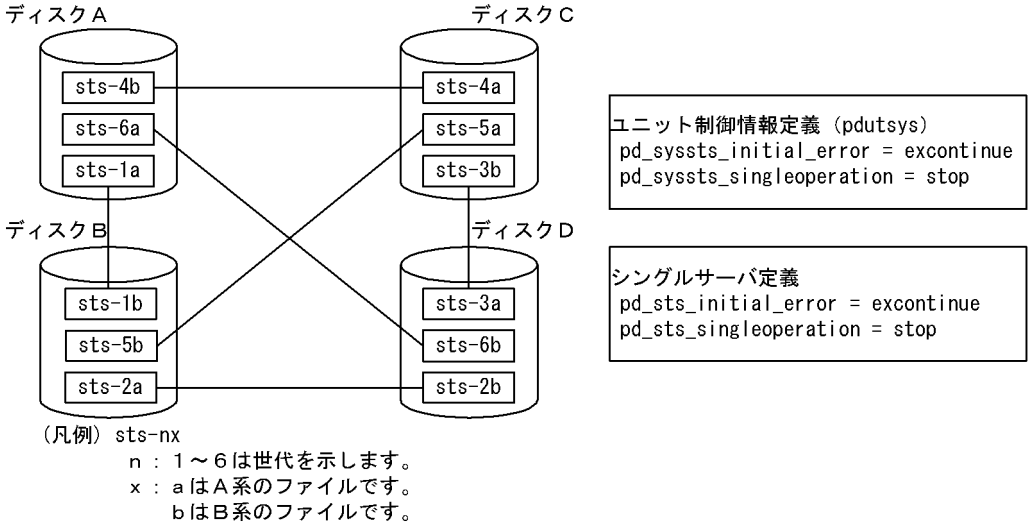
1. ステータスファイルは 3 組（二重化×3=6 ファイル）以上用意し，ディスク障害によってすべてのステータスファイルが障害とならないように配置します。
2. 容量不足による HiRDB の異常終了を防ぐため，ステータスファイルの容量は見積もった容量の 1.2 倍以上の容量を準備することをお勧めします。
3. ステータスファイルには，HiRDB の再開始処理でシステムの状態を回復するために必要な情報を格納しています。予備ファイルがない状態で現用ファイルに障害が発生すると，システムの状態が回復できません。したがって，常に予備ファイルがあるように運用し，現用ファイルの障害に備えてください。

(a) お勧めする構成

ディスク障害発生時から回復時までの安全性を考えると，ステータスファイルは四つのディスクに 6 組（二重化×6=12 ファイル）用意し，次の図のように配置することをお勧めします。また，片系運転中に正常な系に障害が発生すると，HiRDB を再開始できなくなるため，ステータスファイルの片系運転は適用しない（pd_syssts_singleoperation 及び pd_sts_singleoperation に stop を指定）ことをお勧めします。

四つのディスクにステータスファイルを 6 組配置する例を次の図に示します。

図 7-5　四つのディスクに 6 組のステータスファイルを配置する例



［説明］

このように配置すると，あるディスクで障害が発生した後に，更に別のディスクで障害が発生しても，残りの二つのディスクに両系とも正常なファイルが残るため，障害が発生していないディスクのステータスファイルを現用として HiRDB を稼働し続けることができます。例えば，ディスク A に障害が発生し，その後ディスク B にも障害が発生した場合でも，ディスク C 及び D にある両系のステータスファイル（sts-3a と sts-3b）を現用ファイルとして稼働し続けます。この状態で，更に現用ファイルの片系に障害が発生した場合，HiRDB は異常終了しますが，現用ファイルの片系ファイルが正常のため，障害が発生したディスクのどれか一つを回復すると HiRDB を再開始できます。

(3) ステータスファイルの定義

pdstsinit コマンドで作成したステータスファイルをどの論理ファイルに対応させるかを pd_syssts_file_name_1～7 及び pd_sts_file_name_1～7 オペランドで定義します。

ユニット用ステータスファイルは，pd_syssts_file_name_1～7 オペランドで定義します。サーバ用ステータスファイルは，pd_sts_file_name_1～7 オペランドで定義します。

なお，pd_syssts_file_name_2～7 オペランド又は pd_sts_file_name_2～7 オペランドに，実体のないステータスファイルを定義しておくと，HiRDB 稼働中にステータスファイルを追加できます。ただし，この場合，次に示すオペランドを指定しておく必要があります。

ユニット用ステータスファイルの場合

- pd_syssts_initial_error
- pd_syssts_last_active_file

サーバ用ステータスファイルの場合

- pd_sts_initial_error
- pd_sts_last_active_file

(4) ステータスファイルの片系運転

予備ファイルがない状況で現用ファイルの片系に障害が発生した場合，正常な系（片方の系）だけで処理を続行することを**ステータスファイルの片系運転**といいます。ステータスファイルが片系運転になると，KFPS01044-I メッセージが出力されます。

片系運転中に現用ファイルに障害が発生すると，HiRDB を再開始できなくなるため，ステータスファイルの片系運転の適用は推奨しません。ステータスファイルの組数を増やし，予備ファイルがない状況が発生しないような運用をしてください。

なお，ステータスファイルの片系運転に対し，両方の系のステータスファイルで処理を続行すること（通常の処理形態）を**ステータスファイルの両系運転**といいます。

(a) ステータスファイルの片系運転適用のメリット及びデメリット

メリット

予備ファイルがない状況で現用ファイルの片系に障害が発生しても，処理を続行できます。このため，ステータスファイルの障害によって HiRDB が停止する可能性が低くなります。

デメリット

片系運転中に正常な系に障害が発生したり，又はステータスファイルの更新中に HiRDB が異常終了したりすると，現用ファイルの内容が失われるため，HiRDB を再開始できなくなります。

(b) 指定方法

ユニット用ステータスファイルの片系運転をする場合は，ユニット制御情報定義で pd_syssts_singleoperation = continue を指定してください。サーバ用ステータスファイルの片系運転をする場合は，サーバ定義で pd_sts_singleoperation = continue を指定してください。なお，pd_syssts_singleoperation と pd_sts_singleoperation の指定値を同じにしてください。

- ほかのオペランドとの関連

  pd_syssts_singleoperation 及び pd_syssts_initial_error オペランド，又は pd_sts_singleoperation 及び pd_sts_initial_error オペランドの指定値の組み合わせによって，HiRDB の起動時にステータスファイルの障害を検知した場合の HiRDB の動作が決定します。したがって，これら二つのオペランドの指定値は一緒に考えるようにしてください。HiRDB の起動時にステータスファイルの障害を検知し

た場合の HiRDB の動作については，マニュアル「HiRDB システム定義」の，pd_syssts_initial_error 又は pd_sts_initial_error オペランドの説明を参照してください。

(c) 適用の目安

ステータスファイルの片系運転の適用の目安を次に示します。

- HiRDB が再開始できない状態を避けることを重視した運用の場合は，適用しないでください。
- HiRDB がオンラインダウンとなる状態を避けることを重視した運用の場合は，適用してください。
- 系切り替え構成を適用している場合など，HiRDB の再開始を自動で行う運用の場合は，適用しないでください。

(d) 片系運転適用時の注意

片系運転適用の有無による HiRDB の動作及び HiRDB 管理者の処置について次の表に示します。ステータスファイルに障害が発生したときの対処方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

表 7-4 片系運転適用の有無による HiRDB の動作及び HiRDB 管理者の処置

<table>
<tr><th colspan="2" rowspan="2">条件</th><th colspan="2">ステータスファイルの片系運転
(pd_syssts_singleoperation 又は pd_sts_singleoperation オペランドの指定値)</th></tr>
<tr><th>適用する（continue 指定）</th><th>適用しない（省略又は stop 指定）</th></tr>
<tr><td rowspan="2">予備ファイルがある</td><td>現用ファイルに障害が発生</td><td colspan="2">HiRDB の動作
ステータスファイルをスワップします。
HiRDB 管理者の処置
障害が発生したステータスファイルの障害対策をしてください。</td></tr>
<tr><td>現用ファイルの両系に，同時に障害が発生</td><td colspan="2">HiRDB の動作
異常終了します。HiRDB は再開始できません。
HiRDB 管理者の処置
マニュアル「HiRDB システム運用ガイド」の「ステータスファイルに障害が発生したときの対処方法」を参照してください。</td></tr>
<tr><td rowspan="2">予備ファイルがない</td><td>現用ファイルの片系に障害が発生</td><td>HiRDB の動作
片系運転を適用し，処理を続行します。
HiRDB 管理者の処置
早急に予備のファイルを作成して，両系運転の状態に戻してください。</td><td>HiRDB の動作
異常終了します。
HiRDB 管理者の処置
予備ファイルを作成し，HiRDB を再開始してください。</td></tr>
<tr><td>現用ファイルの両系に，同時に障害が発生</td><td colspan="2">HiRDB の動作
異常終了します。HiRDB は再開始できません。
HiRDB 管理者の処置
マニュアル「HiRDB システム運用ガイド」の「ステータスファイルに障害が発生したときの対処方法」を参照してください。</td></tr>
</table>

条件		ステータスファイルの片系運転 (pd_syssts_singleoperation 又は pd_sts_singleoperation オペランドの指定値)	
		適用する（continue 指定）	適用しない（省略又は stop 指定）
	片系運転中に，正常な系に障害が発生	HiRDB の動作 異常終了します。HiRDB は再開始できません。 HiRDB 管理者の処置 マニュアル「HiRDB システム運用ガイド」の「ステータスファイルに障害が発生したときの対処方法」を参照してください。	－

（凡例）－：該当しません。

(5) ステータスファイルの障害に関する注意事項（重要）

- 現用ファイルの両系に同時に障害が発生した場合，HiRDB が異常終了し，再開始できなくなります。対策として，物理ディスクの多重化（ミラーリング）が考えられます。
- HiRDB の開始前に，現用ファイル（終了時の現用ファイル）を削除，又は pdstsinit コマンドでステータスファイルを初期化した場合，HiRDB は再開始できなくなります。

7.4　RD エリアの配置

ここでは，次に示す RD エリアを配置するときの考慮点について説明します。

- システム用 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
- リスト用 RD エリア

7.4.1　システム用 RD エリアの配置

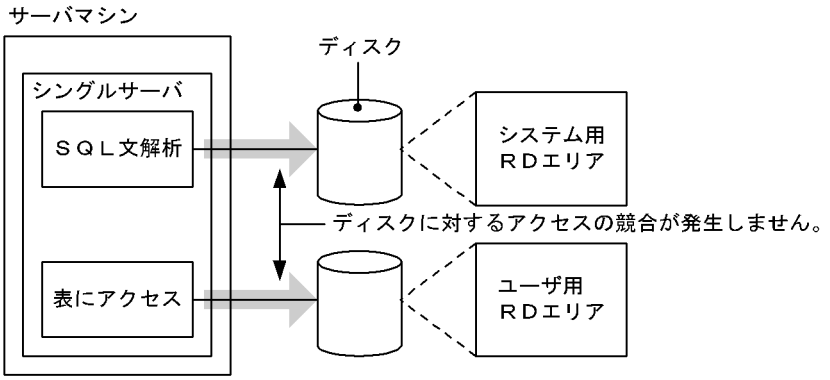
システム用 RD エリアは，ユーザ用 RD エリアの配置を考慮して配置します。システム用 RD エリアを配置するときの考慮点を次に示します。

- **ユーザ用 RD エリアを配置するディスクとは異なるディスクに配置するようにします。**

システム用 RD エリアのうち，特にデータディクショナリ用 RD エリアとデータディレクトリ用 RD エリアは，SQL 文の解析などのために HiRDB にアクセスされることが多くなります。このため，ユーザ用 RD エリアを配置するディスクと同じディスクに配置すると，SQL 文の解析などのためのアクセスと，表に対するアクセスがディスク上で競合するため，どちらか一方が他方のアクセスが終了するまで待たされることになります。

ディスクアクセスの競合を発生させないためのシステム用 RD エリアの配置例を次の図に示します。

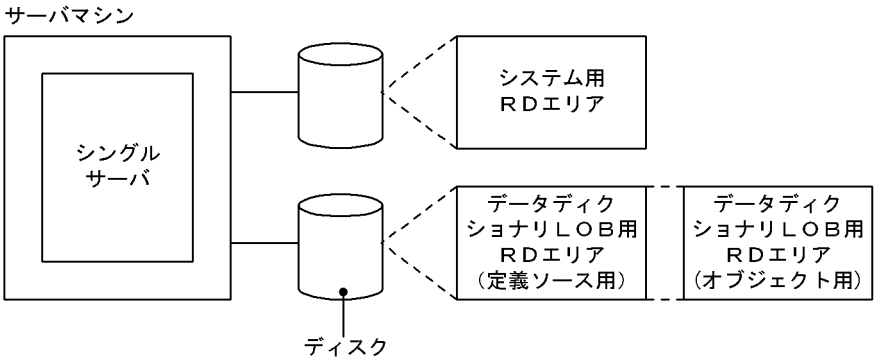
図 7-6　システム用 RD エリアの配置例（HiRDB/シングルサーバの場合）



7.4.2　データディクショナリ LOB 用 RD エリアの配置

ディスクに対するアクセスの競合をなくすため，ほかの RD エリアを配置するディスクとは異なるディスクに配置するようにします。データディクショナリ LOB 用 RD エリアの配置例を次の図に示します。

図 7-7　データディクショナリ LOB 用 RD エリアの配置例（HiRDB/シングルサーバの場合）



データディクショナリ用 RD エリアとの関連

ストアドプロシジャ又はストアドファンクションを管理するディクショナリ表をほかのディクショナリ表とは別のデータディクショナリ用 RD エリアに格納できます。

7.4.3　ユーザ用 RD エリアの配置

(1)　システムログファイルとの関連

システムログファイルを配置したディスクとは異なるディスクに，ユーザ用 RD エリアを配置するようにします。このようにすることで，シンクポイント時のシステムログファイルとユーザ用 RD エリアを構成する HiRDB ファイルへの入出力処理を複数のディスクに分散できるため，シンクポイントでの処理時間を削減できます。

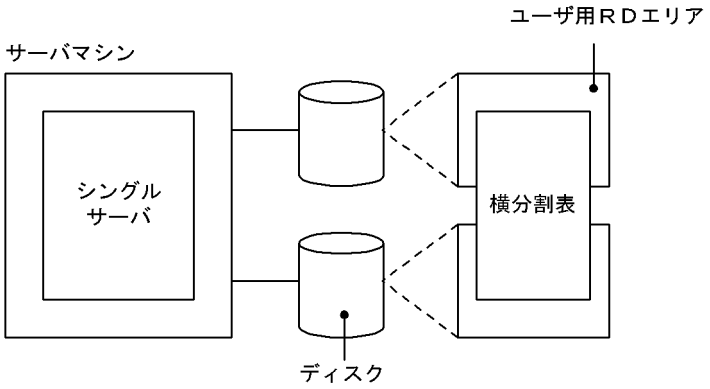
(2)　システム用 RD エリアとの関連

システム用 RD エリアを配置したディスクとは異なるディスクにユーザ用 RD エリアを配置するようにします。

(3)　表を横分割した場合

表を横分割した場合，横分割表を格納する RD エリアを異なるディスクに配置します。ユーザ用 RD エリアの配置例を次の図に示します。

図 7-8　ユーザ用 RD エリアの配置例（HiRDB/シングルサーバの場合）

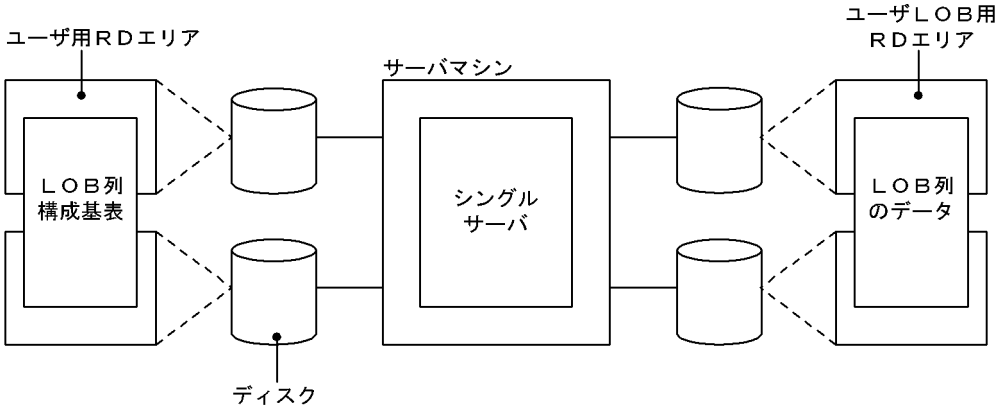


7.4.4　ユーザ LOB 用 RD エリアの配置

ディスクに対するアクセスの競合をなくすため，ユーザ LOB 用 RD エリア以外の RD エリアを配置したディスクとは異なるディスクにユーザ LOB 用 RD エリアを配置するようにします。

また，表を横分割した場合，横分割表を格納する RD エリアを異なるディスクに配置します。ユーザ LOB 用 RD エリアの配置例を次の図に示します。

図 7-9　ユーザ LOB 用 RD エリアの配置例（HiRDB/シングルサーバの場合）



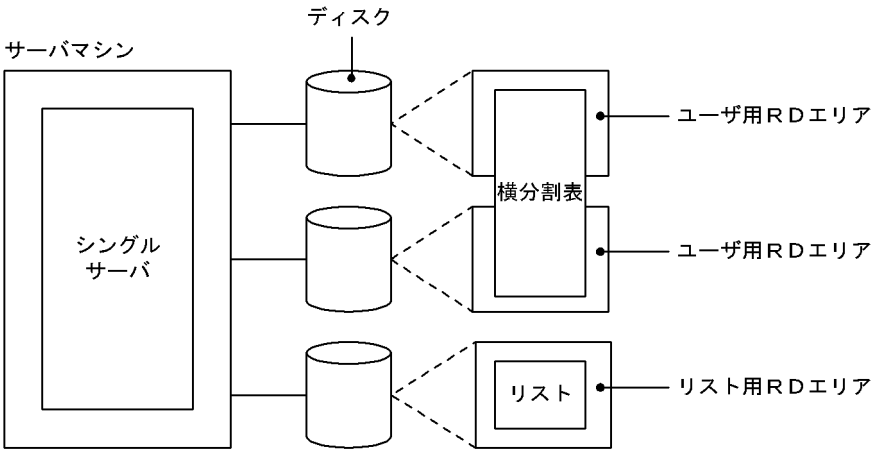
7.4.5　リスト用 RD エリアの配置

ディスクに対するアクセスの競合をなくすため，リスト用 RD エリア以外の RD エリアを配置したディスクとは異なるディスクにリスト用 RD エリアを配置するようにします。

なお，リスト用 RD エリアを一つ以上作成すれば，すべてのユーザ用 RD エリアに格納されている表に対するリストを作成できます。

リスト用 RD エリアの配置例を次の図に示します。

図 7-10　リスト用 RD エリアの配置例（HiRDB/シングルサーバの場合）



8 HiRDB/パラレルサーバの設計

この章では，HiRDB/パラレルサーバのシステム構成，HiRDB ファイルシステム領域の設計方法，システムファイルの設計方法及び RD エリアの配置方法の考慮点について説明します。

8.1　HiRDB/パラレルサーバのシステム設計

ここでは，HiRDB/パラレルサーバのシステム設計とシステム構成について説明します。

8.1.1　システム設計

(1)　サーバ構成

フロントエンドサーバ，ディクショナリサーバ及びバックエンドサーバをそれぞれ一つのサーバマシンに配置するのが，HiRDB/パラレルサーバの基本的なサーバ構成です。ただし，各サーバマシンの CPU の処理負荷が高くない場合，一つのサーバマシンに複数のサーバを配置してもかまいません。一つのサーバマシンに複数のサーバを配置すると，必要とする共用メモリが増加します。共用メモリが不足すると，ユニットが開始できなくなるため，メモリ所要量を十分見積もってください。

設置できるサーバ数の範囲を次の表に示します。

表 8-1　設置できるサーバ数の範囲

項目	設置できるサーバ数の範囲
システムマネジャ数	1
フロントエンドサーバ数	1～1,024
ディクショナリサーバ数	1
バックエンドサーバ数	1～16,382
1 ユニットに作成できるサーバ数	1～34

(2)　システムマネジャの設置

次に示す理由のため，システムマネジャを定義するサーバマシンは HiRDB 管理者が運用しやすい場所に設置することをお勧めします。

- HiRDB 管理者はコマンドで HiRDB を操作しますが，大部分のコマンドはシステムマネジャを定義するサーバマシンから入力する必要がある
- HiRDB システム定義ファイルを共用化するときは，システムマネジャを定義するサーバマシンに共用化するファイルを設置するため

  HiRDB システム定義ファイルの共用化については，「HiRDB システム定義ファイルの共用化（HiRDB/パラレルサーバの場合）」を参照してください。

(3)　フロータブルサーバの設置

HiRDB は結合処理のような複雑な検索処理をする場合，データベースを持たないバックエンドサーバを優先的に使用して，処理性能を向上します。サーバマシンに余裕があり，複雑な検索処理をする場合にフロータブルサーバの設置を検討してください。フロータブルサーバを設置する場合，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。作業表用ファイル用の HiRDB ファイルシステム領域の名称は，バックエンドサーバ定義の pdwork オペランドで指定します。

(4)　フロントエンドサーバの複数化

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，フロントエンドサーバを複数設定します。これを**マルチフロントエンドサーバ**といいます。マルチフロントエンドサーバについては，「マルチフロントエンドサーバの設定」を参照してください。

(5)　HiRDB/パラレルサーバが使用するメモリ

HiRDB/パラレルサーバが使用するメモリについて説明します。

HiRDB/パラレルサーバは次に示すメモリを使用します。

- 共用メモリ
- プロセス固有メモリ

(a)　メモリ所要量

HiRDB/パラレルサーバが必要とするメモリ所要量をサーバマシンごとに見積もってください。HiRDB/パラレルサーバのメモリ所要量については，「HiRDB/パラレルサーバのメモリ所要量の見積もり」を参照してください。

(b)　共用メモリのページ固定

HiRDB では，次に示す共用メモリを実メモリ上に固定できます。

- ユニットコントローラ用共用メモリ
- グローバルバッファ用共用メモリ
- 動的変更したグローバルバッファが使用する共用メモリ
- インメモリデータバッファ用共用メモリ

共用メモリを実メモリ上に固定すると，ページの入出力が少なくなるため，性能が安定します。

前提条件

共用メモリのページ固定をするための，OS ごとの前提条件を次に示します。

OS	前提条件
AIX	64 ビットモードであること。

OS	前提条件
Linux	なし。

動作環境の設定

AIX の場合，オペレーティングシステムパラメタの設定が必要となります。オペレーティングシステムパラメタの設定については，「AIX 固有のパラメタ指定」を参照してください。

ページ固定の方法

共用メモリのページ固定の方法について，共用メモリの種別ごとに説明します。

- ユニットコントローラ用共用メモリ

  システム共通定義，又はユニット制御情報定義の pd_shmpool_attribute オペランドに fixed を指定します。

- グローバルバッファ用共用メモリ

  システム共通定義，又はユニット制御情報定義の pd_dbbuff_attribute オペランドに fixed を指定します。

- 動的変更したグローバルバッファが使用する共用メモリ

  システム共通定義，又はユニット制御情報定義の pd_dbbuff_attribute オペランドに fixed を指定します。これによって，pdbufmod コマンドを実行して動的変更したグローバルバッファが使用する共用メモリも実メモリ上に固定されます。

- インメモリデータバッファ用共用メモリ

  pdmemdb コマンドの-p オプションに fixed を指定します。

注意事項

実メモリ上に連続した領域が確保できなかった場合，共用メモリのページ固定ができません。ページ固定に失敗した場合の HiRDB の動作を次に示します。

OS	ページ固定に失敗した場合の HiRDB の動作			
	ユニットコントローラ用	グローバルバッファ用	動的変更したグローバルバッファ用	インメモリデータバッファ用
AIX	○※	○※	○※	○※
Linux	○	×	×	×

（凡例）

○：ページ固定をしないで共用メモリを確保し，処理を続行します。

×：HiRDB，又はコマンドが異常終了します。

注※

AIX の場合，ページ固定に失敗したときでもシステムコールは正常終了します。そのため，HiRDB からはページ固定に失敗したことが分かりません。ページ固定ができているかどうかは，次の手順で確認してください。

1. HiRDB 稼働中に pdls -d mem コマンドを実行し，次の共用メモリセグメントの識別子を確認します。

・ユニットコントローラ用共用メモリの場合は，SHM-OWNER に MANAGER と表示されている共用メモリセグメント

・上記以外の共用メモリの場合は，SHM-OWNER にユニット名を（）で囲んだ文字列，又は HiRDB サーバ名が表示されている共用メモリセグメント

2. OS の ipcs -s コマンドを実行して，1.で確認した共用メモリセグメントの識別子を持つ共用メモリの SID 値を確認します。

3. 2.で確認した SID 値に対して OS の svmon コマンドを実行して，該当する共用メモリの実メモリページ数と固定されているページ数が一致するかどうかを確認します。

(c) Hugepage 機能を用いた共用メモリの固定（Linux 限定）

Linux 版の HiRDB では，Linux の Hugepage 機能を用いて，次に示す共用メモリを実メモリ上に固定できます。

- ユニットコントローラ用共用メモリ
- グローバルバッファ用共用メモリ
- 動的変更したグローバルバッファが使用する共用メモリ
- インメモリデータバッファ用共用メモリ

Linux の Hugepage 機能を用いて共用メモリを固定すると，共用メモリのページサイズが通常のページサイズ（4KB）から hugepages のページサイズ（2MB）に拡大されます。ページサイズが拡大すると，仮想アドレスを実アドレスに変換するための管理領域（PTE：Page Table Entry）を通常よりも小さくできるため，メモリ不足を防止できます。また，仮想アドレスと物理アドレスの変換回数が少なくなり，ページフォルトの発生を抑えることができます。これによって，トランザクション性能が向上することがあります。

前提条件

Linux の Hugepage 機能を有効にする必要があります。Hugepage 機能については，OS のマニュアルを参照してください。

動作環境の設定

OS のオペレーティングシステムパラメタで，Hugepage 機能を有効にする設定をしてください。OS のオペレーティングシステムパラメタの設定をしないと，共用メモリの確保と固定化に失敗し，HiRDB は起動しません。OS のオペレーティングシステムパラメタに指定する値の詳細については，「Linux の Hugepage 機能の指定」を参照してください。

ページ固定の方法

共用メモリのページ固定の方法について，共用メモリの種別ごとに説明します。

- ユニットコントローラ用共用メモリ

システム共通定義，又はユニット制御情報定義の pd_shmpool_attribute オペランドに hugepage を指定します。

- グローバルバッファ用共用メモリ

  システム共通定義，又はユニット制御情報定義の pd_dbbuff_attribute オペランドに hugepage を指定します。

- 動的変更したグローバルバッファが使用する共用メモリ

  システム共通定義，又はユニット制御情報定義の pd_dbbuff_attribute オペランドに hugepage を指定します。これによって，pdbufmod コマンドを実行して動的変更したグローバルバッファが使用する共用メモリも実メモリ上に固定されます。

- インメモリデータバッファ用共用メモリ

  pdmemdb コマンドの-p オプションに hugepage を指定します。

注意事項

サーバマシン上に十分な hugepages が確保されていないと，共用メモリの確保に失敗します。各共用メモリの確保に失敗した場合，次に示すメッセージを出力して HiRDB の開始処理を中止するか，又はコマンドが異常終了します。

- **ユニットコントローラ用共用メモリの確保に失敗した場合**

  KFPO00113-E メッセージ

- **グローバルバッファ用共用メモリの確保に失敗した場合**

  KFPH23015-E 及び KFPH23005-E メッセージ

- **動的変更したグローバルバッファ用共用メモリの確保に失敗した場合**

  KFPH27031-E メッセージ

- **インメモリデータバッファ用共用メモリの確保に失敗した場合**

  KFPH23208-E メッセージ

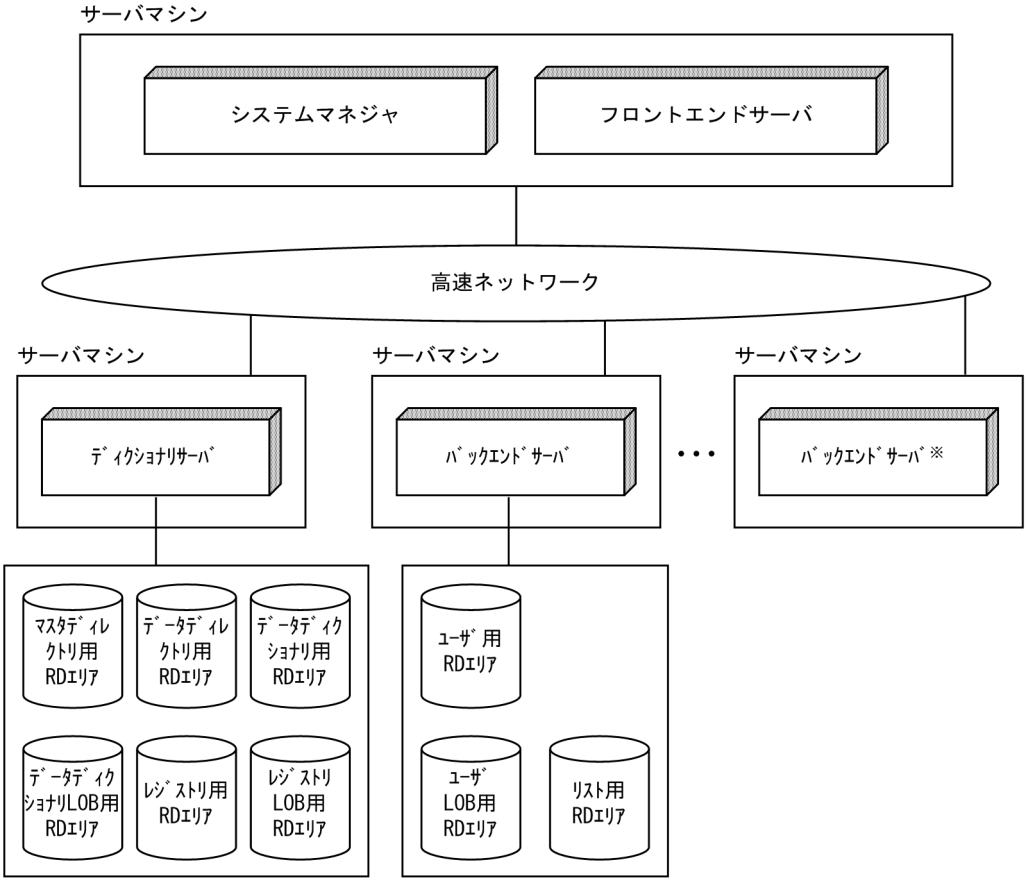
上記の理由で HiRDB の開始処理が中止した場合は，マニュアル「HiRDB メッセージ」を参照して，エラーとなった原因を調査し，対策してください。

8.1.2 HiRDB/パラレルサーバのシステム構成

HiRDB/パラレルサーバのシステム構成例を次の図に示します。

HiRDB/パラレルサーバのシステム構成は，HiRDB システム定義で定義します。HiRDB システム定義の定義例については，マニュアル「HiRDB システム定義」を参照してください。

図 8-1　HiRDB/パラレルサーバのシステム構成例



注※　ＲＤエリアを作成しないで，フロータブルサーバとして使用します。

8.1.3　マルチフロントエンドサーバの設定

HiRDB/パラレルサーバでは複数の SQL を複数のバックエンドサーバを使用して並列に処理しています。フロントエンドサーバは，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や検索結果の編集処理などをしています。したがって，高トラフィックなシステムではフロントエンドサーバの負荷が高くなり，そのため処理性能が向上しなくなります。こういう場合は，フロントエンドサーバを複数設定して，フロントエンドサーバの負荷を分散させてください。これを**マルチフロントエンドサーバ**といいます。

メリット

フロントエンドサーバが稼働するサーバマシンの処理能力ネックを解消し，スケーラブル性をより向上します。

適用基準

SQL 処理の CPU 負荷が高く，一つのサーバマシンだけで処理しきれない場合に適用します。

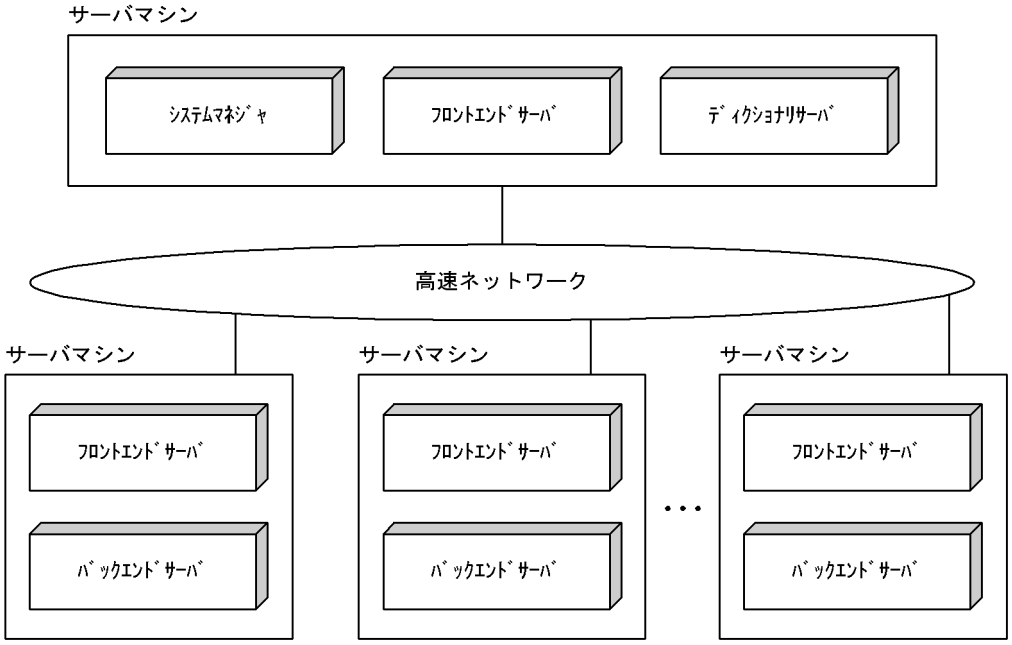
規則

フロントエンドサーバは最大 1024 個設定できます。

サーバマシンとの関係

一つのユニットに複数のフロントエンドサーバを設定できません。また，特定のユニットにフロントエンドサーバを設定しないこともできます。マルチフロントエンドサーバの構成例を次の図に示します。

図 8-2　マルチフロントエンドサーバの構成例



(1)　接続するフロントエンドサーバの選択

複数あるフロントエンドサーバのうち，UAP がどのフロントエンドサーバに接続するかは，次に示すどちらかが決定します。

- クライアントユーザ

  接続するフロントエンドサーバをクライアント環境定義の PDFESHOST オペランドなどで指定します。

- HiRDB

  接続するフロントエンドサーバを HiRDB が自動的に決定します。

  接続するフロントエンドサーバをクライアント環境定義で指定しない場合，HiRDB が任意のフロントエンドサーバに UAP を接続させます。

(2)　環境設定

マルチフロントエンドサーバを実行するために，特に指定は必要ありません。

ただし，次に示すオペランドの指定値に注意してください。このオペランドの指定値の目安については，マニュアル「HiRDB システム定義」を参照してください。

- pd_max_dic_process
- pd_max_bes_process

(3) HiRDB 管理者の運用

HiRDB システム定義のオペランドの指定やシステムファイルの作成個数などの環境設定方法は変わりますが，運用方法は変わりません。

(4) マルチフロントエンドサーバ構成での，挿入又は更新時刻による並べ替え

マルチフロントエンドサーバ構成で，表定義時に CURRENT_TIMESTAMP を既定値とする DEFAULT 句を指定した時刻印型の列を含む表を，行の挿入又は更新時刻で並べ替える場合，注意が必要です。

マルチフロントエンドサーバの場合，UAP と接続したフロントエンドサーバが現在の時刻印を取得し，その値を時刻印列の既定値として設定します。ただし，フロントエンドサーバがあるユニット間でシステムの時刻が異なることがあるため，時刻印列の値で並べ替えた順番と，実際に行を挿入又は更新した時刻で並べ替えた順番が一致しない場合があります。

順番を一致させるには，表定義時に CURRENT_TIMESTAMP USING BES を既定値とする DEFAULT 句を時刻印列に指定します。USING BES を指定すると，その列に既定値を挿入又は既定値で更新時，挿入又は更新する行を格納する RD エリアを管理するバックエンドサーバで現在の時刻印を取得し，その値を挿入，又はその値で更新します。そのため，時刻印型の列で並べ替えた順番と，実際に行を挿入又は更新した時刻で並べ替えた順番は，行を格納する RD エリアを管理するバックエンドサーバがあるユニット単位で一致させることができます。

USING BES の指定有無と現在の時刻印を取得するサーバを次の表に示します。

表 8-2 USING BES の指定有無と現在の時刻印を取得するサーバ

USING BES の指定有無	現在の時刻印を取得するサーバ
なし	UAP と接続したフロントエンドサーバ
あり	挿入，又は更新する行を格納する RD エリアを管理するバックエンドサーバ

注意事項

- 表を分割していて，表格納用 RD エリアを管理するバックエンドサーバが複数のユニットにわたっている場合，時刻印型の列の値で並べ替えた順番と，実際に行を挿入又は更新した時刻で並べ替えた順番とが一致しないことがあります。
- 共用表の場合，排他モードで表に排他を掛けた後でないと，時刻印型の列に既定値を挿入したり，既定値で更新したりできません。
- データベース作成ユティリティ（pdload）を使用して表にデータを格納する場合，起動したユニットでユティリティを起動した時刻印が設定されます。

(5) クライアントユーザの運用

接続するフロントエンドサーバをクライアントユーザが決める場合は，接続するフロントエンドサーバをクライアント環境定義で指定します。**高速接続機能**を使用する場合と **FES ホストダイレクト接続機能**を使

用する場合とで指定するクライアント環境定義が異なります。指定する必要があるクライアント環境定義を次の表に示します。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。

表 8-3　マルチフロントエンドサーバ時に指定するクライアント環境定義

クライアント環境定義のオペランド	接続するフロントエンドサーバを指定しない場合	接続するフロントエンドサーバを指定する場合	
		FES ホストダイレクト接続	高速接続
PDHOST	○	○	○
PDFESHOST	－	○	○
PDNAMEPORT	○	○	○
PDSERVICEPORT	－	－	○
PDSERVICEGRP	－	○	○
PDSRVTYPE	－	－	△

（凡例）

○：必ず指定します。

△：HiRDB サーバが Linux 版又は Windows 版の場合，指定が必要です。

－：指定する必要はありません。

(a)　どのフロントエンドサーバを指定するかの目安

- アクセスする RD エリアを管理するバックエンドサーバがあるサーバマシンのフロントエンドサーバを指定することをお勧めします。
- 用途に応じて接続するフロントエンドサーバを使い分けることをお勧めします。例えば，一般の情報検索処理用，バッチ UAP 処理用，OLTP 下の UAP 処理用などにフロントエンドサーバを使い分けることをお勧めします。

(b)　HiRDB サーバへの接続時間

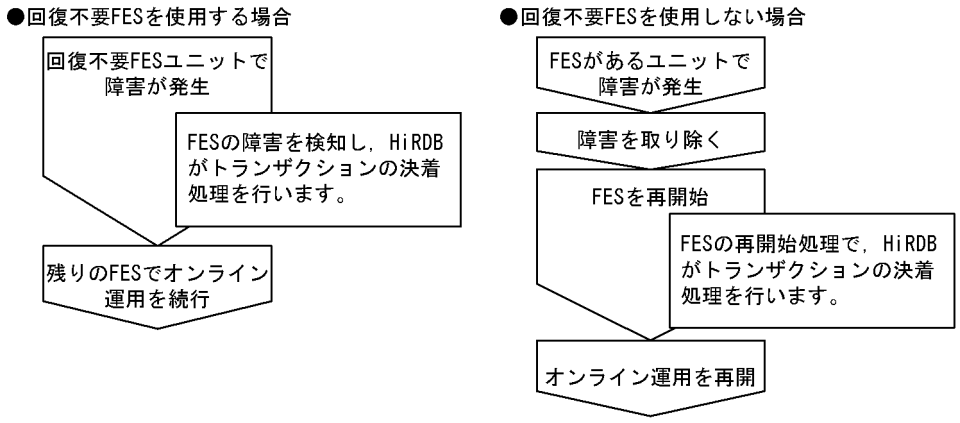
HiRDB サーバへの接続時間は，次に示す順番どおりに短縮されます（1 が最も短縮されます）。

1. 高速接続機能
2. FES ホストダイレクト接続機能
3. 接続するフロントエンドサーバを指定しない場合

8.1.4　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。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 8-3　回復不要 FES を使用する場合としない場合の運用



（凡例）FES：フロントエンドサーバ

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。

メリット

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。

適用基準

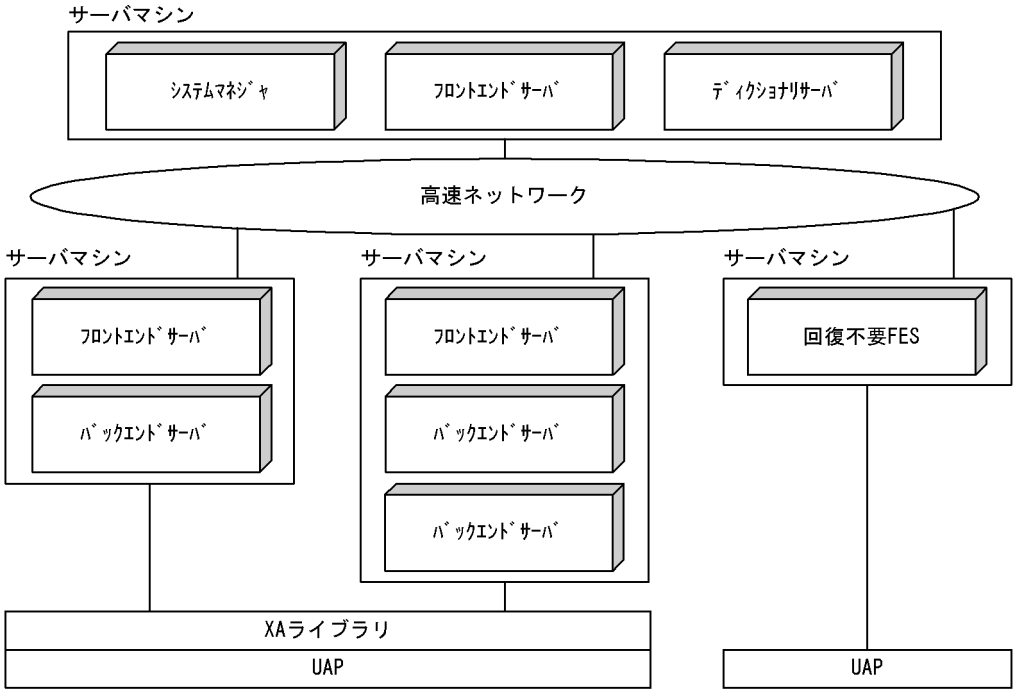
24 時間連続稼働が必要なシステムの場合に適用をお勧めします。

ほかのフロントエンドサーバとの関係

- 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。

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 8-4　回復不要 FES を使用したシステムの構成例



- 回復不要 FES では，反映側 Datareplicator の同期点処理方式に二相コミット方式を利用（反映システム定義 commitment_method オペランドに fxa_sqle を指定）した反映処理を実行できません。反映側 Datareplicator の同期点処理方式に二相コミット方式を利用する場合，反映側 HiRDB に回復不要 FES 以外のフロントエンドサーバを一つ以上配置し，反映側 Datareplicator にクライアント環境変数 PDFESHOST 及び PDSERVICEGRP を設定して，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。

ほかの機能との関連

- 回復不要 FES ユニットでは，系切り替え機能を適用できません。系切り替え機能を適用するシステムの場合，回復不要 FES ユニットのユニット制御情報定義の pd_ha_unit オペランドに必ず nouse を指定してください。

(1)　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。

(2)　注意事項

1. HiRDB を起動する場合に回復不要 FES ユニットが開始しないとき，HiRDB は pd_start_level オペランドの指定値に関係なく，そのユニットを除いて HiRDB を開始します。すべてのフロントエンドサーバを回復不要 FES にする場合，一つ以上のフロントエンドサーバが開始しないと HiRDB システムの起動は完了しません。
2. 回復不要 FES ユニットは独自に縮退起動をするため，pd_start_skip_unit オペランドに回復不要 FES ユニットの名称を指定しても無視します。

3. 回復不要 FES ユニットが異常終了した場合，フロントエンドサーバ及びユニットのステータス情報は STOP(A)になりますが，通常の STOP(A)とは異なり，pdstop コマンドで HiRDB のシステムマネジャとほかのユニットを正常停止，又は計画停止できます。また，回復不要 FES ユニットを強制停止した場合，フロントエンドサーバ及びユニットのステータス情報は STOP(F)になりますが，pdstop コマンドで HiRDB のシステムマネジャとほかのユニットを正常停止，又は計画停止できます。
4. 回復不要 FES ユニットは，次の場合以外，常に正常開始でユニットを開始します。
   - ユニットを正常終了以外で停止していて，かつ前回稼働時に pdstart オペランドの-k オプションに stls を指定していなかった場合
5. 回復不要 FES ユニットのステータス情報が STOP(A)になっていると，その回復不要 FES に CONNECT した UAP からの SQL 要求を HiRDB が受け付けなくなります。この場合，KFPS01820-E メッセージに表示される回復不要 FES のプロセス終了状態は「c800」になります。また，SQL で操作しようとしたデータを持つバックエンドサーバやディクショナリサーバなどのサーバプロセスの終了状態は，KFPS01820-E メッセージに「c900」と表示されることがあります。KFPS01820-E メッセージが表示された場合，プロセス終了状態が「c800」と表示されたフロントエンドサーバがあるユニットを pdstop -z で停止し，STOP(A)になった原因を対策してから再度開始してください。
6. 回復不要 FES ユニットが稼働しているのに，ネットワーク障害などでそのユニットのステータス情報が STOP(A)になった場合，障害が回復してシステムマネジャからそのユニットに通信できるようになると，システムマネジャがそのユニットを自動的に強制停止してから再度開始します。自動的に強制停止してから再度開始する契機を次に示します。
   - ユニットの稼働状況を監視するユニット監視プロセスが，ステータス情報が STOP(A)となった回復不要 FES ユニットが稼働中であることを確認し，KFPS05288-I メッセージが出力されたとき
   - システムマネジャがあるユニットが再度開始する際に全ユニットの稼働状況を確認し，ステータス情報が STOP(A)となった回復不要 FES ユニットが稼働中であることを確認したとき

   自動的に強制停止してから再度開始する処理が行われた場合，回復不要 FES ユニットで KFPS05110-I メッセージが出力されていることを確認してください。このメッセージが出力されていれば，回復不要 FES ユニットの起動処理が正常に終了しています。自動的に強制停止し，再度開始する契機が発生してから，システム共通定義の pd_system_complete_wait_time オペランドに指定した時間が経過しても KFPS05110-I メッセージが出力されない場合，起動処理が正常に終了していません。この場合の対処方法を次に示します。

   (1) システムマネジャがあるユニット，及びディクショナリサーバがあるユニットの稼働状態を pdls -d ust コマンドで確認します。

   それらのユニットが稼働していない場合，pdstart コマンドで開始します。

   それらのユニットが稼働している場合，又はそれらのユニットを開始しても KFPS05110-I メッセージが出力されない場合，(2) の対処をします。

   (2) 回復不要 FES ユニットの稼働状態を pdls -d ust コマンドで確認し，実行結果に応じて，次に示す対処をします。

回復不要 FES ユニットの稼働状態（pdls -d ust コマンドの実行結果）	対処方法
STOP（停止状態）	pdstart -q コマンドを実行して，回復不要 FES ユニットを再開始してください。
PAUSE（プロセスサーバプロセスの再起動中断状態）	1. KFPS00715-E メッセージ，及びそれ以前に syslogfile に出力されたメッセージを参照して，障害要因を取り除いた後，pdrpause コマンドを実行してください。 2. pdstart -q コマンドを実行して，回復不要 FES ユニットを再開始してください。
STARTING（開始途中）	1. pdstop -z コマンドを実行して，回復不要 FES ユニットを強制終了してください。 2. pdstart -q コマンドを実行して，回復不要 FES ユニットを再開始してください。
ONLINE（稼働状態）	
STOPPING（停止途中）	

なお，システムマネジャからそのユニットに通信できるようになる前にユニットが停止した場合は，システムマネジャは強制停止及び再度開始をしません。

7. 回復不要 FES から分岐して，ほかのサーバで実行しているトランザクションは，コミット決着時に分岐先のサーバ間で決着の同期合わせを行います。このため，同期合わせのときに，分岐先サーバのどれかがトランザクション処理を実行できない状態（系切り替え中，サーバ停止状態，サーバ開始準備中，又はサーバ停止準備中）になっていると，トランザクション第 1 状態が READY 又は COMMIT で待ち合わせを行うことがあります。この場合，トランザクション処理を実行できない状態になっている原因を該当するサーバで対策して，トランザクションの決着処理が続行できるようにしてください。
8. 回復不要 FES 機能を使用する FES に接続して実行したトランザクションは，トランザクション第 1 状態，第 2 状態に関係なく，pdcmt，pdrbk，pdfgt コマンドを使用してトランザクションを強制的に終了できないことがあります。この場合，マニュアル「HiRDB システム運用ガイド」の「未決着状態のトランザクションを決着する方法」を参照してトランザクションを自動決着させてください。

8.2 HiRDB ファイルシステム領域の設計

HiRDB のシステム構築時に，HiRDB ファイルを作成する HiRDB ファイルシステム領域を作成します。ここでは，HiRDB ファイルシステム領域を作成するときの設計方針について説明します。

HiRDB ファイルシステム領域は，次に示す用途ごとに作成することをお勧めします。これによって，用途やアクセス特性の異なるファイルへの入出力が競合することを回避できます。また，通常ファイルを使用する場合，用途を明示することによる書き込み性能の向上のほか，用途に応じたデバイスを割り当てられます。

- RD エリア用
- 共用 RD エリア用
- システムファイル用
- 作業表用ファイル用
- ユティリティ用
- リスト用 RD エリア用

8.2.1 RD エリア用の HiRDB ファイルシステム領域の設計

RD エリア用の HiRDB ファイルシステム領域の設計方針について説明します。

(1) 信頼性向上のための方針

1. 更新系処理に対する信頼性は，通常ファイルよりキャラクタ型スペシャルファイルの方が高いです。また，通常ファイルは，OS が異常終了すると使用できなくなる場合があります。したがって，次に示す条件を満たすユーザ用 RD エリア用の 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に作成することをお勧めします。
   - 更新処理が主体となる表を格納するユーザ用 RD エリア
   - 重要度の高いデータを格納するユーザ用 RD エリア
2. 必要な HiRDB ファイルシステム領域の大きさを見積もって，その大きさ以上の HiRDB ファイルシステム領域を作成してください。
3. RD エリア用の HiRDB ファイルシステム領域は，次に示すサーバを定義するサーバマシンに作成します。
   - ディクショナリサーバ
   - バックエンドサーバ
4. 次に示す RD エリアを作成する HiRDB ファイルシステム領域は，ディクショナリサーバを定義するサーバマシンに作成します。
   - システム用 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

5. 次に示す RD エリアを作成する HiRDB ファイルシステム領域は，バックエンドサーバを定義するサーバマシンに作成します。
   - ユーザ用 RD エリア
   - ユーザ LOB 用 RD エリア
6. 系切り替え機能を使用するときは，RD エリア用の HiRDB ファイルシステム領域をキャラクタ型スペシャルファイルに作成してください。

(2) 性能向上のための方針

1. RD エリアを作成する HiRDB ファイルシステム領域は，次に示す RD エリア用ごとに作成することをお勧めします。
   - システム用 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
2. システムファイル用の HiRDB ファイルシステム領域と，RD エリア用の HiRDB ファイルシステム領域は，別々のハードディスクに作成することをお勧めします。これによって，シンクポイントダンプを取得するときに入出力の分散ができ，シンクポイントダンプの取得処理時間を短縮できます。
3. プリフェッチ機能を使用しないときは，シーケンシャルリードに関してはキャラクタ型スペシャルファイルよりも通常ファイルの方が速いです。
4. ランダムな 1 ページリードは，通常ファイルよりもキャラクタ型スペシャルファイルの方が速いです。
5. ライト処理は通常ファイルよりもキャラクタ型スペシャルファイルの方が速いです。
6. 通常ファイルは階層構造で構成されるため，ファイルが大きくなると階層が増えます。階層が増えたファイルにアクセスするときは入出力回数が増えるため，アクセス効率が下がります。
7. 次の表に示すとおりに HiRDB ファイルシステム領域を割り当てることをお勧めします。このようにすると，入出力時間を削減できます。

表 8-4 性能を向上するための HiRDB ファイルシステム領域の割り当て方法

HiRDB ファイルシステム領域の条件	割り当てるファイル
システム用 RD エリアを作成する 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
データディクショナリ LOB 用 RD エリアを作成する HiRDB ファイルシステム領域	

HiRDB ファイルシステム領域の条件	割り当てるファイル
ユーザ LOB 用 RD エリアを作成する HiRDB ファイルシステム領域	
更新の多い表又は少量検索が主体となる表を格納するユーザ用 RD エリアを作成する HiRDB ファイルシステム領域	
データ量が多くて，全件検索又はクラスタキーによる大量キー順検索が主体となる表を格納するユーザ用 RD エリアを作成する HiRDB ファイルシステム領域（ただし，更新をほとんどしない場合）	• 通常ファイル（プリフェッチ機能未使用時） • キャラクタ型スペシャルファイル（プリフェッチ機能使用時）

8.2.2　システムファイル用の HiRDB ファイルシステム領域の設計

システムファイル用の HiRDB ファイルシステム領域の設計方針について説明します。

(1)　信頼性向上のための方針

1. 更新系処理に対する信頼性は，通常ファイルよりもキャラクタ型スペシャルファイルの方が高いです。通常ファイルは，システムダウン後にファイルシステム自体が使えなくなることがあります。このため，システムファイル用の 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に作成します。
2. システムファイル用の HiRDB ファイルシステム領域は二つ以上作成してください。一つしか作成しないと，システムファイルがあるハードディスクに障害が発生した場合，HiRDB が稼働できなくなります。
3. システムファイル用の HiRDB ファイルシステム領域は別々のハードディスクに作成してください。そうすれば，どちらかのハードディスクに障害が発生しても，HiRDB を再開始できます。
4. 必要な HiRDB ファイルシステム領域の大きさを見積もって，その大きさ以上の HiRDB ファイルシステム領域を作成してください。

(2)　性能向上のための方針

システムファイル用の HiRDB ファイルシステム領域と，RD エリア用の HiRDB ファイルシステム領域は，別々のハードディスクに作成することをお勧めします。これによって，シンクポイントダンプを取得するときに入出力の分散ができ，シンクポイントダンプの取得処理時間を短縮できます。

8.2.3　作業表用ファイル用の HiRDB ファイルシステム領域の設計

作業表用ファイル用の HiRDB ファイルシステム領域の設計方針について説明します。

(1) 設計方針

1. 作業表用ファイルは通常ファイルに割り当ててもかまいませんが，系切り替え機能を使用する場合は，キャラクタ型スペシャルファイルに割り当てると共用できるため，ディスク量を削減できます。
2. 作業表用ファイル用の HiRDB ファイルシステム領域長は，その HiRDB ファイルシステム領域に作成する作業表用ファイルの総容量より大きくしてください。なお，pdfmkfs コマンドで-a オプションを指定すると，HiRDB ファイルシステム領域を自動的に拡張できます。作業表用ファイルの総容量が HiRDB ファイルシステム領域長に達した場合，自動で HiRDB ファイルシステム領域を拡張できるため，-a オプションを指定することをお勧めします。※

   作業表用ファイルの容量については，「作業表用ファイルの容量の見積もり」を参照してください。
3. 作業表用ファイル用の HiRDB ファイルシステム領域は，次に示すサーバを定義するサーバマシンに作成してください。
   - ディクショナリサーバ
   - バックエンドサーバ

注※

HiRDB 再開始時に作業表用ファイル用の HiRDB ファイルシステム領域のディスク占有サイズを削減したい場合は，HiRDB 再開始前に pdfmkfs コマンドを実行して作業表用ファイル用の HiRDB ファイルシステム領域を再初期化してください。

(2) 最大使用量を調べる方法

作業表用ファイル用の HiRDB ファイルシステム領域の最大使用量を次に示す方法で調べられます。

pdfstatfs -d 作業表用の HiRDB ファイルシステム領域名

-d：

HiRDB ファイルシステム領域の最大使用量を表示するオプションです。出力情報の peak capacity が最大使用量です。なお，上記の最大使用量は pdfstatfs コマンドでクリアできます。

pdfstatfs -c 作業表用の HiRDB ファイルシステム領域名

-c：

HiRDB ファイルシステム領域の最大使用量をクリアするオプションです。

8.2.4 ユティリティ用の HiRDB ファイルシステム領域の設計

ユティリティ用（バックアップファイル，アンロードデータファイル，又はアンロードログファイル作成用）の HiRDB ファイルシステム領域の設計方針について説明します。ユティリティ用の HiRDB ファイルシステム領域には，次に示すファイルを作成します。

- 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
- 差分バックアップ管理ファイル

(1) 設計方針

1. バックアップファイル作成用にする場合は，キャラクタ型スペシャルファイルに HiRDB ファイルシステム領域を作成してください。
2. バックアップファイル作成用にする場合は，HiRDB ファイルシステム領域長をバックアップ対象 RD エリアの総容量より大きくしてください。RD エリアの容量については，「RD エリアの容量の見積もり」を参照してください。
3. 差分バックアップ管理ファイル作成用の HiRDB ファイルシステム領域は，システムマネジャがあるサーバマシンに作成してください。
4. 系切り替え機能使用時は，アンロードログファイルを共有ディスク（キャラクタ型スペシャルファイル）上に作成してください。
5. アンロードログファイル作成用にする場合は，pdfmkfs コマンドのオプションに次に示す値を指定してください。
   - -k オプション：使用目的には UTL（ユティリティ用の HiRDB ファイルシステム領域）を指定します。
   - -n オプション：HiRDB ファイルシステム領域長には次に示す計算式の値を指定します。

     (アンロードするシステムログファイルの総レコード数※1×システムログファイルのレコード長)※2×作成するアンロードログファイル数×1.2÷1048576
   - -l オプション：最大ファイル数には，作成するアンロードログファイル数を指定します。
   - -e オプション：最大増分回数には，作成するアンロードログファイル数×24 を指定します。

注※1

システムログファイルの自動拡張機能を適用している場合，pd_log_auto_expand_size オペランドの拡張上限サイズに指定した値で計算してください。

注※2

システムログファイルの概算値です。

(2) 最大使用量を調べる方法

ユティリティ用の HiRDB ファイルシステム領域の最大使用量を次に示す方法で調べられます。

pdfstatfs -d ユティリティ用の HiRDB ファイルシステム領域名

-d：

HiRDB ファイルシステム領域の最大使用量を表示するオプションです。出力情報の peak capacity が最大使用量です。なお，上記の最大使用量は pdfstatfs コマンドでクリアできます。

pdfstatfs　-c　ユティリティ用の HiRDB ファイルシステム領域名

-c：

HiRDB ファイルシステム領域の最大使用量をクリアするオプションです。

8.2.5　リスト用 RD エリア用の HiRDB ファイルシステム領域の設計

リスト用 RD エリア用の HiRDB ファイルシステム領域の設計方針について説明します。

(1)　設計方針

1. リストは検索の一時的な中間結果を保存するものなので，ほかの RD エリアほど信頼性は要求されません。したがって，リスト用 RD エリア用の HiRDB ファイルシステム領域は通常ファイルに作成してもかまいません。
2. 系切り替え機能を使用するときは，キャラクタ型スペシャルファイルに作成すると HiRDB ファイルシステム領域を共用できるので，ディスク容量を削減できます。
3. リスト用 RD エリア用の HiRDB ファイルシステム領域は，基表と同じバックエンドサーバに作成してください。

(2)　性能向上のための方針

1. リスト用 RD エリア用の HiRDB ファイルシステム領域を RAID に作成するときは，キャラクタ型スペシャルファイルに作成した方が処理時間が短縮されます。RAID 以外のディスクに作成する場合は，通常ファイルに作成した方が処理時間が短縮されます。
2. リスト用 RD エリア用の HiRDB ファイルシステム領域は，次に示す HiRDB ファイルシステム領域とは別々のハードディスクに作成することをお勧めします。別々のハードディスクに作成すると，リストを検索するときに入出力を分散できるため，処理時間を短縮できます。
   - ユーザ用 RD エリア用の HiRDB ファイルシステム領域
   - ユーザ LOB 用 RD エリア用の HiRDB ファイルシステム領域
   - 作業表用ファイル用の HiRDB ファイルシステム領域

8.2.6　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 8-5　HiRDB ファイルシステム領域の最大長

HiRDB の種類	条件		HiRDB ファイルシステム領域の最大長（単位：メガバイト）
AIX 版	ラージファイル未使用	通常ファイル	2,047
		キャラクタ型スペシャルファイル	
	ラージファイル使用	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	ラージファイル未使用	通常ファイル	2,047
		キャラクタ型スペシャルファイル	
	ラージファイル使用	通常ファイル	1,048,575
		キャラクタ型スペシャルファイル	

8.3　システムファイルの設計

ここでは，システムファイルの設計方針について説明します。

8.3.1　システムログファイルの設計

システムログファイルの設計方針について説明します。

(1)　設計方針

1. システムログファイルはシステムマネジャを除いた各サーバに必要です。
2. サーバ内の全システムログファイルのレコード長及びレコード数を同じにしてください。
3. 各サーバに作成できるシステムログファイルは，2～200 グループです（ただし，6 グループ以上作成することを推奨します）。
4. 一つのサーバに対して，次の式を満たすようにシステムログファイルを作成します。

   100（単位：ギガバイト）×（200-サーバに割り当てるシステムログファイルグループ数）≧サーバに割り当てるシステムログファイルの総容量×3

   これは，システムログファイルの容量不足によって異常終了した HiRDB を再開始する場合に必要になります。システムログファイルが容量不足になった状況や運用によっては，サーバに割り当てられている 3 倍の容量のシステムログファイルを作成し，サーバに追加して対処する必要があるためです。
5. アンロードする回数を少なくしたい場合は，システムログファイルの 1 ファイル容量を大きくします。
6. HiRDB 運用ディレクトリがあるディスクに大量に入出力が発生するファイル（システムログのアンロードログファイルなど）を作成しないでください。
7. 一つのシステムログファイルの容量は，次に示す条件を満たす必要があります。

   一つのシステムログファイルの容量（バイト）≧↑（a＋368）÷c↑×c×b×d

   a：pd_log_max_data_size オペランドの値

   b：pd_log_sdinterval オペランドの値

   c：pd_log_rec_leng オペランドの値

   d：pd_spd_assurance_count オペランドの値
8. 全システムログファイルの容量（二重化している場合は片系の容量だけを対象とする）は，次に示す二つの条件を満たす必要があります。

条件 1

「システムログファイルの総容量の求め方」で見積もった容量以上の値にしてください。

条件 2

長大トランザクションが終了するまでは，長大トランザクション開始以降のシステムログファイルの上書きができません。また，シンクポイントダンプファイルの有効保証世代とカレント世代のシステムログファイルは上書きできません。そのため，これらの分のシステムログファイル容量を確保してください。次の計算式で求めます。

全システムログファイルの総容量（バイト）≧3×a×（b＋1）

a：

データベースを更新するトランザクションのうち，実行時間が最大のトランザクション実行中に該当するサーバで出力されることのあるシステムログ量

システムログ量の見積もり式については，「システムログファイルの容量の見積もり」を参照してください。

b：pd_spd_assurance_count オペランドの値

シンクポイントダンプファイルの有効保証世代数です。

(a)　システムログファイルの世代数と運用への影響

システムログファイルの総容量が同じ場合，システムログファイルの世代数によって，1 世代当たりの容量が変化します。システムログファイルの世代数と運用への影響を次の表に示します。システムログファイルの総容量は同じとします。

表 8-6　システムログファイルの世代数と運用への影響

比較項目	システムログファイルの構成	
	世代数を少なくした場合	世代数を多くした場合
システムログファイル 1 世代当たりの容量	大きくなります。	小さくなります。
スワップ間隔	システムログファイル 1 世代当たりの容量が大きくなるため，スワップ間隔は長くなります。	システムログファイル 1 世代当たりの容量が小さくなるため，スワップ間隔は短くなります。
アンロード回数	スワップ間隔が長くなるため，アンロード回数は少なくなります。	スワップ間隔が短くなるため，アンロード回数は多くなります。
ディスク障害などでシステムログファイルの数世代が使用できなくなった場合のシステムログ容量への影響	• システムログファイル 1 世代当たりの容量が大きくなるため，ディスク障害時にデータベースの回復に用いるログ量は多くなり，データベース回復に掛かる時間は長くなります。 • システムログ容量の減少幅が大きいため，システムログの容量が減少したことによる HiRDB の稼働に及ぼす影響は大きくなります。	• システムログファイル 1 世代当たりの容量が小さくなるため，ディスク障害時にデータベースの回復に用いるログ量は少なくなり，データベース回復に掛かる時間は短くなります。 • システムログ容量の減少幅が小さいため，システムログの容量が減少したことによる HiRDB の稼働に及ぼす影響は小さくなります。

通常の運用では，システムログファイルの世代数を少なくした場合の方がスワップ間隔及びアンロード回数の点で利点があります。ただし，障害発生時には，世代数を多くした場合の方が障害の影響が小さくなります。

(2) 信頼性向上のための方針

(a) システムログファイルの二重化

システムログファイルを二重化すると，HiRDB は両方の系に同じシステムログを取得します。取得したシステムログを読み込むとき，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からシステムログを読み込めるため，システムの信頼性を向上できます。なお，二重化する場合，ディスクをミラー化して二重化するより，HiRDB 管理下で二重化することをお勧めします。システムログファイルを二重化する場合は，それぞれの系のファイルを別々のハードディスクに作成してください。

システムログファイルを二重化する場合は，サーバ定義で次に示すオペランドを指定してください。

- pd_log_dual = Y
- pdlogadpf オペランドの-b オプション（B 系のシステムログファイル名を指定します）

(b) システムログファイルの片系運転

システムログファイルの片系運転は，システムログファイルを二重化する場合に適用されます。

システムログファイルに障害が発生して，両系とも使用できるシステムログファイルがない場合でも，HiRDB のユニットを異常終了しないで正常な系だけで処理を続行できます。これを**システムログファイルの片系運転**といいます。システムログファイルの片系運転をする場合は，サーバ定義で pd_log_singleoperation = Y を指定してください。

システムログファイルの片系運転に対し，両方のシステムログファイルで処理を続行すること（通常の処理形態）を**システムログファイルの両系運転**といいます。

(c) システムログファイルの自動オープン

HiRDB を再開始するときに，上書きできる状態のシステムログファイルがない場合，予約のファイルがあれば HiRDB が予約のファイルをオープンして上書きできる状態にし，処理を続行します。これを**システムログファイルの自動オープン**といいます。

システムログファイルの自動オープンをする場合は，サーバ定義で pd_log_rerun_reserved_file_open = Y を指定してください。

(3) 可用性向上のための方針

(a) 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スワップ時に，スワップ先にできる状態のシステムログファイルがないと HiRDB（ユニット）は異常終了します。これを予防するため，HiRDB にはシステムログファイルの空き容量を監

視する機能（**システムログファイルの空き容量監視機能**）があります。この機能は，システムログファイルの空き率が**警告値**未満になったときに作動します。次に示す二つのレベルのどちらかを選択できます。

レベル 1：

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，警告メッセージ KFPS01162-W を出力します。

レベル 2：

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，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します。このとき，KFPS01160-E メッセージを出力します。これによって，システムログの出力量を抑えます。

レベル 2 を選択した場合，システムログファイルの空き容量が不足したときにサーバ内の全トランザクションが強制終了されます。このため，システムログファイルの設計をより正確に行う必要があります。

システムログファイルの空き容量監視機能については，マニュアル「HiRDB システム運用ガイド」を参照してください。

(b) 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。

(c) シンクポイントダンプ有効化のスキップ回数監視機能

UAP が無限ループしてデータベースを更新し続けるとシンクポイントが有効化できないため，上書きできない状態のシステムログファイルが増えてしまいます。上書きできない状態のシステムログファイルが増えて全システムログファイルが上書きできない状態になると，HiRDB が異常終了します。

また，上書きできない状態のシステムログファイルが，全システムログファイルの半分以上になったときに HiRDB が異常終了又は強制終了すると，HiRDB を再開始するときのロールバック処理でシステムログファイルが不足します。この場合，システムログファイルを新規追加しないと，HiRDB を再開始できません。そして，この再開始処理に要する時間も長くなります。

このようなことを防ぐために，HiRDB では**シンクポイントダンプ有効化のスキップ回数監視機能**を設けています。

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) システムログの並列出力機能（AIX 版及び Linux 版限定）

システムログファイルを二重化している場合に，**システムログの並列出力機能**を使用すると，両系へのログの出力処理を並列して実行できるため，ログ出力に掛かる時間を短縮できます。システムログの並列出力機能を使用するには，**aio ライブラリ**（AIX の場合は Asynchronous I/O Subsystem，Linux の場合は libaio）が必要になります。Asynchronous I/O Subsystem 又は libaio については，OS のマニュアルを参照してください。

(a) お勧めの使い方

システムログの並列出力機能は，サーバごとに適用するかどうか定義できますが，すべてのサーバに適用することをお勧めします。また，ログ出力に掛かる時間がより短縮されるため，A 系と B 系をそれぞれ別デバイスに配置することをお勧めします。

(b) 環境設定（システム定義の設定）

サーバ定義で pd_log_dual_write_method=parallel を指定してください。ただし，次の場合は指定値に関係なく，システムログの並列出力機能は適用されません。

- システムログファイルが二重化されていない（pd_log_dual オペランドに Y が指定されていない）
- システムログファイルがキャラクタ型スペシャルファイルに配置されていない

(c) 環境設定（aio ライブラリの設定）

システムログの並列出力機能を適用するサーバが稼働するすべてのサーバマシンに aio ライブラリを導入し，必要な設定を行ってください。aio ライブラリの導入と設定方法については，OS のマニュアルを参照してください。

(b)で説明した環境設定を行っている状態で，aio ライブラリの導入と設定が正しく行われていない場合，HiRDB を開始できません。

(d) 注意事項

1. システムログの並列出力機能は，通常ファイルに配置されたシステムログファイルには適用されません。そのため，システムログファイルを追加する場合，キャラクタ型スペシャルファイルに配置してください。
2. システムログの並列出力機能が適用されるのは，両系の現用ファイルがキャラクタ型スペシャルファイルに配置されており，かつ両系の現用ファイルにシステムログが出力できる状態の場合（クローズ状態，予約状態，又は障害発生時ではない）だけです。現用ファイルが次の条件に該当する場合，システム定義に関係なく，システムログの並列出力を行いません。
   - A 系又は B 系のどちらかの現用ファイルが通常ファイルに配置されている
   - A 系又は B 系のどちらかの現用ファイルにログが出力できない状態である

3. 系切り替え機能使用時など，システムログの並列出力機能を適用するサーバが複数のサーバマシンで稼働する場合，aio ライブラリを有効化していないサーバマシンがあると，待機系ユニットの開始，又は系切り替えに失敗します。全サーバマシンで aio ライブラリを有効化してください。

(5) システムログファイルのレコード長（既に HiRDB を稼働している場合）

レコード長が 1024 以外の場合，システムログの格納効率を良くするために 1024 に変更することをお勧めします。

システムログファイルのレコード長の変更方法については，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) システムログファイルの定義

作成したシステムログファイルをどのファイルグループに対応させるかをサーバ定義の pdlogadfg 及び pdlogadpf オペランドで定義します。

8.3.2 シンクポイントダンプファイルの設計

シンクポイントダンプファイルの設計方針について説明します。

(1) 設計方針

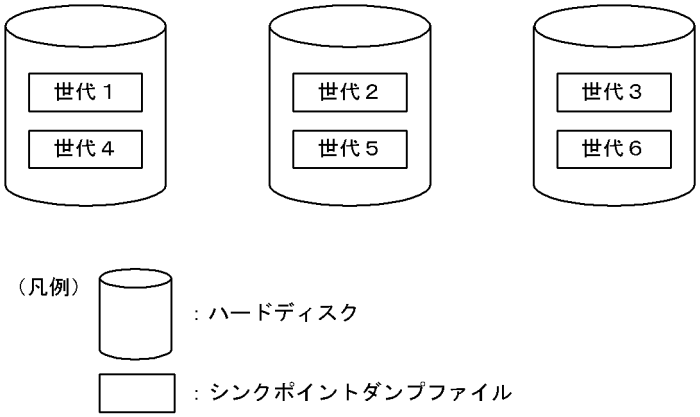
1. シンクポイントダンプファイルはシステムマネジャを除いた各サーバに必要です。
2. 各サーバに作成できるシンクポイントダンプファイルは，2～60 グループです。pdlogadfg -d spd オペランドに ONL を指定した場合は，2～30 グループです。
3. HiRDB は pdlogadfg -d spd オペランドの指定順にシンクポイントダンプファイルを使用します。
4. 各サーバに 4 個以上のシンクポイントダンプファイルを作成することをお勧めします。
5. シンクポイントダンプファイルの容量が不足すると，HiRDB を開始できません。このため，シンクポイントダンプファイルのレコード数は，システム共通定義の最大同時接続数（pd_max_users オペランド）の指定値より大きな値にしてください。詳細なシンクポイントダンプファイルの容量計算式については，「シンクポイントダンプファイルの容量の見積もり」を参照してください。

(2) 信頼性向上のための方針

(a) ファイル構成例

ハードディスク障害に備えて，シンクポイントダンプファイルをそれぞれ別々のハードディスクに作成してください。そのような構成を取れない場合は，隣り合わせの世代を別々のハードディスクに作成するような構成にしてください。例を次の図に示します。

図 8-5　隣り合わせの世代を別々のハードディスクに作成する構成例



(b)　シンクポイントダンプファイルの二重化

シンクポイントダンプファイルを二重化すると，HiRDB はA系及びB系の両方に同じシンクポイントダンプを取得します。取得したシンクポイントダンプを読み込むとき，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からシンクポイントダンプを読み込めるため，システムの信頼性を向上できます。また，二重化している場合，有効保証世代数を 1 世代にすると，信頼性を損ねることなく上書きできない状態のシンクポイントダンプファイル数を削減できます。

シンクポイントダンプファイルを二重化する場合は，サーバ定義で次に示すオペランドを指定してください。

- pd_spd_dual = Y
- pdlogadpf オペランドの-b オプション（B 系のシステムログファイル名を指定します）

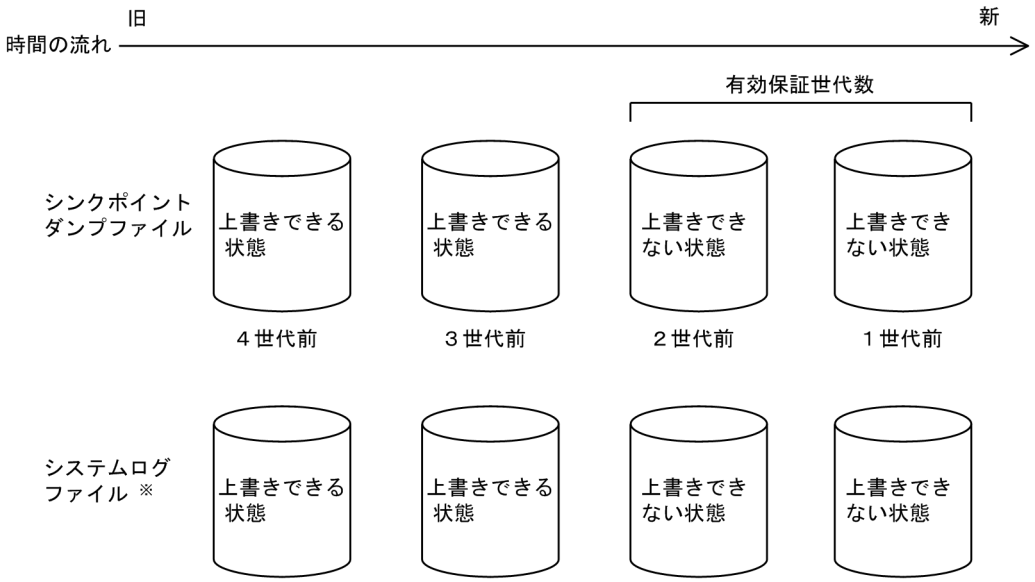
(c)　シンクポイントダンプファイルの片系運転

シンクポイントダンプファイルの片系運転は，シンクポイントダンプファイルを二重化する場合に適用されます。シンクポイントダンプファイルは，片方のファイルに異常が発生しても，正常な系だけで処理を続行します。これをシンクポイントダンプファイルの片系運転といいます。

(d)　シンクポイントダンプファイルの有効保証世代数

一つのシンクポイントダンプファイルには，HiRDB が 1 回に取得するシンクポイントダンプが格納されます。HiRDB は，シンクポイントダンプファイルを世代という概念で管理しています。HiRDB 管理者は，何世代前までのシンクポイントダンプファイルに対応するシステムログファイルを上書きできない状態にするかを指定できます。これを**シンクポイントダンプファイルの有効保証世代数**といいます。シンクポイントダンプファイルの有効保証世代数を次の図に示します。

図 8-6　シンクポイントダンプファイルの有効保証世代数



注※　シンクポイントダンプファイルに対応するシステムログファイルを示します。

［説明］

有効保証世代を 2 とすると，2 世代前までのシンクポイントダンプファイル及びそのシンクポイントダンプファイルに対応するシステムログファイルが，上書きできない状態になります。3 世代前より前の世代のシンクポイントダンプファイル及びそのシンクポイントダンプファイルに対応するシステムログファイルは，上書きできる状態になります。

シンクポイントダンプファイルの運用に必要なファイル数は，**有効保証世代数＋ 1** となります。シンクポイントダンプファイルの有効保証世代数は，サーバ定義の pd_spd_assurance_count オペランドで指定してください。

なお，シンクポイントダンプファイルを二重化している場合，必要な有効保証世代数は 1 世代をお勧めします。二重化していない場合，2 世代をお勧めします。

(e)　シンクポイントダンプファイルの縮退運転

シンクポイントダンプファイルに障害が発生して，運用に必要なファイル数（有効保証世代数＋ 1）を下回った場合でも，最低二つのファイルで処理を続行できます。これを**シンクポイントダンプファイルの縮退運転**といいます。

シンクポイントダンプファイルの縮退運転をする場合は，サーバ定義の pd_spd_reduced_mode オペランドを指定してください。

(f)　シンクポイントダンプファイルの自動オープン

シンクポイントダンプファイルに障害が発生して，運用に必要なファイル数（有効保証世代数＋ 1）を下回った場合，予約のファイルがあれば HiRDB が予約のファイルをオープンして上書きできる状態にし，処理を続行します。これを**シンクポイントダンプファイルの自動オープン**といいます。

シンクポイントダンプの自動オープンをする場合は，サーバ定義で pd_spd_reserved_file_auto_open = Y を指定してください。

(3) シンクポイントダンプファイルの定義

作成したシンクポイントダンプファイルをどのファイルグループに対応させるかを pdlogadfg 及び pdlogadpf オペランドで定義します。

なお，pdlogadfg オペランドだけを指定しておくと，HiRDB 稼働中にシンクポイントダンプファイルを追加できます。

8.3.3 ステータスファイルの設計

ステータスファイルの設計方針について説明します。

(1) 設計方針

1. 両系のファイルに障害が起きないように，A 系と B 系のファイルは別々のディスクに作成します。
2. ステータスファイルの容量不足による HiRDB の異常終了を防ぐため，見積もったファイル容量よりも大きい容量の予備ファイルを幾つか作成してください。ステータスファイルは，容量が満杯になると予備ファイルとスワップしますが，満杯になったステータスファイルと予備ファイルの容量が同じ場合，スワップ先のファイルでも容量不足になり，HiRDB は異常終了します。そのため，例えばステータスファイルを 6 組作成する場合，そのうちの 2 組以上のファイル容量をほかのファイル容量より大きくすることをお勧めします。
3. 各サーバマシンにユニット用ステータスファイルが必要です。
4. システムマネジャを除いた各サーバにサーバ用ステータスファイルが必要です。
5. A 系と B 系のファイルのレコード長及びレコード数を同じにしてください。
6. 一つのユニットに作成できるユニット用ステータスファイルは 1～7 組です。
7. 一つのサーバに作成できるサーバ用ステータスファイルは 1～7 組です。

(2) 信頼性向上のための方針

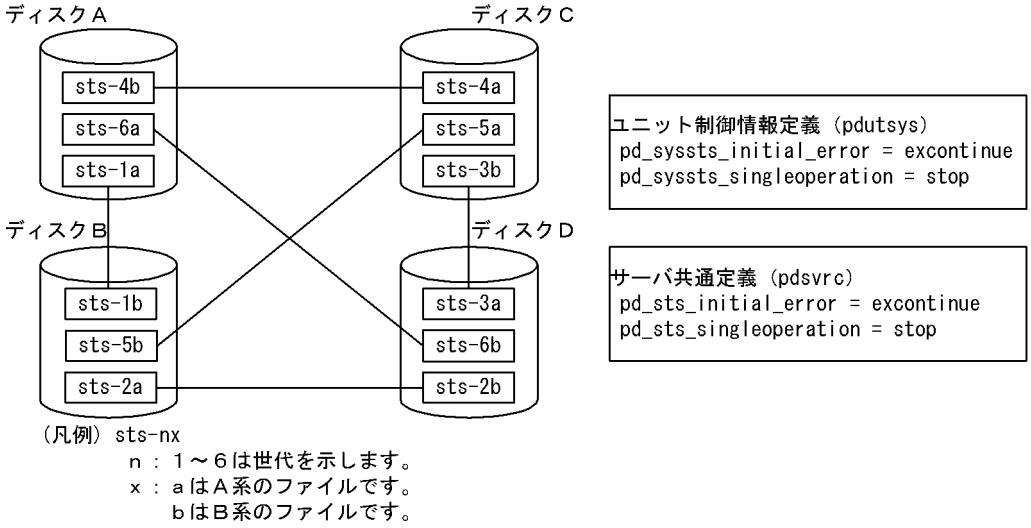
1. ステータスファイルは 3 組（二重化×3=6 ファイル）以上用意し，ディスク障害によってすべてのステータスファイルが障害とならないように配置します。
2. 容量不足による HiRDB の異常終了を防ぐため，ステータスファイルの容量は見積もった容量の 1.2 倍以上の容量を準備することをお勧めします。
3. ステータスファイルには，HiRDB の再開始処理でシステムの状態を回復するために必要な情報を格納しています。予備ファイルがない状態で現用ファイルに障害が発生すると，システムの状態が回復できません。したがって，常に予備ファイルがあるように運用し，現用ファイルの障害に備えてください。

(a) お勧めする構成

ディスク障害発生時から回復時までの安全性を考えると，ステータスファイルは四つのディスクに 6 組（二重化×6=12 ファイル）用意し，次の図のように配置することをお勧めします。また，片系運転中に正常な系に障害が発生すると，HiRDB を再開始できなくなるため，ステータスファイルの片系運転は適用しない（pd_syssts_singleoperation 及び pd_sts_singleoperation に stop を指定）ことをお勧めします。

四つのディスクにステータスファイルを 6 組配置する例を次の図に示します。

図 8-7　四つのディスクに 6 組のステータスファイルを配置する例



［説明］

このように配置すると，あるディスクで障害が発生した後に，更に別のディスクで障害が発生しても，残りの二つのディスクに両系とも正常なファイルが残るため，障害が発生していないディスクのステータスファイルを現用として HiRDB を稼働し続けることができます。例えば，ディスク A に障害が発生し，その後ディスク B にも障害が発生した場合でも，ディスク C 及び D にある両系のステータスファイル（sts-3a と sts-3b）を現用ファイルとして稼働し続けます。この状態で，更に現用ファイルの片系に障害が発生した場合，HiRDB は異常終了しますが，現用ファイルの片系ファイルが正常のため，障害が発生したディスクのどれか一つを回復すると HiRDB を再開始できます。

(3) ステータスファイルの定義

pdstsinit コマンドで作成したステータスファイルをどの論理ファイルに対応させるかを pd_syssts_file_name_1～7 及び pd_sts_file_name_1～7 オペランドで定義します。

ユニット用ステータスファイルは，pd_syssts_file_name_1～7 オペランドで定義します。サーバ用ステータスファイルは，pd_sts_file_name_1～7 オペランドで定義します。

なお，pd_syssts_file_name_2～7 オペランド又は pd_sts_file_name_2～7 オペランドに，実体のないステータスファイルを定義しておくと，HiRDB 稼働中にステータスファイルを追加できます。ただし，この場合，次に示すオペランドを指定しておく必要があります。

ユニット用ステータスファイルの場合

- pd_syssts_initial_error
- pd_syssts_last_active_file

サーバ用ステータスファイルの場合

- pd_sts_initial_error
- pd_sts_last_active_file

(4) ステータスファイルの片系運転

予備ファイルがない状況で現用ファイルの片系に障害が発生した場合，正常な系（片方の系）だけで処理を続行することを**ステータスファイルの片系運転**といいます。ステータスファイルが片系運転になると，KFPS01044-I メッセージが出力されます。

片系運転中に現用ファイルに障害が発生すると，HiRDB を再開始できなくなるため，ステータスファイルの片系運転の適用は推奨しません。ステータスファイルの組数を増やし，予備ファイルがない状況が発生しないような運用をしてください。

なお，ステータスファイルの片系運転に対し，両方の系のステータスファイルで処理を続行すること（通常の処理形態）を**ステータスファイルの両系運転**といいます。

(a) ステータスファイルの片系運転適用のメリット及びデメリット

メリット

予備ファイルがない状況で現用ファイルの片系に障害が発生しても，処理を続行できます。このため，ステータスファイルの障害によって HiRDB が停止する可能性が低くなります。

デメリット

片系運転中に正常な系に障害が発生したり，又はステータスファイルの更新中に HiRDB が異常終了したりすると，現用ファイルの内容が失われるため，HiRDB を再開始できなくなります。

(b) 指定方法

ユニット用ステータスファイルの片系運転をする場合は，ユニット制御情報定義で pd_syssts_singleoperation = continue を指定してください。サーバ用ステータスファイルの片系運転をする場合は，サーバ定義で pd_sts_singleoperation = continue を指定してください。なお，pd_syssts_singleoperation と pd_sts_singleoperation の指定値を同じにしてください。

- ほかのオペランドとの関連

  pd_syssts_singleoperation 及び pd_syssts_initial_error オペランド，又は pd_sts_singleoperation 及び pd_sts_initial_error オペランドの指定値の組み合わせによって，HiRDB の起動時にステータスファイルの障害を検知した場合の HiRDB の動作が決定します。したがって，これら二つのオペランドの指定値は一緒に考えるようにしてください。HiRDB の起動時にステータスファイルの障害を検知し

た場合の HiRDB の動作については，マニュアル「HiRDB システム定義」の，pd_syssts_initial_error 又は pd_sts_initial_error オペランドの説明を参照してください。

(c) 適用の目安

ステータスファイルの片系運転の適用の目安を次に示します。

- HiRDB が再開始できない状態を避けることを重視した運用の場合は，適用しないでください。
- HiRDB がオンラインダウンとなる状態を避けることを重視した運用の場合は，適用してください。
- 系切り替え構成を適用している場合など，HiRDB の再開始を自動で行う運用の場合は，適用しないでください。

(d) 片系運転適用時の注意

片系運転適用の有無による HiRDB の動作及び HiRDB 管理者の処置について次の表に示します。ステータスファイルに障害が発生したときの対処方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

表 8-7 片系運転適用の有無による HiRDB の動作及び HiRDB 管理者の処置

<table>
<tr><th colspan="2" rowspan="2">条件</th><th colspan="2">ステータスファイルの片系運転
(pd_syssts_singleoperation 又は pd_sts_singleoperation オペランドの指定値)</th></tr>
<tr><th>適用する（continue 指定）</th><th>適用しない（省略又は stop 指定）</th></tr>
<tr><td rowspan="2">予備ファイルがある</td><td>現用ファイルに障害が発生</td><td colspan="2">HiRDB の動作
ステータスファイルをスワップします。
HiRDB 管理者の処置
障害が発生したステータスファイルの障害対策をしてください。</td></tr>
<tr><td>現用ファイルの両系に，同時に障害が発生</td><td colspan="2">HiRDB の動作
異常終了します。HiRDB は再開始できません。
HiRDB 管理者の処置
マニュアル「HiRDB システム運用ガイド」の「ステータスファイルに障害が発生したときの対処方法」を参照してください。</td></tr>
<tr><td rowspan="2">予備ファイルがない</td><td>現用ファイルの片系に障害が発生</td><td>HiRDB の動作
片系運転を適用し，処理を続行します。
HiRDB 管理者の処置
早急に予備のファイルを作成して，両系運転の状態に戻してください。</td><td>HiRDB の動作
異常終了します。
HiRDB 管理者の処置
予備ファイルを作成し，HiRDB を再開始してください。</td></tr>
<tr><td>現用ファイルの両系に，同時に障害が発生</td><td colspan="2">HiRDB の動作
異常終了します。HiRDB は再開始できません。
HiRDB 管理者の処置
マニュアル「HiRDB システム運用ガイド」の「ステータスファイルに障害が発生したときの対処方法」を参照してください。</td></tr>
</table>

条件		ステータスファイルの片系運転 (pd_syssts_singleoperation 又は pd_sts_singleoperation オペランドの指定値)	
		適用する（continue 指定）	適用しない（省略又は stop 指定）
	片系運転中に，正常な系に障害が発生	HiRDB の動作 異常終了します。HiRDB は再開始できません。 HiRDB 管理者の処置 マニュアル「HiRDB システム運用ガイド」の「ステータスファイルに障害が発生したときの対処方法」を参照してください。	－

（凡例）－：該当しません。

(5) ステータスファイルの障害に関する注意事項（重要）

- 現用ファイルの両系に同時に障害が発生した場合，HiRDB が異常終了し，再開始できなくなります。対策として，物理ディスクの多重化（ミラーリング）が考えられます。
- HiRDB の開始前に，現用ファイル（終了時の現用ファイル）を削除，又は pdstsinit コマンドでステータスファイルを初期化した場合，HiRDB は再開始できなくなります。

8.4 RD エリアの配置

ここでは，次に示す RD エリアを配置するときの考慮点について説明します。

- システム用 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
- リスト用 RD エリア

8.4.1 システム用 RD エリアの配置

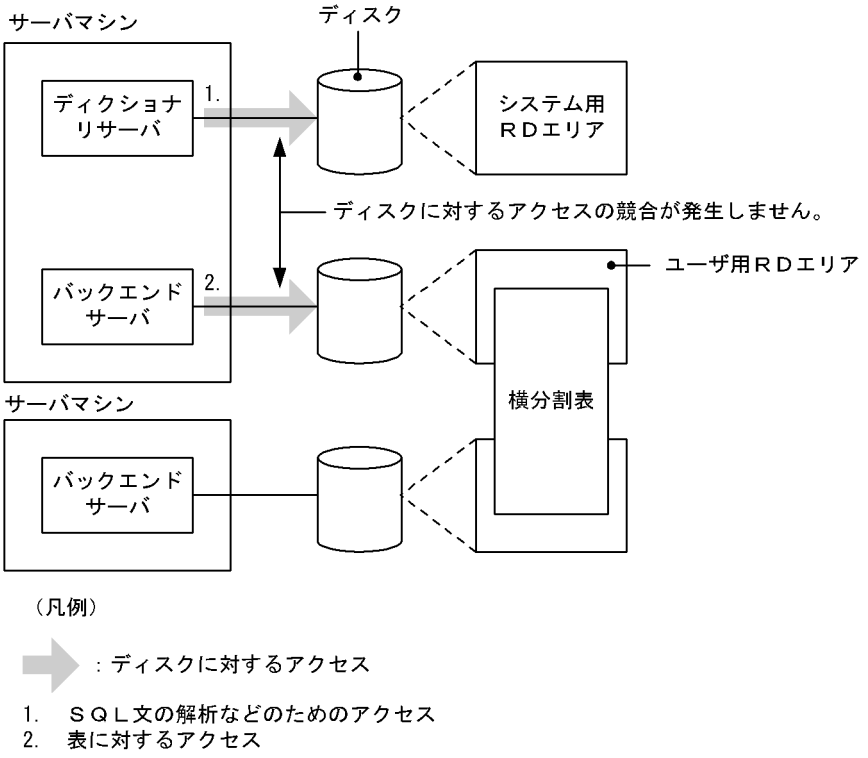
システム用 RD エリアは，ユーザ用 RD エリアの配置を考慮して配置します。システム用 RD エリアを配置するときの考慮点を次に示します。

- **システム用 RD エリアはディクショナリサーバに配置します。**
- **ディクショナリサーバとバックエンドサーバが同一のサーバマシンにある場合は，ユーザ用 RD エリアを配置するディスクとは異なるディスクにシステム用 RD エリアを配置するようにします。**

システム用 RD エリアのうち，特にデータディクショナリ用 RD エリアとデータディレクトリ用 RD エリアは，SQL 文の解析などのために HiRDB にアクセスされることが多くなります。このため，ユーザ用 RD エリアを配置するディスクと同じディスクに配置すると，SQL 文の解析などのためのアクセスと，表に対するアクセスがディスク上で競合するため，どちらか一方が他方のアクセスが終了するまで待たされることになります。

ディスクアクセスの競合を発生させないためのシステム用 RD エリアの配置例を次の図に示します。

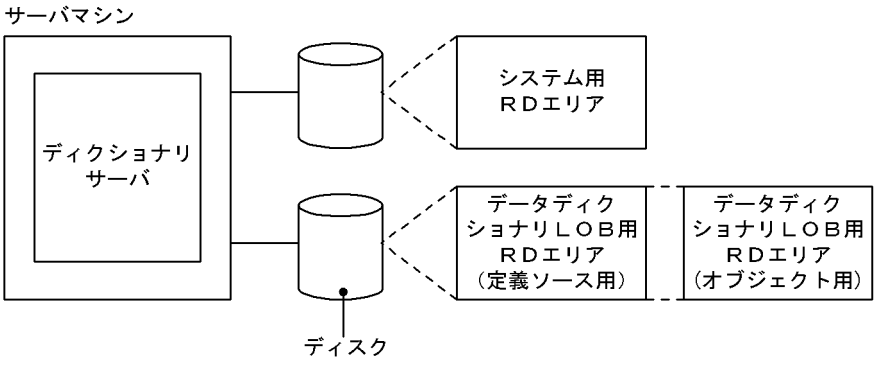
図 8-8　システム用 RD エリアの配置例（HiRDB/パラレルサーバの場合）



8.4.2　データディクショナリ LOB 用 RD エリアの配置

ディスクに対するアクセスの競合をなくすため，ほかの RD エリアを配置するディスクとは異なるディスクに配置するようにします。データディクショナリ LOB 用 RD エリアの配置例を次の図に示します。

図 8-9　データディクショナリ LOB 用 RD エリアの配置例（HiRDB/パラレルサーバの場合）



データディクショナリ用 RD エリアとの関連

ストアドプロシジャ又はストアドファンクションを管理するディクショナリ表をほかのディクショナリ表とは別のデータディクショナリ用 RD エリアに格納できます。

8.4.3　ユーザ用 RD エリアの配置

(1)　システムログファイルとの関連

システムログファイルを配置したディスクとは異なるディスクに，ユーザ用 RD エリアを配置するようにします。このようにすることで，シンクポイント時のシステムログファイルとユーザ用 RD エリアを構成する HiRDB ファイルへの入出力処理を複数のディスクに分散できるため，シンクポイントでの処理時間を削減できます。

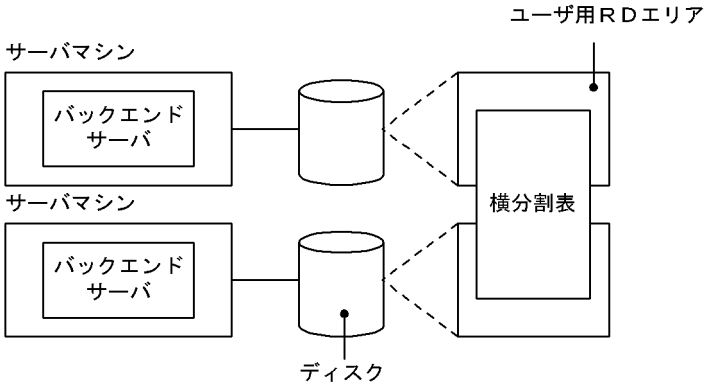
(2)　システム用 RD エリアとの関連

システム用 RD エリアを配置したディスクとは異なるディスクにユーザ用 RD エリアを配置するようにします。

(3)　表を横分割した場合

表を横分割した場合，横分割表を格納する RD エリアを異なるバックエンドサーバに配置します。また，横分割表を格納する RD エリアを異なるディスクに配置します。ユーザ用 RD エリアの配置例を次の図に示します。

図 8–10　ユーザ用 RD エリアの配置例（HiRDB/パラレルサーバの場合）



(4)　フロータブルサーバの設置

複数のバックエンドサーバにわたる結合処理やソート処理などの，表に対する複雑な問い合わせ処理をするときは，ユーザ用 RD エリアの配置に注意します。

すべてのバックエンドサーバにユーザ用 RD エリアを配置すると，あるバックエンドサーバがユーザ用 RD エリアに対するアクセスのほか，複雑な問い合わせ処理をするため，このバックエンドサーバの負荷が高くなります。このため，システム全体のスループットが低下することがあります。

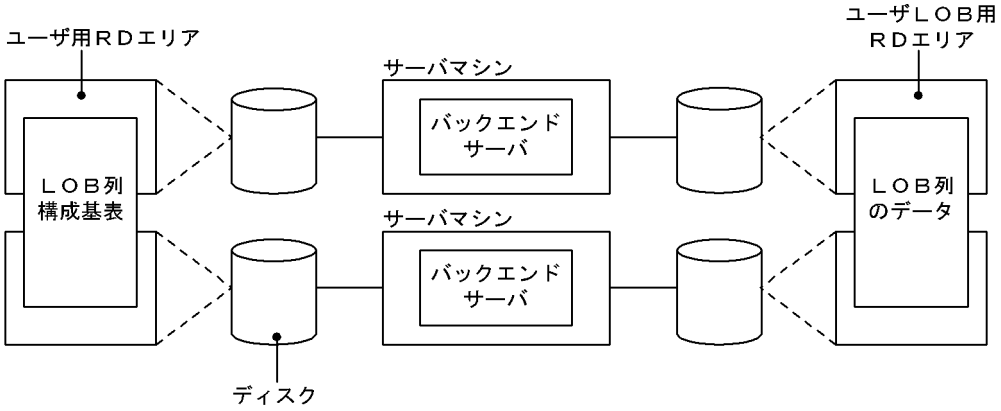
よって，サーバマシンの台数に余裕がある場合，ユーザ用 RD エリアを配置しないバックエンドサーバ（フロータブルサーバ）を設置します。フロータブルサーバを設置すると，複雑な問い合わせ処理をフロータブルサーバに割り当てるため，各バックエンドサーバの負荷を軽減できます。

8.4.4　ユーザ LOB 用 RD エリアの配置

ディスクに対するアクセスの競合をなくすため，ユーザ LOB 用 RD エリア以外の RD エリアを配置したディスクとは異なるディスクにユーザ LOB 用 RD エリアを配置するようにします。

また，LOB データを格納しているユーザ LOB 用 RD エリアと，LOB 列構成基表を格納しているユーザ用 RD エリアとは同一のバックエンドサーバに配置します。ユーザ LOB 用 RD エリアの配置例を次の図に示します。

図 8-11　ユーザ LOB 用 RD エリアの配置例（HiRDB/パラレルサーバの場合）

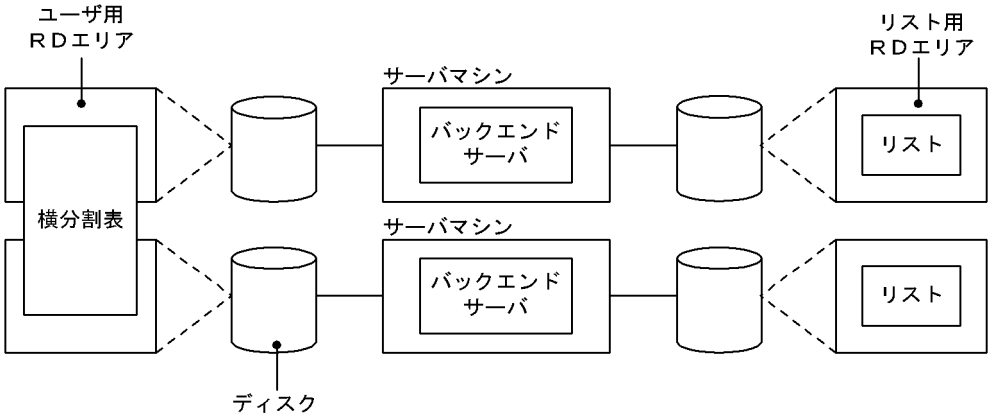


8.4.5　リスト用 RD エリアの配置

基表があるバックエンドサーバにリスト用 RD エリアを配置します。リスト用 RD エリアを一つ以上作成すれば，そのバックエンドサーバにあるすべての表に対するリストを作成できます。

また，ディスクに対するアクセスの競合をなくすため，リスト用 RD エリア以外の RD エリアを配置したディスクとは異なるディスクにリスト用 RD エリアを配置するようにします。リスト用 RD エリアの配置例を次の図に示します。

図 8-12　リスト用 RD エリアの配置例（HiRDB/パラレルサーバの場合）



8.5　ユニット数又はサーバ数が多いシステムを構築する場合の考慮点

ここでは，ユニット数又はサーバ数が多いシステムを構築，運用する場合の考慮点について説明します。目安として，10 ユニット以上，又は 10 サーバ以上のシステムを構築して運用する場合にお読みください。

なお，ここでいうサーバとは，フロントエンドサーバ，ディクショナリサーバ，及びバックエンドサーバのことです。これらのサーバが 10 サーバ以上の場合にお読みください。

8.5.1　システム構築時の考慮点

(1)　システム定義の設定

ユニット数又はサーバ数が多いシステムを構築する場合，HiRDB で使用するポートを固定して通信負荷を軽減する必要があります。そのため，次の表に示すシステム定義のオペランドを指定してください。

表 8-8　通信負荷を軽減するために指定する必要があるオペランド

項番	オペランド名	説明
1	pd_name_fixed_port_lookup=Y	このオペランドに Y を指定して，自ユニットの共用メモリ情報を使って，ほかのユニットと通信をするようにしてください。また，併せて項番 2 のオペランドを指定してください。
2	• pd_mlg_port 又は pdunit の-m オプション • pd_alv_port 又は pdunit の-a オプション • pd_trn_port 又は pdunit の-t オプション • pd_scd_port 又は pdunit の-s オプション • pd_name_port 又は pdunit の-p オプション	－
3	pd_ipc_conn_nblock_time	HiRDB のサーバ間通信を行う際に，稼働していないサーバがあると，このオペランドに指定した時間まで次の処理を待ち合わせます。そのため，このオペランドの指定値が大き過ぎると，性能低下の原因になります。 ネットワークの負荷が高くないシステムの場合は，このオペランドに 2（秒）を指定してください。 ネットワーク負荷の高いシステムの場合は，次に示すどちらかの処置をしてください。 • ネットワーク環境に合わせて，このオペランドの指定値を決める • ネットワーク環境を見直して，ネットワーク環境の増強を検討する
4	pd_bes_connection_hold=Y	特にありません。
5	pd_bes_conn_hold_trn_interval	UAP の接続時間（SQL の CONNECT から DISCONNECT までの間）が短い場合は，このオペランドに 0 を指定してください。

（凡例）－：該当しません。

(2)　高速接続機能の設定

高速接続機能を使用して通信負荷を軽減してください。高速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。

(3)　影響分散スタンバイレス型系切り替え機能の設定

一つの HA グループに定義できるユニット数の上限は 32 です。そのため，ユニット数が 33 以上の場合は，HA グループを複数定義してください。影響分散スタンバイレス型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。

(4)　ファイルのオープン数の設定

ユニット数又はサーバ数が多い場合，ファイルのオープン数の上限値について考慮する必要があります。単一プロセスがオープン又はロックできるファイル数の物理限界値を設定するオペレーティングシステムパラメタ（AIX の場合は nofiles_hard，Linux の場合は hard nofile）の指定値を検討してください。オペレーティングシステムパラメタについては，「OS のオペレーティングシステムパラメタの見積もり」を参照してください。

(5)　Listen キュー不足の発生を抑える設定

通信時に使用する TCP/IP の Listen キューが不足することがあります。Listen キューの指定値が小さ過ぎないか確認してください。Listen キューの指定方法については，「Listen キュー指定値」を参照してください。

(6)　ポート数不足の発生を抑える設定

通信時に使用するポート数が不足することがあります。ポート数不足の発生を抑えるようにしてください。設定方法については，「ポート数不足を回避する方法」を参照してください。

8.5.2　システム運用時の考慮点

トランザクション又はコマンド（ユティリティを含む）の実行時，次に示す現象が発生して通信エラーとなることがあります。その結果，トランザクション又はコマンドがエラーとなったり，ホスト間監視によって，ユニットの異常が検知（KFPS05289-E メッセージが出力）されたりすることがあります。

- リモートシェルが使用するポート数が不足する

  HiRDB のコマンドを実行すると，内部的にリモートシェルが実行されます。そのため，コマンドの実行対象となるユニットやサーバ数が多い場合，リモートシェルが使用するポート数が不足することがあります。

- システムで使用するポート数が不足する

ユニット又はサーバ数が多い場合，HiRDB のサーバ間通信の接続数が増加し，システムで使用するポート数が不足することがあります。

- **ネットワーク帯域が不足する**

  ユニット又はサーバ数が多い場合，HiRDB のサーバ間通信の接続数が増加し，ネットワーク帯域が不足することがあります。

このような現象が発生した場合，次に示す方法で通信負荷を軽減してください。

(1) コマンドを実行するときにユニット名を指定する

次に示すコマンドを実行するときは，ユニット名を指定してください。ユニット名を指定すると，ユニット単位でコマンドが実行されるため，コマンドを実行する際に使用されるポート数を削減できます。

- pdaudbegin，pdaudend，pdaudrm，pdaudswap
- pdcancel
- pdcat
- pdrisechk
- pdstscls，pdstsinit，pdstsopen，pdstsrm，pdstsswap
- pdstbegin，pdstend
- pdcmt，pdrbk，pdfgt
- pdls（サーバ名を指定してもよい）

ただし，ユニット単位にコマンドを実行した場合でも，そのコマンド処理の延長でほかのユニットに処理要求が発生するときは一時的にポートが不足することがあります。その場合はしばらく時間を置いてからコマンドを再度実行してください。

(2) コマンドを実行するときにサーバ名を指定する

次に示すコマンドを実行するときは，サーバ名を指定してください。サーバ名を指定すると，サーバ単位でコマンドが実行されるため，コマンドを実行する際に使用されるポート数を削減できます。

- pdchprc
- pdclttrc
- pdjarsync
- pdlogadpf，pdlogchg，pdlogcls，pdloginit，pdlogls，pdlogopen，pdlogrm，pdlogswap，pdlogsync，pdlogunld，pdlogatul
- pdobils
- pdpfresh

ただし，サーバ単位にコマンドを実行した場合でも，そのコマンド処理の延長でほかのユニットに処理要求が発生するときは一時的にポートが不足することがあります。その場合はしばらく時間を置いてからコマンドを再度実行してください。

(3) pddbst コマンドの実行時に通信負荷を軽減する方法

pddbst コマンドを実行する場合は，状態解析を RD エリア単位で実行してください。

(4) pdload コマンドの実行時に通信負荷を軽減する方法

pdload コマンドを実行する場合，入力ファイルを分割格納条件ごとに作成し，RD エリア単位に pdload コマンドを実行すると，通信負荷を軽減できます。

また，作成した複数の入力ファイルを 1 か所（同一サーバマシン上）に配置するのではなく，表格納 RD エリアがあるサーバマシンに配置すると，通信負荷を軽減できます。

(5) pdrorg コマンドの実行時に通信負荷を軽減する方法

pdrorg コマンドで表の再編成，表のアンロード，表のリロードを実行する場合，RD エリア単位又はサーバ単位で pdrorg コマンドを実行すると，通信負荷を軽減できます。

pdrorg コマンドでインデクスの再作成，インデクスの再編成，インデクスの一括作成を実行する場合，インデクス単位又はサーバ単位で pdrorg コマンドを実行すると，通信負荷を軽減できます。

(6) pdcopy 又は pdrstr コマンドの実行時に通信負荷を軽減する方法

pdcopy 又は pdrstr コマンドを実行する場合，次に示す方法で通信負荷を軽減できます。

- -s オプションにサーバ名を一つだけ指定してコマンドを実行する
- -r オプションに複数の RD エリア名を指定する場合，同じバックエンドサーバに属する RD エリアだけを指定してコマンドを実行する

また，コマンドの処理対象となるサーバマシンにバックアップファイルを配置すると，通信負荷を軽減できます。

(7) SQL の実行時に通信負荷を軽減する方法

一つのトランザクションで複数のバックエンドサーバ下のデータをアクセスする場合，サーバ間でデータ通信が発生するため，サーバ間のデータ通信経路を極力減らすようにしてください。次に示す条件を一つでも満たす場合，複数のバックエンドサーバ下のデータをアクセスします。

- 複数のバックエンドサーバに横分割した表を指定した SQL を発行する
- FROM 句に二つ以上の表を指定した SQL を発行する
- 副問合せを指定した SQL を発行する

- 集合演算を指定した SQL を発行する
- 共用表のデータを更新する SQL を発行する
- 同一トランザクション内に複数の SQL を発行する場合，それぞれの SQL の FROM 句に異なるバックエンドサーバに定義された表を指定する

これらの条件を一つでも満たす場合は，次に示す対策をして，サーバ間のデータ通信経路を削減してください。

- 横分割表の分割数を少なくする
- SQL の探索条件に分割キーに対する条件を指定する
- 表の結合時は，対象となる表の分割を合わせ，結合キーを分割キーとする
- トランザクションがアクセスする複数の表のデータを同じバックエンドサーバに格納する

(8) フロータブルサーバの使用時に通信負荷を軽減する方法

フロータブルサーバを使用すると通信負荷が上がるため，フロータブルサーバを極力使用しないでください。次に示す条件を一つでも満たす場合，フロータブルサーバが使用されます。

- FROM 句に二つ以上の表を指定した SQL を発行する（結合方式にネストループジョインを適用する場合を除く）
- 副問合せを指定した SQL を発行する
- 集合演算を指定した SQL を発行する
- ORDER BY 句を指定した SQL を発行する（ORDER BY のためのソート処理をしなくても，インデクスを検索することで ORDER BY に含まれる列のソート順を保証できる場合を除く）
- GROUP BY 句を指定した SQL を発行する
- DISTINCT を指定した SQL を発行する
- ビュー表，WITH 句，又は FROM 句に導出表を指定した SQL を発行する（ビュー表，WITH 句を指定した SQL で内部導出表を作成しない場合を除く）

  内部導出表を作成する条件については，マニュアル「HiRDB SQL リファレンス」を参照してください。
- FOR READ ONLY 句を指定した SQL を発行する

これらの条件を一つでも満たす場合は，次に示す対策をすると，使用するフロータブルサーバ数を削減できることがあります。

- SQL 最適化オプションに"FLTS_ONLY_DATA_BES"を指定する
- SQL 最適化オプションに"SORT_DATA_BES"を指定する

SQL 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。

(9) 共用表の使用時に通信負荷を軽減する方法

複数のフロントエンドサーバを経由して共用表を更新すると，通信負荷が上がります。そのため，共用表の更新をする場合は，HiRDB クライアントが接続するフロントエンドサーバができるだけ同じになるようにしてください。

(10) ユティリティの同時実行数に関する注意事項

バックエンドサーバ数が多い場合，ユティリティの同時実行数が多いと，ユティリティが異常終了することがあります。この場合，同時に実行するユティリティの数を減らすなどの対処をしてください。

(11) 接続ユーザ情報ファイルが出力されない場合の対処方法

KFPS05120-W メッセージが出力された場合に，$PDDIR/spool/cnctusrinf 下に接続ユーザ情報ファイルが出力されないことがあります。この場合，pdls -d act，pdls -d prc，及び pdls -d trn コマンドを各ユニットに対して実行してください。

接続ユーザ情報ファイルについては，マニュアル「HiRDB システム運用ガイド」を参照してください。

(12) 制限事項

- ユニット数が 65 以上の場合，MIB パフォーマンス情報監視機能は使用できません。
- リモートシェルが使用できるポート数の上限を超えた場合，次に示すコマンドは実行できません。
  - pdconfchk
  - pdls -d rpc

8.5.3 コマンドの実行時にエラーが発生したときの対処方法

コマンドの実行時にエラーが発生したときの対処方法を次の表に示します。

表 8-9 コマンドの実行時にエラーが発生したときの対処方法

実行したコマンド	エラーの内容と対処方法
pdchgconf	pdchgconf コマンドは，リモートシェルが使用できるポート数の上限を超えた場合には実行できません。この場合，HiRDB を pdstop コマンドで正常終了してから構成変更をしてください。
pdtrndec	pdtrndec コマンドは，リモートシェルが使用できるポート数の上限を超えた場合には実行できません。この場合，未決着状態のトランザクションを手動で決着させてください。未決着状態のトランザクションを手動で決着させる方法については，マニュアル「HiRDB システム運用ガイド」の「未決着状態のトランザクションがあるときの対処方法」を参照してください。

実行したコマンド	エラーの内容と対処方法
pdprgcopy 又は pdprgrenew	pdprgcopy 又は pdprgrenew コマンドは，リモートシェルが使用できるポート数の上限を超えた場合には実行できません。この場合，HiRDB がインストールされているサーバマシンに，それぞれ修正版をインストールして入れ替えを行ってください。
pdstart	pdstart コマンドは，リモートシェルが使用できるポート数の上限を超えた場合には実行できません。この場合，HiRDB/パラレルサーバを構成する各サーバマシンで pdstart -q コマンドを実行して HiRDB を開始してください。
pdorend	pdorend コマンドでオンライン再編成の追い付き反映をする際に，サーバ数が多いと通信負荷が大きくなり，-w オプションに指定した追い付き反映処理の最大待ち時間を超えてエラーになることがあります。この場合，次に示す対処をしてください。 • -w オプションの指定値を大きくする • pdorend コマンド実行中のオンライン業務のトランザクション量を抑える

9 マルチ HiRDB の設計

この章では，マルチ HiRDB のシステム設計で検討する項目について説明します。

9.1　マルチ HiRDB のシステム設計

マルチ HiRDB のシステムを設計する場合に，通常の HiRDB と設計方法が異なる箇所についてだけ説明します。

9.1.1　マルチ HiRDB のインストール

マルチ HiRDB をインストールする場合に気を付けることについて説明します。

(1)　HiRDB 管理者の登録

マルチ HiRDB のサーバでは，HiRDB ごとに別々の HiRDB 管理者を登録してください。HiRDB 管理者の登録については，「HiRDB 管理者の登録」を参照してください。

(2)　HiRDB 運用ディレクトリの作成

マルチ HiRDB のサーバでは，HiRDB ごとに異なる HiRDB 運用ディレクトリを作成してください。HiRDB 運用ディレクトリについては，「HiRDB 運用ディレクトリの作成」を参照してください。

9.1.2　マルチ HiRDB の環境設定

(1)　HiRDB サーバの環境変数の設定

マルチ HiRDB のサーバで別々に設定した HiRDB 管理者は，それぞれが操作する対象の HiRDB を環境変数 PDDIR で判別します。HiRDB 管理者別に，HiRDB 運用ディレクトリを環境変数 PDDIR に指定します。

また，環境変数 PATH に各 HiRDB の$PDDIR/bin を設定すると，PATH 中で先に指定した HiRDB の運用コマンドしか使用できません。そこで，各 HiRDB を操作し分けるには，それぞれの HiRDB 用のウィンドウを用意し，それぞれのウィンドウで環境変数を設定して運用することをお勧めします。

HiRDB 管理者が設定する環境変数については，「環境変数の設定」を参照してください。

(2)　HiRDB システム定義の設定

HiRDB ごとに HiRDB システム定義を作成してください。HiRDB システム定義に指定する次に示す情報は，HiRDB ごとに変えてください。

- HiRDB 識別子（システム共通定義の pd_system_id オペランド）
- HiRDB のポート番号（システム共通定義の pd_name_port オペランド）
- ユニット識別子（ユニット制御情報定義の pd_unit_id オペランド）

(3) クライアント環境定義の設定

クライアントからどの HiRDB にアクセスするかは，クライアント環境定義の PDNAMEPORT オペランドで指定します。アクセスする HiRDB のポート番号を PDNAMEPORT オペランドに指定してください。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。

(4) ユティリティ専用ユニットの設置

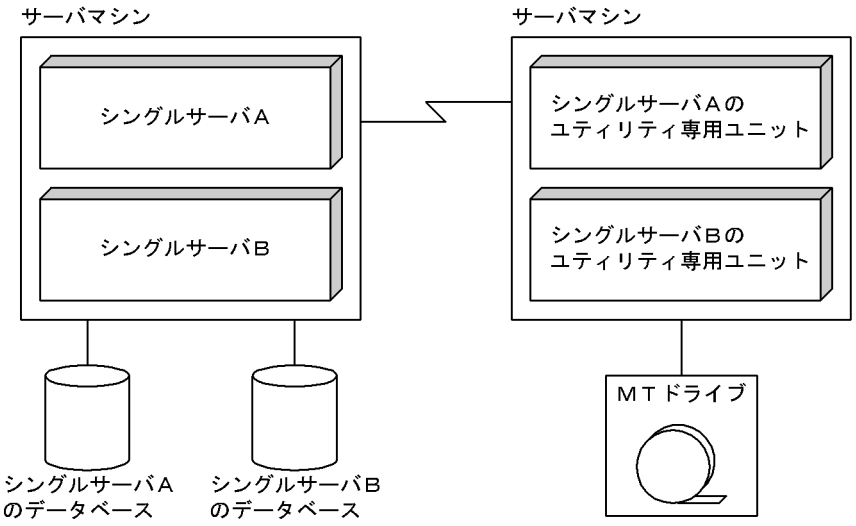
マルチ HiRDB にユティリティ専用ユニットを設置する場合，1.又は 2.のどちらかのシステム構成が選択できます。

1. HiRDB ごとにユティリティ専用ユニットを持つシステム構成
2. 複数の HiRDB でユティリティ専用ユニットを共用するシステム構成

HiRDB ごとに別々の業務をする場合は，1. のシステム構成をお勧めします。相互系切り替え構成で系切り替え機能を使用する場合は，2. のシステム構成をお勧めします。

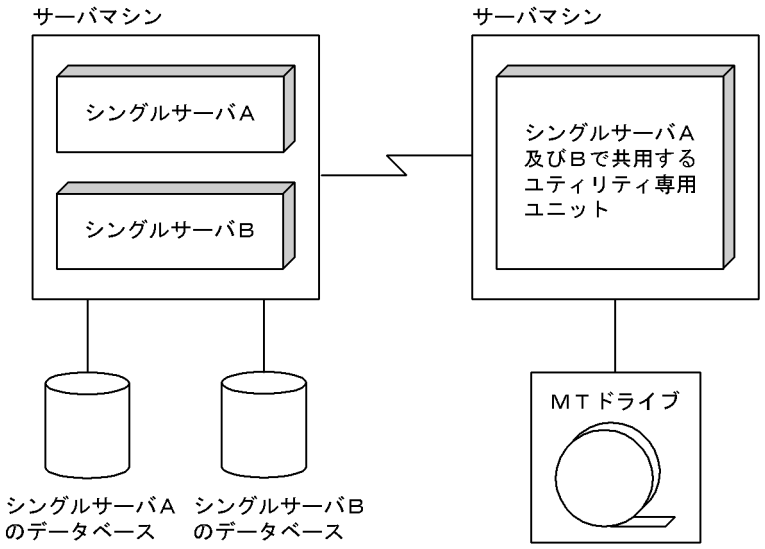
マルチ HiRDB でのユティリティ専用ユニットの設置例を次の図に示します。

図 9-1　マルチ HiRDB でのユティリティ専用ユニットの設置例（HiRDB ごとにユティリティ専用ユニットを持つシステム構成）



注　それぞれのHiRDBに対応するユティリティ専用ユニットを設置します。

図 9-2　マルチ HiRDB でのユティリティ専用ユニットの設置例（複数の HiRDB でユティリティ専用ユニットを共用するシステム構成）



(5)　ライブラリの共用化

マルチ HiRDB を構成する各 HiRDB で，インストールディレクトリ（/opt/HiRDB_S 又は/opt /HiRDB_P）下のライブラリの一部を共用化できます。pdmemsv -s コマンドでライブラリの一部を共用化すると，必要なメモリを削減できます。

ただし，ライブラリの一部を共用化するには，次に示す前提条件があります。

- マルチ HiRDB を構成する HiRDB のバージョンが同じであること
- マルチ HiRDB を構成する HiRDB が使用する文字コードが同じであること
- マルチ HiRDB を構成する HiRDB が使用するロード種別が同じであること（POSIX ライブラリ版を使用する場合は全 HiRDB で使用すること）

なお，ライブラリを共用化する場合，インストールディレクトリと HiRDB 運用ディレクトリを格納するボリュームを別々にしてください。こうすると，インストールディレクトリを格納するハードディスクに障害が発生して HiRDB が稼働できなくなった場合，pdmemsv -d コマンドでライブラリの共用化を解除すれば，HiRDB が稼働できるようになります。ライブラリを共用化する手順を次に示します。

```
# pdsetup /USERS/DB1            1
# pdsetup -d /USERS/DB1         2
# PDDIR=”USERS/DB1”             3
# export PDDIR
# $PDDIR/bin/pdmemsv -s         4
# pdsetup /USERS/DB1            5
```

〔説明〕

スーパユーザがこの作業を実行してください。

1. インストールディレクトリ以外のディレクトリに，HiRDB を構築します。

2. 一度，OS への登録を解除します。このとき，メッセージ KFPS00036-Q に対して，n を返答します。

3. 環境変数 PDDIR を設定します。

4. ライブラリを共用化します。

5. もう一度，HiRDB を OS に登録します。

注意事項

HiRDB を再インストールする場合，pdmemsv -d コマンドでライブラリの共用化をいったん解除してください。再インストール後に，再度 pdmemsv -s コマンドを実行してライブラリを共用化してください。

また，pdsetup -d コマンドを実行した際に KFPS00036-Q メッセージに y で応答し，OS への登録を解除した場合，pdmemsv コマンドを前記の手順に従って再実行してください。

9.2.1　マルチ HiRDB 環境下で HiRDB をバージョンアップするときの注意事項

ここでは，マルチ HiRDB 環境下で HiRDB をバージョンアップするときの注意事項について説明します。

(1)　全 HiRDB を同時にバージョンアップする場合

マルチ HiRDB 環境で，全 HiRDB を同時にバージョンアップする場合の手順の概略を次に示します。バージョンアップ手順の詳細については，「HiRDB のバージョンアップ」を参照してください。

〈手順〉

1. 旧バージョンの全 HiRDB を pdsetup -d コマンドで OS から削除します。KFPS00036-Q メッセージには，y で応答します（n で応答すると，元の環境がそのまま残るため，新バージョンの HiRDB をインストール，セットアップしても，新バージョンに入れ替わりません）。
2. 新バージョンの HiRDB をインストールします。
3. 新バージョンの全 HiRDB を pdsetup コマンドで OS に登録します。

(2)　一部の HiRDB だけをバージョンアップする場合

マルチ HiRDB 環境で，一部の HiRDB だけをバージョンアップする場合の手順の概略を次に示します。バージョンアップ手順の詳細については，「HiRDB のバージョンアップ」を参照してください。

〈手順〉

1. バージョンアップ対象の HiRDB を pdsetup -d コマンドで OS から削除します。KFPS00036-Q メッセージには，y で応答します（n で応答すると，元の環境がそのまま残るため，新バージョンの HiRDB をインストール，セットアップしても，新バージョンに入れ替わりません）。
2. 新バージョンの HiRDB をインストールします。
3. バージョンアップ対象の HiRDB を pdsetup コマンドで OS に登録します。

10 グローバルバッファ，ローカルバッファの設計

この章では，グローバルバッファ，ローカルバッファを設計する上で検討する項目について説明します。

10.1　グローバルバッファの割り当て

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集まりのことで，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には次に示す種類があります。

- インデクス用グローバルバッファ
- データ用グローバルバッファ
- LOB 用グローバルバッファ

また，HiRDB の稼働中に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。動的変更をするには，pdbufmod コマンドを使用します。グローバルバッファの動的変更の詳細は，マニュアル「HiRDB システム運用ガイド」を参照してください。

それぞれのグローバルバッファと割り当て方法について説明します。

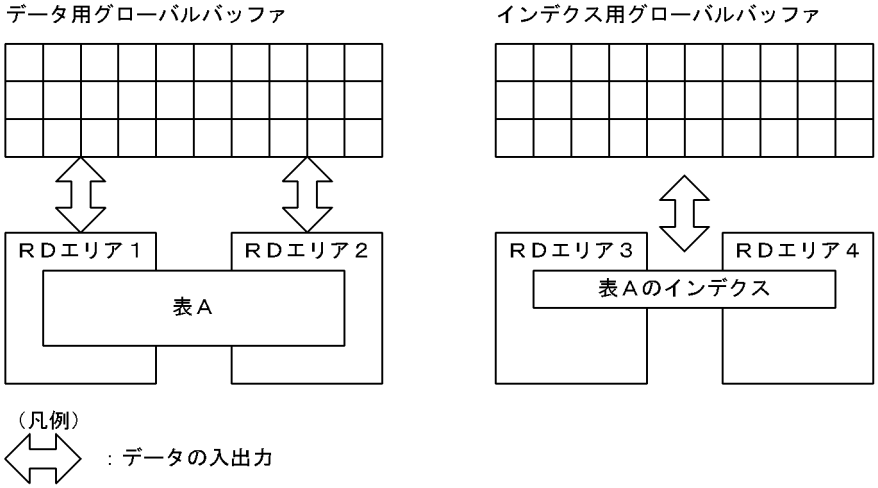
10.1.1　インデクス用グローバルバッファの割り当て

アクセス頻度が高いインデクスがある場合，特に，クラスタキーや UNIQUE を指定したインデクスがある場合は，そのインデクスに専用のグローバルバッファを割り当てるようにします。これでインデクスが常駐できるため，インデクスにアクセスするときの入出力回数が削減できます。

インデクス専用のグローバルバッファを割り当てると，このグローバルバッファは，表の行を格納したユーザ用 RD エリアのグローバルバッファとは独立して管理されます。このため，グローバルバッファ内でインデクスページとデータページが共用されることがありません。しかし，あるインデクスに対して，表やほかのインデクスと同一のグローバルバッファを割り当てると，ほかの表のデータなどが一時的に大量に入力された場合，このインデクスの情報がグローバルバッファ上から追い出されてしまうことがあります。

インデクス専用のグローバルバッファの概要を次の図に示します。

図 10–1　インデクス用グローバルバッファの概要



10.1.2　データ用グローバルバッファの割り当て

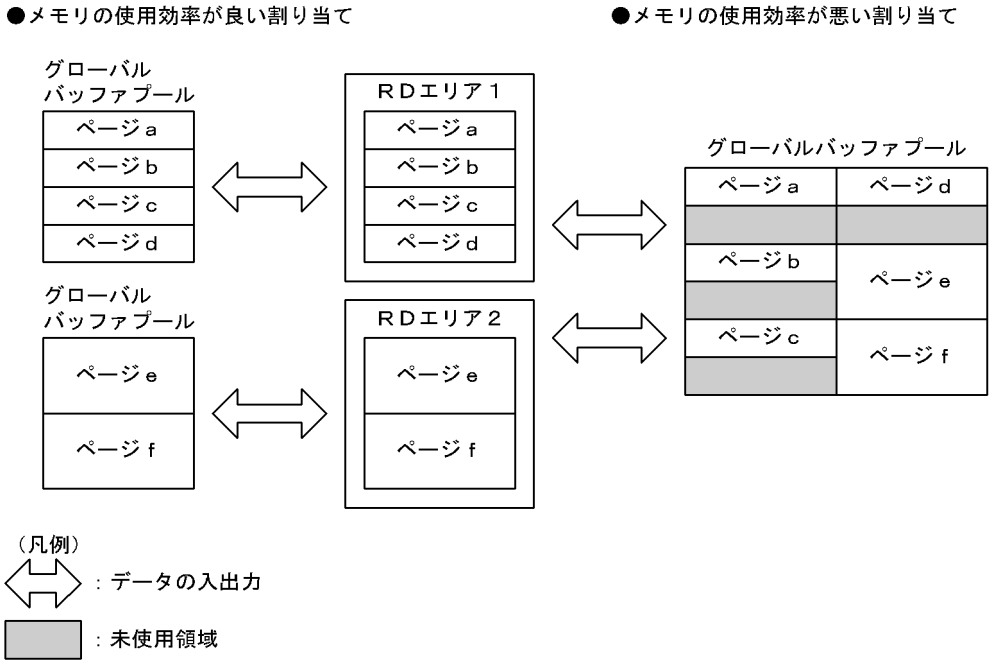
(1)　異なるページ長の RD エリアが複数ある場合

異なるページ長の RD エリアが複数ある場合は，同じか又は近いページ長の RD エリアをまとめて一つのグローバルバッファに割り当てます。これによって，メモリの使用効率が良くなります。

異なるページ長の RD エリアをまとめて一つのグローバルバッファに割り当てると，ページ長が最も大きい RD エリアに合わせてグローバルバッファが確保されます。このため，ページ長が小さい RD エリアに対してデータページの入出力をした場合，グローバルバッファ一面に使用されない領域ができることになり，メモリの使用効率が下がります。

グローバルバッファの割り当ての例を次の図に示します。

図 10-2　グローバルバッファの割り当ての例



HiRDB/パラレルサーバの場合は，サーバごとにページ長が最も大きい RD エリアに合わせてグローバルバッファが確保されます。例えば，ページ長が最も大きい RD エリアのページ長がバックエンドサーバ 1 は 4096 バイト，バックエンドサーバ 2 は 8192 バイトの場合，割り当てられるグローバルバッファサイズはバックエンドサーバ 1 が 4096 バイト，バックエンドサーバ 2 が 8192 バイトになります。

(2)　一つのグローバルバッファに複数の RD エリアを割り当てる場合

複数の RD エリアを構成する HiRDB ファイルが一つの HiRDB ファイルシステム領域に含まれる場合は，これらの RD エリアを一つのグローバルバッファに割り当てるようにします。

(3)　UAP からのアクセス方法が異なる RD エリアが複数ある場合

同じページ長でも，UAP からのアクセス方法が異なる RD エリアが複数ある場合，例えば，用途が異なる RD エリア，順次処理が多くて更新が少ない RD エリア，追加や更新が多い RD エリアなどがある場合は，それぞれ異なるグローバルバッファを割り当てるようにします。

(4)　RD エリアの追加が予想される場合

データベース構成変更ユティリティ（pdmod）で追加できる 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
- 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を管理するディクショナリ表を格納するデータディクショナリ用 RD エリア
- リスト用 RD エリア

また，フレキシブルハッシュ分割した表を格納しているRDエリアは，ALTER TABLEでRDエリアを追加できます。追加したRDエリアを使用するためには，グローバルバッファを割り当てる必要があります。このため，RDエリアの追加が予想される場合は，追加が予想されるRDエリアの最大ページを見込んで，システム共通定義のpdbufferオペランドで-oオプションを指定したグローバルバッファを用意しておきます。

なお，あらかじめグローバルバッファを割り当てていなかった場合は，システム共通定義のpdbufferオペランドで，追加したRDエリアへの割り当てを再度定義します。これによって，追加したRDエリアを使用できます。

(5) リスト用RDエリアにグローバルバッファを割り当てるときの注意

リスト用RDエリアにグローバルバッファを割り当てる場合，ユーザ用RDエリアにグローバルバッファを割り当てるときの設計に加え，次に示す点に注意してください。

1. リスト用RDエリアのグローバルバッファと，表やインデクスのグローバルバッファを共用したときに大量のリストを作成すると，表やインデクスのデータがグローバルバッファから追い出されてしまうことがあります。したがって，なるべく共用しないで，リスト用RDエリア専用のグローバルバッファを割り当ててください。
2. リスト用RDエリアのグローバルバッファのバッファ面数には，なるべく「同時にアクセスするリスト数×1.5」以上の値を指定してください。
3. リスト用RDエリアのグローバルバッファと，表やインデクスのグローバルバッファとを共用する場合，ページ長が同じか又はページ長が近いRDエリアを共用してください。
4. リスト用RDエリアのグローバルバッファにプリフェッチ機能を指定した場合，次に示すSQL文が実行されたときにセグメントサイズ分のページを一括して入力します。
   - SELECT文でリストを介した表の検索をする場合，リストページを一括して入力します。
   - ASSIGN LIST文でリストからリストを作成する場合，FROM句に指定したリストのリストページを一括して入力します。

10.1.3 LOB用グローバルバッファの割り当て

次に示す場合に，LOB用RDエリアにグローバルバッファを割り当てるようにします。これによって，LOB用RDエリアに格納されているデータの入出力回数が削減されます。

- 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

なお，RDエリア間のバッファリングの干渉を避けるため，単独のRDエリアに割り当てることをお勧めします。次に示すLOB用RDエリアにグローバルバッファを割り当てられます。

- データディクショナリ LOB 用 RD エリア
- ユーザ LOB 用 RD エリア
- レジストリ LOB 用 RD エリア

10.1.4 グローバルバッファの割り当て方法

(1) インデクス用グローバルバッファを割り当てる場合

システム共通定義の pdbuffer オペランドの-i オプションに，インデクス用グローバルバッファを割り当てるインデクスを，「認可識別子．インデクス識別子」の形式で指定します。

クラスタキーの場合はインデクス識別子を HiRDB が決定するため，クラスタキーを指定した表を定義した後に，ディクショナリ表の SQL_INDEXES 表の INDEX_NAME 列を検索してインデクス識別子を確認してください。クラスタキーのインデクス識別子は，次のように表示されます。

(CLUSTER 表番号)

ディクショナリ表の検索方法と SQL_INDEXES 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。

定義したインデクスにインデクス用グローバルバッファを割り当てる場合は，いったん HiRDB を正常終了し，pdbuffer オペランドを指定してインデクス用グローバルバッファを割り当ててください。この作業を行わない場合，定義したインデクスはインデクス格納 RD エリアに割り当てられているデータ用グローバルバッファを使用します。

●グローバルバッファ面数を見積もるときの考え方

グローバルバッファ面数は，インデクスの総ページ数（インデクスの格納ページ数として算出した値）以上とすることを原則とし，そこからインデクスの重要度によってグローバルバッファ面数を減らしてください。

インデクスの使用ページ数は，データベース状態解析ユティリティ（pddbst）で確認できます。

(2) データ用グローバルバッファを割り当てる場合

システム共通定義の pdbuffer オペランドの-r オプションでグローバルバッファに割り当てる RD エリア名を指定します。

(3) LOB 用グローバルバッファを割り当てる場合

次に示す手順で割り当てます。

1. システム共通定義の pdbuffer オペランドの-r オプションでグローバルバッファに割り当てる LOB 用 RD エリア名を指定します。

2. システム共通定義の pdbuffer オペランドの-b オプションでグローバルバッファに割り当てる LOB 用 RD エリア名を指定します。

(4) グローバルバッファの定義例

RD エリアの構成

RD エリアの構成は次のとおりとします。

RD エリアの種類	RD エリア名
マスタディレクトリ用 RD エリア	RDMAST
データディレクトリ用 RD エリア	RDDIR
データディクショナリ用 RD エリア	RDDIC
ユーザ用 RD エリア	USER01，USER02，USER03
ユーザ LOB 用 RD エリア	ULOB03
データディクショナリ LOB 用 RD エリア	DICLOB01，DICLOB02
リスト用 RD エリア	LIST01

定義例

グローバルバッファの定義例を次に示します。

```
pdbuffer -a DGB1 -n 1000 -r RDMAST,RDDIR,RDDIC              1
pdbuffer -a DGB2 -n 1000 -r USER01,USER02
pdbuffer -a DGB3 -n 1000 -r USER03
pdbuffer -a DGB4 -n 1000 -r ULOB03
pdbuffer -a DGB5 -n 1000 -r DICLOB01
pdbuffer -a DGB6 -n 1000 -r DICLOB02
pdbuffer -a DGB7 -n 1000 -r LIST01

pdbuffer -a LGB1 -n 1000 -b ULOB03                          2
pdbuffer -a LGB2 -n 1000 -b DICLOB01
pdbuffer -a LGB3 -n 1000 -b DICLOB02

pdbuffer -a IGB1 -n 1000 -i USER1.INDX01                    3
pdbuffer -a IGB2 -n 1000 -i USER1.INDX02
```

〔説明〕

1. データ用グローバルバッファの定義です。作成するすべての RD エリアを-r オプションで指定します。
2. LOB 用グローバルバッファの定義です。
   -b オプションに指定している RD エリアは，-r オプションでも指定しておく必要があります。
3. インデクス用グローバルバッファの定義です。-i オプションにインデクスの認可識別子とインデクス識別子を指定します。

この定義例に出てくる pdbuffer オペランドのオプションを簡単に説明します。

-a：グローバルバッファの名称を指定します。

-n：グローバルバッファの面数を指定します。

-r：データ用グローバルバッファを割り当てる RD エリアを指定します。

-b：LOB 用グローバルバッファを割り当てる LOB 用 RD エリアを指定します。

-i：インデクス用グローバルバッファを割り当てるインデクスを指定します。

10.2　グローバルバッファのバッファ面数の設定

10.2.1　グローバルバッファのバッファ面数の設定するときの考慮点

(1)　共用メモリの上限を考慮した設定

グローバルバッファのバッファ面数を設定するには，共用メモリの上限を超えないように設定します。現状の共用メモリの上限を超えたバッファ面数を設定する必要がある場合は，OS の共用メモリセグメント一つ当たりの上限を sam コマンドで再設定して，一つの共用メモリセグメント内に収まるバッファ面数を設定します。ただし，サーバマシンで確保できる共用メモリには限界があることに注意してください。

共用メモリでは，一つの共用メモリセグメントに確保できる容量が OS によって決められています。一つの共用メモリセグメントに確保できないグローバルバッファを定義すると，複数の共用メモリセグメントが確保されるため，共用メモリに対するアクセスのオーバヘッドが大きくなります。

(2)　バッファヒット率を考慮した設定

グローバルバッファは，共用メモリ上に確保されます。必要以上にグローバルバッファのバッファ面数を設定すると，共用メモリが増加して，システムのディスク容量及びメモリを圧迫します。また，グローバルバッファを検索するためのオーバヘッドも大きくなります。このため，必要最低限の入出力性能が得られるように設定する必要があります。

必要最低限の入出力性能が得られるように設定するには，グローバルバッファ全体のヒット率（更新バッファヒット率＋参照バッファヒット率）及び参照バッファヒット率が高くなるように設定します。このためには，次に示す方法があります。

- グローバルバッファのバッファ面数を大きくします。
- グローバルバッファに割り当てる RD エリア又はインデクスを異なるグローバルバッファに分けます。

なお，上記の方法でバッファ面数を設定して HiRDB を稼働してから性能向上を図る場合は，pdbufls コマンド又は統計解析ユティリティ（pdstedit）を指定します。

pdbufls コマンドの場合には，編集項目である，グローバルバッファ全体のヒット率を高くなるように面数を設定します。

統計解析ユティリティ（pdstedit）の場合には，編集項目である，更新バッファヒット率及び参照バッファヒット率を参照して，グローバルバッファ全体のヒット率が高くなるように面数を設定します。

(3)　設定方法

システム共通定義の pdbuffer オペランドの-n オプションでバッファ面数を指定します。

10.3　プリフェッチ機能の指定

プリフェッチ機能とは，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。

10.3.1　プリフェッチ機能の効果

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。

10.3.2　適用基準

プリフェッチ機能は次に示す SQL 文又はユティリティの場合にページを一括して入力します。

- インデクスを使用しない SELECT，UPDATE，DELETE 文の場合に**データページ**を一括して入力します。
- インデクスを使用した昇順検索をする SELECT，UPDATE，DELETE 文（=条件，IN 条件を除く）の場合にインデクスリーフページを一括して入力します。
- クラスタキーを使用した昇順検索をする SELECT，UPDATE，DELETE 文（=条件，IN 条件を除く）の場合にインデクスリーフページ及びデータページを一括して入力します。
- ローカルバッファを使用しないデータベース再編成ユティリティ（pdrorg）のアンロードの場合にインデクスリーフページ及びデータページを一括して入力します。

10.3.3　指定方法

(1)　グローバルバッファの場合

プリフェッチ機能を動作させるには，システム共通定義の 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。

(2)　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。

10.3.4　指定上の考慮点

- プリフェッチ機能を使用する場合，グローバルバッファ（又はローカルバッファ）とは別に一括入力専用のバッファが取られます。これによって，グローバルバッファ用の共用メモリが増加します。グローバルバッファが使用する共用メモリの計算式については，「HiRDB のメモリ所要量」を参照してください。
- プリフェッチ機能が有効に動作しているかどうかは統計解析ユティリティ（pdstedit）又は pdbufls コマンドのプリフェッチヒット率を参照してください。

10.4　非同期 READ 機能の指定

10.4.1　非同期 READ 機能の効果と指定方法

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 DB 処理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，入出力の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。

(1)　非同期 READ 機能の効果

プリフェッチ機能と同じですが，プリフェッチ機能だけを使用した場合と比べて，非同期 READ 機能は処理負荷が高いジョイン処理などで有効です。非同期 READ 機能は，入出力処理時間が長いキャラクタ型スペシャルファイルを使用している場合，特に効果があります。逆に，入出力時間が掛からない通常ファイル又は日立ディスクアレイシステムのディスクなどを使用している場合は余り効果が得られないことがあります。

(2)　指定方法

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）をしていることが前提です。

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。0 を指定するか，又はこのオペランドを省略した場合，非同期 READ 機能は動作しません。

(3)　指定上の考慮点

プリフェッチ機能を使用する場合，グローバルバッファとは別に一括入力専用のバッファが 2 面取られます。これによって，グローバルバッファ用の共用メモリが増加します。グローバルバッファが使用する共用メモリの計算式については，「HiRDB のメモリ所要量」を参照してください。

10.5　デファードライト処理の指定

10.5.1　デファードライト処理の効果と指定方法

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，システム共通定義の 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。なお，次の RD エリアはデファードライト処理の対象ではありません。

- データディクショナリ LOB 用 RD エリア
- ユーザ LOB 用 RD エリア
- レジストリ LOB 用 RD エリア
- リスト用 RD エリア

(1)　デファードライト処理の効果

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します。

(2)　指定方法

pd_dbsync_point オペランドに sync を指定するか，指定を省略してください。また，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出力比率を指定します。

(3)　指定上の考慮点

1. グローバルバッファに割り当てる RD エリアに格納された表やインデクスに対する更新が多い場合は，デファードライトトリガでの更新ページの出力比率を低めに設定します。
2. グローバルバッファに対する更新が多くても，同じデータに対する更新がほとんど発生しない場合は，デファードライトトリガでの更新ページの出力比率を高めに設定します。
3. HiRDB を稼働してから，更に性能の向上を図る場合は，pdbufls コマンドを使用します。それぞれの編集項目である各グローバルバッファの更新要求ヒット率を参照して，次に示すように設定します。
   - 更新要求バッファヒット率が高い場合は，デファードライトトリガでの更新ページの出力比率を低く設定します。
   - 更新要求バッファヒット率が低い場合は，デファードライトトリガでの更新ページの出力比率を高く設定します。

(4) 注意

デファードライトトリガでの更新ページ出力比率を必要以上に高くすると，デファードライト処理でのディスクへの書き込みが多くなります。このため，同時に実行しているトランザクションが入出力待ちになることがあり，レスポンスタイムが悪くなることがあります。

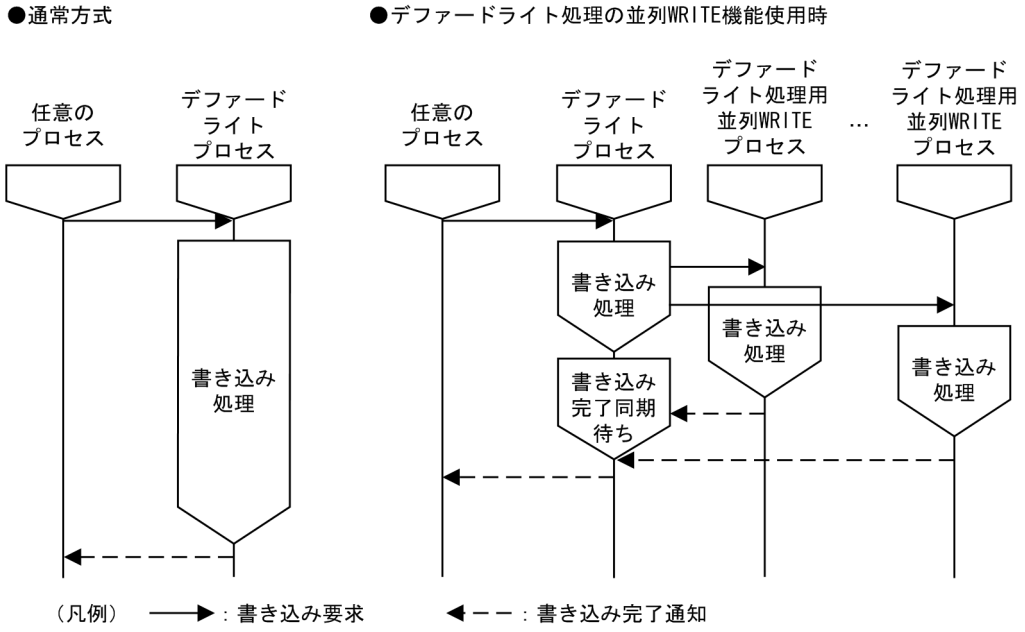
また，必要以上に低くすると，シンクポイントダンプの出力時に，データベースに書き出すページ数が多くなることがあります。このため，同時に実行しているトランザクションが入出力待ちになることがあり，レスポンスタイムが悪くなることがあります。

10.6　デファードライト処理の並列 WRITE 機能の指定

10.6.1　デファードライト処理の並列 WRITE 機能の効果と指定方法

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。並列 WRITE 機能を使用する場合としない場合の概要を次の図に示します。

図 10–3　デファードライト処理の並列 WRITE 機能の概要



(1)　デファードライト処理の並列 WRITE 機能の効果

書き込み処理を複数のデファードライト処理用並列 WRITE 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。

(2)　指定方法

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。

(3)　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，デファードライト処理用並列 WRITE プロセス数が増加するため，CPU 利用率が上がります。

10.7　コミット時反映処理の設定

10.7.1　コミット時反映処理の効果と指定方法

コミット時反映処理とは，グローバルバッファ上で更新されたページを COMMIT 文発行時にディスクに書き込む処理のことです。

(1)　コミット時反映処理の効果

COMMIT 文の発行時にデータベースの更新内容をディスクに書き込むため，トランザクションの完了時点でデータベースの内容が保証されます。そのため，全面回復処理時に，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く，全面回復処理の時間が短縮できます。

(2)　指定方法

pd_dbsync_point オペランドに commit を指定します。

ただし，LOB 用 RD エリアはこのオペランドの影響を受けません。ディレクトリ部は COMMIT 文発行時点で反映されます。データ部は LOB 用グローバルバッファを割り当てているかどうかによって処理が異なります。LOB 用グローバルバッファを割り当てていない場合は更新要求時にすぐに反映されます。LOB 用グローバルバッファを割り当てている場合は COMMIT 文発行時点で反映されます。ただし，グローバルバッファが満杯になったときはその時点で反映されます。

(3)　指定上の考慮点

pdbufls コマンドで取得した情報を参照して，ディスクへの出力回数が多くて更新要求ヒット率が低い場合には，グローバルバッファのバッファ面数を大きく設定します。

10.8 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）によって，**グローバルバッファの LRU 管理方式**を選択できます。

10.8.1 LRU の管理方式

LRU の管理方式には，次に示す 2 種類があります。

- 参照バッファ及び更新バッファの独立した LRU での管理
- グローバルバッファの一括した LRU での管理

(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。

(a) 適用基準

参照バッファ及び更新バッファをそれぞれ独立した LRU で管理した方がよい場合を次に示します。

- 検索処理に比べて，更新処理の割合が比較的少ない場合で更新バッファヒット率が高い場合（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）

(b) 指定方法

システム共通定義の pd_dbbuff_lru_option オペランドに SEPARATE を指定します。

(c) 注意事項

- 大量の更新処理が発生した場合，参照バッファヒット率が低下し，検索処理が遅くなります。
- 次のどちらかに該当する場合は，無条件に pd_dbbuff_lru_option オペランドに MIX を仮定します。そのため，参照バッファ及び更新バッファの独立した LRU での管理はできません。
  - pd_dbsync_point オペランドに commit を指定している場合
  - pd_dbbuff_binary_data_lru オペランドに N を指定している場合

(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。

(a) 適用基準

グローバルバッファを一括した LRU で管理した方がよい場合を次に示します。

- 検索処理に比べて，更新処理の割合が多い場合，突発的な大量検索又は大量更新が発生する場合（オンライン業務とバッチ業務など，大量検索，大量更新が共存する場合）

(b) 指定方法

システム共通定義の pd_dbbuff_lru_option オペランドに MIX を指定します。

システム共通定義の 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。

(c) 注意事項

- 更新バッファヒット率が高い場合には，大量検索によって更新バッファが一時的にメモリから追い出されます。このような場合には，更新処理の延長でファイルの読み込みが発生し，処理が遅くなる場合があります。
- pd_dbsync_point=sync の指定又は省略時には，検索処理の延長でファイルへの書き込みが発生し，検索処理が遅くなる場合があります。

10.8.2 UAP ごとの LRU 管理抑止設定

OLTP 環境で，UAP の大量検索や大量更新によってグローバルバッファにキャッシュされた直近の内容がメモリから追い出され，OLTP 性能を一時的に低下させてしまうことがあります。このとき，大量検索や大量更新をする UAP を特定できるのであれば，UAP ごとに LRU 管理を抑止する設定をすることで，OLTP 性能の低下を回避できます。

参考

LRU 管理を抑止できるのは UAP からのアクセスだけです。コマンドやユティリティからのアクセスの場合は，LRU 管理が行われます。ただし，次のコマンドは LRU 管理が抑止されます。

- データベース状態解析ユティリティ（pddbst）
  次のどちらかの条件に該当する場合，LRU 管理が抑止されます。
  ・-s オプションを指定しない
  ・-s オプションと-b オプションを指定する
- 空きページ解放ユティリティ（pdreclaim）
  globalbuffer_lru 文の値が no の場合，LRU 管理が抑止されます。

(1) 適用基準

OLTP 環境で，グローバルバッファを使用して大量検索や大量更新をする UAP を実行する場合に適用することをお勧めします。

(2) 適用の効果

LRU 管理を抑止した UAP がアクセスしたページはアクセス頻度に関係なく，最も古い時にアクセスしたページとしてグローバルバッファ上にキャッシュされます。そのため，LRU 管理を適用した UAP がアクセスしたページより先にメモリから追い出されるようになり，LRU 管理を適用した UAP がアクセスしたページはメモリから追い出されなくなります。ただし，LRU 管理を抑止する UAP と LRU 管理を抑止しない UAP が同じページにアクセスする場合，そのページは LRU 管理されます。

(3) 指定方法

クライアント環境定義の PDDBBUFLRU オペランドに NO を指定します。

(4) 注意事項

1. LRU 管理を抑止する UAP がアクセスしたページは，アクセス頻度に関係なく，バッファ不足が発生すると，メモリからの追い出し対象となります。そのため，LRU 管理を抑止する UAP はバッファヒット率の低下に伴う入出力回数の増加によって，レスポンス性能が低下することがあります。
2. UAP は 1～4 面のバッファを同時に確保します。そのため，LRU 管理を抑止する設定をした場合でも，UAP ごとにグローバルバッファにキャッシュされているページの中で 1～4 ページはキャッシュから追い出されるおそれがあります。
3. 更新を行う UAP に対して LRU 管理を抑止した場合，DB への書き込み回数が多く，ログ出力契機が頻繁に発生するため，LRU 管理を抑止しない場合に比べて，出力されるログ量が多くなります。容量が不足しないように次のようにしてください。
   - システムログファイルの容量を再度見積もりする
   - ログレスモードで実行できる場合，クライアント環境定義の PDDBLOG に NO を指定する

   LRU 管理を抑止する場合のログ量は，次に示す計算式で求めます。なお，pd_log_rec_leng オペランドの指定値を 1024 にすると，LRU 管理を抑止する場合に出力されるログ量を最小に抑えられます。

更新GET数※×pd_log_rec_lengオペランドの値

   注※

   更新 GET 数は，UAP 統計レポートの DIDUC の値，又は UAP に関する統計情報の DIDUC の値で確認できます。
4. LRU 管理を抑止する UAP でも，ロールバックが発生した場合は LRU 管理されることがあります。LRU 管理されるかどうかはロールバックのタイミングによって次のように異なります。

ロールバックのタイミング		LRU 管理されるかどうか
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 文が実行される直前のコミット時点までロールバックされるとき	LRU 管理は抑止されます。
HiRDB によって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り，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るとき	LRU 管理は抑止されます。
	UAP の異常終了時に HiRDB によって直前のコミット時点までロールバックされるとき	LRU 管理が行われます。
	ディレードリランによってロールバックされるとき	LRU 管理が行われます。

10.8.3 UAP がアクセスするバイナリデータの LRU 管理抑止設定

サイズの大きなバイナリデータを大量にアクセスする UAP を実行する場合，バイナリデータがグローバルバッファにキャッシュされると，グローバルバッファにキャッシュされた直近の内容がメモリから追い出され，性能が一時的に低下することがあります。このとき，バイナリデータのアクセス頻度が低い場合は，バイナリデータの分岐行ページの LRU 管理だけを抑止することで，性能の低下を回避できます。

なお，この設定は BINARY 型のバイナリデータの場合に有効です。BLOB 型のバイナリデータの場合は無効となります。

参考

LRU 管理を抑止できるのは UAP からのアクセスだけです。コマンドやユティリティからのアクセスの場合は，LRU 管理が行われます。ただし，次のコマンドは LRU 管理が抑止されます。

- プラグインが提供するコマンド
- pddbst
  次のどちらかの条件に該当する場合，LRU 管理が抑止されます。
  ・-s オプションを指定しない
  ・-s オプションと-b オプションを指定する
- pdreclaim
  globalbuffer_lru 文の値が no の場合，LRU 管理が抑止されます。

また，次のコマンドの場合，LRU 管理は抑止できませんが，グローバルバッファにキャッシュされた直近の内容がメモリから追い出されることを回避できます。

- pdload，pdrorg，pdrbal
  -n オプションを指定してローカルバッファを使用すると，グローバルバッファから基本行内のデータが追い出されるのを回避できます。
- pdpgbfon

-b オプションを省略すると，バイナリデータの分岐行ページをアクセスしません。

(1) 適用基準

次のすべての条件を満たす場合に適用することをお勧めします。

- BINARY 型や，BINARY 型の属性を含む抽象データ型，XML 型などのサイズの大きなバイナリデータを含む表がある場合
- バイナリデータへのアクセスはまれである場合

注意事項

頻繁にバイナリデータをアクセスする場合には，適用しないでください。

(2) 適用の効果

バイナリデータが格納された分岐行はアクセス頻度に関係なく，最も古い時にアクセスしたページとしてグローバルバッファ上にキャッシュされます。そのため，分岐行ページが基本行ページより先にメモリから追い出されるようになり，基本行内のデータがメモリから追い出されなくなります。

(3) 指定方法

システム共通定義の pd_dbbuff_binary_data_lru オペランドに N を指定します。

(4) 注意事項

1. LRU 管理を抑止した場合，UAP がアクセスするバイナリデータの分岐行ページは，アクセス頻度に関係なく，バッファ不足が発生するとメモリからの追い出し対象となります。そのため，バイナリデータの分岐行をアクセスする UAP はバッファヒット率の低下に伴う入出力回数の増加によって，レスポンス性能が低下することがあります。
2. UAP は 1～4 面のバッファを同時に確保します。そのため，LRU 管理を抑止する設定をした場合でも，UAP ごとにグローバルバッファにキャッシュされているページの中で 1～4 ページはキャッシュから追い出されるおそれがあります。
3. LRU 管理を抑止した場合，バイナリデータの分岐行ページの更新を行う UAP を実行するときに DB への書き込み回数が多く，ログ出力契機が頻繁に発生します。そのため，LRU 管理を抑止しない場合に比べて，出力されるログ量が多くなります。容量が不足しないように次のようにしてください。
   - システムログファイルの容量を再度見積もる
   - ログレスモードで実行できる場合，クライアント環境定義の PDDBLOG に NO を指定する

   LRU 管理を抑止する場合のログ量は，次に示す計算式で求めます。なお，pd_log_rec_leng オペランドの指定値を 1024 にすると，LRU 管理を抑止する場合に出力されるログ量を最小に抑えられます。

バイナリデータをアクセスするUAPの更新GET数※×pd_log_rec_lengオペランドの値

注※

更新 GET 数は，UAP 統計レポートの DIDUC の値，又は UAP に関する統計情報の DIDUC の値で確認できます。

4. LRU 管理を抑止する UAP でも，ロールバックが発生した場合は LRU 管理されることがあります。LRU 管理されるかどうかはロールバックのタイミングによって次のように異なります。

ロールバックのタイミング		LRU 管理されるかどうか
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 文が実行される直前のコミット時点までロールバックされるとき	LRU 管理は抑止されます。
HiRDB によって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り，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るとき	LRU 管理は抑止されます。
	UAP の異常終了時に HiRDB によって直前のコミット時点までロールバックされるとき	LRU 管理が行われます。
	ディレードリランによってロールバックされるとき	LRU 管理が行われます。

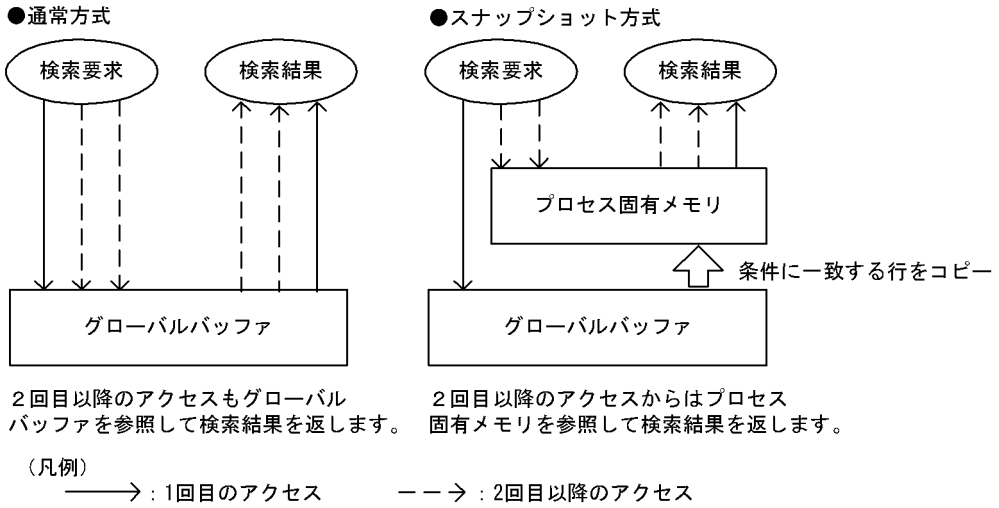
5. この機能を適用した場合，無条件に pd_dbbuff_lru_option オペランドに MIX を仮定します。そのため，参照バッファ及び更新バッファの独立した LRU での管理はできません。

10.9　スナップショット方式によるページアクセス

10.9.1　スナップショット方式によるページアクセスの効果と指定方法

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。スナップショット方式の概要を次の図に示します。

図 10–4　スナップショット方式の概要



(1)　スナップショット方式によるアクセスの効果

最初に探索条件に一致する行をプロセス固有メモリにコピーするため，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。

(2)　指定方法

pd_pageaccess_mode オペランドに SNAPSHOT（省略値）を指定します。

(3)　指定上の考慮点

スナップショット方式を指定すると，表又はインデクスの格納 RD エリアのページサイズに基づいて，動的にプロセス固有メモリが確保されます。確保されるプロセス固有メモリの計算式については，HiRDB/シングルサーバの場合「スナップショット方式指定時に必要なメモリ所要量の求め方」を参照してください。HiRDB/パラレルサーバの場合「スナップショット方式指定時に必要なメモリ所要量の求め方」を参照してください。

(4) スナップショット方式の適用可否

検索時にスナップショット方式を適用するかどうかを次の表に示します。

適用可否が×になっている場合は，システム定義の pd_pageaccess_mode オペランドで SNAPSHOT を指定しても，スナップショット方式が適用されません。

表 10-1 検索時のスナップショット方式の適用可否

<table>
<tr><th colspan="3" rowspan="2">条件</th><th colspan="2">適用可否</th></tr>
<tr><th>表</th><th>インデクス</th></tr>
<tr><td colspan="3">次のどれかに該当する
• ホールダブルカーソルを使用
• 対象表がディクショナリ表
• プラグインインデクスを使用（PLUGIN INDEX SCAN，PLUGIN KEY SCAN）
• 行識別子を使用（ROWID FETCH）</td><td>×</td><td>×</td></tr>
<tr><td rowspan="6">上記以外</td><td rowspan="2">テーブルスキャン（TABLE SCAN）</td><td>次の列を指定している検索
• 定義長 256 バイト以上の VARCHAR，MVARCHAR，NVARCHAR 型の列
• 繰返し列
• 抽象データ型の列
• LOB 列
• 定義長 256 バイト以上の BINARY 型の列</td><td>×</td><td>－</td></tr>
<tr><td>上記以外</td><td>○</td><td>－</td></tr>
<tr><td rowspan="3">インデクススキャン（INDEX SCAN，MULTI COLUMNS INDEX SCAN）</td><td>システム定義に pd_indexlock_mode=KEY を指定した場合に次の条件を満たさない検索
• WITHOUT LOCK NOWAIT 指定の検索
• LOCK TABLE 前提の検索</td><td>×</td><td>×</td></tr>
<tr><td>WITHOUT LOCK WAIT 指定の検索</td><td>×</td><td>×</td></tr>
<tr><td>上記以外</td><td>×</td><td>○</td></tr>
<tr><td rowspan="2">キースキャン（KEY SCAN，MULTI COLUMNS KEY SCAN）</td><td>システム定義に pd_indexlock_mode=KEY を指定した場合に次の条件を満たさない検索
• WITHOUT LOCK NOWAIT 指定の検索
• LOCK TABLE 前提の検索</td><td>－</td><td>×</td></tr>
<tr><td>上記以外</td><td>－</td><td>○</td></tr>
</table>

（凡例）

○：適用します。ただし，ページ内のヒット行が 1 件のときは適用されません。

×：適用しません。

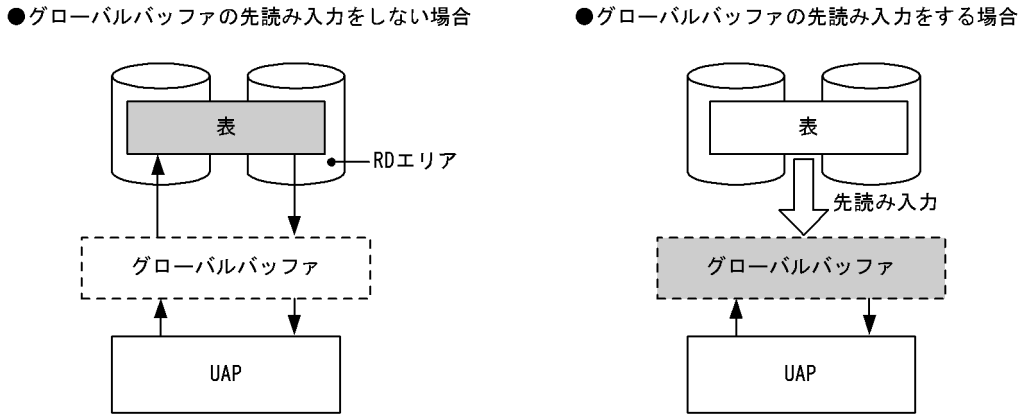
－：条件に依存しません。又は該当しません。

10.10　グローバルバッファの先読み入力

10.10.1　グローバルバッファの先読み入力の効果と実行方法

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みしておく機能です。概要を次の図に示します。

図 10–5　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。

10.11　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。ローカルバッファには次に示す種類があります。

- **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。
- **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。

ローカルバッファは，UAP ごとに **UAP 環境定義**で定義できます。UAP に専用のローカルバッファを割り当てることで，他 UAP によるグローバルバッファの占有やバッファの排他処理による待ち状態を避けられます。UAP 環境定義の詳細は，マニュアル「HiRDB システム定義」を参照してください。

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

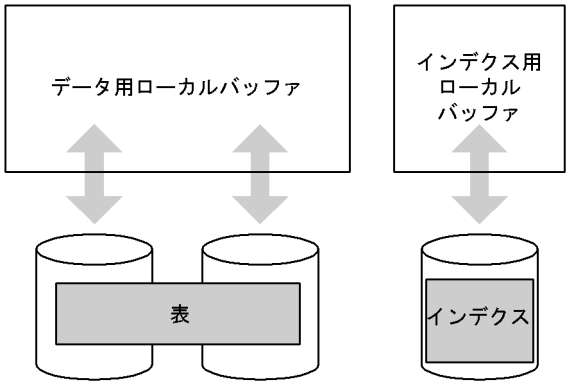
なお，HiRDB に常時接続する UAP はシステムへの影響（メモリの圧迫，サーバプロセスの占有など）が大きいため，ローカルバッファを定義しないでください。

10.11.1　インデクス用ローカルバッファの割り当て

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。

表データとインデクスデータそれぞれにローカルバッファを割り当てた場合のインデクス用ローカルバッファの概要を次の図に示します。

図 10-6　インデクス用ローカルバッファの概要



10.11.2　データ用ローカルバッファの割り当て

(1)　異なるページ長の RD エリアが複数ある場合

異なるページ長の RD エリアが複数ある場合は，同じか又は近いページ長の RD エリアをまとめて一つのローカルバッファに割り当てます。これによって，メモリの使用効率が良くなります。

異なるページ長の RD エリアをまとめて一つのローカルバッファに割り当てると，ページ長が最も大きい RD エリアに合わせてローカルバッファが確保されます。このため，ページ長が小さい RD エリアに対してデータページの入出力をした場合，1 面のローカルバッファに使用されない領域ができることになり，メモリの使用効率が下がります。

(2)　UAP からのアクセス方法が異なる RD エリアが複数ある場合

同じページ長でも，UAP からのアクセス方法が異なる RD エリアが複数ある場合，例えば，用途が異なる RD エリア，順次処理が多くて更新が少ない RD エリア，追加や更新が多い RD エリアなどがある場合は，それぞれ異なるローカルバッファを割り当てるようにします。

10.11.3　ローカルバッファの割り当て方法

インデクス用ローカルバッファを割り当てる場合，pdlbuffer オペランドの-i オプションでインデクス用のローカルバッファを割り当てるインデクスの名称（認可識別子．インデクス識別子）を指定します。

データ用ローカルバッファを割り当てる場合，pdlbuffer オペランドの-r オプションでデータ用のローカルバッファを割り当てる RD エリアの名称を指定します。

ローカルバッファの定義例を次に示します。

```
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。

10.11.4　ローカルバッファ使用時の注意

ローカルバッファの使用時にサーバプロセスが異常終了すると，アボートコード Phb3008 を出力して HiRDB（HiRDB/パラレルサーバの場合はユニット）が異常終了することがあります。サーバプロセスの異常終了時に更新ページがあると，トランザクション回復プロセスで回復処理ができないことがあります。その場合，HiRDB の再開始時に回復処理を行います。ローカルバッファを使用している場合に障害が発生したときの HiRDB の処理と対処方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。

11 表の設計

この章では，表を設計する上で検討する項目について説明します。

11.1　表を設計するときの検討項目

HiRDB のデータベースはリレーショナルデータベースです。その論理構造である表をどのように設計するかを検討します。

まず，表を**正規化**しておくことが必要です。ただし，同じように正規化した表であっても，ユーザ用 RD エリアへの格納の仕方などによって，表に対する処理性能が異なります。また，処理性能よりも操作性を重視する場合もあるため，期待する効果を考慮した表の設計が必要になります。表を設計するときの検討項目を次の表に示します。

表 11-1　データベースの表を設計するときの検討項目

設計作業ごとの検討項目		長所	短所	記載箇所
表の正規化		表の格納効率や処理効率が向上します。	表の検索時に，正規化後の表同士の結合検索が必要になる場合は，処理性能が落ちることがあります。	「表の正規化」を参照してください。
表の横分割	表の横分割の指定	• RD エリアごとの運用ができます。 • HiRDB/パラレルサーバの場合は，表にアクセスする処理を複数の 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及び負荷の分散が図れます。	• RD エリアの数が増えます。 • この表に作成するインデクスを横分割しないと，インデクスでの排他制御によって，同時実行性が低下することがあります。	「表の横分割」を参照してください。
	キーレンジ分割	• 表のデータをどの RD エリアに格納したかが分かります。 • 各業務に関連する部分のデータを別々の RD エリアにまとめて格納し，RD エリアごとの運用ができます。	キーレンジを意識しないと，RD エリアに均等に格納できません。	
	フレキシブルハッシュ分割※	• キーレンジを意識しなくても，データを均等に RD エリアに格納できます。 • RD エリア，ハッシュ関数を変更しやすくなります。 • CPU，ディスクの追加に対応できます。	• 表のデータをどの RD エリアに格納したかが分かりません。 • 特定のキーに偏りや重複があるとデータが均等に格納できません。 • キーの一意性のチェックができません。	
	FIX ハッシュ分割※	• キー値によって,データを格納する RD エリアが一意に決定されます。 • キーレンジを意識しなくても，データを均等に RD エリアに格納できます。 • CPU，ディスクを追加しやすくなります。 • 表分割ハッシュ関数を使用した UAP を作成し，入力データを RD エリアごとに格納できます。	表にデータが格納されているとユーザ用 RD エリアの追加及びハッシュ関数の変更ができません。	
表のマトリクス分割		• キーレンジ分割したデータを，さらに別の列値で分割できるため，通常のキーレンジ分割	通常の横分割に比べ，RD エリアを更に細分化できるため，運用や管理が複雑になります。	「表のマトリクス分割」を参

設計作業ごとの検討項目	長所	短所	記載箇所
	よりも，SQL の高速処理，運用時間の短縮が期待できます。 • キーレンジ分割とハッシュ分割の組み合わせで分割できるため，適用業務の幅が広がります。		照してください。
トリガの定義	ある表への操作を契機として，自動的に SQL 文を実行するようにできます。	特にありません。	「トリガの定義」を参照してください。
ビュー表の作成	• ほかのユーザに実表のアクセス権限を与えないで，ビュー表だけのアクセス権限を与えると，アクセス可能な実表の範囲を行又は列単位で制限できます。 • 複雑な問い合わせ指定をした場合に検索できるデータであらかじめビュー表を作成すると，表を参照する操作が手軽になります。 • ビュー表を通して実表を参照又は更新できます。	特にありません。	「ビュー表の作成」を参照してください。
FIX 属性の指定	• 行単位インタフェースを使用する場合は列数が多くてもアクセス性能を向上できます。 • FIX 属性の表に対する入力データにナル値を許さないように制約できます。 • 列数の多い表の場合はディスク所要量を削減できます。	特にありません。	「FIX 属性の指定」を参照してください。
主キー（プライマリキー）の指定	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。	特にありません。	「主キー（プライマリキー）の指定」を参照してください。
クラスタキーの指定	• 範囲を指定した行の検索，更新，削除などをするときやクラスタキー順の検索，更新などをするときに入出力時間を削減できます。 • クラスタキーに UNIQUE を指定すると，クラスタキーの構成列の値がすべての行で重複しないように制約できます。 • 表を作成するときの入力データがクラスタキーの昇順又は降順に並んでいるかどうかをデータベース作成ユティリティ（pdload）で確認できます。 • 表を再編成するときに，アンロードした行のクラスタキーと，リロードするクラスタキー	• クラスタキーを構成する列の値を更新できません。 • クラスタキーを構成する列の値にナル値を挿入できません。 • クラスタキーを指定した表にデータを追加するとき，追加しようとするキー値に近接するキー値を持ったページを探すためのオーバヘッドが発生します。	「クラスタキーの指定」を参照してください。

設計作業ごとの検討項目	長所	短所	記載箇所
	が一致しているかどうかをデータベース再編成ユティリティ（pdrorg）で確認できます。		
サプレスオプションの指定	• ディスク所要量を削減できます。 • 全件検索などの検索処理での入出力時間を削減できます。	特にありません。	「サプレスオプションの指定」を参照してください。
ノースプリットオプションの指定	データの格納効率を向上できるため，ディスク所要量を削減できます。	特にありません。	「ノースプリットオプションの指定」を参照してください。
バイナリデータ列の指定	文書，画像，音声などの可変長データを指定できます。	特にありません。	「バイナリデータ列の指定」を参照してください。
文字集合の指定	表の列ごとに異なる文字集合の文字列データを格納できます。これによって，次のことができます。 • VOS3 システムから HiRDB に移行した場合に，データベースに格納していた文字データを VOS3 システムの文字列データの照合順で検索，代入，比較ができます。 • UTF-16 で文字データの検索，代入，比較ができます。	特にありません。	「文字集合の指定」を参照してください。
WITHOUT ROLLBACK オプションの指定	採番業務で使用する表を定義する場合，表に対する更新処理完了を契機に更新行に対する排他が解除されるため，排他待ちの発生を削減できます。	特にありません。	「WITHOUT ROLLBACK オプションの指定」を参照してください。
改竄防止機能の指定	表データを誤って，又は不当に更新されることを防止できます。	改竄防止表がある RD エリアに対して使用できない機能や，SQL，ユティリティ，コマンドの実行に制限があります。	「改竄防止機能の指定」を参照してください。

設計作業ごとの検討項目	長所	短所	記載箇所
繰返し列を含む表	• 複数の表の結合が不要になります。 • 重複する情報がなくなるため，ディスク容量を削減できます。 • 関連データ（繰返しデータ）が近くに格納されるため，別の表にするよりもアクセス性能が優れています。	特にありません。	「繰返し列を含む表」を参照してください。
抽象データ型を含む表	複雑な構造のデータを表に格納して，通常の表データと同様に操作できます。	特にありません。	「抽象データ型を含む表」を参照してください。
共用表	• 複数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。 • トランザクションの多重実行時など，並列処理の効率が上がります。	共用表を更新する場合，全バックエンドサーバで，更新する共用表がある RD エリアに排他を掛けるので，共用 RD エリアのほかの表にアクセスする業務があると，デッドロックが発生することがあります。	「共用表」を参照してください。
参照制約	複数の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ます。	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。	「参照制約」を参照してください。
検査制約	データの追加又は更新時のチェックを自動化できます。	検査制約が定義された表を更新する場合，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。	「検査制約」を参照してください。
圧縮表	• 表データの格納効率が向上し，データベースの容量を削減できます。 • HiRDB がデータ圧縮をするため，UAP 開発時にデータ圧縮処理が不要になります。	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。	「圧縮表」を参照してください。
一時表	• トランザクション又は SQL セッションごとに専用の表を作成するため，ほかのユーザの影響を受けないで処理できます。 • 一時表に中間処理結果を格納してさらに加工して最終的な結果を得るなど，複雑な処理ができます。 • 一時表は自動的に削除されるため，後処理が不要です。	HiRDB 開始時，又は一時表に最初に INSERT 文が実行された時に，一時表用 RD エリアを初期化するオーバヘッドが発生します。	「一時表」を参照してください。

注※

次に示す場合は，**ハッシュ分割表のリバランス機能**を使用することをお勧めします。

- 表をハッシュ分割する場合
- データ量の増加が見込まれる場合

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。

11.2　表の正規化

11.2.1　表の正規化の概要

表を正規化することは，表の格納効率や処理効率の向上を図る上で重要です。表を正規化する際は，表の構成列を検討します。ここでは，次に示す正規化について説明します。

- 表の格納効率を向上させる正規化
- 表の処理効率を向上させる正規化

(1)　表の格納効率を向上させる正規化

一つの表に同じような情報を持つ列が複数ある場合は，この表を複数の表に分けて，それぞれの表に同じような情報を持つ列がなくなるように正規化します。これによって，表に対するデータの格納効率が良くなります。これを次の図の例で説明します。

図 11-1　表に同じような情報を持つ列が複数ある場合

● 正規化前

ZAIKO

商品番号	商品コード	商品名	単価	在庫量
SNO	SCODE	SNAME	TANKA	SURYO
01010	101	ブラウス	3500	62
01011	101	ブラウス	3500	85
02021	202	ポロシャツ	3640	67
03530	353	スカート	4760	18
03531	353	スカート	4760	56
04121	412	セーター	8400	22
05910	591	ソックス	250	300
05911	591	ソックス	250	90
05912	591	ソックス	250	280
06710	671	トレーナー	4500	45
06711	671	トレーナー	4500	76

● 正規化後

ZAIKO

SNO	SNAME	TANKA	SURYO
01010	ブラウス	3500	62
01011	ブラウス	3500	85
02021	ポロシャツ	3640	67
03530	スカート	4760	18
03531	スカート	4760	56
04121	セーター	8400	22
05910	ソックス	250	300
05911	ソックス	250	90
05912	ソックス	250	280
06710	トレーナー	4500	45
06711	トレーナー	4500	76

SHOHIN

SCODE	SNAME
101	ブラウス
202	ポロシャツ
353	スカート
412	セーター
591	ソックス
671	トレーナー

〔説明〕

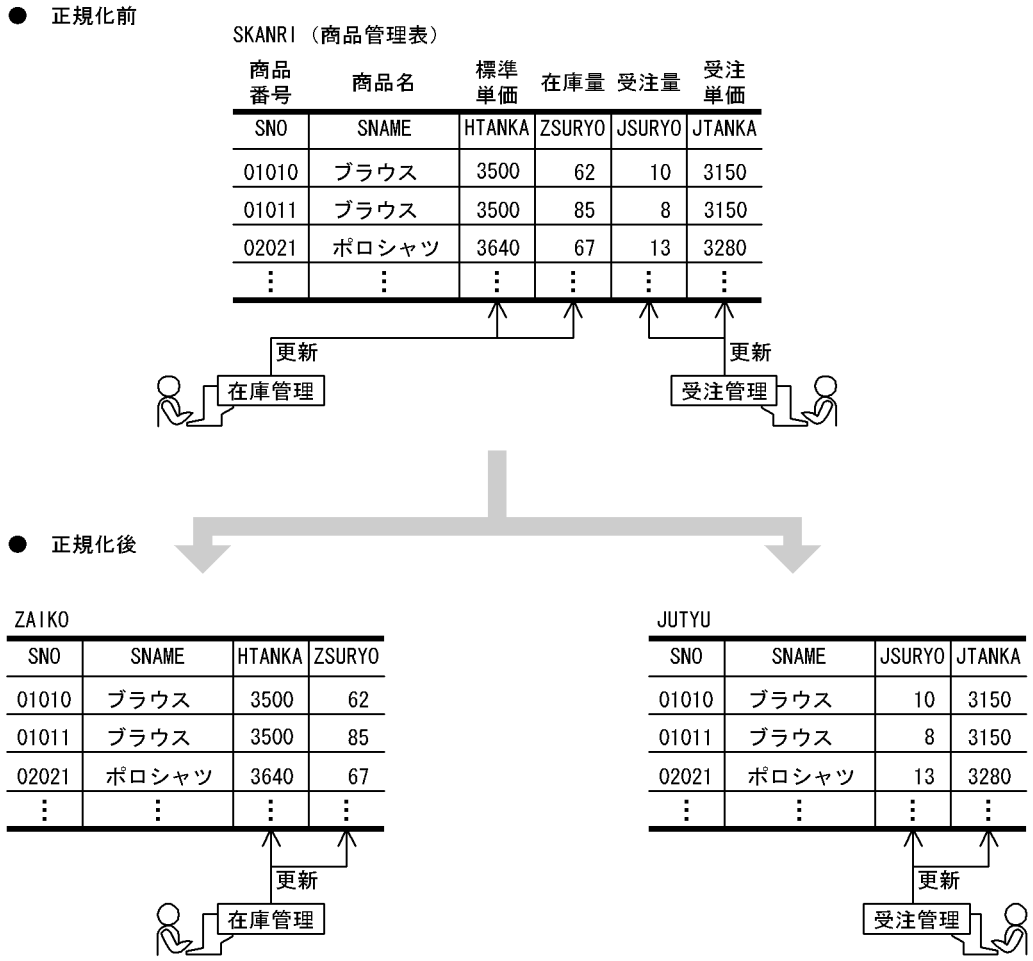
正規化前の ZAIKO 表の SCODE の列と SNAME の列は 1 対 1 で対応し，それぞれの列の情報は冗長になっています。このような場合は，ZAIKO 表から，SCODE の列と SNAME の列で構成される SHOHIN 表をほかに作成します。このとき，SHOHIN 表では，SCODE の列と SNAME の列に重複した情報を持たないようにします。

(2)　表の処理効率を向上させる正規化

(a)　複数の業務で同一の表を使用する場合

複数の業務で同一の表を使用する場合は，それぞれの業務で使用する列によって，この表を業務ごとの表に正規化します。これによって，それぞれの表に対する同時実行性が向上します。これを次の図の例で説明します。

図 11-2　複数の業務で同一の表を使用する場合



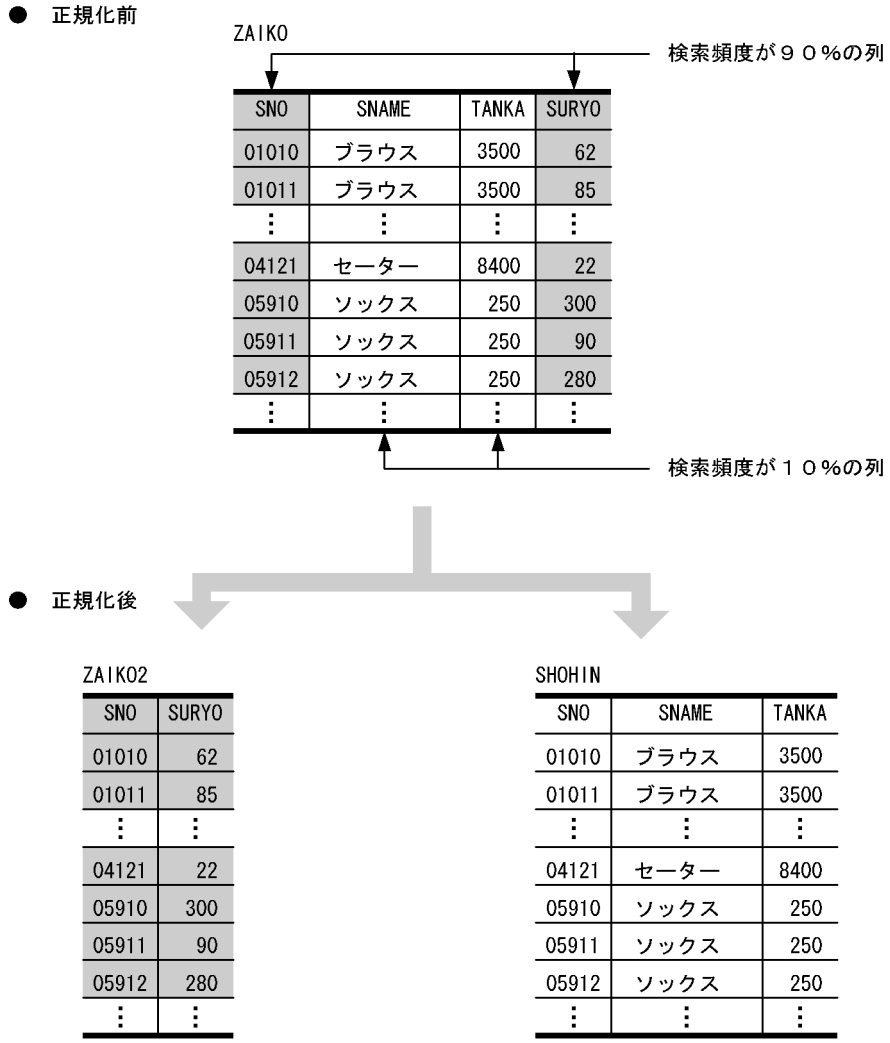
〔説明〕

正規化前の SKANRI 表（商品管理表）を在庫管理業務と受注管理業務で使用しています。このような場合は，SKANRI 表を在庫管理業務だけで使用する ZAIKO 表と，受注管理業務だけで使用する JUTYU 表に正規化します。

(b)　アクセス頻度の高い列と低い列がある場合

一つの表に，アクセス頻度が高いと考えられる列と低いと考えられる列がある場合は，この表をアクセス頻度が高い列で構成した表と，アクセス頻度が低い列で構成した表に正規化します。これを次の図の例で説明します。

図 11-3　アクセス頻度の高い列と低い列がある場合



〔説明〕

正規化前の ZAIKO 表を検索する場合，SNO 及び SURYO の列と，SNAME 及び TANKA の列の間で，検索頻度の比率が 9：1 であるとします。このような場合は，ZAIKO 表を検索頻度の高い列の集まり（ZAIKO2 表）と検索頻度の低い列の集まり（SHOHIN 表）の二つの表に正規化します。

例えば，ZAIKO 表を全件検索するのに，10000 回の物理的な入出力が必要であるとします。ZAIKO 表を ZAIKO2 表と SHOHIN 表に分割したことで，ZAIKO2 表の検索が 4500 回（=5000×0.9），SHOHIN 表の検索が 500 回（=5000×0.1），合計 5000 回の物理的な入出力で済むことになり，全体としての表の処理効率が向上します。

11.3　表の横分割

ここでは，表の横分割の設計方法について説明します。

11.3.1　表の横分割の概要

一つの表を複数のユーザ用 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。なお，表を横分割する RD エリアは，それぞれ異なるディスクに配置することを原則とします。

(1)　適用基準

次に示す場合に表を横分割することをお勧めします。

- データ量が多い場合
- 特定の時間帯にアクセスが集中する場合
- 表の分割単位でユーザ用 RD エリアの運用（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）をする場合

(2)　定義方法

定義系 SQL の CREATE TABLE で定義します。定義例は「横分割表の作成」を参照してください。

11.3.2　表の横分割の種類

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

キーレンジ分割とは，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。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。

(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。

(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。なお，境界値指定の場合，マトリクス分割もできます。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。

(2) ハッシュ分割

ハッシュ分割とは，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です。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。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。ハッシュ分割は，境界値指定のキーレンジ分割と組み合わせてマトリクス分割ができます。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。

ハッシュ分割には**フレキシブルハッシュ分割**と **FIX ハッシュ分割**があります。

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。

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。

(a) 分割キーの選択方法

分割キーには次に示すようなキーを指定してください。

- キー値の偏りが少ない
- キーの値に重複が少ない

また，ハッシュ分割では，分割キーに単一列と複数列が選択できます。単一列を指定した場合，分割列のキー値の種類が少なかったり，キー値に偏りがあるとデータを均等に分割できないことがあります。この場合，分割する列名を複数指定して，データを RD エリアに均等に分割させるようにします。

(b) ハッシュ関数の種類

ハッシュ分割で使用する**ハッシュ関数**には，次のものがあります。

- HASH0
- HASH1
- HASH2
- HASH3
- HASH4
- HASH5

- HASH6
- HASHA
- HASHB
- HASHC
- HASHD
- HASHE
- HASHF
- HASHZ

リバランス表でない場合，又はマトリクス分割表の第 2 次元にハッシュ関数を指定する場合

HASH0～HASH6 又は HASHZ のどれかを指定してください。HASH6 が最も均等にハッシングされるので，通常は HASH6 を指定してください。ただし，分割キーのデータによっては均等にならない場合もあるので，そのときにはほかのハッシュ関数を指定してください。

リバランス表の場合

HASHA～HASHF のどれかを指定してください。HASHF が最も均等にハッシングされるので，通常は HASHF を指定してください。ただし，分割キーのデータによっては均等にならない場合もあるので，そのときにはほかのハッシュ関数を指定してください。

各ハッシュ関数の詳細については，マニュアル「HiRDB SQL リファレンス」の「CREATE TABLE（表定義）」の「オペランド」のハッシュ関数名の説明を参照してください。

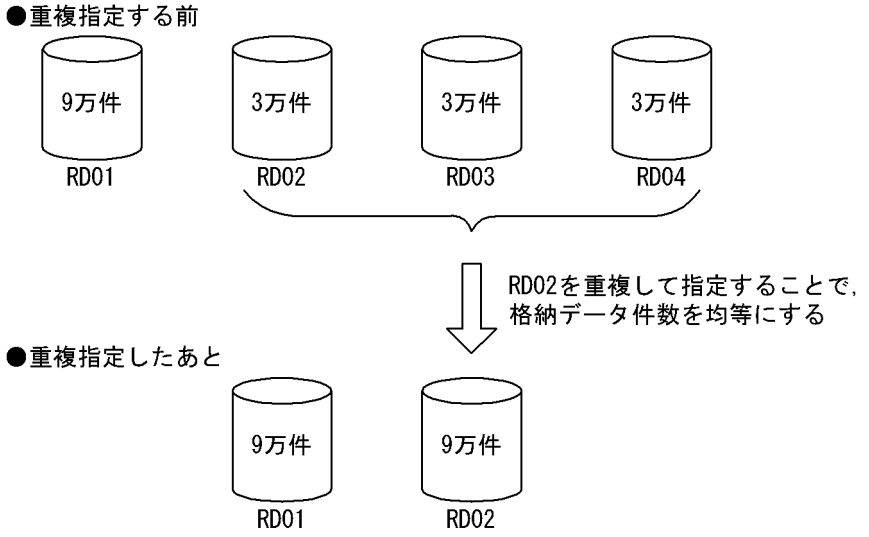
(c) ハッシュ関数の選択方法

実際にデータベースにデータを格納して選択する方法

この場合のハッシュ関数の選択手順を次に示します。

1. 分割キーに対応して有効なハッシュ関数を指定します。
2. データベース状態解析ユティリティ（pddbst）で RD エリアごとに格納されている行数を確認します。
3. RD エリアごとに格納している行数に偏りがある場合には，ハッシュ関数を変更し，RD エリアごとの格納行数が均等になるようにします。
4. 3.の方法で格納行数が均等にならない場合は，格納行数の少ない RD エリアを重複して指定することで，格納行数が均等になるようにします。この場合の例を次の図に示します。

図 11-4　ハッシュ分割で表格納用 RD エリアを重複指定する例



表分割ハッシュ関数を使用した UAP を作成し，ハッシュ関数を選択する方法

この場合のハッシュ関数の選択手順を次に示します。

1. HiRDB からライブラリとして提供されている，表分割ハッシュ関数（分割キーのデータ値を入力すると分割条件指定順序を出力する関数）を使用して，RD エリアごとのデータ件数の偏りを求める UAP を作成します。
2. ハッシュ関数ごとに，表分割ハッシュ関数が出力する分割条件指定順序ごとの件数を求め，最も偏りが少ないハッシュ関数を選択します。

表分割ハッシュ関数を使用した 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。

(d)　ハッシュ関数が使用されるタイミング

ハッシュ関数は次に示すときに使用されます。

- 表単位のデータロード時
- データの追加時
- 表単位のデータのリロード時

(3)　キーレンジ分割，フレキシブルハッシュ分割及び FIX ハッシュ分割の相違点

キーレンジ分割，フレキシブルハッシュ分割及び FIX ハッシュ分割の相違点を次の表に示します。

表 11-2　キーレンジ分割，フレキシブルハッシュ分割及び FIX ハッシュ分割の相違点

相違点	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
データベース設計	キーレンジを考慮して，データベースを設計する必要があります。	キーレンジを考慮しないでデータベースを設計できます。	キーレンジを考慮しないでデータベースを設計できます。
検索	検索条件によって，該当データが存在する可能性のあるバックエンドサーバだけ検索します。※1	対象表のある全バックエンドサーバで検索します。	分割列に対して次に示す探索条件を指定した場合に，該当データが存在する可能性のある RD エリアだけ検索します。※1 ハッシュ関数に HASH1～HASH6，HASHA～HASHF を指定した場合 • 比較述語（=） • IN 述語 ハッシュ関数に HASH0，HASHZ を指定した場合 • 比較述語（=） • IN 述語 • 比較述語（<，>，<=，>=）による範囲条件 • BETWEEN 述語 •（前方一致比較の）LIKE 述語 •（前方一致比較の）SIMILAR 述語
データ増加時の対応	キーが増加するデータの場合，データの格納が特定の RD エリアに偏ります。	データが増加しても，常に均等に RD エリアに格納されます。	データが増加しても，常に均等に RD エリアに格納されます。
RD エリア閉塞時の運用	閉塞している RD エリアをアクセスしない検索条件であれば SQL を実行できます。※2	検索する表が格納されている RD エリアを一つでも閉塞すると，検索条件にかかわらず SQL を実行できません。	閉塞している RD エリアをアクセスしない検索条件であれば SQL を実行できます。※2
表の分割数の変更	表の再作成及び表の再編成が必要です。	ALTER TABLE で RD エリアを追加でき，表の再編成は必須ではありません。	表の再作成及び表の再編成が必要です。ただし，表にデータが入っていないときだけ ALTER TABLE で RD エリアを追加できます。
RD エリア単位のデータロード・リロード	該当する RD エリアに格納するデータかどうかをチェックします。	該当する RD エリアに格納するデータかどうかをチェックしません。	該当する RD エリアに格納するデータかどうかをチェックします。
データロード時の RD エリア単位による入力データファイルの作成方法	キーレンジを考慮して，入力データを RD エリアごとに分類します。	RD エリアごとのデータ件数が均等になるように，任意の方法で分類します。	表分割ハッシュ関数※3 を使用したアプリケーションを作成し，入力データを RD エリアごとに分類します。

相違点	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
分割キーの更新	同値更新だけできます。	更新できます。	同値更新だけできます。
クラスタキーの UNIQUE 定義及び UNIQUE 指定のインデクス定義	UNIQUE を指定できます。	UNIQUE を指定できません。	UNIQUE を指定できます。
ALTER TABLE による分割格納条件の変更	次の分割方法の場合に変更できます。 • 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）	変更できません。ただし，ALTER TABLE で RD エリアの追加はできます。	変更できません。ただし，ALTER TABLE で RD エリアの追加はできます。

注※1

ASSIGN LIST 文の場合，検索条件に該当しないバックエンドサーバにも負荷が掛かります。

注※2

ASSIGN LIST 文の場合，表全体が閉塞扱いになります。

注※3

表分割ハッシュ関数を使用した 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。

(4) 表の横分割定義時の指定規則

表の横分割定義時の指定規則を次に示します。

- キーレンジ分割の場合
  - 指定できる分割キー※1は 1 個です。分割キーの更新はできません。
  - 格納条件指定※2の場合，同じ RD エリアを複数指定できません。境界値指定※3の場合，同じ RD エリアを複数指定できますが，連続して同じ RD エリアを指定することはできません。
- ハッシュ分割の場合
  - 指定できる分割キー※1は最大 16 個です。ただし，同じ分割キーを重複して指定することはできません。フレキシブルハッシュ分割は，分割キーの更新ができますが，FIX ハッシュ分割は分割キーの更新はできません。

注※ 1

次のデータ型の列及び繰返し列は，分割キーに指定できません。

- 定義長が 256 バイト以上の CHAR，VARCHAR，MCHAR，MVARCHAR 型
- 定義長が 28 文字以上の NCHAR，NVARCHAR 型
- BLOB 型

- BINARY 型
- 抽象データ型
- 既定値に CURRENT_TIMESTAMP USING BES を指定した TIMESTAMP 型

注※2

格納条件を複数指定した場合，格納条件の指定順に条件を評価し，最初に真となった格納条件に指定した RD エリアに格納します。すべての条件で真とならない場合，格納条件を指定していない RD エリアに格納します。ただし，格納条件を指定していない RD エリアがない場合，どの RD エリアにも格納されません。また，条件を評価した結果，行が 1 行も格納されない RD エリアがある指定の表定義はできません。

注※3

境界値には定数を指定します。ただし，長さが 0 の文字列定数は指定できません。境界値を複数指定する場合，昇順となるように指定してください。また，境界値を指定しない RD エリアを最後に必ず指定してください。

(5) キーレンジ分割（格納条件指定）の例

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 11-5　キーレンジ分割（格納条件指定）の例

● USR01に格納された横分割表

ZAIKO（101L～353Mのデータ）

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
353L	スカート	赤	4760	18
353M	スカート	緑	4760	56

● USR02に格納された横分割表

ZAIKO（411M～617Mのデータ）

SCODE	SNAME	COL	TANKA	ZSURYO
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
671L	トレーナー	白	4500	45
671M	トレーナー	青	4500	76

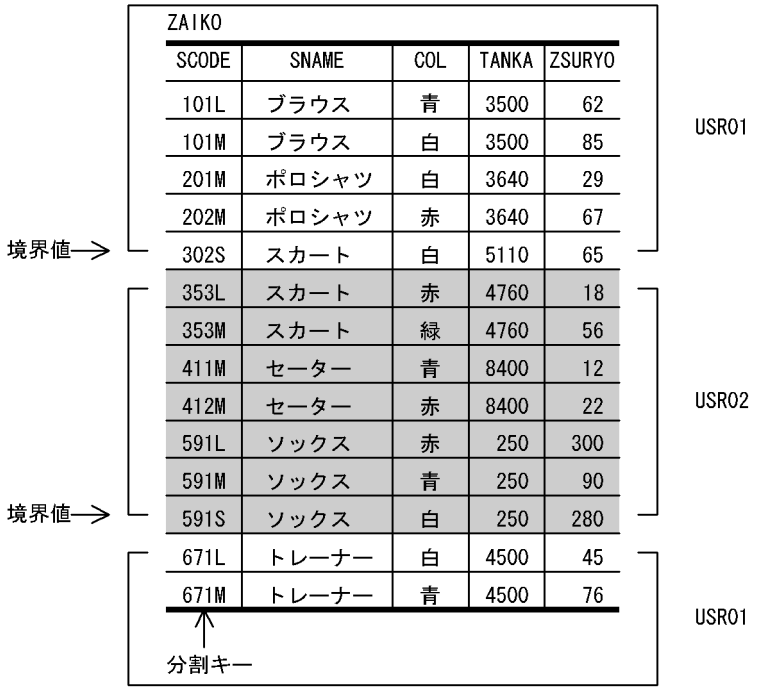
［説明］

ZAIKO 表の商品コード列（SCODE）の範囲（100L～399S と 400L～699S）を条件として，複数のユーザ用 RD エリア（USR01 と USR02）に横分割します。

(6)　キーレンジ分割（境界値指定）の例

キーレンジ分割（境界値指定）の例を次の図に示します。

図 11-6　キーレンジ分割（境界値指定）の例



● USR01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

● USR02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〕

ZAIKO 表の商品コード列（SCODE）の 302S，591S を境界値として，複数のユーザ用 RD エリア（USR01 と USR02）に横分割します。

(7)　フレキシブルハッシュ分割及び FIX ハッシュ分割の例

フレキシブルハッシュ分割，FIX ハッシュ分割の例を次の図に示します。

図 11-7　フレキシブルハッシュ分割，FIX ハッシュ分割の例

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
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
671L	トレーナー	白	4500	45
671M	トレーナー	青	4500	76

分割キー

● USR01に格納された横分割表

ZAIKO

SCODE	SNAME	COL	TANKA	ZSURYO
101L	ブラウス	青	3500	62
353M	スカート	緑	4760	56
411M	セーター	青	8400	12
412M	セーター	赤	8400	22
591M	ソックス	青	250	90
591S	ソックス	白	250	280
671M	トレーナー	青	4500	76

● USR02に格納された横分割表

ZAIKO

SCODE	SNAME	COL	TANKA	ZSURYO
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
353L	スカート	赤	4760	18
591L	ソックス	赤	250	300
671L	トレーナー	白	4500	45

〔説明〕

ZAIKO 表の商品コード列（SCODE）を分割キーとして，ハッシュ関数 HASH6 を使用して複数のユーザ用 RD エリア（USR01 と USR02）に横分割します。

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。

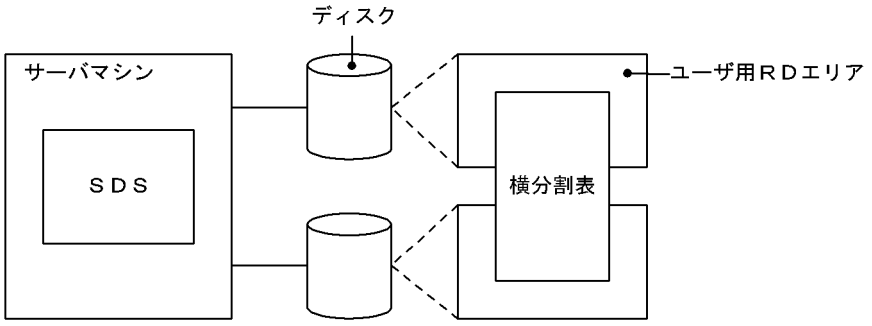
11.3.3　表の横分割の形態

表の横分割の基本的な形態には，次に示す 2 種類があります。

- サーバ内の横分割（HiRDB/シングルサーバでの形態）
- サーバ間の横分割（HiRDB/パラレルサーバでの形態）

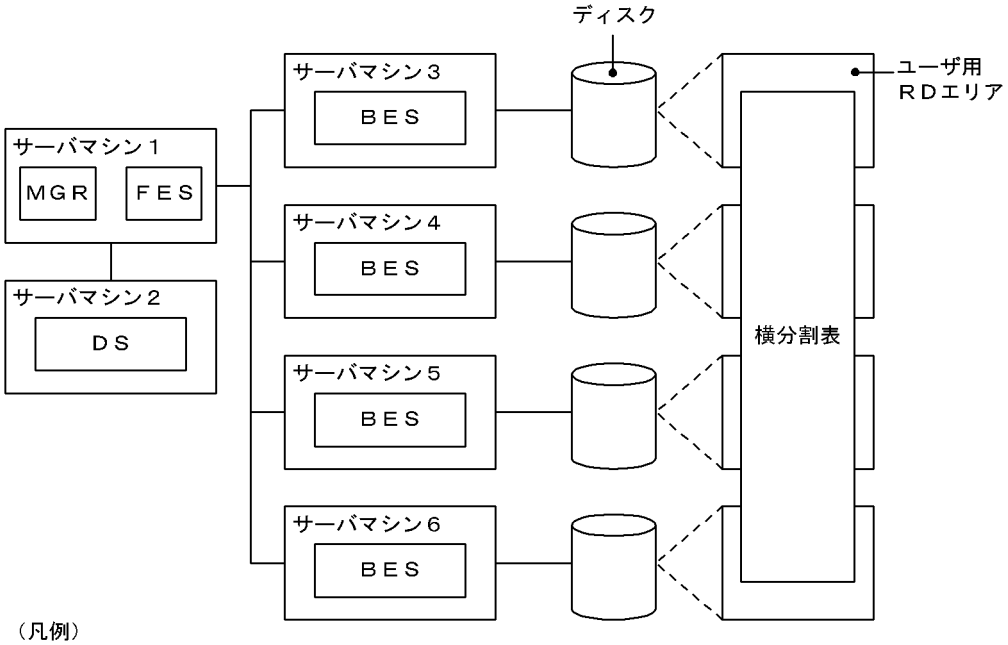
それぞれの形態を次の図に示します。

図 11–8　表の横分割の形態（HiRDB/シングルサーバの場合）



（凡例）
　ＳＤＳ：シングルサーバ

図 11–9　表の横分割の形態（HiRDB/パラレルサーバの場合）



（凡例）
　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
　ＢＥＳ：バックエンドサーバ

11.3.4　表の横分割の効果

表を横分割して得られる効果を次に示します。

(1)　HiRDB/シングルサーバの場合

操作性の向上

ユーザ用 RD エリアごとに，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。

キーレンジ分割の場合

ディクショナリ表の SQL_DIV_TABLE 表を検索することで，表のデータをどのユーザ用 RD エリアに格納したかが分かります。このため，ユーザ用 RD エリアに障害が発生した場合に，どのデータが利用できないかが分かります。なお，ディクショナリ表の検索方法と SQL_DIV_TABLE 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。

(2) HiRDB/パラレルサーバの場合

性能の向上

- 表にアクセスする処理を複数のユーザ用 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が図れます。
- 表にアクセスする処理の負荷を複数のバックエンドサーバに分散できます。

操作性の向上

効果は HiRDB/シングルサーバの場合と同様です。

11.3.5 設計上の考慮点

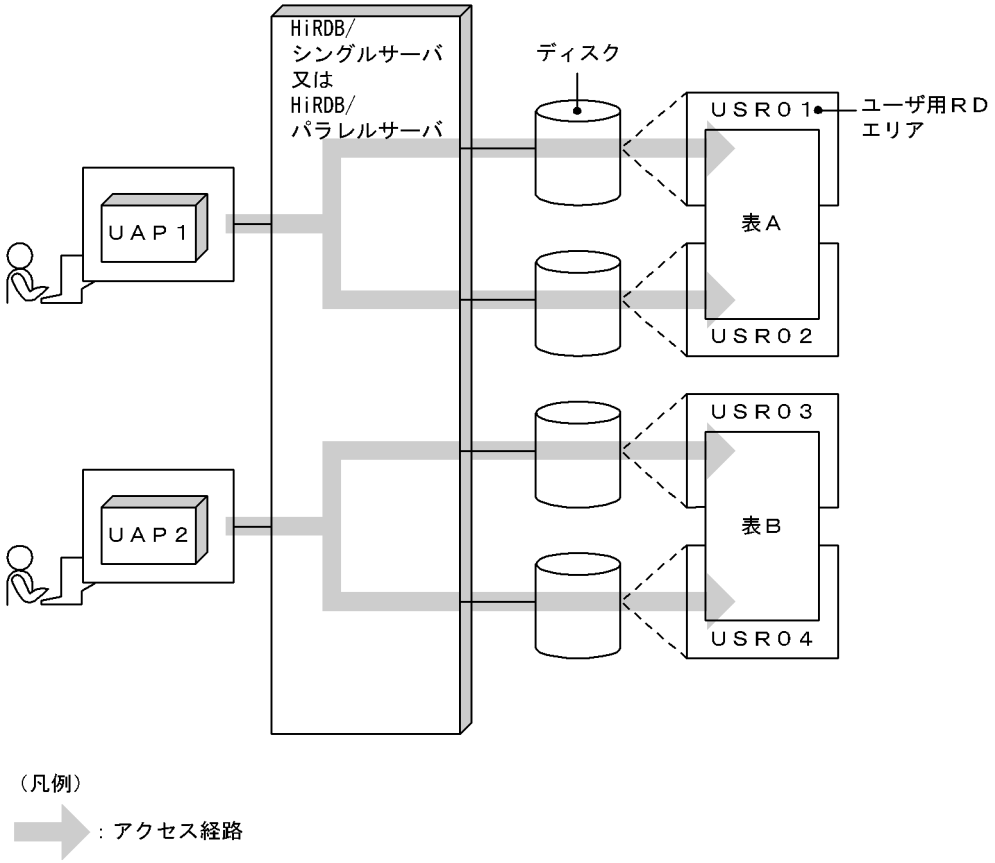
(1) HiRDB/シングルサーバと HiRDB/パラレルサーバでの共通の考慮点

HiRDB/シングルサーバと HiRDB/パラレルサーバでの共通の考慮点を次に示します。

(a) ディスクに対するアクセスの競合を考慮した横分割

複数の UAP がそれぞれ別々の表に同時にアクセスする場合は，これらの表をそれぞれ異なるディスク上の異なるユーザ用 RD エリアにわたって横分割します。ディスクに対するアクセスの競合を考慮した横分割の概要を次の図に示します。

図 11-10　ディスクに対するアクセスの競合を考慮した横分割の概要



〔説明〕

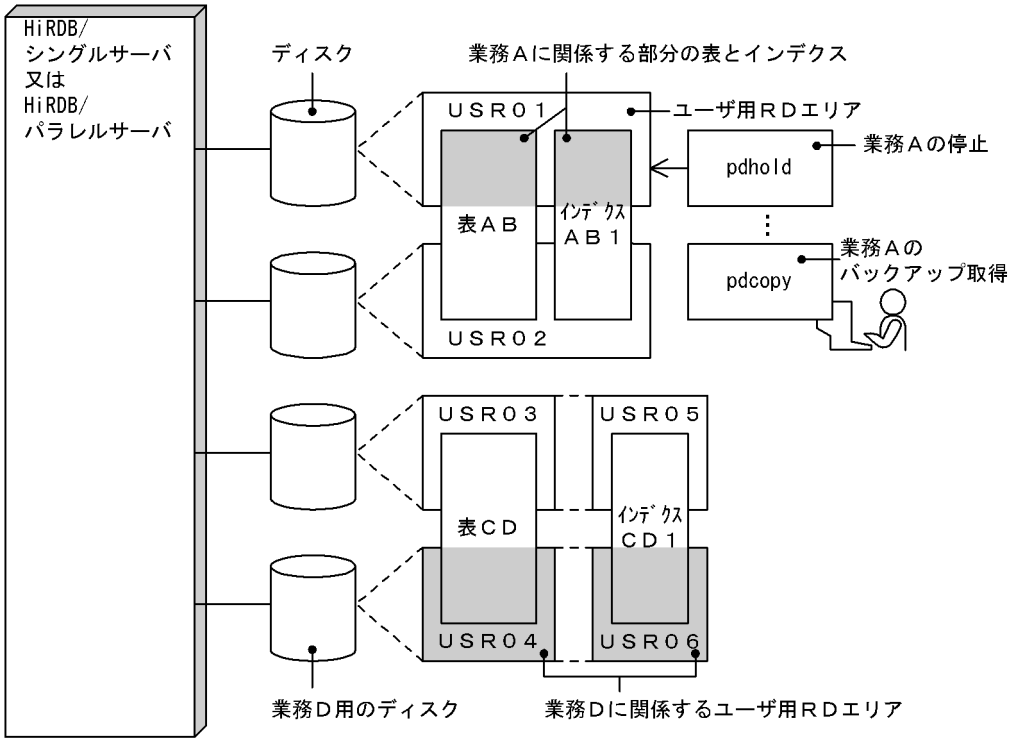
表 A と表 B をそれぞれ異なるディスク上のユーザ用 RD エリア USR01～USR02 と USR03～USR04 にわたって横分割しています。このため，UAP1 と UAP2 が同時に表 A と表 B にアクセスしても，ディスクに対するアクセスの競合による待ちが発生しないため，待ち時間を短くできます。

しかし，一つのディスク上のユーザ用 RD エリアに複数の表を格納した場合，これらの表に対して複数の UAP が同時にアクセスすると，ディスクに対するアクセスの競合が発生します。このため，この表にアクセスできた一つの UAP のアクセスが終了するまで，ほかの UAP が待たされることになり，待ち時間が長くなります。

(b)　操作性を考慮した横分割

操作性を考慮した横分割の概要を次の図を基に説明します。

図 11-11　操作性を考慮した横分割の概要



〔説明〕

- 表とインデクスの同一ユーザ用 RD エリアへの格納

  検索性能よりはむしろ，表の作成，表の再編成，ユーザ用 RD エリアのバックアップの取得，RD エリアの回復などの運用の操作性を重視する場合は，横分割した表とそれに対応するインデクスを同じユーザ用 RD エリアに格納します。これによって，ユーザ用 RD エリアごとの独立した運用が便利になります。

  図「操作性を考慮した横分割の概要」の例では，表 AB のうち，業務 A に関係する部分の表とインデクスを専用のユーザ用 RD エリア USR01 にまとめて格納しています。これによって，例えば，業務 A を停止する場合は，pdhold コマンド（RD エリアの閉塞）で運用できます。また，データベース複写ユティリティ（pdcopy）を使用した業務単位でのバックアップが取得しやすくなります。

- 関連するユーザ用 RD エリアの同一ディスクへの配置

  横分割した表とそれに対応するインデクスをそれぞれ異なるユーザ用 RD エリアに格納する場合は，これらの互いに関連するユーザ用 RD エリアを同一のディスクに配置します。これによって，ディスクごとに独立したユーザ用 RD エリアの運用ができます。

  図「操作性を考慮した横分割の概要」の例では，表 CD のうち，業務 D に関係する部分の表とインデクスをそれぞれ格納したユーザ用 RD エリア USR04 と USR06 を同一のディスクに配置しています。これによって，ディスクごとに業務の運用ができます。

(2)　HiRDB/パラレルサーバ固有の考慮点

HiRDB/パラレルサーバ固有の考慮点を次に示します。

(a) ディスクに対するアクセスの負荷を考慮した横分割

- 複数のバックエンドサーバにわたる横分割

  一つのバックエンドサーバのディスクに複数のユーザ用 RD エリアを配置したとき，それぞれのユーザ用 RD エリアに格納された表のアクセス頻度がどれも高い場合，このバックエンドサーバでのディスクに対するアクセスの負荷が高くなります。

  このため，アクセス頻度の高い表は，複数のバックエンドサーバの異なるディスク上のユーザ用 RD エリアにわたって横分割します。このとき，それぞれのバックエンドサーバでの表に対するアクセス頻度が均等になるようにします。

- 複数のサーバマシンにわたるディスクアクセスの並列化

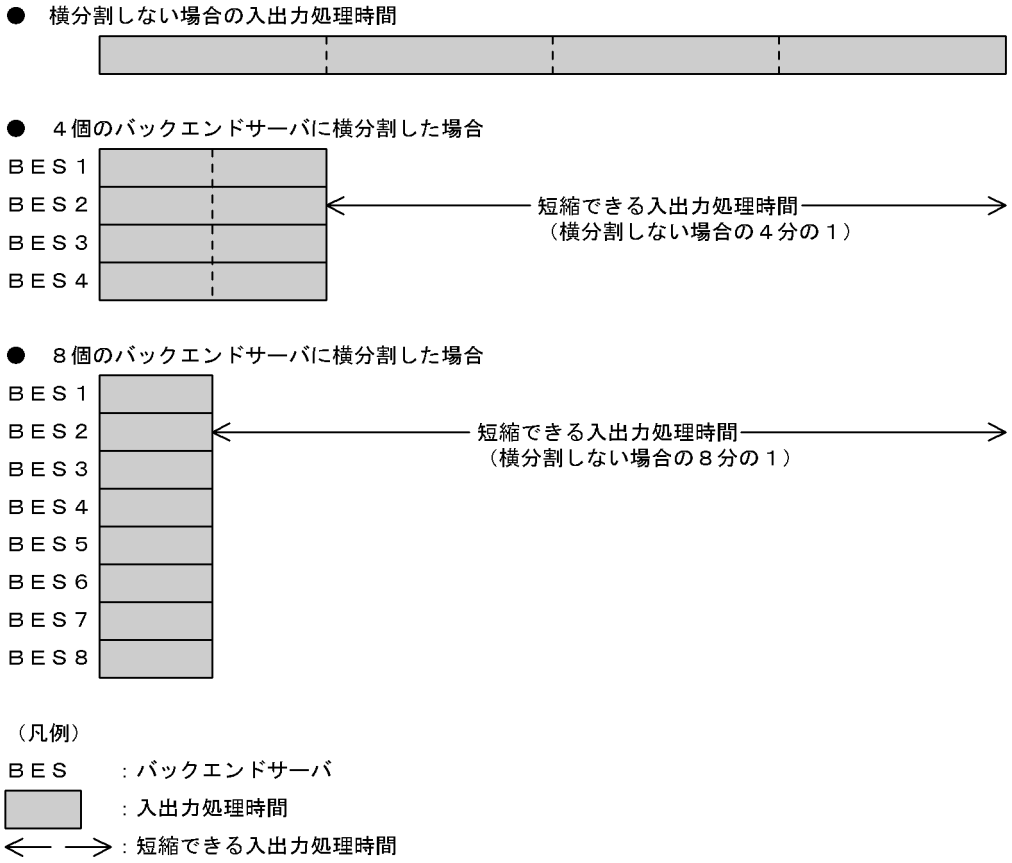
  あるサーバマシンの一つのバックエンドサーバのユーザ用 RD エリアに格納された表に対する処理が，入出力が主体で CPU の負荷が低い場合，ディスクに対するアクセスの負荷が複数のサーバマシンにわたって均等にならないため，並列処理の効率が下がります。

  このため，サーバマシン内の CPU ビジー率に余裕がある場合は，このサーバマシンに更にバックエンドサーバとユーザ用 RD エリアを配置して，ディスクに対するアクセスの並列度を向上させるようにします。

(b) 入出力処理の並列度を考慮した横分割

表をできるだけ多くのサーバマシンにわたって横分割すると，並列処理によって表に対する入出力処理時間が短縮できます。横分割できるサーバマシン数に限界があれば，各サーバマシンにバックエンドサーバとディスクを増やして表を横分割することで，入出力処理の並列効果が得られます。表を横分割するバックエンドサーバの数に伴う入出力処理性能の概要を次の図に示します。

図 11–12　表を横分割するバックエンドサーバの数に伴う入出力処理性能の概要



ただし，表を横分割するバックエンドサーバの数が多過ぎると，各バックエンドサーバでの処理結果をフロントエンドサーバに返すための通信量が増加します。このため，データベースの運用や SQL 処理（容量の大きい表から大量のデータを検索する SQL 処理かどうか）を考慮して，表を横分割するバックエンドサーバの数を決定する必要があります。

(c)　表に対するアクセス頻度を考慮した横分割

それぞれのバックエンドサーバでの表に対するアクセス頻度が均等になるように表を横分割します。

アクセス頻度を均等にするための考慮点を次に示します。

キーレンジ分割の場合

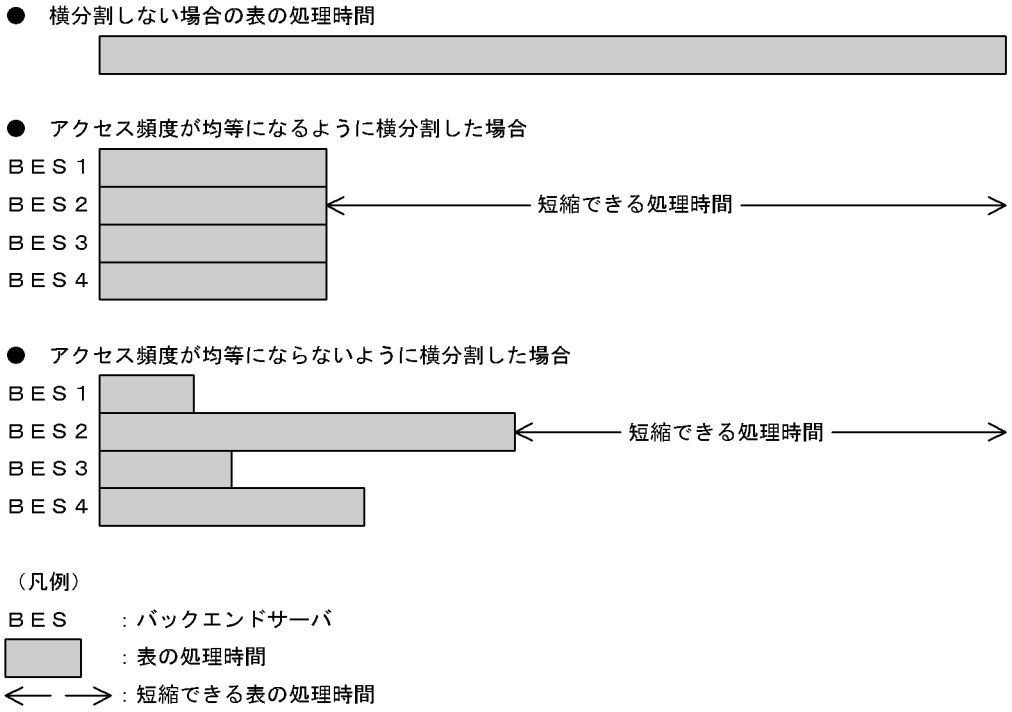
- 表の横分割を定義する場合に，分割キーに UNIQUE を指定して，データ量が均等になるようにします。
- 表を横分割する場合に，あるキーレンジのデータに対するアクセス回数が，ほかのキーレンジのデータに対するアクセス回数よりも多いと予想できるときは，アクセス回数が多いと予想できるキーレンジのデータを更にキーレンジで分割します。

フレキシブルハッシュ分割，FIX ハッシュ分割の場合

- ハッシュ関数を変更して，データ量が均等になるように調整します。
- 分割キーに偏りや重複がないものを選択して，データ量が均等になるように調整します。

複数のバックエンドサーバにわたって表を横分割する場合でも，表に対するアクセス頻度が均等になるように横分割した場合と，均等にならないように横分割した場合とでは，表に対する並列処理の性能に差が生じます。表に対するアクセス頻度の違いによる並列処理性能の違いを次の図に示します。

図 11–13　表に対するアクセス頻度の違いに伴う並列処理性能の違い



［説明］

アクセス頻度が均等になるように表を横分割した場合と，均等にならないように表を横分割した場合とでは，削減できる処理時間が異なります。均等にならないように表を横分割した場合は，バックエンドサーバ BES2 での処理が終了するまで，表に対する処理は終了しないため，並列処理の効果が得られません。

(d)　複雑な検索処理を考慮した横分割

大量のデータの検索や結合処理をするなど，複雑な検索処理を考慮した表の横分割をする場合，次に示す手順で設計します。

1. ディスクに対する処理時間とディスクの台数の決定

   データの規模と処理のパターンから，ディスクへのアクセス頻度（利用率）を求め，これを基にデータをディスクに配分して，ディスクに対する処理時間の目標値を決めます。なお，結合処理をする場合は，結合処理に必要なワークディスク（ソートマージ用に使用する作業表用ファイルを作成する HiRDB ファイルシステム領域の数）を除いてデータを配分します。結合処理に必要な時間は，ディスクに対する処理時間の目標値から除きます。ディスクへのデータ配分から，ディスクの台数を決定します。

2. サーバマシンの台数の決定

データの処理パターンから，サーバマシンでの処理のオーバヘッド時間を求めます。ディスクに対する処理時間とサーバマシンでのオーバヘッド時間が，それぞれ均等になるようにサーバマシンの台数（バックエンドサーバを配置するサーバマシンの台数）を決定します。

3. 結合処理をするサーバマシンの台数の決定

データの処理パターンから，結合処理をするときに掛かるサーバマシンでのオーバヘッド時間を求めます。この時間とディスクに対する処理時間を基に，必要とするフロータブルマシンの台数を決定します。

フロータブルマシンとは，結合処理のような複雑な検索処理をするための専用のバックエンドサーバ（**フロータブルサーバ**）を配置するサーバマシンのことです。フロータブルサーバとするバックエンドサーバは，ユーザ用 RD エリアを割り当てないバックエンドサーバとして定義します。

4. ワークディスクの台数の決定

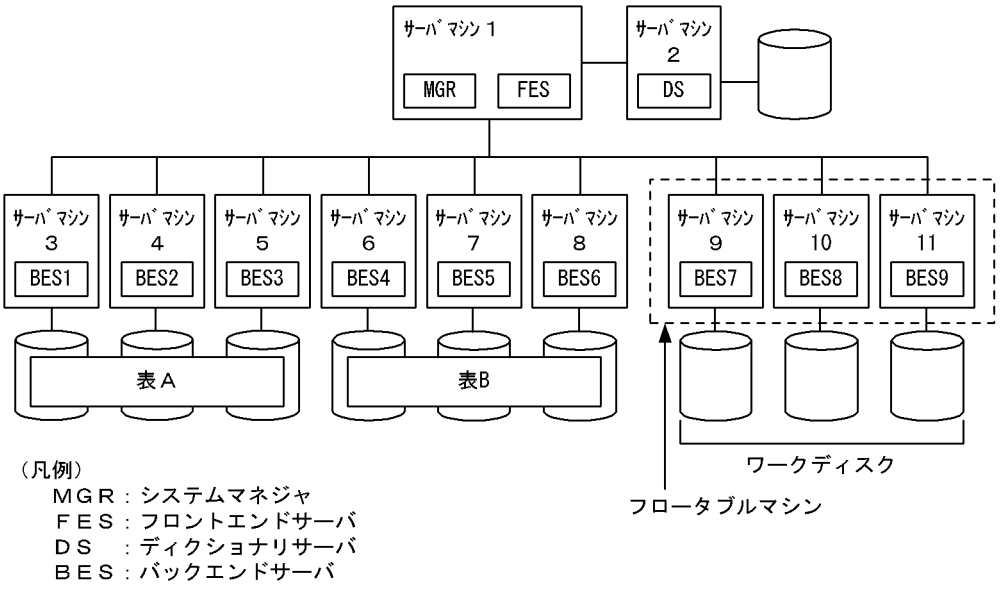
フロータブルマシンには，結合処理の対象となるデータが各バックエンドサーバから均等に分配されます。このときに予想されるデータ量を基に，ワークディスク（作業表用ファイルを作成する HiRDB ファイルシステム領域）の台数を求めます。

5. システム構成の決定

以上で決定したサーバマシンとディスクの台数から，システム全体の構成を決定します。

以上の手順で設計した，複雑な検索処理を考慮した横分割のシステム構成の概念を次の図に示します。

図 11–14　複雑な検索処理を考慮した横分割のシステム構成



注　ＢＥＳ７～ＢＥＳ９はフロータブルサーバ

［説明］

バックエンドサーバ BES1～BES3 とバックエンドサーバ BES4～BES6 が，それぞれ表 A と表 B から結合処理の対象となるデータを読み出します。フロータブルサーバ BES7～BES9 は，バックエンドサーバ BES1～BES6 が読み出したデータを受け取り，並列に突き合わせる処理をします。

このようなシステム構成にすることで，バックエンドサーバ BES1～BES6 での負荷を軽減でき，処理時間を短縮できます。なお，フロータブルサーバを配置しなかった場合は，フロータブルサーバで実行していた結合処理をバックエンドサーバ BES1～BES6 のどれかが実行することになります。

11.3.6　表を横分割する場合の注意

1. 表を横分割した場合は，この表に作成するインデクスも横分割してください。インデクスを一つのユーザ用 RD エリアに格納して，表を横分割して運用すると，インデクスでの排他制御によって，同時実行性が低下することがあります。インデクスの横分割については，「インデクスの横分割」を参照してください。
2. 次に示す場合は，**ハッシュ分割表のリバランス機能**を使用することをお勧めします。
   - 表をハッシュ分割する場合
   - データ量の増加が見込まれる場合

   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。

11.4　表のマトリクス分割

11.4.1　表のマトリクス分割の効果と定義方法

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※

注※

指定できるハッシュ関数は HASH0～HASH6 と HASHZ です。HASHA～HASHF は指定できません。

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。

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要する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多大となる
- 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，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

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多大となる
- 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

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で，ハッシュ分割に指定する RD エリア名を重複指定することをお勧めします。同じ RD エリア名を重複指定することによって，分割数はそのままで，実際に使用する RD エリア数を減らすことができます。また，各 RD エリアに格納するデータ量も均等にできます。RD エリア名を重複指定する場合については，図「ハッシュ分割で表格納用 RD エリアを重複指定する例」を参照してください。

- 1RD エリアのサイズを一定にすることで各 RD エリアの再編成処理の時間を一定にしたいが，第 1 次元分割列の境界値による分割ではデータ量を均等に分ける境界値の指定が難しいため，一定にできない
- 分割数が検索時の性能に影響しているため，分割数を減らしたい

(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）

定義時の規則を次に示します。

- **第 2 次元分割列が境界値指定のキーレンジ分割の場合**
  - 分割キーの指定可能数は 2（第 1 次元分割列の分割キー＋第 2 次元分割列の分割キー）です。第 1 次元分割列と第 2 次元分割列で同じ分割キーは指定できません。
- **第 2 次元分割列がハッシュ分割の場合**
  - 分割キーの指定可能数は 2～16 です。フレキシブルハッシュ分割のとき，第 2 次元分割列の分割キーだけは更新できます。FIX ハッシュ分割のとき，分割キーの更新はできません。

定義例は「マトリクス分割の例」を参照してください。

(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，登録日と店番号によって，表をマトリクス分割します。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	100 より大きい
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。

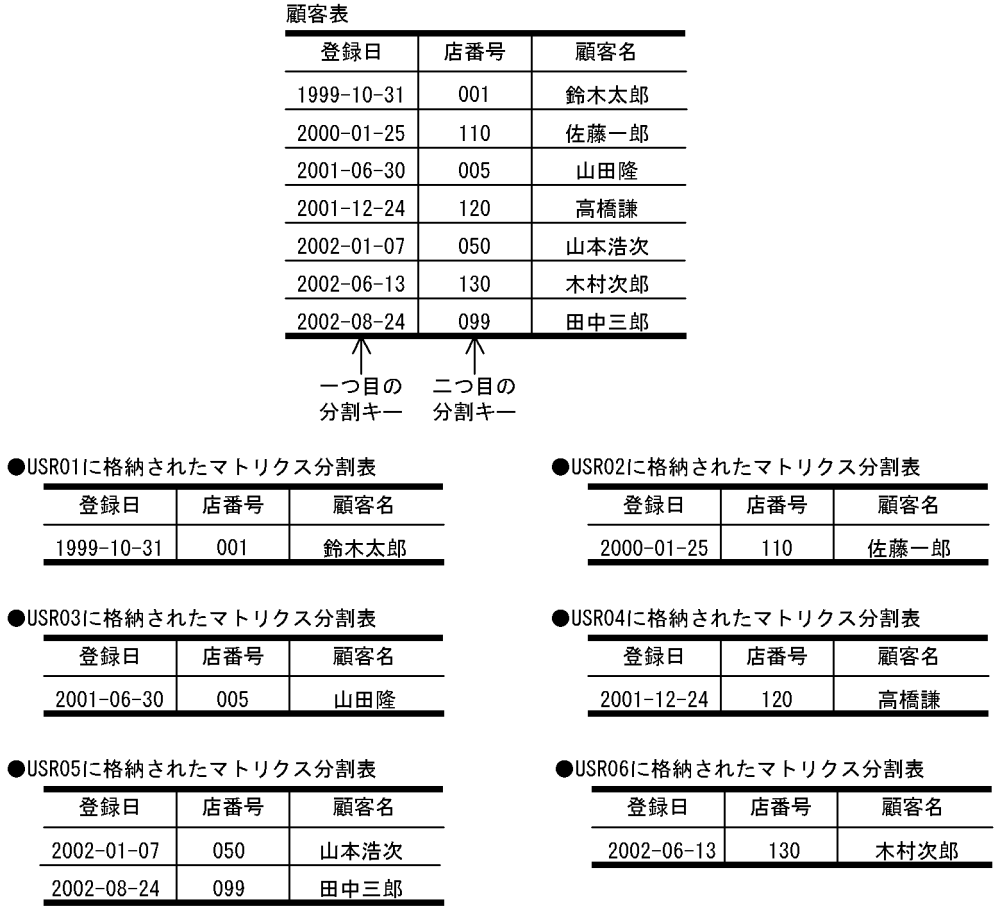
```
CREATE FIX TABLE 顧客表
  (登録日 DATE, 店番号 INT, 顧客名 NCHAR(10))
  PARTITIONED BY MULTIDIM(
  登録日 (('2000-12-31'),('2001-12-31')), …1.
  店番号 ((100))                          …2.
  )IN ((USR01,USR02),(USR03,USR04),(USR05,USR06))
```

〔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 11-15　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三分割することによって，表をマトリクス分割します。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（ハッシュ関数で三分割）		
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 11-16　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

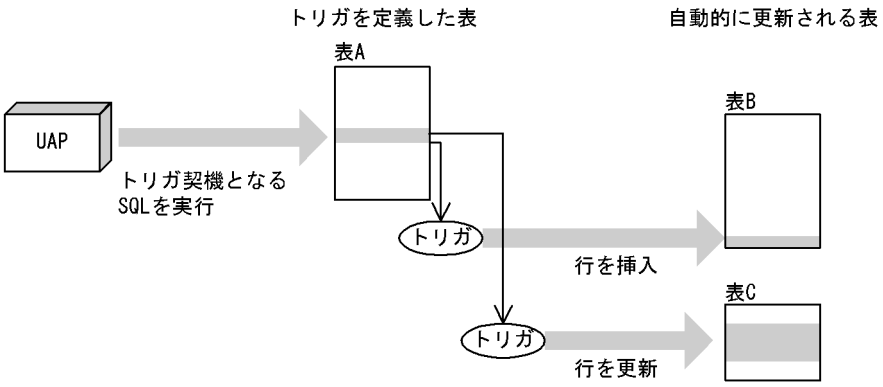
●USR09に格納されたマトリクス分割表

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

11.5　トリガの定義

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 11–17　トリガの概要



〔説明〕

UAP からトリガ契機となる SQL が実行されると，トリガを定義した表 A でトリガが呼び出され，トリガ動作の探索条件を満たしている場合，自動的にトリガ SQL 文（この場合，表 B への行の挿入や表 C への行の更新）が実行されます。

前提条件

トリガを定義する場合，データディクショナリ LOB 用 RD エリアを作成しておく必要があります。データディクショナリ LOB 用 RD エリアは，データベース構成変更ユティリティ（pdmod）で作成します。

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「トリガの管理」を参照してください。

11.5.1　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）

例えば，商品管理表の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という場合，トリガを使用しないと，商品管理表を更新する UAP は常に商品管理履歴表も更新する必要があります。トリガを使用すると，商品管理履歴表への操作を自動化できるため，商品管理表を更新する UAP は商品管理履歴表の更新を考慮する必要がありません。このように，トリガを適切に使用すると UAP を作成するときの負荷を軽減できます。

11.5.2 トリガの定義

(1) 定義の準備

トリガを定義すると，指定したトリガ SQL 文を基に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，「データディクショナリ LOB 用 RD エリアの容量の見積もり」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。

(2) 定義方法

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，他ユーザが所有する表には定義できません。次の項目を指定します。

  - トリガ動作の実行時期

    トリガ動作は表への操作の前（BEFORE），又は後（AFTER）に実行できます。なお，トリガ動作時期に BEFORE を指定したトリガを **BEFORE トリガ**，AFTER を指定したトリガを **AFTER トリガ**といいます。

  - トリガ動作の実行契機

    トリガ動作を実行する契機には，INSERT 文，DELETE 文，又は UPDATE 文の実行があります。

  - トリガを定義する表

    トリガは実表にだけ定義できます。

  - トリガ契機となる SQL の実行前後の行の名称（新旧値別名）

    トリガ契機となる SQL で更新する行に，SQL 実行前の名称（旧値相関名），又は実行後の名称（新値相関名）を指定します。ここで指定した名称を使ってトリガの動作内容を指定できます。

  - トリガ動作

トリガ動作には次の三つの要素があります。

・トリガ SQL 文（自動的に実行させる SQL 文）

・トリガ動作の探索条件（トリガ SQL 文が実行される条件）

・動作が実行されるのが行単位か文単位かの種別

トリガ SQL 文はトリガ動作の探索条件を満たす場合にだけ実行され，条件を省略した場合はトリガ契機となる SQL が実行されると常にトリガ SQL 文が実行されます。

- 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

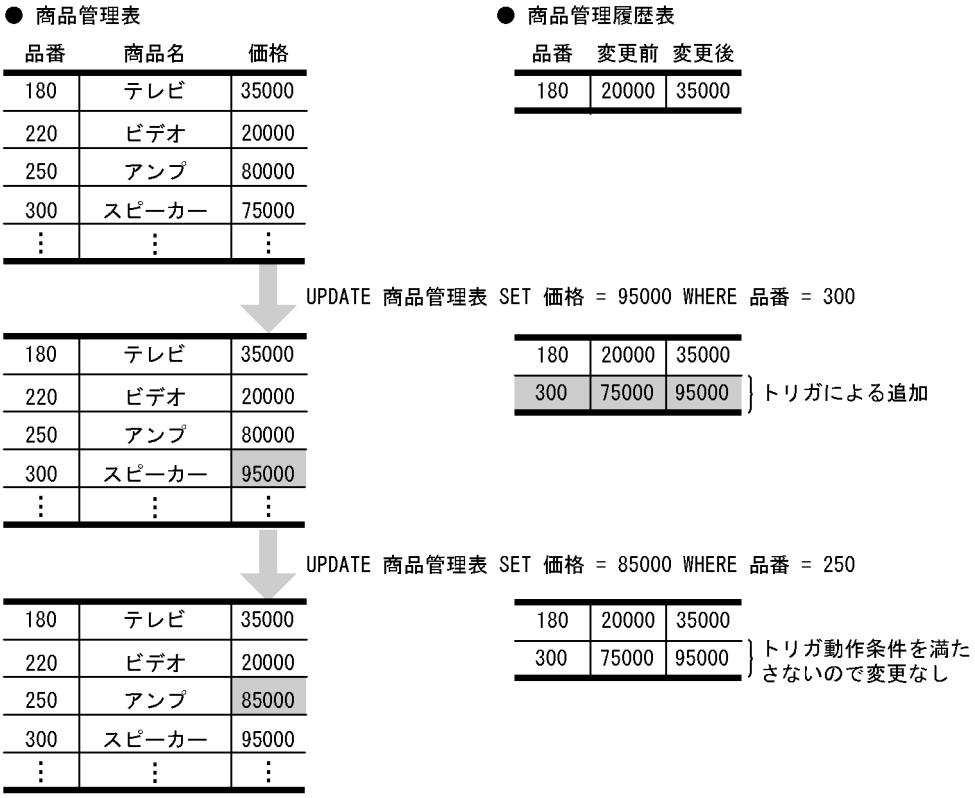
  トリガを削除します。

(3) トリガの定義例

(a) トリガを使用した例

商品管理表の価格が更新された場合，変更前に比べ変更後の価格上昇が 10,000 円を超えるとき，商品管理履歴表に更新前後の価格を挿入するトリガの定義例とトリガ動作を次に示します。

```
CREATE TRIGGER TR1          …トリガ名
       AFTER          …トリガ動作の実行時期
       UPDATE OF 価格          …トリガ動作の実行契機
       ON 商品管理表          …トリガを定義する表
       REFERENCING OLD ROW AS X1          …更新前の行に指定する名称
                   NEW ROW AS Y1          …更新後の行に指定する名称
       FOR EACH ROW          …行単位か文単位かの種別
       WHEN(Y1.価格 - X1.価格 > 10000)          …トリガ動作の探索条件
       INSERT INTO 商品管理履歴表 VALUES          …トリガSQL文
                       (X1.品番, X1.価格, Y1.価格)
```



(b) トリガ動作に SQL 制御文（代入文）を使用した例

代入文は，指定した値を指定した列に代入する SQL です。トリガでは表に対する操作の前に動作するトリガ動作で代入文を使用できます。トリガ動作に代入文を使用すると列と列の関係付けができます。

社員表の職種列が更新されると，更新された内容によって手当て列の値を変更するトリガの定義例とトリガ動作を次に示します。

- 社員表に挿入される行に対して，職種が A の場合は給料の 8%，B の場合は 10%，A と B 以外の場合は 0%を手当ての列に設定するトリガ

```
CREATE TRIGGER 手当て設定トリガ1
       BEFORE
       INSERT
       ON 社員表
       REFERENCING NEW ROW AS X1
       FOR EACH ROW
       SET X1.手当て=CASE X1.職種 WHEN 'A' THEN X1.給料*0.08
                                  WHEN 'B' THEN X1.給料*0.1
                                  ELSE 0 END
```

- 職種と給料が更新された行に対して，職種が A の場合は給料の 8%，B の場合は 10%，A と B 以外の場合は 0%を手当ての列に設定するトリガ

```
CREATE TRIGGER 手当て設定トリガ2
       BEFORE
       UPDATE OF 職種,給料
       ON 社員表
       REFERENCING NEW ROW AS X1
```

```
        FOR EACH ROW
        SET X1.手当て=CASE X1.職種 WHEN 'A' THEN X1.給料*0.08
                                   WHEN 'B' THEN X1.給料*0.1
                                   ELSE 0 END
```

● 社員表

社員番号	氏名	職種	給料	手当て
S99245	佐藤　一郎	A	190000	15200
L98003	鈴木　浩一	B	220000	22000

INSERT INTO 社員表 VALUES ('R97023','田中　次郎','A','180000,0)

INSERT実行後

S99245	佐藤　一郎	A	190000	15200
L98003	鈴木　浩一	B	220000	22000
R97023	田中　次郎	A	180000	14400

UPDATE 社員表 SET 職種='C' WHERE 社員番号='S99245'

UPDATE実行後

S99245	佐藤　一郎	C	190000	0
L98003	鈴木　浩一	B	220000	22000
R97023	田中　次郎	A	180000	14400

〔説明〕

INSERT 文が契機となり，手当て設定トリガ 1 が実行され，行が追加されました。INSERT 文では手当てに 0 を設定していますが，代入文の結果が格納されます。

次に，UPDATE 文が契機となり，手当て設定トリガ 2 が実行され，手当てが 0 に変更されました。

(c)　トリガ動作に SQL 制御文（複合文）を使用した例

複合文は，複数の SQL をまとめて一つの SQL として実行する SQL です。ある表に対する更新を契機に複数の表を更新する場合，トリガ動作に複合文を使用すると一つのトリガを定義するだけで複数の表を更新できます。

在庫マスタ表が更新された後，その更新内容を広島在庫及び仙台在庫に反映するトリガの定義例を次に示します。複合文を使用しない場合，トリガを二つ定義しなければなりません。

```
CREATE TRIGGER 地方在庫表更新トリガ
       AFTER
       UPDATE OF 在庫数
       ON 在庫マスタ
       REFERENCING NEW ROW 更新後
                   OLD ROW 更新前
       BEGIN
         UPDATE 広島在庫 SET 在庫数=更新後.在庫数
                WHERE 商品コード=更新前.商品コード;
         UPDATE 仙台在庫 SET 在庫数=更新後.在庫数
                WHERE 商品コード=更新前.商品コード;
       END
```

(d) トリガ動作に SQL 診断文（SIGNAL 文）を使用した例

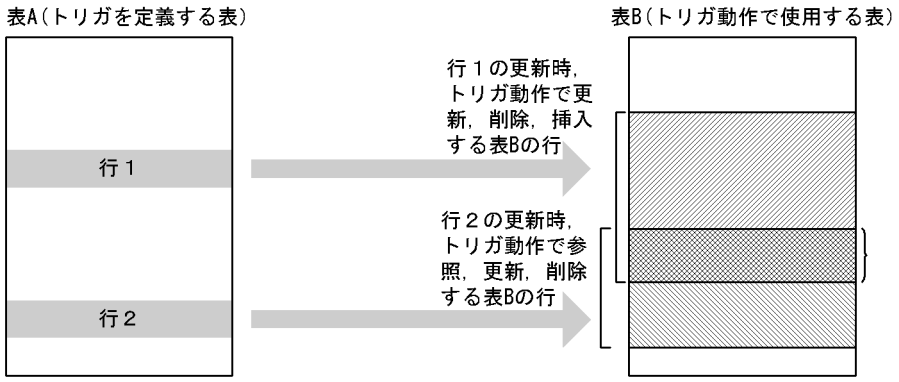
SIGNAL 文はエラーを発生させる SQL です。表への操作の前に SIGNAL 文を指定したトリガ動作を実行させれば，操作が不正な場合に SIGNAL 文を実行してその操作を抑止できます。

社員情報表を更新する前に，自分の情報以外を更新しようとした場合に SIGNAL 文でエラーを発生させ，更新を抑止するトリガの定義例を次に示します。

```
CREATE TRIGGER 更新抑止トリガ
       BEFORE
       UPDATE
       ON 社員情報
       REFERENCING OLD ROW AS X1
       WHEN(X1.社員名<>USER) SIGNAL SQLSTATE '99001'
```

11.5.3 トリガの使用上の注意

行単位トリガの定義によっては，トリガ契機となる SQL 実行時に HiRDB の内部処理に依存して異なる結果（更新の場合は異なる更新内容）になる場合があります。



〔説明〕

表 A の行 1 の更新時，トリガ動作で更新，削除，又は挿入する表 B の行と，行 2 の更新時，トリガ動作で参照，更新，又は削除する表 B の行に同一の行（重なり合った部分）があります。行 1 と行 2 の更新順序は HiRDB の内部処理に依存するため，この部分で異なる結果が得られる場合があります。

11.5.4 トリガの管理

(1) トリガの定義

トリガを定義すると，トリガを定義する表を使用する関数，手続き及びトリガの SQL オブジェクトは無効になるため，再作成する必要があります。トリガを定義する前にディクショナリ表の

SQL_ROUTINE_RESOURCES 表を参照すると，SQL オブジェクトが無効になる関数，手続き及びトリガを確認できます。無効になる SQL オブジェクトを確認して，再作成してください。

(a) トリガの定義によって SQL オブジェクトが無効になる関数，手続き及びトリガの確認

トリガの定義によって SQL オブジェクトが無効になる関数，手続き及びトリガを確認する SQL の例を次に示します。なお，無効になるのがトリガの場合は TRIGGER_NAME としてそのトリガ識別子が得られます。関数及び手続きの場合は TRIGGER_NAME が NULL になります。

```
SELECT DISTINCT B.ROUTINE_SCHEMA, B.ROUTINE_NAME, B.SPECIFIC_NAME, A.TRIGGER_NAME
FROM MASTER.SQL_ROUTINE_RESOURCES B LEFT JOIN MASTER.SQL_TRIGGERS A
ON B.ROUTINE_SCHEMA=A.TRIGGER_SCHEMA
AND B.SPECIFIC_NAME=A.SPECIFIC_NAME
WHERE B.BASE_TYPE='R'
AND B.BASE_OWNER='トリガを定義する表の所有者の認可識別子'
AND B.BASE_NAME='トリガを定義する表の識別子'
AND (B.列名※ ='Y'
  OR (  B.INSERT_OPERATION IS NULL
    AND B.UPDATE_OPERATION IS NULL
    AND B.DELETE_OPERATION IS NULL))
```

注※

INSERT を契機とするトリガを定義すると無効になる SQL オブジェクトを検索する場合には，列名に INSERT_OPERATION を，UPDATE を契機とするトリガの場合は UPDATE_OPERATION を，DELETE を契機とするトリガの場合は DELETE_OPERATION を指定します。

(2) トリガの SQL オブジェクトの再作成

トリガが使用している表，インデクスなどのリソースが定義，定義変更，又は削除されるとトリガの SQL オブジェクトは無効になります。また，トリガが使用している表にインデクスを定義したり，又はトリガが使用している表のインデクスを削除したりすると，トリガの SQL オブジェクトのインデクス情報が無効になります。

トリガの SQL オブジェクトが無効又は SQL オブジェクトのインデクス情報が無効の場合，トリガ契機となる SQL は実行できません。トリガの SQL オブジェクトの無効又は SQL オブジェクトのインデクス情報の無効を解除するには，定義系 SQL の ALTER TRIGGER 又は ALTER ROUTINE でトリガの SQL オブジェクトを再作成する必要があります。

(a) トリガが使用しているリソースを確認する方法

ディクショナリ表の SQL_ROUTINE_RESOURCES，SQL_TRIGGER_USAGE，及び SQL_ROUTINE_PARAMS を参照すると，トリガが使用しているリソースの情報を得ることができます。

- トリガ動作条件で使用するリソースを確認する SQL の例

```
SELECT B.* FROM MASTER.SQL_TRIGGERS A, MASTER.SQL_TRIGGER_USAGE B
WHERE A.TRIGGER_SCHEMA='スキーマ名'
  AND A.TRIGGER_NAME='トリガ識別子'
```

```
  AND A.TRIGGER_SCHEMA=B.TRIGGER_SCHEMA
  AND A.TRIGGER_NAME=B.TRIGGER_NAME
```

- 新旧値別名を指定したトリガ SQL 中で使用する列リソースを確認する SQL の例

```
SELECT B.* FROM MASTER.SQL_TRIGGERS A, MASTER.SQL_ROUTINE_PARAMS B
WHERE A.TRIGGER_SCHEMA='スキーマ名'
  AND A.TRIGGER_NAME='トリガ識別子'
  AND A.TRIGGER_SCHEMA=B.ROUTINE_SCHEMA
  AND A.SPECIFIC_NAME=B.SPECIFIC_NAME
```

- 上記以外でトリガが使用するリソースを確認する SQL の例

```
SELECT B. * FROM MASTER.SQL_TRIGGERS A, MASTER.SQL_ROUTINE_RESOURCES B
WHERE A.TRIGGER_SCHEMA='スキーマ名'
  AND A.TRIGGER_NAME='トリガ識別子'
  AND A.TRIGGER_SCHEMA=B.ROUTINE_SCHEMA
  AND A.SPECIFIC_NAME=B.SPECIFIC_NAME
```

(b) 表の列削除の前に，削除されるトリガを確認する方法

トリガ契機で指定されている列がすべて削除された場合，そのトリガは削除されます。表の列が削除される前に，削除されるトリガを確認する SQL の例を次に示します。

```
SELECT A.TRIGGER_SCHEMA, A.TRIGGER_NAME
  FROM MASTER.SQL_TRIGGERS A
WHERE A.N_UPDATE_COLUMNS>0
  AND A.TABLE_SCHEMA='列削除する表の所有者の認可識別子'
  AND A.TABLE_NAME='列削除する表の表識別子'
  AND NOT EXISTS(SELECT * FROM MASTER.SQL_TRIGGER_COLUMNS B
                 WHERE B.TRIGGER_SCHEMA=A.TRIGGER_SCHEMA
                   AND B.TRIGGER_NAME=A.TRIGGER_NAME
                   AND B.TABLE_SCHEMA=A.TABLE_SCHEMA
                   AND B.TABLE_NAME=A.TABLE_NAME
                   AND B.COLUMN_NAME NOT IN('削除する列名',…))
```

(c) 表，インデクスなどの定義，定義変更，削除の前に SQL オブジェクトが無効又は SQL オブジェクトのインデクス情報が無効になる関数，手続き及びトリガを確認する方法

表，インデクスなどを定義，定義変更，又は削除する前に，SQL オブジェクトが無効又は SQL オブジェクトのインデクス情報が無効になる関数，手続き及びトリガを確認する SQL の例を次に示します。なお，無効になるのがトリガの場合は TRIGGER_NAME としてそのトリガ識別子が得られます。関数及び手続きの場合は TRIGGER_NAME が NULL になります。

- 表（ビュー表も含む）の定義変更及び削除，又はインデクス定義（インデクスを定義する表のスキーマ名と表識別子を指定）の場合

```
SELECT DISTINCT B.ROUTINE_SCHEMA, B.ROUTINE_NAME, B.SPECIFIC_NAME, A.TRIGGER_NAME
  FROM MASTER.SQL_ROUTINE_RESOURCES B LEFT JOIN MASTER.SQL_TRIGGERS A
  ON B.ROUTINE_SCHEMA=A.TRIGGER_SCHEMA
    AND B.SPECIFIC_NAME=A.SPECIFIC_NAME
```

```
WHERE B.BASE_TYPE IN('R','V')
  AND B.BASE_OWNER='表（ビュー）の所有者の認可識別子'
  AND B.BASE_NAME='表（ビュー）の表識別子'
```

- インデクス削除の場合

```
SELECT DISTINCT B.ROUTINE_SCHEMA, B.ROUTINE_NAME, B.SPECIFIC_NAME, A.TRIGGER_NAME
  FROM MASTER.SQL_ROUTINE_RESOURCES B LEFT JOIN MASTER.SQL_TRIGGERS A
  ON B.ROUTINE_SCHEMA=A.TRIGGER_SCHEMA
    AND B.SPECIFIC_NAME=A.SPECIFIC_NAME
WHERE B.BASE_TYPE ='I'
  AND B.BASE_OWNER='インデクスの所有者の認可識別子'
  AND B.BASE_NAME='インデクスの識別子'
```

- 関数，又は手続き削除の場合

```
SELECT DISTINCT B.ROUTINE_SCHEMA, B.ROUTINE_NAME, B.SPECIFIC_NAME, A.TRIGGER_NAME
  FROM MASTER.SQL_ROUTINE_RESOURCES B LEFT JOIN MASTER.SQL_TRIGGERS A
  ON B.ROUTINE_SCHEMA=A.TRIGGER_SCHEMA
    AND B.SPECIFIC_NAME=A.SPECIFIC_NAME
WHERE B.BASE_TYPE ='P'
  AND B.BASE_OWNER='関数又は手続きの所有者の認可識別子'
  AND B.BASE_NAME='ルーチン識別子'
```

- トリガ削除の場合

```
SELECT DISTINCT B.ROUTINE_SCHEMA, B.ROUTINE_NAME, B.SPECIFIC_NAME, A.TRIGGER_NAME
  FROM MASTER.SQL_ROUTINE_RESOURCES B LEFT JOIN MASTER.SQL_TRIGGERS A
  ON B.ROUTINE_SCHEMA=A.TRIGGER_SCHEMA
    AND B.SPECIFIC_NAME=A.SPECIFIC_NAME
WHERE B.BASE_TYPE ='T'
  AND B.BASE_OWNER='トリガの所有者の認可識別子'
  AND B.BASE_NAME='トリガ識別子'
```

- スキーマ削除の場合

```
SELECT DISTINCT B.ROUTINE_SCHEMA, B.ROUTINE_NAME, B.SPECIFIC_NAME, A.TRIGGER_NAME
  FROM MASTER.SQL_ROUTINE_RESOURCES B LEFT JOIN MASTER.SQL_TRIGGERS A
  ON B.ROUTINE_SCHEMA=A.TRIGGER_SCHEMA
    AND B.SPECIFIC_NAME=A.SPECIFIC_NAME
WHERE B.BASE_OWNER='スキーマ名'
```

- ユーザ定義型削除の場合

```
SELECT B.ROUTINE_SCHEMA, B.ROUTINE_NAME, B.SPECIFIC_NAME, A.TRIGGER_NAME
  FROM MASTER.SQL_ROUTINE_RESOURCES B LEFT JOIN MASTER.SQL_TRIGGERS A
  ON B.ROUTINE_SCHEMA=A.TRIGGER_SCHEMA
    AND B.SPECIFIC_NAME=A.SPECIFIC_NAME
WHERE B.BASE_NAME='削除するデータ型識別子'
  AND B.BASE_TYPE='D'
UNION
SELECT B.ROUTINE_SCHEMA, B.ROUTINE_NAME, B.SPECIFIC_NAME, A.TRIGGER_NAME
  FROM MASTER.SQL_ROUTINES C INNER JOIN MASTER.SQL_ROUTINE_RESOURCES B
  ON C.SPECIFIC_NAME=B.BASE_NAME
 LEFT JOIN MASTER.SQL_TRIGGERS A
  ON B.ROUTINE_SCHEMA=A.TRIGGER_SCHEMA
    AND B.SPECIFIC_NAME=A.SPECIFIC_NAME
WHERE C.ROUTINE_ADT_OWNER='削除するユーザ定義型の所有者の認可識別子'
```

```
AND C.ROUTINE_ADT_NAME='削除するデータ型識別子'
AND B.BASE_TYPE='P'
```

(d) 表，インデクスなどの定義，定義変更，削除の後に SQL オブジェクトが無効又は SQL オブジェクトのインデクス情報が無効になった関数，手続き及びトリガを確認する方法

表，インデクスなどを定義，定義変更，又は削除した後に，SQL オブジェクトが無効又は SQL オブジェクトのインデクス情報が無効になったトリガを確認するには，それぞれディクショナリ表の SQL_TRIGGER 表の TRIGGER_VALID 列，INDEX_VALID 列を参照します。TRIGGER_VALID 列の内容が'N'の場合，そのトリガの SQL オブジェクトは無効になっています。また，INDEX_VALID 列の内容が'N'の場合，そのトリガの SQL オブジェクトのインデクス情報は無効になっています。

表，インデクスなどを定義，定義変更，又は削除した後に，SQL オブジェクトが無効又は SQL オブジェクトのインデクス情報が無効になった関数，手続き及びトリガを確認する SQL の例を次に示します。なお，無効になるのがトリガの場合は TRIGGER_NAME としてそのトリガ識別子が得られます。関数及び手続きの場合は TRIGGER_NAME が NULL になります。

```
SELECT 'TRIGGER', TRIGGER_SCHEMA AS "SCHEMA", TRIGGER_NAME AS "NAME",
 TRIGGER_VALID AS "OBJECT_VALID", INDEX_VALID
  FROM MASTER.SQL_TRIGGERS
WHERE TRIGGER_VALID='N' OR INDEX_VALID='N'
UNION
SELECT 'ROUTINE', ROUTINE_SCHEMA, ROUTINE_NAME, ROUTINE_VALID, INDEX_VALID
  FROM MASTER.SQL_ROUTINES
WHERE ROUTINE_VALID='N' OR INDEX_VALID='N'
```

11.5.5 障害時の回復方法

トリガのソースはデータディクショナリ用 RD エリアに，トリガの 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。データディクショナリ用 RD エリアのログ取得モードは ALL，データディクショナリ LOB 用 RD エリアのログ取得モードは PARTIAL です。そのため，障害が発生した場合，ソースはバックアップとログを入力して回復すれば最新時点に回復できますが，SQL オブジェクトはバックアップ時点にしか回復できません。そこで次の運用をする必要があります。

- 常に最新のバックアップを取得しておく

  常にデータディクショナリ LOB 用 RD エリアのバックアップを取得し，障害発生時に最新のバックアップから回復します。pdcopy コマンドで-M x，又は-M r を指定してください。

  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。

- トリガの SQL オブジェクトを再作成する

  データディクショナリ LOB 用 RD エリアの最新のバックアップがない場合，そのデータディクショナリ LOB 用 RD エリアを pdmod コマンドで再初期化します。その後，ALTER ROUTINE に ALL を指定して実行し，全トリガの SQL オブジェクトを再作成します。

11.6　ビュー表の作成

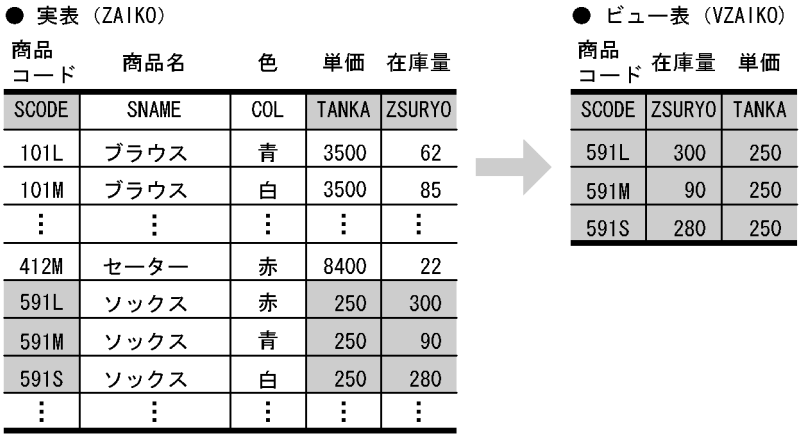
11.6.1　ビュー表作成の概要

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（**ビュー表**）を作成できます。

(1)　実表とビュー表の関係

実表とビュー表の関係を次の図に示します。

図 11-18　実表とビュー表の関係



〔説明〕

実表（ZAIKO）から，商品名（SNAME）がソックスの行で，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）を作成した例です。

商品を扱うある支店では，「商品名」が「ソックス」で「商品コード」「在庫量」「単価」のデータを参照だけすればよい場合，実表（ZAIKO）にはアクセス不可とし，ビュー表（VZAIKO）に対して参照だけを許す権限（SELECT 権限）を設定します。これによって，必要な情報だけを参照でき，データは保護されます。

(2)　ビュー表を作成したときの効果

ビュー表を作成したときの効果を次に示します。

機密性の向上

自分の表の機密保護を図りたい場合にビュー表を作成します。ビュー表を作成することで，表の特定の列と行だけを表示できます。このため，アクセス権限をビュー表に設定することで，行又は列レベルの機密保護が図れます。

操作性の向上

- 複雑な問い合わせ指定をして表を検索しなくてもよいように，複雑な問い合わせ指定をした場合に検索できるデータであらかじめビュー表を作成します。これによって，表を参照する操作が手軽になります。
- ビュー表を通して実表を参照したり更新したりできます。これによって，実表の定義が変更されても SQL 文を変更する必要はなく，場合によってはビュー表の定義を変更する必要がなくなります。

(3) ビュー表の作成方法

定義系 SQL の CREATE VIEW を実行してビュー表を作成します。CREATE VIEW で次に示すビュー表を定義できます。

- 実表の特定の行及び列を選択したビュー表
- 実表の列から集合関数，日付演算，時刻演算，連結演算，スカラ関数，又は四則演算で求めた値を列とするビュー表
- 最大 128 個の実表を基にした一つのビュー表
- グループ分け検索の結果を基にしたビュー表
- ほかのユーザが所有する実表を基にしたビュー表（ほかのユーザが所有する実表の SELECT 権限を与えられている場合に限る）

規則

1. 一つのビュー表には 30,000 個まで列を定義できます。
2. ビュー表に対して列の追加，及びインデクスを定義できません。
3. 自分の所有する実表から定義したビュー表の所有者はビュー表に対するすべてのアクセス権限（行の検索，追加，削除，更新）を持ちます。
4. ほかのユーザが所有する実表から定義したビュー表の所有者は，ビュー表に対するアクセス権限として，実表に対して付与されているアクセス権限だけを持ちます。ただし，ビュー表の定義に次に示すどれかの指定をした場合は，機密保護機能を使用するかどうかに関係なく行の検索しかできません。

- ビュー表の列に実表の同じ列を複数指定した場合
- ビュー表の列に定数，USER 値関数，CURRENT_DATE 値関数，CURRENT_TIME 値関数，四則演算，日付演算，時刻演算，連結演算，又はスカラ関数の結果を指定した場合
- 複数の実表を指定した場合
- DISTINCT，集合関数（COUNT(*)，AVG，MAX，MIN，SUM），グループ分け（GROUP BY 句），又はグループ条件（HAVING 句）を指定した場合

  機密保護機能を使用しない場合，これら以外のビュー表はほかのユーザが自由に更新できます。ただし，読み込み専用のビュー表（READ ONLY 指定）の場合，機密保護機能を使用しなくてもほかのユーザがビュー表を更新できません。

(4) ビュー表の削除方法

定義系 SQL の DROP VIEW を実行してビュー表を削除します。ビュー表を削除すると，削除したビュー表に関するすべてのアクセス権限が取り消されます。

11.7 FIX 属性の指定

FIX 属性とは，行長が固定の表に付ける属性のことです。

11.7.1 FIX 属性を指定したときの効果

表に FIX 属性を指定したときの効果を次に示します。

性能の向上

- 任意の列を取り出す性能が，列の定義順序に関係なく一定になります。また，FIX 属性を指定しない場合と比べて，列の取り出し時間を短縮できます。
- UAP で行単位インタフェースが使用できるため，列数が多くてもアクセス性能を向上できます。

操作性の向上

FIX 属性の表の列を更新する場合に入力データにナル値があったときは，エラーとしてチェックアウトできます。

ディスク所要量の削減

FIX 属性を指定していない表と比べて，物理的な行長が 1 列で 2 バイト短くなります。このため，列数の多い表の場合はディスク所要量を削減できます。

11.7.2 適用基準

ナル値を持つ列がない，かつ可変長の列がない場合は，表定義時に FIX 属性を指定します。

また，上記の条件を満たさない場合でも，次に示すことを検討してください。

- 将来，表に列の追加が見込まれる場合は，表定義時に予備列を定義してください。予備列を定義しておくと，表にデータが格納されている状態でも，列の追加ができます。
- 0（数値データ）又は空白（文字データ）を使用して，ナル値を不要にしてください。ただし，ナル値は探索条件や集合関数での扱いが，ほかの値と異なるため注意してください。
- 最大値の小さい可変長データや実長の取り得る範囲が狭い可変長データを固定長にしてください。ただし，探索条件での扱いが異なるため注意してください。

11.7.3 指定方法

表に FIX 属性を指定するには，定義系 SQL の CREATE TABLE で FIX を指定（CREATE FIX TABLE と指定）します。

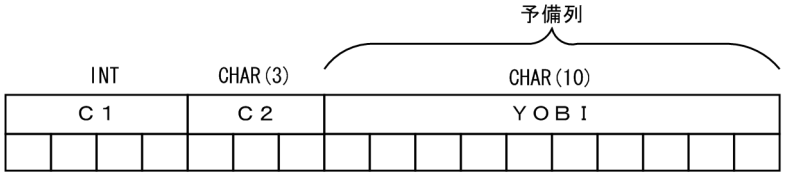
11.7.4　注意

予備列は，将来の列追加のために確保しておく領域（列）であり，ユーザは予備列に対して任意の値で挿入，更新を行うことはできません。また，予備列には HiRDB が予備列の定義長分のナル文字（0x00）を格納するため，表格納用 RD エリアの容量を見積もる際には，予備列も含めて表の格納ページ数を見積もる必要があります。予備列の使い方を次の図に示します。

図 11-19　予備列の使い方

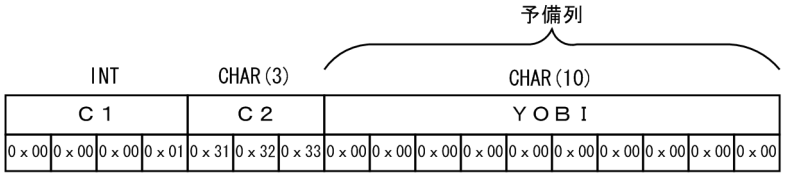
■予備列を含む表の定義

```
CREATE FIX TABLE TABLE01 (C1 INT,C2 CHAR(3),YOBI CHAR(10) FOR RESERVED)
```



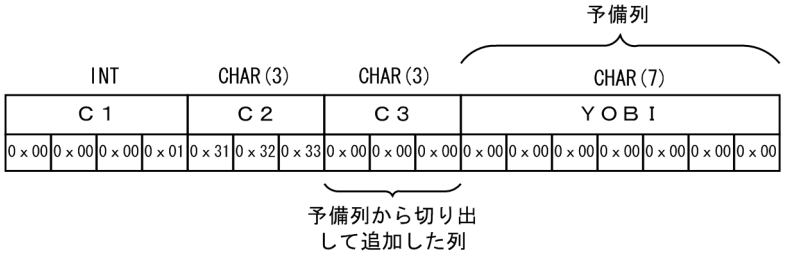
■予備列を含む表へのデータ挿入

```
INSERT INTO TABLE01 VALUES(1,' 123' )
```



■予備列を含む表への列追加

```
ALTER TABLE TABLE01 ADD C3 CHAR(3) INTO YOBI
```



11.8 主キー（プライマリキー）の指定

11.8.1 主キーの効果と指定方法

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。主キーを定義すると，指定した列に対してインデクスが作成されます。

(1) 主キーを定義したときの効果

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。

(2) 適用基準

行を一意に識別できる列に主キーを定義します。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。

(3) 指定方法

表に主キーを定義するには，定義系 SQL の CREATE TABLE の PRIMARY KEY オプションを指定します。

なお，主キーは定義系 SQL の ALTER TABLE でも追加，及び削除ができます。追加する場合は，ADD PRIMARY KEY オプションを指定します。削除する場合は，DROP PRIMARY KEY オプションを指定します。

11.9　クラスタキーの指定

11.9.1　クラスタキーの効果と指定方法

クラスタキーとは，特定の列の値の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。表の一つの列又は複数の列にクラスタキーを指定しておくと，クラスタキーの昇順又は降順に行を格納できます。

クラスタキーを定義すると，指定した列に対してインデクスが作成されます。

(1)　クラスタキーを定義したときの効果

表にクラスタキーを指定したときの効果を次に示します。

性能の向上

範囲を指定して行の検索，更新，削除などをするとき又はクラスタキー順に検索，更新などをするときに，入出力時間を削減できます。

操作性の向上

- UNIQUE 指定のクラスタキーを定義すると，そのクラスタキーには一意性制約が適用されます。このため，行を挿入するときに，クラスタキーを構成している列の値がすべての行で重複しないようにできます。なお，フレキシブルハッシュ分割表には，UNIQUE 指定のクラスタキーを定義できません。
- PRIMARY 指定のクラスタキーを定義すると，そのクラスタキーには一意性制約及び非ナル値制約が適用されます。このため，行を挿入するときに，クラスタキーを構成している列の値がすべての行で重複しないようにできます。また，クラスタキーを構成している列の値にナル値を格納しないようにできます。なお，フレキシブルハッシュ分割表には，PRIMARY 指定のクラスタキーを定義できません。
- 表を作成するときに，入力データがクラスタキーの昇順又は降順に並んでいるかどうかをデータベース作成ユティリティ（pdload）で確認できます。
- 表を再編成するときに，アンロードした行のクラスタキーと，リロードするクラスタキーが一致しているかどうかをデータベース再編成ユティリティ（pdrorg）で確認できます。

(2)　適用基準

クラスタキーを指定するとよい場合を次に示します。

- 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
- キーを変更しない表の場合
- 行長が固定の表の場合

(3) 指定方法

表にクラスタキーを定義するには，定義系 SQL の CREATE TABLE で CLUSTER KEY オプションを指定します。

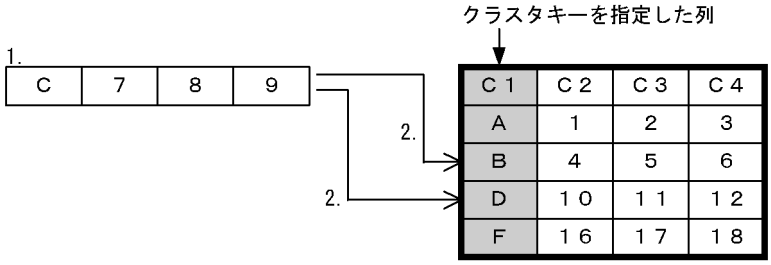
(4) 設計上の考慮点

データを追加した後の検索効果を上げるため，表を格納しているページ内に未使用領域を設定しておきます。設定方法については，「ページ」を参照してください。

(5) 注意

- クラスタキーを構成する列の値は更新できません。
- クラスタキーを構成する列にナル値は挿入できません。
- クラスタキーを指定した表にデータを追加するとき，追加しようとするキー値に近接するキー値を持ったページを探すためのオーバヘッドが発生します。この概要を次の図に示します。

図 11-20　クラスタキーを指定した表にデータを追加するときのオーバヘッドの概要



〔説明〕

1. 列Ｃを持つデータを追加します。
2. 列Ｃに近接するキー値を探すオーバヘッドが発生します。

11.10　サプレスオプションの指定

11.10.1　サプレスオプションの効果と指定方法

サプレスオプションとは，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のことです。

サプレスオプションを指定した場合には，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。

(1)　サプレスオプションを指定したときの効果

サプレスオプションを指定したときの効果を次に示します。

性能の向上

- 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，ディスク所要量を削減できます。
- ディスク所要量の削減で，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。

(2)　適用基準

次に示す場合にサプレスオプションを指定することをお勧めします。

- 表中に DECIMAL データが多く，有効けた数が多い場合
- 全件検索などの検索業務が多く，更新業務が少ない場合

(3)　指定方法

サプレスオプションを指定するには，定義系 SQL の CREATE TABLE で SUPPRESS オプションを指定します。

(4)　注意

- DECIMAL データの有効けた数が定義長と等しい場合又は定義長が 1 の場合には，「定義長＋ 1」の長さで格納されます。このため，サプレスオプション指定時よりも格納するデータ長が長くなります。
- FIX 属性を指定している表では，サプレスオプションを指定できません。

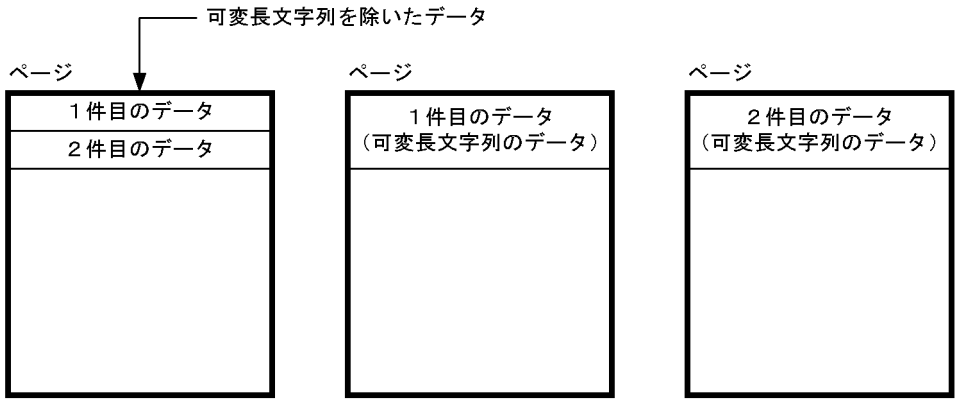
11.11　ノースプリットオプションの指定

11.11.1　ノースプリットオプションの適用基準と指定方法

表に次に示すデータ型が定義されていて，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。このときのデータ格納方式を次の図に示します。

- VARCHAR
- MVARCHAR
- NVARCHAR

図 11-21　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



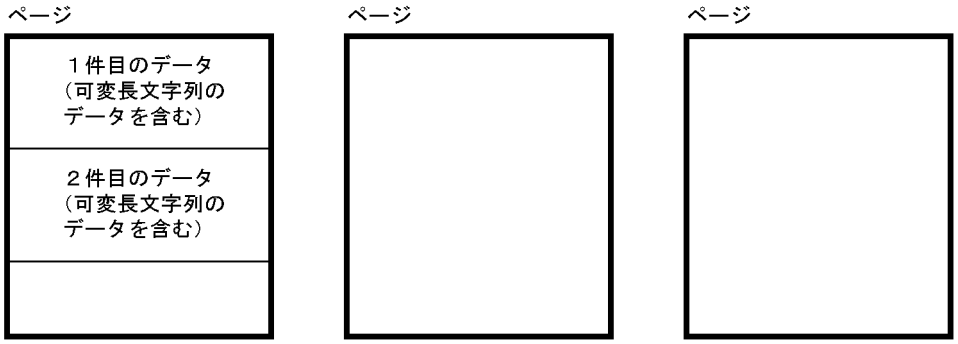
〔説明〕

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。

(1)　適用基準

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 11-22　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。

(2)　指定方法

ノースプリットオプションを指定するには，定義系 SQL の ALTER TABLE，CREATE TABLE 又は CREATE TYPE で NO SPLIT オプションを指定します。

(3)　注意

- 1 行のデータ長の合計がページ長を超える場合は，ノースプリットオプションを指定してもスプリット（1 行のデータを複数のページに格納）します。
- 可変長文字列の実際のデータ長が 255 バイト以下の場合にノースプリットオプションを指定すると，指定しないときに比べて列データ長が 1 バイト長くなります。
- 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分岐しないため，ノースプリットオプションを指定しない場合と比べて 1 ページに格納される行数が少なくなります。このため，インデクススキャンでノースプリットオプションが適用された可変長文字列型の列データを取り出さない検索をすると，ノースプリットオプションを指定しないときに比べてアクセスするページ数が多くなり，検索性能が低下することがあります。ただし，キースキャン及びテーブルスキャンの場合は影響はありません。

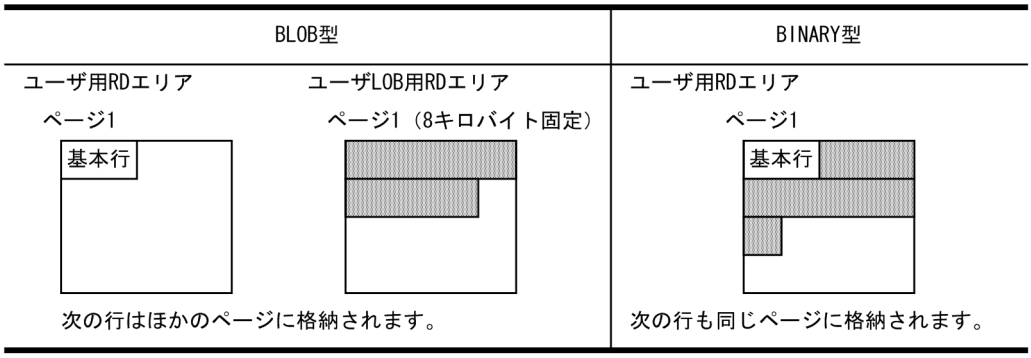
11.12 バイナリデータ列の指定

文書，画像，音声などの可変長バイナリデータを格納する列に指定するデータ型には，次の二つがあります。

- BLOB 型（BLOB 型を指定した列を LOB 列といいます）
- BINARY 型

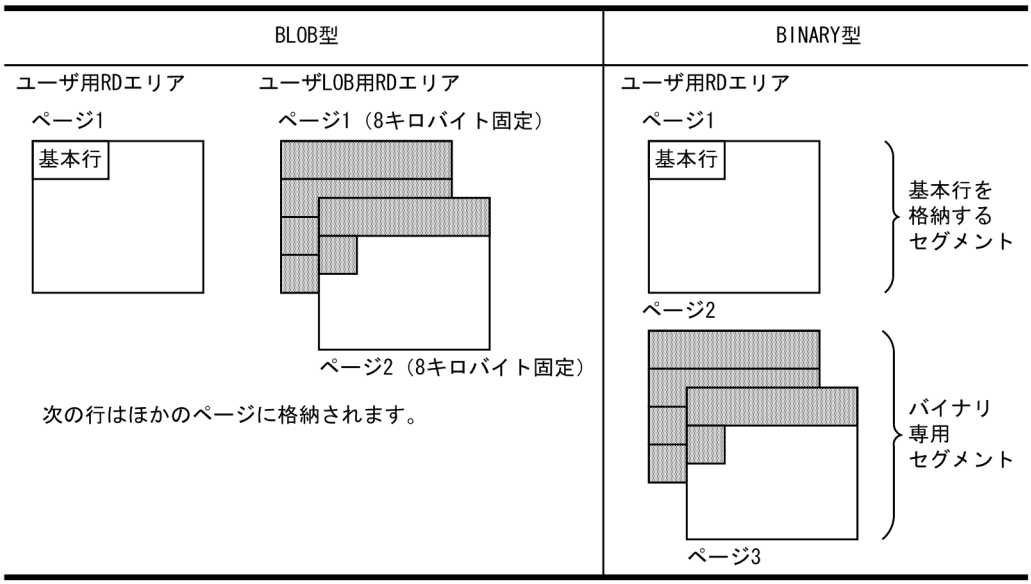
それぞれのデータの格納方法の比較を次に示します。

データ長（基本行＋バイナリデータ長）が 1 ページ以下の場合



（凡例）：バイナリデータ

データ長（基本行＋バイナリデータ長）が 1 ページより大きい場合



（凡例）：バイナリデータ

［説明］

BLOB 型を指定した列はほかの属性の列とは別に，ページ長が 8 キロバイト固定のユーザ LOB 用 RD エリアに格納します。

BINARY 型を指定した列は表を構成するすべての列データが 1 ページに格納できる場合，実際のデータ長に関係なく 1 行を 1 ページ内に格納します。ただし，1 行が 1 ページに格納できない場合は，1 行が複数ページになり，バイナリ専用セグメントに格納します。

11.12.1 BLOB 型

(1) 設計上の考慮点

- BLOB 型を指定する場合はユーザ LOB 用 RD エリアを作成してください。
- BLOB 型のデータは 8 キロバイトバウンダリされるので，データの格納効率は BINARY 型が優位です。ただし，8 キロバイトのバウンダリが無視できるような長大データサイズになると余り差はありません。

(2) 指定方法

定義系 SQL の CREATE TABLE の列のデータ型で，BLOB 型を指定します。

(3) 注意

BLOB 型は次の項目には使用できません。

- FIX 属性の表
- インデクスの定義
- 分割キー

11.12.2 BINARY 型

(1) 設計上の考慮点

BINARY 型は，BLOB 型に比べて 1 ページに格納する行数が減るため，ある列を探索条件に指定して，BINARY 型を指定したデータを取り出さない検索を実行した場合，BLOB 型に比べて入出力回数が増えます。そのため，検索性能が低下することがあります。ただし，BINARY 型以外の列にインデクスを定義してインデクススキャンにすると BINARY 型と BLOB 型の性能差はなくなります。

(2) 指定方法

定義系 SQL の CREATE TABLE の列のデータ型で，BINARY 型を指定します。

(3) 注意

- BINARY 型は次の項目には使用できません。
  - FIX 属性の表
  - インデクスの定義
  - 分割キー

- 外への参照列

- バイナリデータ長が 1 ページより大きい場合，基本行を格納するセグメントと異なるバイナリ専用セグメントに格納します。データ長が 1 ページより大きいバイナリデータ格納時，基本行を格納するセグメントに未使用ページがある場合でも，バイナリ専用セグメントに未使用ページがなければ RD エリアが容量不足となる場合があります。RD エリアが容量不足になった場合，データベース解析ユティリティ（pddbst）でバイナリ専用セグメント，及び基本行を格納するセグメントのそれぞれの使用状況を参照することで，原因がバイナリデータによるものか，又は基本行のデータによるものかが確認できます。
- BINARY 型の列を定義する場合は，列の定義長（列の最大長）の見積もりを詳細に行ってください。列の定義長を不要に大きくすると，列の定義長に比例してメモリを使用するデータロードを実行できなくなることがあります（BINARY 列を定義している表のデータロードを実行する場合，BINARY 列の定義長分のメモリを確保します）。

11.12.3 BLOB 型と BINARY 型の使い分け

バイナリデータの運用方法によってお勧めするデータ型を次の表に示します。

表 11-3 バイナリデータの運用による推奨データ型

バイナリデータの運用方法		推奨するデータ型の平均データサイズ		説明
		32,000 バイト以下の場合	32,000 バイトより大きい場合	
射影列にバイナリデータ列を指定する頻度	高い	BINARY	BLOB	• 32,000 バイト以下の場合，BINARY 型の方がブロック転送もでき，行全体のデータが近くに集まって配置されるので，性能的に優位です。 • 32,000 バイトより大きい場合，BLOB 型の方が長大データに特化した内部メモリ削減などの処理をするので，性能的に優位です。
	低い	BLOB※	BLOB	BLOB 型の方が BINARY 型より基本行のデータサイズが小さいので，データ件数が多くなれば BLOB 型が性能的に優位です。ただし，BINARY 型以外の列にインデクスを定義してインデクススキャンにすることで BINARY 型と BLOB 型の性能差はなくなります。また，インデクススキャンにすることを推奨しますが，セグメントサイズを大きくすることでも性能差を小さくできます。
SQL 記述の柔軟性		BINARY	同等	32,000 バイト以下の場合，BINARY 型の SQL 記述はインデクス定義を除いて，ほぼ VARCHAR と同等の記述ができます。そのため，BLOB 型より SQL 記述範囲が広がります。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。

バイナリデータの運用方法	推奨するデータ型の平均データサイズ		説明
	32,000 バイト以下の場合	32,000 バイトより大きい場合	
データ格納効率	BINARY		格納効率は BINARY 型が優位です。ただし，8 キロバイトのバウンダリが無視できるような長大データサイズになると余り差はありません。
追加更新を頻繁にする場合	BLOB		連結演算するデータサイズが大きいほど BLOB 型が性能的に優位です。
部分抽出を頻繁にする場合	同等	BLOB	BINARY 型の格納済みデータサイズが大きいデータに対して部分抽出をすると非常に性能が悪くなります。さらに，部分抽出回数を多くすると，性能は劣化します。大きなデータに対して部分抽出を頻繁にする必要がある場合には，BLOB 型をお勧めします。
運用性を重視する場合	BINARY		BLOB 型はユーザ LOB 用 RD エリアをバックアップするなどの特別な運用が必要になります。
上記使用方法で判断できない場合，又は将来的に方針変更の可能性があり，判断できない場合	BINARY※	BLOB	32,000 バイトより大きいデータを扱う場合は BLOB 型をお勧めします。比較的データサイズが小さい場合は BINARY 型をお勧めします。

注※

データサイズがページサイズに近い大きさで分岐しない場合，BINARY 型で大量のテーブルスキャンをすると，BLOB 型に比べて非常に性能が悪くなります。これを防ぐため，テーブルスキャンからインデクススキャンへ変更してください。また，インデクススキャンにすることを推奨しますが，セグメントサイズを大きくすることでも性能差を小さくできます。

11.13　文字集合の指定

文字集合とは，文字データに対する属性です。文字集合は，次の三つの属性を持ちます。

- 使用形式

  文字を表現する規約のことです。例えば，ある文字集合では A を 1 バイトのコード X'41'と表現しますが，別の文字集合では，これを X'C1'と表現します。このように，文字を表現する規約のことを使用形式といいます。

- 文字レパートリ

  表現できる文字の集合のことです。例えば，ある文字集合ではバックスラッシュを表現できますが，別の文字集合ではこれを表現できません。このように，表現できる文字の集まりを文字レパートリといいます。

- 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のことです。例えば，ある文字集合では'1'＞'A'ですが，別の文字集合では'A'＞'1'という照合順になります。どの文字集合にも，既定の照合順があります。

なお，文字集合指定を省略して仮定された文字集合を，既定文字集合といいます。

11.13.1　文字集合を定義したときの効果

文字集合を定義すると，表の列ごとに異なる文字集合の文字列データを格納できます。これによって，文字集合に EBCDIK を指定した場合は，VOS3 システムから HiRDB に移行したときに，データベースに格納していた文字データを VOS3 システムの文字列データの照合順で検索，代入，比較ができます。また，文字集合に UTF16 を指定した場合は，UTF-16 で文字データの検索，代入，比較ができます。

11.13.2　HiRDB で使用できる文字集合

HiRDB で使用できる文字集合を次に示します。

- EBCDIK

  EBCDIK を使用する場合には，HiRDB セットアップ時，文字コード種別に sjis を指定します。

- UTF16

  UTF16 を使用する場合には，HiRDB セットアップ時，文字コード種別に utf-8，又は utf-8_ivs を指定します。

11.13.3　指定方法

文字集合は文字データ型に指定します。文字集合を指定する場合の形式や規則については，マニュアル「HiRDB SQL リファレンス」を参照してください。

11.13.4　注意事項

- 入力データの文字集合と列の文字集合が異なる場合はデータロードできません。
- 文字集合に UTF16 を指定した場合，データベースに格納されるデータはビッグエンディアン形式になります。文字集合に UTF16 を指定した列に対して，埋込み変数，又は？パラメタを使用して操作する場合は，埋込み変数，？パラメタに指定する値をビッグエンディアン形式にしてください。これらをリトルエンディアン形式にした場合，文字コード変換をする必要があるため，性能が悪くなります。

11.14　WITHOUT ROLLBACK オプションの指定

11.14.1　WITHOUT ROLLBACK オプションの効果と適用基準

WITHOUT ROLLBACK オプションとは，表に対しての更新処理（追加，削除を含む）の完了を契機に，その更新行に対する排他が解除され，それ以降はロールバックされなくなるオプションのことです。

(1)　WITHOUT ROLLBACK オプションを指定したときの効果

表に WITHOUT ROLLBACK オプションを指定したときの効果を次に示します。

性能の向上

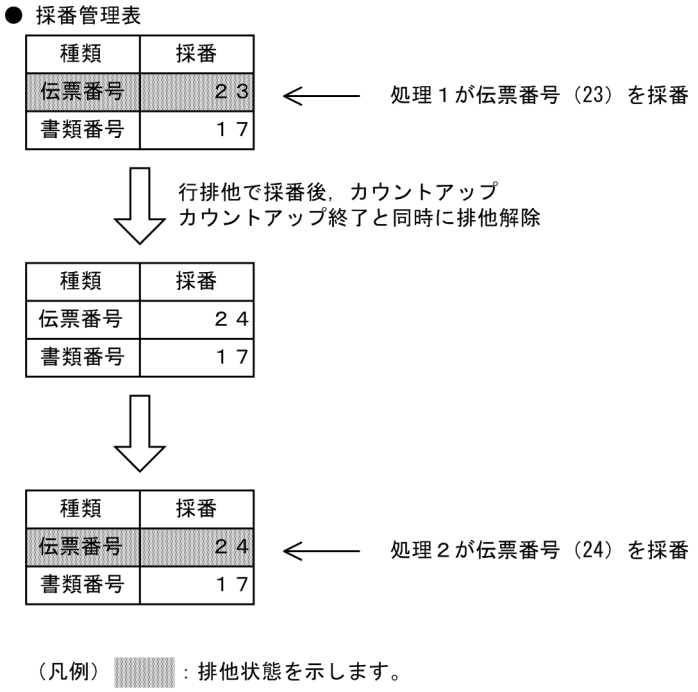
更新処理完了時に排他が解除されるので，排他待ちの発生を削減できます。

(2)　適用基準

採番業務のような，更新処理が集中する表に適しています。

伝票番号や書類番号などの採番業務では，一つの表で番号を管理し，採番するごとにカウントアップするという方法が考えられます。しかし，この方法では処理が集中すると，COMMIT 発行まで排他が解除されないため，排他待ちが多く発生することが考えられます。このような処理をする場合，その表に WITHOUT ROLLBACK オプションを指定すると，カウントアップが終了した時点で排他が解除されるため，排他待ちの発生を削減できます。採番業務の例を次の図に示します。

図 11–23　採番業務の例



この例では，1 行で 1 種類の番号を管理しています。

図「採番業務の例」に示した採番管理表の定義例を次に示します。

```
CREATE TABLE 採番管理表 (種類 NCHAR(4),
                         採番 INT)
                    :
              WITHOUT ROLLBACK
```

ただし，次に示す要因で欠番が発生する可能性がありますので，欠番が発生しても問題がない業務で使用してください。

- 表定義時に WITHOUT ROLLBACK オプションを指定した表の行を更新したトランザクションが ROLLBACK 文や SQL 実行中のエラーによりロールバックした場合，取得した番号を使用した業務の表に対してはロールバックされます。この場合，整合性が保たれますが，取得した番号はロールバックされません。このため，ロールバックしたトランザクションで取得した値は HiRDB システム内で使われずに次の番号が取得され，欠番となります。

(3) 注意

- データベース作成ユティリティ（pdload）及びデータベース再編成ユティリティ（pdrorg）をログ取得モードを指定して実行した場合，WITHOUT ROLLBACK オプションを指定した表でも，通常の表と同様にロールバックされます。
- 採番業務のように更新処理が集中する表の場合，専用の RD エリア及びグローバルバッファを割り当ててください。
- システム共通定義 pd_idx_without_rollback に N を指定した場合，WITHOUT ROLLBACK オプションを指定している表にインデクスを定義すると，行更新時にインデクス構成列に対して同値更新となるときだけ行更新ができます。また，行挿入及び行削除時は，行排他は解除されず，通常の表と同様にロールバックされます。
- 表定義時に WITHOUT ROLLBACK オプションを指定した表の行を更新したトランザクションが完了する前に HiRDB システムが異常終了した場合，HiRDB システムの再開始後に採番した値が戻るときがあります。採番した値を HiRDB システム内に閉じて使用する場合，採番した値が戻っても，その値を使用したトランザクションもロールバックするため，HiRDB システム内で一意性が保てます。HiRDB システム外で使用する場合は採番した値が戻ると一意性が保てなくなります。HiRDB システム外で使用する場合は，更新する SQL 文に WRITE IMMEDIATE オペランドを指定してください。WRITE IMMEDIATE オペランドを指定すると，HiRDB システムが異常終了しても値が戻りません。ただし，WRITE IMMEDIATE オペランドを指定すると 1 行の更新ごとにシステムログの書き出しが行われ，その書き出し時間が SQL の実行時間に加算されます。

11.15　改竄防止機能の指定

改竄防止機能とは，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する機能です。この機能を適用することで，重要なデータを人為的ミスや不正な改竄から守れます。この機能が適用された表を**改竄防止表**といいます。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 11–4　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。

適用基準

改竄防止機能は，表のデータを誤って，又は不当に更新されることを防止したい表に適用をお勧めします。

11.15.1　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 で 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。既存の表を改竄防止表に定義変更する場合，事前に実行します。

- **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（「定義系 SQL」参照），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。

なお，データベース再編成ユティリティや pdrels コマンドでの RD エリアに対する運用に制限がある※ため，改竄防止表は 1 表 1RD エリアに格納することをお勧めします。

注※

データベース再編成ユティリティで改竄防止表の再編成をする場合，RD エリアのコマンド閉塞が必須です。また，データベース再編成ユティリティが異常終了した場合は，再編成が完了するまで RD エリアの閉塞が解除できないため，該当する RD エリアに別の表やインデクスが定義されている場合は，その表やインデクスが使用できなくなります。詳細は，「制限事項」を参照してください。

11.15.2　制限事項

改竄防止表はデータの更新及び削除ができない表のため，改竄防止表又は改竄防止表格納 RD エリアに対して，SQL 文，ユティリティ，及びコマンドの実行に制限があります。

(1)　定義系 SQL

改竄防止表に対して実行できない定義系 SQL 文があります。制限がある定義系 SQL 文と制限事項を次の表に示します。

表 11-5　制限がある定義系 SQL 文と制限事項

SQL 文	制限事項
CREATE TABLE	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。
ALTER TABLE	• 表名称，及び列名称は変更できません。 • データが存在する表に改竄防止機能は適用できません。既存の表に改竄防止機能を適用する手順については，「非改竄防止表から改竄防止表への変更」を参照してください。 • 改竄防止機能の解除はできません。 • 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※。 • 行削除禁止期間の設定，解除，及び期間の変更はできません。 • 行削除禁止期間が指定されている改竄防止表の場合，行削除禁止期間を指定した挿入履歴保持列は削除できません。 • 分割格納条件は変更できません。
DROP TABLE	改竄防止表にデータがある場合，実行できません。

注※

既存の表に更新可能列を指定して，改竄防止機能を適用するためには，列及び表に対してそれぞれ ALTER TABLE を実行する必要があります。次の手順で改竄防止機能を適用してください。

1. ALTER TABLE で，更新可能とする列の属性を更新可能列に変更
2. ALTER TABLE（CHANGE INSERT ONLY）で，改竄防止表にする表に対して改竄防止機能を適用

(2)　ユティリティ

改竄防止表，又は改竄防止表格納 RD エリアに対して，ユティリティ運用が制限されます。制限されるユティリティと制限事項を次の表に示します。記載されていないユティリティについては，特に制限はありません。

表 11-6　運用時に制限があるユティリティと制限事項

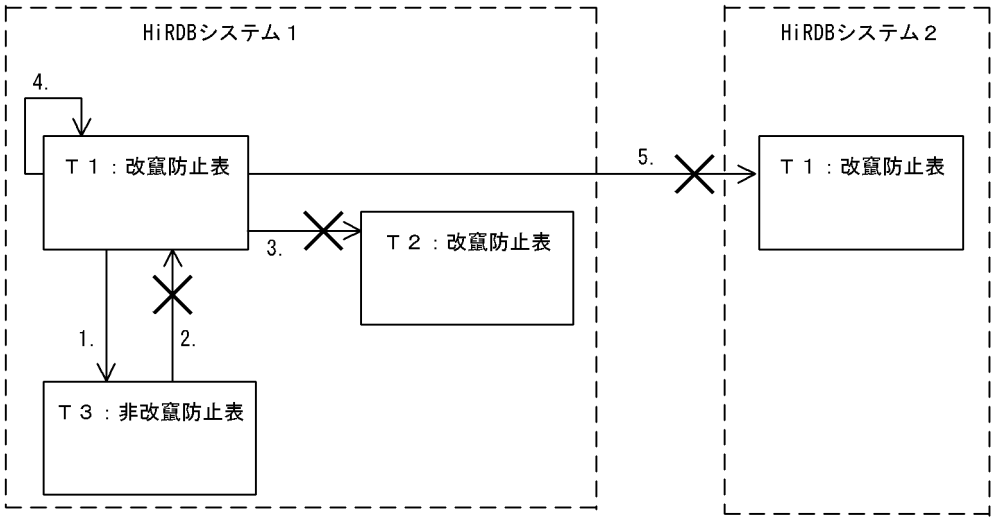
ユティリティ名	制限事項
データベース作成ユティリティ（pdload）	• 作成モード（-d オプション指定）は使用できません。 • 対象となる表がデータ未完状態※の場合は実行できません。
データベース構成変更ユティリティ（pdmod）	• 改竄防止表格納 RD エリアの再初期化（initialize rdarea）は実行できません。 • 次に示す機能は使用できません。 HiRDB ファイルシステム領域の世代登録（create generation） HiRDB ファイルシステム領域の世代削除（remove generation） RD エリアのレプリカ定義（replicate rdarea） RD エリアの構成情報複写（define copy rdarea） RD エリアの統合（recast rdarea）

ユティリティ名	制限事項
データベース再編成ユティリティ（pdrorg）	**表の再編成（-k rorg）** • 処理対象となる表格納 RD エリアがコマンド閉塞となっていない場合は実行できません。 • UOC を利用した再編成（unlduoc 文）はできません。 • 同期点指定の再編成（option job 文）はできません。 • 対象となる表がデータ未完状態※の場合は実行できません。 **表のアンロード（-k unld）** • -W オプションを指定しない場合は実行できません。 **表のリロード（-k reld）** • 処理対象となる表格納 RD エリアがコマンド閉塞となっていない場合は実行できません。 • 対象となる表がデータ未完状態※となっている場合に実行できます（表の再編成で，表のリロードが異常終了した場合での再実行だけ実行できます）。 • 同期点指定の再編成（option job 文）はできません。 • 別表へのリロードは実行できません。詳細は図「別表へのリロード可否」を参照してください。 **インデクスの一括作成（-k ixmk），再作成（-k ixrc），再編成（-k ixor）** 対象となる表がデータ未完状態※の場合は実行できません。
リバランスユティリティ（pdrbal）	対象となる表がデータ未完状態※の場合は実行できません。

注※

改竄防止表に対して表の再編成を実行し，エラーなどによって表のリロードが完了していない表の状態を**データ未完状態**といい，改竄防止表格納 RD エリアごとに状態を保持します。該当する RD エリアがデータ未完状態かどうかは，データベース状態解析ユティリティで，RD エリア単位の状態解析（論理的解析）又は表単位の状態解析を実行することで確認できます。データ未完状態は表の再編成（表のリロード）が正常に完了した場合に解除されます。データ未完状態の解除方法の詳細は，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。

図 11-24　別表へのリロード可否



〔説明〕

1. 改竄防止表 T1 から非改竄防止表 T3 へのリロードは，改竄防止表 T1 の改竄にはならないため，できます。
2. 非改竄防止表 T3 から改竄防止表 T1 へのリロードは，改竄防止表 T1 のデータが改竄されることになるため，できません。
3. 改竄防止表 T1 から改竄防止表 T2 へのリロードは，改竄防止表 T2 のデータが改竄されることになるため，できません。
4. 改竄防止表 T1 自身に対するリロードは，改竄防止表 T1 の改竄にはならないため，できます。
5. HiRDB システム 1 の改竄防止表 T1 から HiRDB システム 2 の改竄防止表 T1 へのリロードは，HiRDB システム 2 の改竄防止表 T1 のデータが改竄されることになるため，できません。

(3) 運用コマンド

改竄防止表，又は改竄防止表格納 RD エリアに対して，コマンド実行が制限されます。制限事項がある運用コマンドを次の表に示します。

表 11–7 制限がある運用コマンドと制限事項

運用コマンド	制限事項
RD エリアの閉塞（pdhold）	データ未完状態の改竄防止表格納 RD エリアの場合，状態を遷移させると表の再編成を完了させるためのリロード実行が不可となるため次のオプションは実行できません。 • 参照可能閉塞：-i • バックアップ閉塞：-b
RD エリアの閉塞解除（pdrels）	データ未完状態の改竄防止表格納 RD エリアの場合，表の再編成が未完了のままで閉塞解除すると該当表のデータが 0 件の状態となるため，実行できません。

(4) 関連製品での制限事項

関連製品での制限事項を次に示します。

- インナレプリカ機能

  改竄防止表格納 RD エリアにはインナレプリカ機能を使用できません。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。

- レプリケーション機能

  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となったり，エラーとなる場合があります。

11.15.3　非改竄防止表から改竄防止表への変更

既存の表を改竄防止表に変更する方法について説明します。次に示すどちらかの方法で，表を改竄防止表に変更できます。

- HiRDB Control Manager の改竄防止ウィザードを使用する方法
- HiRDB のコマンドを使用する方法

なお，データが格納されている表は改竄防止表に変更できません。したがって，表にデータが格納されている場合は，いったん表データをアンロードし，その後に ALTER TABLE で表の定義を変更します。

(1)　HiRDB Control Manager の改竄防止ウィザードを使用する方法

HiRDB Control Manager の改竄防止ウィザードを使用して，改竄防止表に変更する手順を次に示します。なお，手順の画面は Windows 版 HiRDB サーバで実行した例です。UNIX 版 HiRDB サーバで実行する場合は，パス名の表記が異なります。

〈手順〉

表 T1 を次に示す条件を設定した改竄防止表に変更します。

- COL_NOTE 列は更新可能列にします。
- COL_DATE 列を挿入履歴保持列とし，行削除禁止期間を 10 年に設定します。

1. HiRDB Control Manager - Console を起動します。HiRDB Control Manager - Console の起動方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。
2. 操作対象の HiRDB サーバを登録します。管理 HiRDB の登録方法については，マニュアル「HiRDB システム運用ガイド」を参照してください。既に登録してある場合，この手順は必要ありません。
3. タブメニュー［表操作］から［改ざん防止表移行ウィザード］を選択します。
   ［改ざん防止ウィザード - 1/5］画面が表示されます。

改竄防止表に変更する表名を選択して，［パスワード］テキストボックスに，表所有者のパスワードを入力してください。ここで入力する値は，大文字と小文字の区別をします。

4. ［次へ>］ボタンをクリックします。

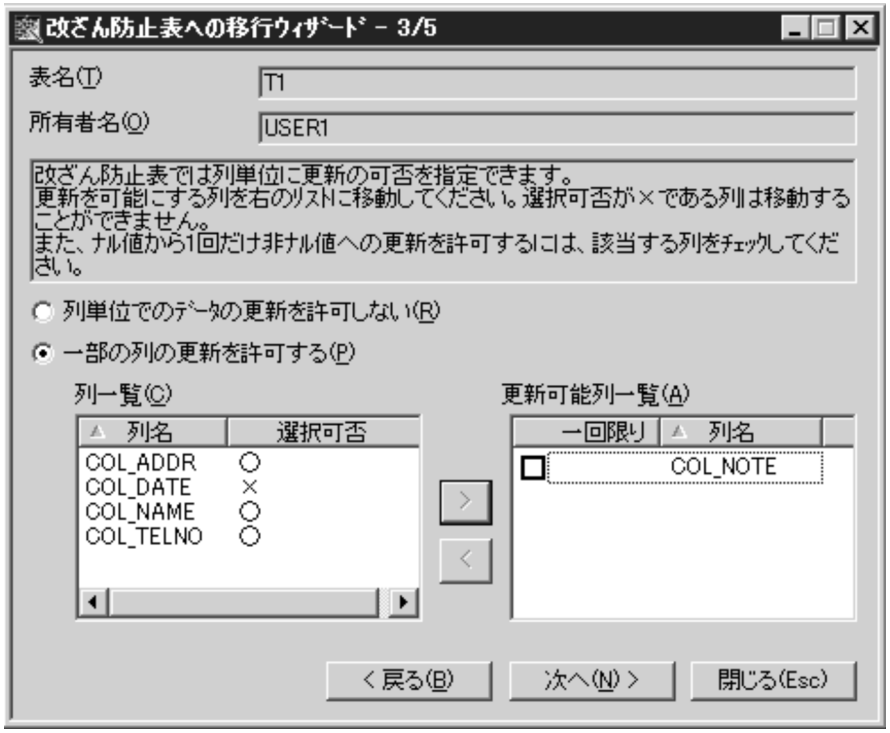
［改ざん防止ウィザード - 2/5］画面が表示されます。

この画面では，行削除禁止期間を設定します。行削除禁止期間を設定しない場合は，［表データの削除を行わない］が選択されていることを確認してください。行削除禁止期間を設定する場合は，［一

定期間後に表のデータの削除を許可する］を選択し，［挿入履歴保持列名］と［行削除禁止期間］を設定してください。

5. ［次へ>］ボタンをクリックします。

［改ざん防止ウィザード - 3/5］画面が表示されます。



この画面では，更新可能列を設定します。更新可能列を設定しない場合は，［列単位でのデータの更新を許可しない］が選択されていることを確認してください。更新可能列を設定する場合は，［一部の列の更新を許可する］を選択してください。次に，［列一覧］リストから更新可能にする列名を選択し，［>］ボタンをクリックして［更新可能列一覧］リストに追加してください。

6. ［次へ>］ボタンをクリックします。

［改ざん防止ウィザード - 4/5］画面が表示されます。

この画面では，変更処理中に使用する一時ファイルを格納するディレクトリについて指定します。[一時ファイル格納先一覧] リストに既定値が設定されていますので，必要に応じて変更してください。

7. [次へ>] ボタンをクリックします。
   [改ざん防止ウィザード - 5/5] 画面が表示されます。

設定した条件を確認します。設定を変更したい場合は，[<戻る] ボタンをクリックして，前の画面に戻ります。

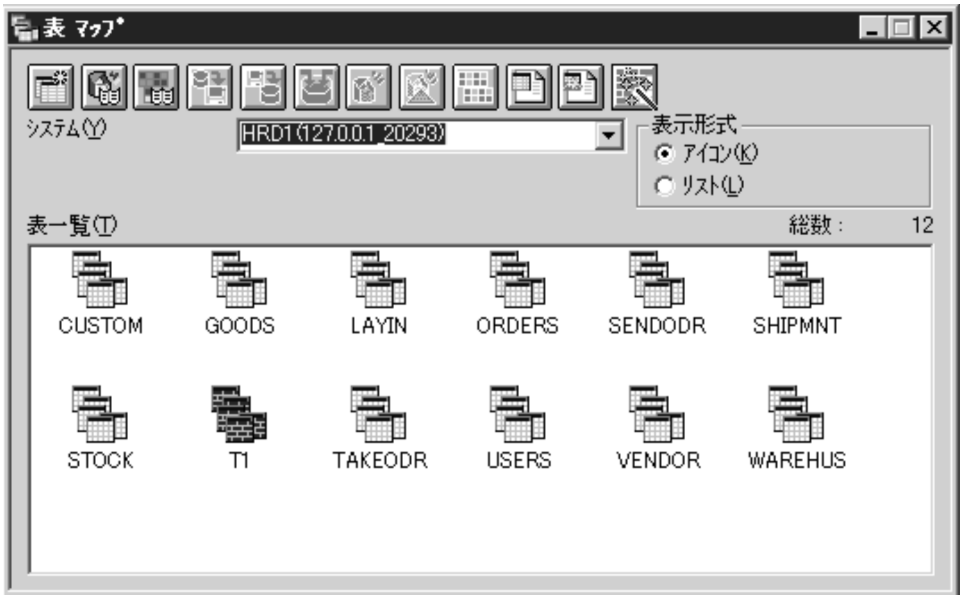
8. 問題がなければ［実行］ボタンをクリックします。変更処理が始まります。

注意事項

改竄防止ウィザード実行後は，ディクショナリ用 RD エリアと表データの回復に必要な RD エリアのバックアップを取得してください。バックアップは，HiRDB Control Manager のバックアップウィザードで実行できます。

参考

HiRDB Control Manager では，改竄防止表は通常表と区別して表示されます。タブメニュー［マップ］から［表］を選択して，［表マップ］画面を表示すると，変更した表が改竄防止属性であることが確認できます。



(2) HiRDB のコマンドを使用する方法

HiRDB のコマンドを使用して，改竄防止表に変更する手順を次に示します。

〈手順〉

RD エリア（RDAREA01）に格納されている表 T1 を改竄防止表に変更します。

1. pdhold コマンドで，非改竄防止表が格納されている RD エリアとデータディクショナリ用 RD エリア（RDDIC01）をバックアップ閉塞します。

```
pdhold -r RDAREA01,RDDIC01 -b
```

2. バックアップ対象の RD エリアが属するサーバ（bes01，dic01）のシステムログファイルをスワップさせます。

```
pdlogswap -d sys -s bes01 -w
pdlogswap -d sys -s dic01 -w
```

3. データベース複写ユティリティ（pdcopy）を実行して RD エリアのバックアップを取得します。
   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。

```
pdcopy -m /hirdb/rdarea/mast/mast01 -M r -p /usr/hirdb/pdcopy/pdcopy01 -b /usr/hirdb/p
dcopy/backup/backup01 -r RDAREA01,RDDIC01
```

4. pdrels コマンドで，データディクショナリ用 RD エリアの閉塞を解除します。

```
pdrels -r RDDIC01
```

5. 非改竄防止表のデータを，データベース再編成ユティリティ（pdrorg）を使用してアンロードします。このとき，後でデータベース作成ユティリティ（pdload）の入力データとして使用できるように，-W オプションを指定してアンロードしてください。制御文ファイル（control_file）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。

```
pdrorg -k unld -t T1 -W bin control_file
```

6. pdrels コマンドで，ユーザ用 RD エリアの閉塞を解除します。なお，9.で再度 RD エリアを閉塞するまで，これ以降 RD エリアへのアクセスは行わないでください。この間に対象となる表が更新された場合，データの不整合が発生するおそれがあります。

```
pdrels -r RDAREA01
```

7. 非改竄防止表のデータを PURGE TABLE ですべて削除します。

```
PURGE TABLE T1
```

8. ALTER TABLE で改竄防止オプションを指定して，改竄防止表に変更します。

```
ALTER TABLE T1 CHANGE INSERT ONLY
```

9. pdhold コマンドで，改竄防止表が格納されている RD エリアを閉塞します。

```
pdhold -r RDAREA01
```

10. 5.でアンロードしたデータを，データベース作成ユティリティ（pdload）でロードします。制御文ファイル（control_file）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。

```
pdload -b -W T1 control_file
```

11. pdrels コマンドで，RD エリアの閉塞を解除します。

```
pdrels -r RDAREA01
```

12. pdhold コマンドで，バックアップ対象の RD エリアをバックアップ閉塞します。

```
pdhold -r RDAREA01,RDDIC01 -b
```

13. バックアップ対象の RD エリアが属するサーバ（bes01，dic01）のシステムログファイルをスワップさせます。

```
pdlogswap -d sys -s bes01 -w
pdlogswap -d sys -s dic01 -w
```

14. データベース複写ユティリティ（pdcopy）を実行して RD エリアのバックアップを取得します。
    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。

```
pdcopy -m /hirdb/rdarea/mast/mast01 -M r -p /usr/hirdb/pdcopy/pdcopy01 -b /usr/hirdb/p
dcopy/backup/backup01 -r RDAREA01,RDDIC01
```

15. pdrels コマンドで，RD エリアの閉塞を解除します。

```
pdrels -r RDAREA01,RDDIC01
```

なお，改竄防止オプションの設定時期はディクショナリ表の SQL_TABLES 表の値で確認できます。SQL_TABLES 表の値の意味を次の表に示します。

表 11-8　SQL_TABLES 表の値の意味

改竄防止オプションの設定	SQL_TABLES	
	INSERT_ONLY 列の値	CHANGE_TIME_INSERT_ONLY の値
なし	NULL	NULL
CREATE TABLE 実行時に指定	Y	NULL
ALTER TABLE 実行時に指定	Y	改竄防止表への変更日時

11.15.4　障害時の運用

改竄防止表格納 RD エリアは再初期化（initialize rdarea）できないため，RD エリアの回復のとき，再初期化を使った回復はできません。データベース回復ユティリティ（pdrstr）で回復してください。また，RD エリア満杯時は expand rdarea 文で RD エリアを拡張してください。

11.16　繰返し列を含む表

11.16.1　繰返し列を含む表の効果と指定方法

HiRDB では，複数の要素から構成される列（**繰返し列**）を含む表を定義できます。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。従来は，このような表を定義する場合，次の図のように作成する必要がありました。繰返し列を定義しない表の例を次の図に示します。

図 11–25　繰返し列を定義しない表の例

社員表

氏名	性別	資格
伊藤栄一	男	情報処理１種
伊藤栄一	男	ネットワーク
伊藤栄一	男	情報処理２種
中村和男	男	情報処理２種
中村和男	男	英語検定２級
河原秀雄	男	シスアド
井上俊夫	男	

扶養家族表

氏名	家族	続柄	扶養
伊藤栄一	虎夫	父	1
伊藤栄一	ウメ	母	1
伊藤栄一	綾子	妻	1
伊藤栄一	太郎	長男	1
伊藤栄一	恵子	次女	1
中村和男	和彦	父	0
中村和男	陽子	妻	1
河原秀雄	直子	母	1

この二つの表をアクセスする場合，結合する必要があります。結合することで，SQL の構文が複雑になるなどのデメリットが発生します。そこで，繰返し列を含む表を作成することで，一つの表として作成できるため，結合が不要になります。

繰返し列を含む表の例を次の図に示します。

図 11–26　繰返し列を含む表の例

社員表

氏名	資格	性別	家族	続柄	扶養
伊藤栄一	情報処理１種	男	虎夫	父	1
	ネットワーク		ウメ	母	1
	情報処理２種		綾子	妻	1 ← 繰返し列の要素
			太郎	長男	1
			恵子	次女	1
中村和男	情報処理２種	男	和彦	父	0
	英語検定２級		陽子	妻	1 ← 行
河原秀雄	シスアド	男	直子	母	1
井上俊夫		男			

注　空白の箇所は，ナル値を表します。

〔説明〕

「資格」,「家族」,「続柄」,「扶養」が繰返し列になります。

(1) 繰返し列を含む表を定義したときの効果

多値多重性のある表を行ごとにまとめた形で表現できます。このため，次に示す効果が期待できます。

- 複数の表の結合が不要になります。
- 重複する情報がなくなるため，ディスク容量を削減できます。
- 関連データ（繰返しデータ）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。

(2) 指定方法

繰返し列を指定するには，定義系 SQL の CREATE TABLE の列定義に ARRAY オプションを指定します。

繰返し列を含む表の定義例を次に示します。この定義例は図「繰返し列を含む表の例」に示した社員表の場合です。なお，社員表には,「続柄」,「扶養」に複数列インデクスが定義されているものとします。

(例)

```
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]);

CREATE INDEX 扶養IDX ON 社員表 (続柄,扶養);
```

注　扶養 IDX は，社員表に付けたインデクス名です。

(3) 注意

- 次に示すデータ型に対して繰返し列を指定できません。
  - BLOB 型
  - BINARY 型
  - 抽象データ型
- クラスタキーを指定した列には，繰返し列を指定できません。
- FIX を指定した場合，繰返し列を指定できません。
- 繰返し列に対して格納条件，ハッシュ分割，サプレスオプションを指定できません。
- キーレンジ分割をする場合，境界値を指定する列に繰返し列を指定できません。
- 繰返し列には非ナル値制約を指定できません。

11.17　抽象データ型を含む表

11.17.1　抽象データ型の効果と継承

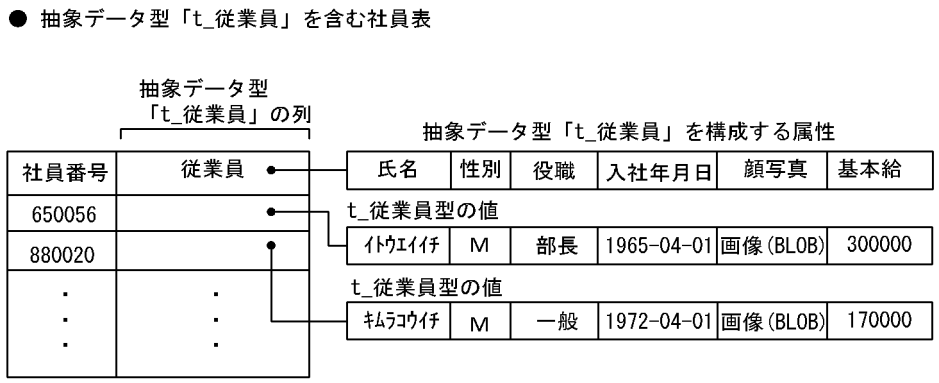
HiRDB では表を構成する列のデータ型として**抽象データ型**を定義できます。

抽象データ型とは，既定義のデータ型で扱えないような複雑なデータを扱いやすく表現するための構造を持ったデータ型のことです。HiRDB では，このようなデータ型をユーザが抽象データ型として定義できます。抽象データ型では，構造を示す属性とその値に対する操作をひとまとまりとし，定義系 SQL によって定義できます。

抽象データ型は，数値型や文字型などの既定義のデータ型と同様に，表を構成する列のデータ型としてを扱えます。

抽象データ型を含む表のデータ構造を次の図に示します。次の図では，社員表中の「従業員」列のデータ型を抽象データ型「t_従業員型」としています。

図 11-27　抽象データ型を含む表のデータ構造



(1)　抽象データ型を定義した場合の効果

- 複雑な構造のデータを一つの値として扱えます。
- データとそれに対する操作を一体化することで，オブジェクト指向のアプリケーションとマッピングしやすくなります。
- データとそのデータに対する操作をひとまとまりとし，操作を外部的なインタフェースとすることで，データの内部情報を意識することなくデータを扱えます。

(2) 継承の概要

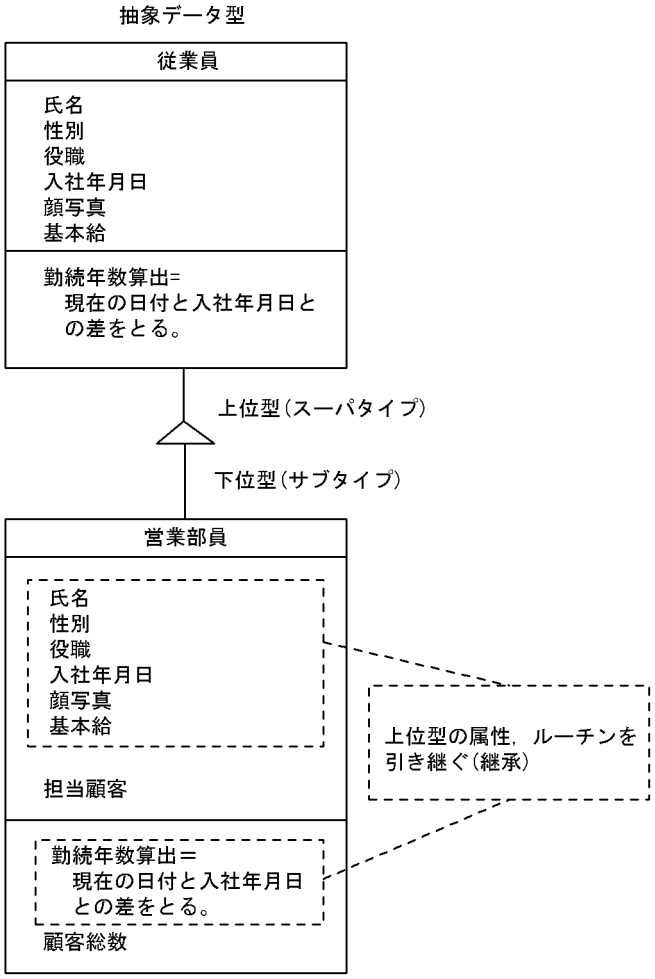
(a) 継承（inheritance）

HiRDB では，既に定義した抽象データ型を基に，その型に定義された属性と操作を受け継いだ新しい抽象データ型を導出し定義できます。基になる型を**スーパタイプ**といい，導出した型を**サブタイプ**といいます。サブタイプがスーパタイプの属性及び関数を引き継ぐことを**継承**といいます。

スーパタイプーサブタイプの関係は階層的に表現できます。これによって，複雑な概念モデルを抽象データ型を用いて階層化して表現できます。

抽象データ型のスーパタイプーサブタイプ関係に基づく階層構造を次の図に示します。次の図では，抽象データ型「従業員」から，サブタイプ「営業部員」を導出しています。

図 11-28　抽象データ型のスーパタイプーサブタイプ関係に基づく階層構造

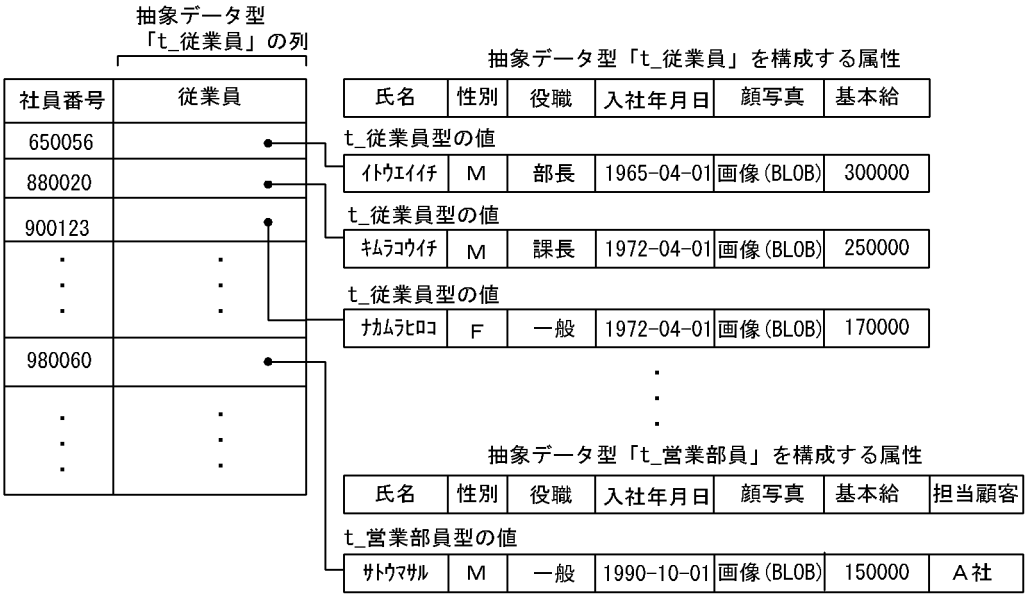


(b) 代替可能性（substitutability）

サブタイプの値は，そのスーパタイプの値として扱うことができます。これを**代替可能性**（substitutability）といいます。代替可能性を利用して値を挿入した抽象データ型を含む表のデータ構造を次の図に示します。

図 11-29　抽象データ型を含む表のデータ構造（代替可能性を利用した場合）

● 抽象データ型「t_従業員」を含む社員表



(c)　多重定義（override）

HiRDB では，上位の抽象データ型（スーパタイプ）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（サブタイプ）の定義で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。多重定義によって，呼び出すルーチンの名称を型によって変更する必要がありません。

(3)　継承を利用したときの効果

継承を利用することで，次に示す効果が期待できます。

- 上位の抽象データ型の特性（データ及び操作）を下位の抽象データ型でも利用できます。
- サブタイプを定義することで，最初から定義し直さなくてもデータ定義を共有できます。これによって，データベースの定義の手間が省けます。
- 多重定義によって，呼び出すルーチンの名称を型によって変更する必要がありません。

(4)　抽象データ型の定義方法

抽象データ型を定義するには，定義系 SQL の CREATE TYPE を実行します。CREATE TYPE では，構造を示すための属性と，値に対する操作を定義します。また，継承を利用する場合は CREATE TYPE でサブタイプ句を指定して定義します。CREATE TYPE の定義例については，「ユーザが定義した抽象データ型を定義した表の作成」を参照してください。

(a) コンストラクタ関数の定義

抽象データ型の値を生成するための関数（**コンストラクタ関数**）を定義します。コンストラクタ関数の実装で，抽象データ型の定義時に HiRDB によって提供される**デフォルトコンストラクタ関数**を利用できます。デフォルトコンストラクタ関数は，すべての属性がナル値である値を生成します。

(b) ルーチンの定義

抽象データ型の定義内に，ある属性の値を操作するインタフェースとして**ルーチン**を定義できます。

(c) 隠蔽レベルの指定

抽象データ型を構成する属性及びルーチンに対するアクセスを制御するため，**隠蔽レベル**を指定できます。隠蔽レベルは，属性及びその抽象データ型の値に対する操作であるルーチンに指定できます。隠蔽レベルには，次に示す 3 種類があります。

- PUBLIC
  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中，アプリケーションからも属性の値へアクセスさせたい場合及びルーチンを使用させたい場合に指定します。
- PRIVATE
  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などに，その抽象データ型の定義中だけで，属性の値へアクセスさせたい場合及びルーチンを使用させたい場合に指定します。SQL で属性の値へアクセスさせたい場合及びルーチンを使用したい場合は，関数を定義する必要があります。
- PROTECTED
  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などに，その抽象データ型の定義中及びその抽象データ型のサブタイプの定義中でだけ属性の値へアクセスさせたい場合やルーチンを使用させたい場合に指定します。

抽象データ型の定義内でいったん隠蔽レベルを指定すると，次に別の隠蔽レベルの指定が出現するまでは直前の隠蔽レベルが有効になります。また，隠蔽レベルの指定がない場合，PUBLIC が仮定されます。隠蔽レベルの違いによって，データへのアクセス，ルーチンの使用権限の範囲が異なります。

隠蔽レベルと権限を次の表に示します。

表 11–9　隠蔽レベルと権限

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	○	○	○	○
PRIVATE	○	×	×	×
PROTECTED	○	○	×	×

（凡例）

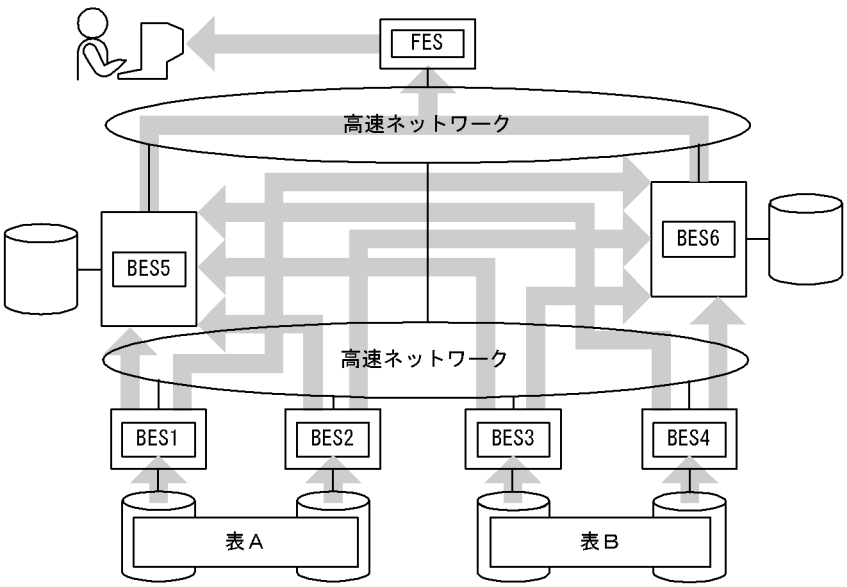
○：属性値へのアクセス及びルーチンを使用できます。

×：属性値へのアクセス及びルーチンを使用できません（SQL エラーになります）。

11.18　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。共用表を更新できるのは**更新可能バックエンドサーバ**だけです。ほかのバックエンドサーバは**参照専用バックエンドサーバ**になります。ただし，共用表の更新には制限があるため，原則としてオンライン中は更新しないでください。共用表の更新については，「共用表の操作」を参照してください。共用表を使用しない結合処理と使用した結合処理を次の図に示します。

図 11–30　共用表を使用しない結合処理



〔説明〕

表 A と表 B の結合処理をします。

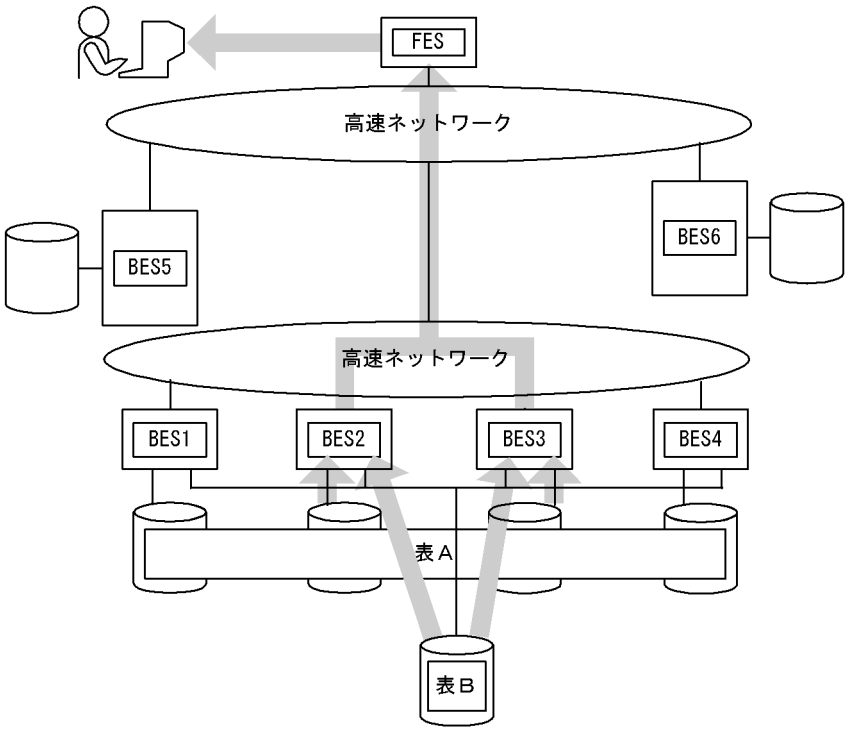
BES1，2：表 A からデータを取り出し，突き合わせ処理のため，BES5 と BES6 にデータを転送します。

BES3，4：表 B からデータを取り出し，突き合わせ処理のため，BES5 と BES6 にデータを転送します。

BES5，6：突き合わせ，結合処理をして FES にデータを転送します。

FES：結合処理した結果をマージして，ユーザに結果を返します。

図 11-31　共用表を使用した結合処理



〔説明〕

表 A と表 B の結合処理をします。共通データがある表 B を共用表とします。検索範囲は BES2，3 のバックエンドサーバにあります。

BES1，4，5，6：特に処理はありません。

BES2，3：表 A 及び表 B からデータを取り出し，マージ処理をして FES にデータを転送します。

FES：ユーザに結果を返します。

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 の互換性を保つことができます。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」を参照してください。

11.18.1　効果と適用基準

(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。

11.18.2 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

11.18.3 共用表の操作

(1) 検索

共用表はすべてのバックエンドサーバから参照できるため，共用表を検索するために最適なバックエンドサーバを HiRDB が決定します。なお，共用表を更新すると全バックエンドサーバで排他が掛かるため，検索処理と更新処理とのデッドロックが発生することがあります。デッドロックを回避するために，共用表の検索時は次のようにすることをお勧めします。

- 排他オプションに WITHOUT LOCK，又は WITHOUT LOCK NOWAIT を指定する
- 共用表を更新するために検索する場合，FOR UPDATE 句を指定する

なお，共用表に対して IN EXCLUSIVE MODE 指定で LOCK 文を実行すると，対象の共用表と共用インデクスが格納されている RD エリアに排他を掛けます。検索する表が LOCK 文の対象ではなくても，アクセスする RD エリアが同じであれば排他制御されます。そのため，WITHOUT LOCK NOWAIT を指定していても，ほかのトランザクションが IN EXCLUSIVE MODE 指定で LOCK 文を実行している場合は，共用表にアクセスできません。そのため，IN EXCLUSIVE MODE 指定の LOCK 文実行中は共用表を検索できません。

HiRDB が決定するバックエンドサーバの割り当て規則については，「共用表を検索するバックエンドサーバの割り当て規則」を参照してください。

(2) 更新

共用表を更新する場合，LOCK 文で IN EXCLUSIVE MODE を指定し，全バックエンドサーバの共用 RD エリアに排他を掛けなければ実行できません。ただし，インデクスキー値を変更しない UPDATE 文は，LOCK 文を発行しないで実行できます。共用表及び共用インデクスの更新は，COMMIT 文発行時にディスクに書き込まれます。

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)。

(a) LOCK 文を発行する更新

LOCK 文を発行する更新の流れを次に示します。

1. IN EXCLUSIVE MODE 指定で LOCK 文を発行します。
   このとき，共用表だけでなく，共用表が格納されている共用 RD エリア及び共用インデクスが格納されている共用 RD エリアにも排他が掛かります。参照専用バックエンドサーバの共用 RD エリアのグローバルバッファは無効化されます。
2. 共用表に対して，INSERT，UPDATE，又は DELETE 文を実行します。
   更新可能バックエンドサーバが更新情報をファイルに反映します。
   LOCK 文が解除されるまで共用 RD エリアに排他が掛かるので，同じ共用 RD エリアにある，別の共用表への操作は待ち状態になります。
3. LOCK 文を解除します。

注意事項

- LOCK 文は UAP の最初に発行してください。LOCK 文発行時，関連する共用 RD エリア内の表に対して，自サーバプロセスでオープン中のカーソルがあると，LOCK 文はエラーになります。
- 共用表を更新する手続き及びトリガを作成する場合，LOCK 文を記述してください。ただし，手続き及びトリガから LOCK 文を実行する場合はトランザクションの開始時点から排他が掛かりません。そのため，エラーになるおそれがあります。
- 全バックエンドサーバで共用表，共用表が格納されている共用 RD エリア，及び共用インデクスが格納されている共用 RD エリアに排他を掛けるため，該当する RD エリアの表又はインデクス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おそれがあります。
- 共用表を更新するトランザクションの決着前に更新可能バックエンドサーバのユニットが異常終了して再開始しない場合に次のような検索をすると，排他タイムアウトエラーになります(KFPA11770-I メッセージが出力されます)。
  ・別のユニットの参照専用バックエンドサーバで，更新対象の共用表又はその表に定義されているインデクスが格納されている RD エリア内の表を検索

(b) LOCK 文を発行しない更新

LOCK 文を発行しない場合，実行できるのはインデクスキー値の変更がない UPDATE 文だけです。また，少量の変更の場合だけにしてください。

LOCK 文を発行しない更新の流れを次に示します。

1. 全バックエンドサーバの状態を同じにするため，更新情報を全バックエンドサーバに配布します。
2. 更新可能バックエンドサーバが更新情報をデータベースに反映します。
   参照専用バックエンドサーバでは，グローバルバッファ上で更新し，COMMIT 文発行までファイルに反映しないで更新情報を保持します。トランザクションがロールバックした場合，グローバルバッファ上で回復します。

注意事項

- 参照専用バックエンドサーバで，グローバルバッファがすべて更新中，かつ COMMIT 文未発行で空きページがない状態になった場合，トランザクションはロールバックします。このため，LOCK 文を発行しない場合は大量のデータを更新しないでください。
- 全バックエンドサーバで更新行に排他を掛けるため，同時に該当する表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おそれがあります。デッドロックを回避するため，UPDATE 文で更新する行はトランザクションで 1 件だけにすることをお勧めします。
- 共用表を更新するトランザクションの決着前に更新可能バックエンドサーバのユニットが異常終了して再開始しない場合に次のような検索をすると，排他タイムアウトエラーになります（KFPA11770-I メッセージが出力されます）。
  ・別のユニットの参照専用バックエンドサーバで，更新対象の共用表又はその表に定義されているインデクスが格納されている RD エリア内の表を検索

11.18.4 共用表の制限事項

- IN EXCLUSIVE MODE 指定の LOCK 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。

11.18.5 共用表を検索するバックエンドサーバの割り当て規則

共用表を検索するバックエンドサーバの割り当て規則について説明します。1SQL 中に共用表だけが指定されている場合と，1SQL 中に共用表を含む複数の表が指定されている場合とで割り当て規則が異なります。それぞれの割り当て規則について説明します。

(1) 1SQL 中に指定されている表が共用表だけの場合

1SQL 中に指定されている表が共用表だけの場合，共用表を検索するバックエンドサーバの割り当ては，同一トランザクション内の直前の SQL でどのような検索を行ったかなどの条件によって決まります。

共用表を検索するバックエンドサーバの割り当て規則（1SQL 中に指定されている表が共用表だけの場合）を次の表に示します。項番 1 が最も優先順位が高くなっています。

表 11-10 共用表を検索するバックエンドサーバの割り当て規則（1SQL 中に指定されている表が共用表だけの場合）

項番	検索条件	割り当てられるバックエンドサーバ
1	次に示すどちらかの条件を満たす場合 • 検索対象の共用表に対して，IN EXCLUSIVE MODE 指定の LOCK TABLE 文を実行している • FOR UPDATE 句による検索を行っている	更新可能バックエンドサーバを割り当てます。
2	同一トランザクション内の（又は複数のトランザクションにわたって※1）異なる SQL 内で共用表※2 を使用している場合	同一トランザクション内の共用表検索のうち，直前に実行した SQL 文内で共用表へのアクセスに使用したバックエンドサーバを割り当てます。 バックエンドサーバの割り当ての例については，例 1 を参照してください。
3	同一トランザクション内の（又は複数のトランザクションにわたって※1）異なる SQL 内で横分割表※3 を使用し，分割列に 1 バックエンドサーバだけの検索が行われる制限条件※4 がある場合	同一トランザクション内の分割列に，1 バックエンドサーバだけの検索が行われる制限条件がある横分割表検索のうち，直前に実行した SQL 文内で横分割表へのアクセスに使用したバックエンドサーバの中から，ランダムに割り当てます。 バックエンドサーバの割り当ての例については，例 2 を参照してください。
4	同一トランザクション内の（又は複数のトランザクションにわたって※1）異なる SQL 内で非分割表※5 を使用している場合	同一トランザクション内の非分割表検索のうち，直前に実行した SQL 文内で非分割表へのアクセスに使用したバックエンドサーバの中から，ランダムに割り当てます。 バックエンドサーバの割り当ての例については，例 3 を参照してください。
5	同一トランザクション内の（又は複数のトランザクションにわたって※1）異なる SQL 内で横分割表を使用している場合	同一トランザクション内の横分割表検索のうち，直前に実行した SQL 文内で横分割表へのアクセスに使用したバックエンドサーバの中から，ランダムに割り当てます。 バックエンドサーバの割り当ての例については，例 4 を参照してください。
6	接続したフロントエンドサーバと同じユニットにバックエンドサーバがある場合	フロントエンドサーバと同じユニット内にあるバックエンドサーバの中からランダムに割り当てます。 バックエンドサーバの割り当ての例については，例 5 を参照してください。
7	項番 1～6 に該当しない場合	バックエンドサーバをランダムに割り当てます。 バックエンドサーバの割り当ての例については，例 6 を参照してください。

注※1

バックエンドサーバ接続保持機能，ホールダブルカーソル，及び AP 単位のローカルバッファの使用時を指します。

注※2

IN EXCLUSIVE MODE 指定の LOCK TABLE 文を実行している共用表，及び FOR UPDATE 句による検索を行う共用表を除きます。

注※3

フレキシブルハッシュ分割，及び 1 バックエンドサーバ内での分割の場合を除きます。

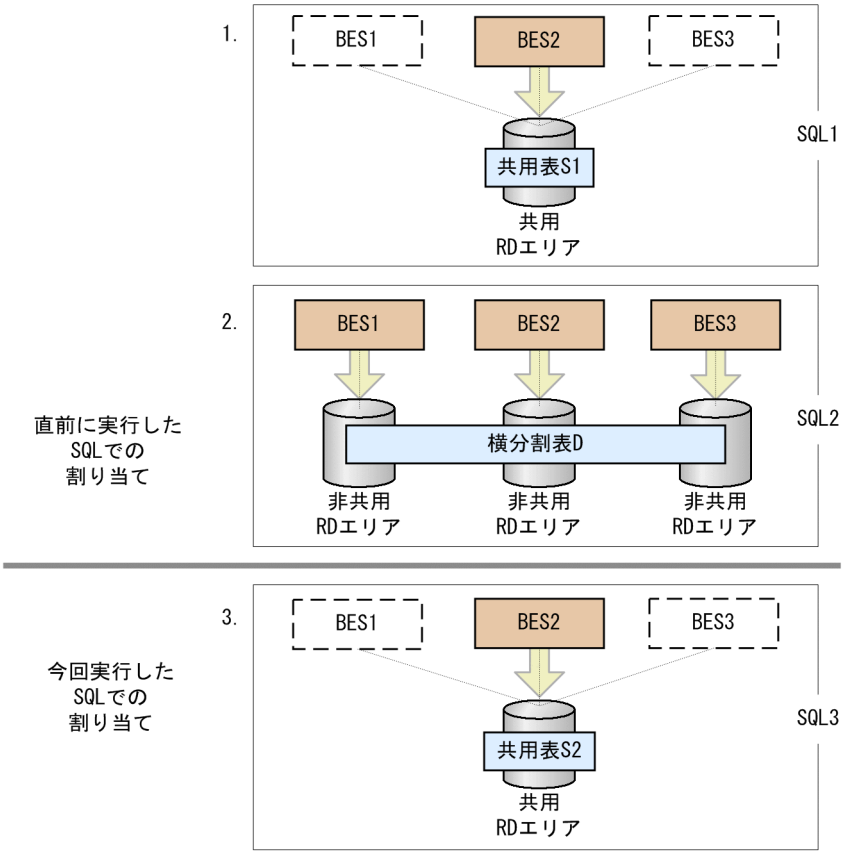
注※4

探索条件中の一つの表の列だけを指定した条件（述語，又は OR 演算された述語）です。

注※5

IN EXCLUSIVE MODE 指定の LOCK TABLE 文を実行している共用表，及び FOR UPDATE 句による検索を行う共用表を含みます。

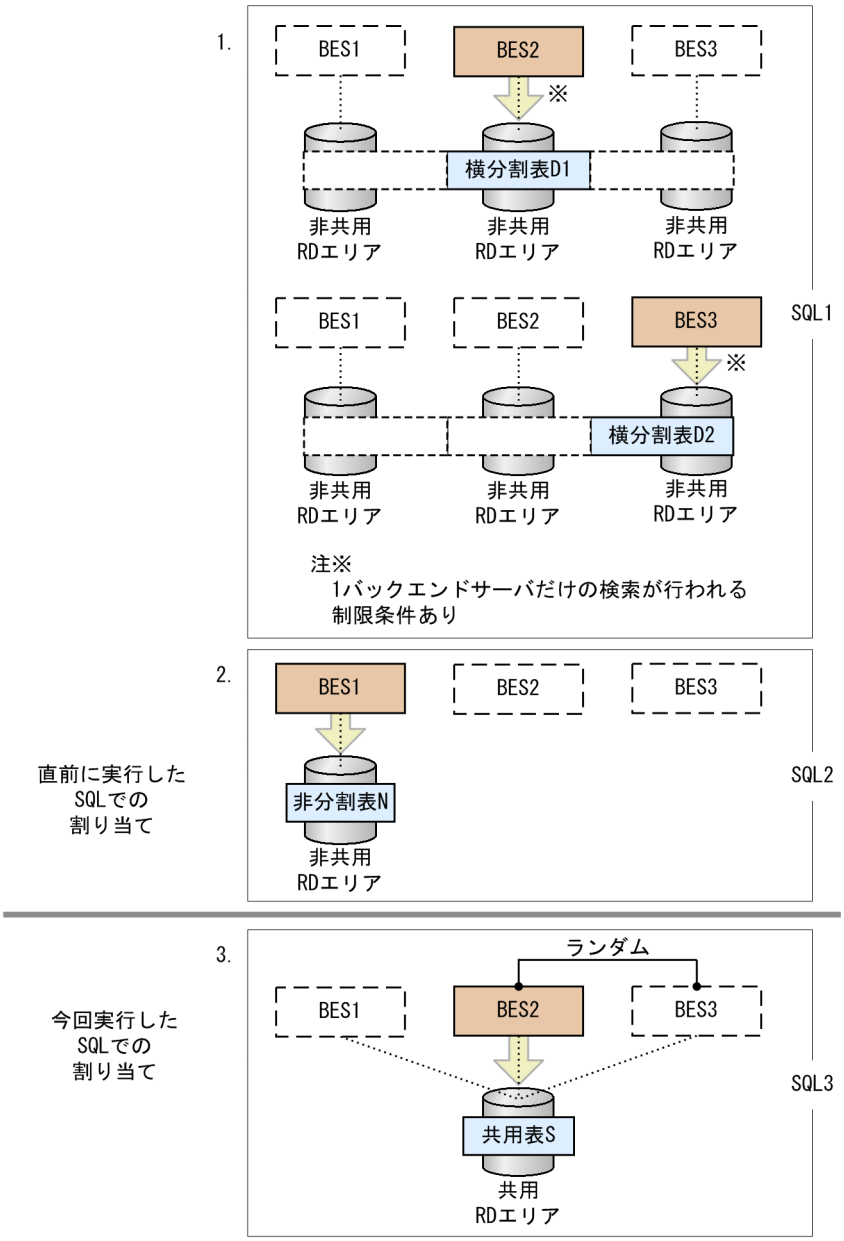
(a) 例 1



〔説明〕

1. SQL1 で共用表 S1 へのアクセスに使用したバックエンドサーバを BES2 とします。
2. SQL2 で横分割表 D へのアクセスに使用したバックエンドサーバを BES1，BES2，及び BES3 とします。
3. SQL1，SQL2 の直後に，SQL3 で共用表 S2 にアクセスする場合，共用表 S1 へのアクセスに使用した BES2 を割り当てます。

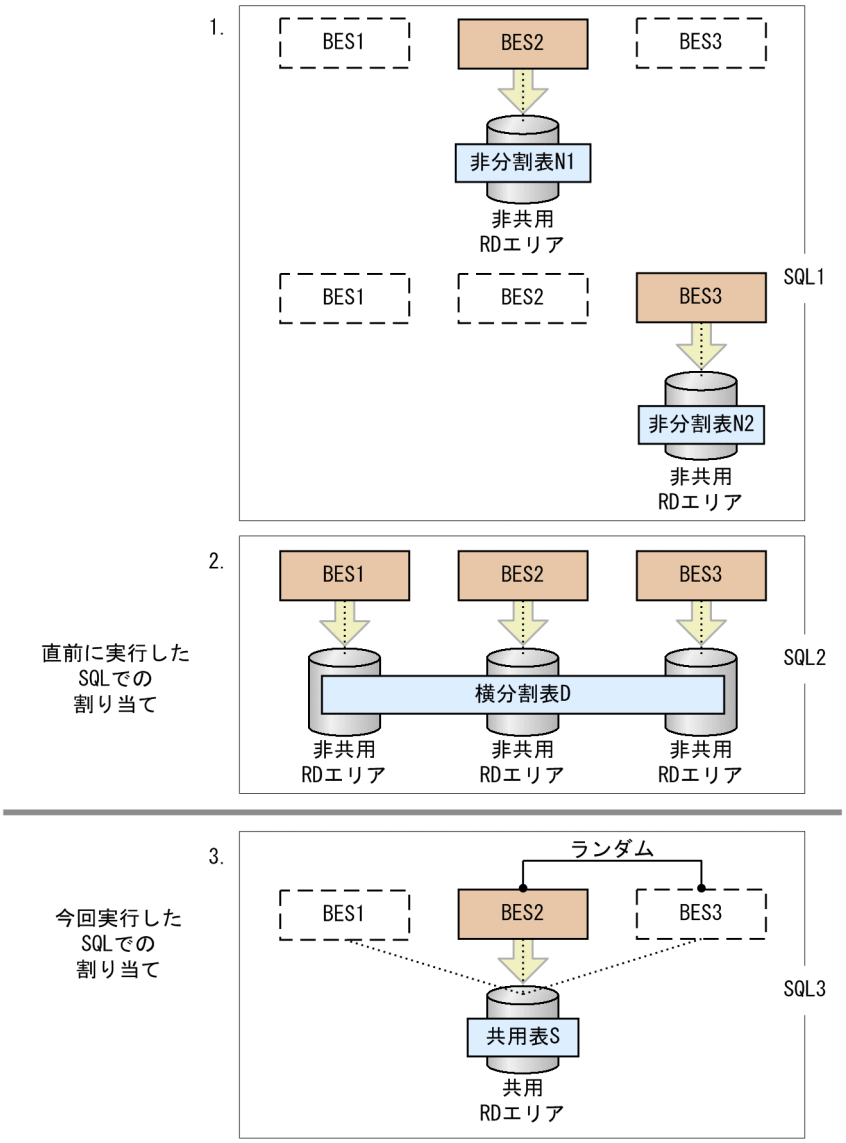
(b) 例 2



〔説明〕

1. SQL1 では，分割列に 1 バックエンドサーバだけの検索が行われる制限条件を付けて横分割表 D1 と，横分割表 D2 を検索します。横分割表 D1 へのアクセスに使用したバックエンドサーバを BES2，横分割表 D2 へのアクセスに使用したバックエンドサーバを BES3 とします。
2. SQL2 で，非分割表 N へのアクセスに使用したバックエンドサーバを BES1 とします。
3. SQL1，SQL2 の直後に，SQL3 で共用表 S にアクセスする場合，横分割表 D1，及び横分割表 D2 へのアクセスに使用したバックエンドサーバのうち，ランダムに選んだ BES2 を割り当てます（ランダムに割り当てるため，BES3 となる場合もあります）。

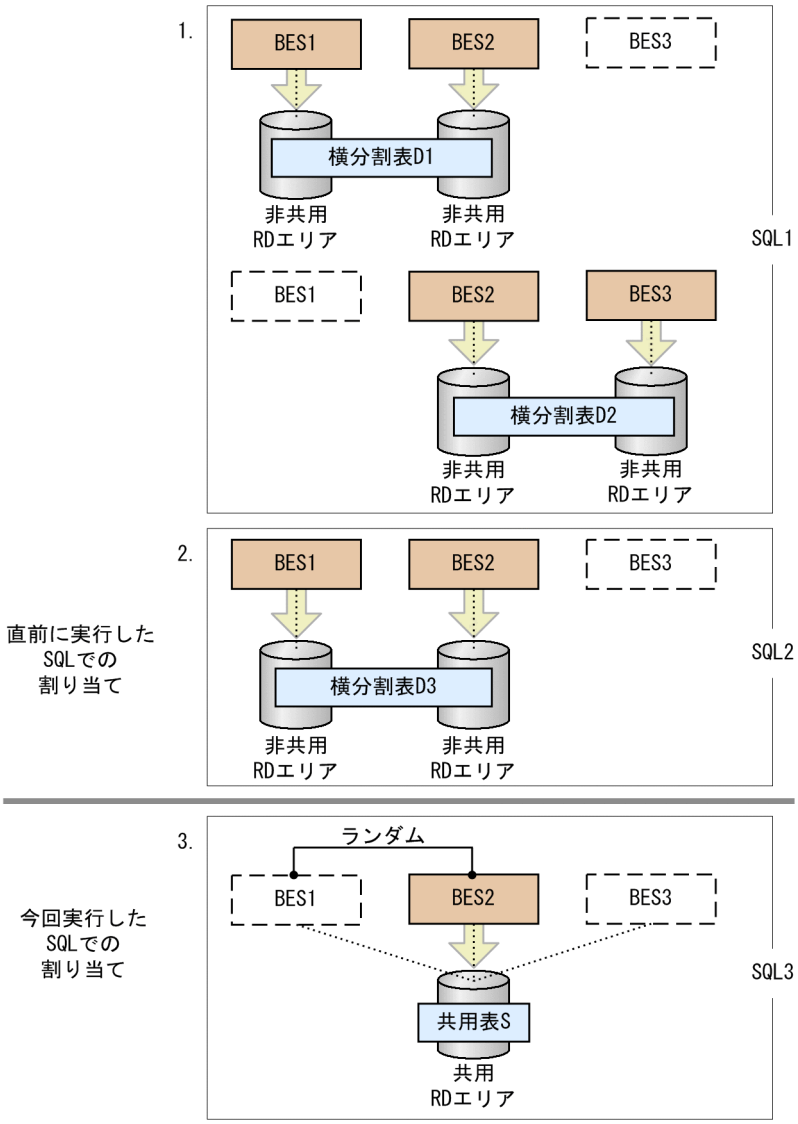
(c) 例 3



〔説明〕

1. SQL1 では，非分割表 N1 と，非分割表 N2 を検索します。非分割表 N1 へのアクセスに使用したバックエンドサーバを BES2，非分割表 N2 へのアクセスに使用したバックエンドサーバを BES3 とします。
2. SQL2 で，横分割表 D へのアクセスに使用したバックエンドサーバを，BES1，BES2，及び BES3 とします。
3. SQL1，SQL2 の直後に，SQL3 で共用表 S にアクセスする場合，非分割表 N1 と，非分割表 N2 へのアクセスに使用したバックエンドサーバのうち，ランダムに選んだ BES2 を割り当てます（ランダムに割り当てるため，BES3 となる場合もあります）。

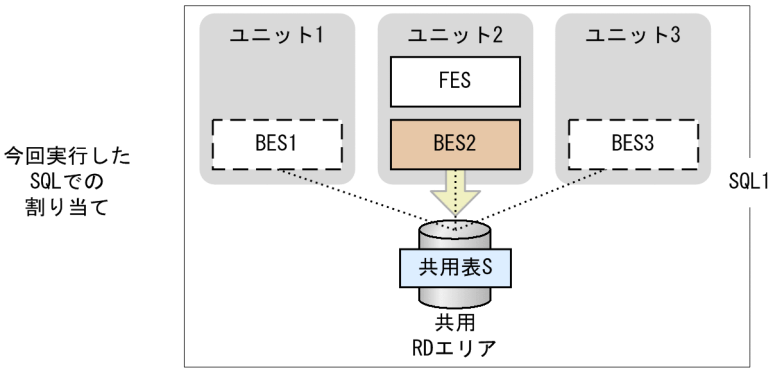
(d) 例 4



〔説明〕

1. SQL1 では，横分割表 D1 と横分割表 D2 を検索します。横分割表 D1 へのアクセスに使用したバックエンドサーバを BES1 及び BES2，横分割表 D2 へのアクセスに使用したバックエンドサーバを BES2 及び BES3 とします。
2. SQL2 で，横分割表 D3 へのアクセスに使用したバックエンドサーバを BES1 及び BES2 とします。
3. SQL1，SQL2 の直後に，SQL3 で共用表 S にアクセスする場合，横分割表 D3 へのアクセスに使用したバックエンドサーバのうち，ランダムに選んだ BES2 を割り当てます（ランダムに割り当てるため，BES1 となる場合もあります）。

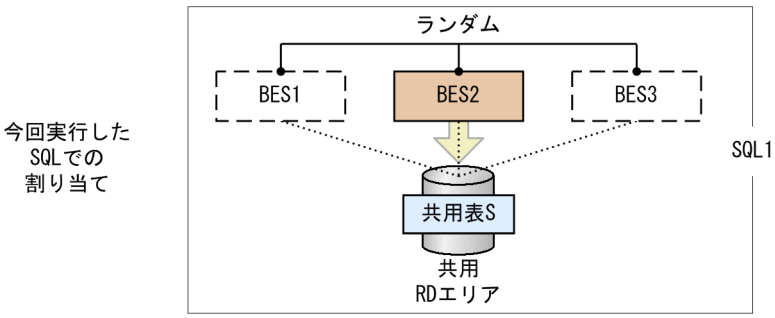
(e) 例5



［説明］

トランザクションを開始した直後に共用表 S にアクセスする場合，フロントエンドサーバと同じユニットにあるバックエンドサーバ BES2 を割り当てます。

(f) 例6



［説明］

トランザクションを開始した直後に共用表 S にアクセスする場合，ランダムに選んだバックエンドサーバ BES2 を割り当てます（ランダムに割り当てるため，BES1 又は BES3 となる場合もあります）。

(2) 1SQL 中に共用表を含む複数の表が指定されている場合

1SQL 中に指定されている表が共用表を含む複数の表の場合，共用表を検索するバックエンドサーバの割り当ては，同一 SQL 内でどのような検索を行ったかなどの条件によって決まります。なお，1SQL 中に共用表と非共用表の両方を含む場合には，非共用表にバックエンドサーバを割り当てた後，共用表にバックエンドサーバを割り当てます。

共用表を検索するバックエンドサーバの割り当て規則（1SQL 中に共用表を含む複数の表が指定されている場合）を次の表に示します。項番 1 が最も優先順位が高くなっています。

表 11-11　共用表を検索するバックエンドサーバの割り当て規則（1SQL 中に共用表を含む複数の表が指定されている場合）

項番	検索条件	割り当てられるバックエンドサーバ
1	次に示すどちらかの条件を満たす場合 • 検索対象の共用表に対して，IN EXCLUSIVE MODE 指定の LOCK TABLE 文を実行している • FOR UPDATE 句による検索を行っている	更新可能バックエンドサーバを割り当てます。
2	同一 SQL 内で共用表※1 を使用している場合	同一 SQL 内で共用表を複数使用している場合は，同じバックエンドサーバを割り当てます。すべての表が共用表の場合は，表「共用表を検索するバックエンドサーバの割り当て規則（1SQL 中に指定されている表が共用表だけの場合）」に従って割り当てます。 バックエンドサーバの割り当ての例については，例 7 を参照してください。
3	同一 SQL 内で横分割表※2 を使用し，分割列に 1 バックエンドサーバだけの検索が行われる制限条件※3 がある場合	同一 SQL 内で使用している横分割表へのアクセスに使用したバックエンドサーバの中から，ランダムに割り当てます。 バックエンドサーバの割り当ての例については，例 8 を参照してください。
4	同一 SQL 内で非分割表※4 を使用している場合	同一 SQL 内で使用している非分割表へのアクセスに使用したバックエンドサーバの中から，ランダムに割り当てます。 バックエンドサーバの割り当ての例については，例 9 を参照してください。
5	同一 SQL 内で横分割表を使用している場合	同一 SQL 内で使用している横分割表へのアクセスに使用したバックエンドサーバの中から，ランダムに割り当てます。 バックエンドサーバの割り当ての例については，例 10 を参照してください。

注※1

IN EXCLUSIVE MODE 指定の LOCK TABLE 文を実行している共用表，及び FOR UPDATE 句による検索を行う共用表を除きます。

注※2

フレキシブルハッシュ分割，及び 1 バックエンドサーバ内での分割の場合を除きます。

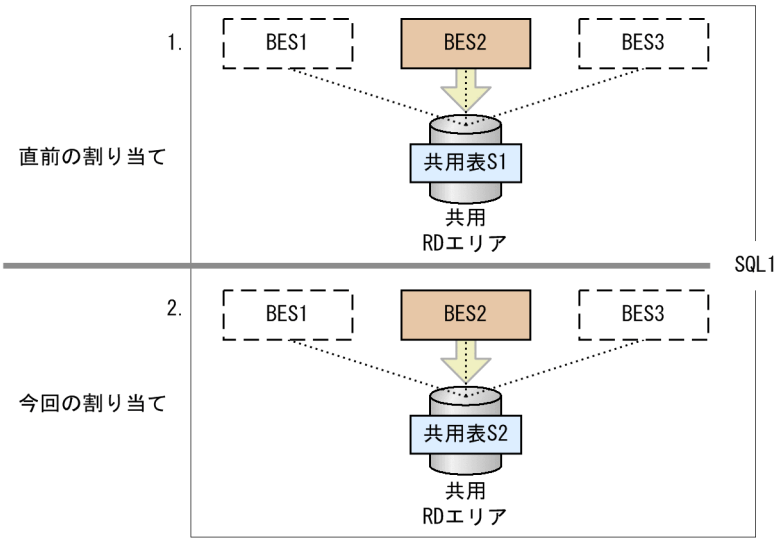
注※3

探索条件中の一つの表の列だけを指定した条件（述語，又は OR 演算された述語）です。

注※4

IN EXCLUSIVE MODE 指定の LOCK TABLE 文を実行している共用表，及び FOR UPDATE 句による検索を行う共用表を含みます。

(a) 例7

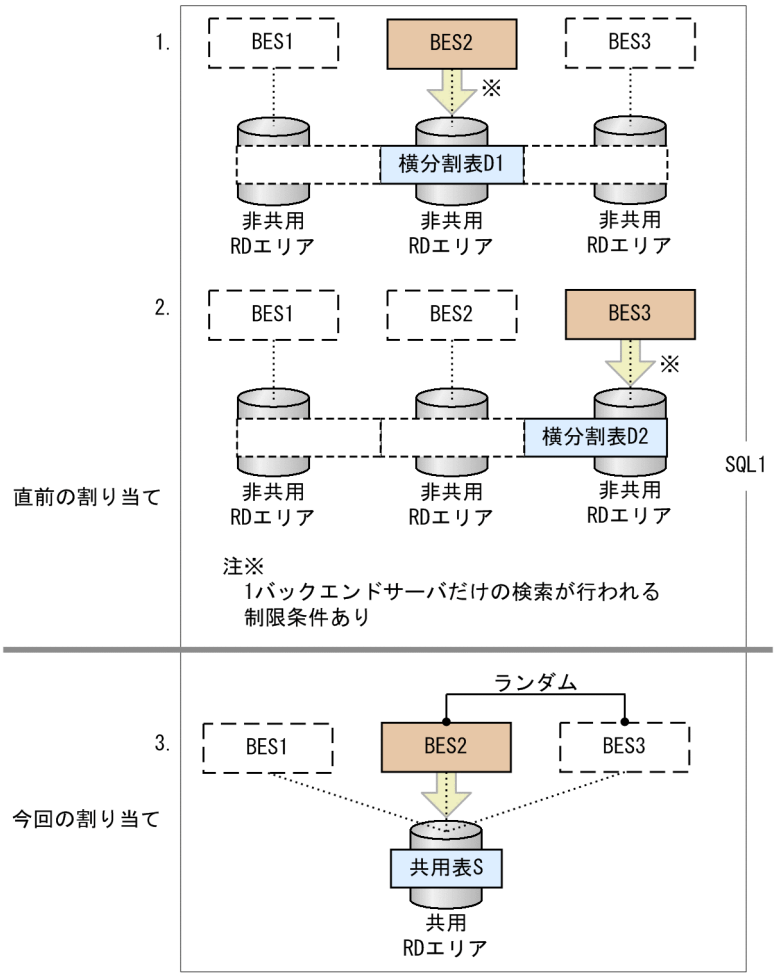


〔説明〕

同一 SQL 内で共用表 S1，及び共用表 S2 を検索します。

1. 共用表 S1 へのアクセスに使用したバックエンドサーバを BES2 とします。
2. 1.の直後に共用表 S2 にアクセスする場合，共用表 S1 へのアクセスに使用したバックエンドサーバ BES2 を割り当てます。

(b) 例 8

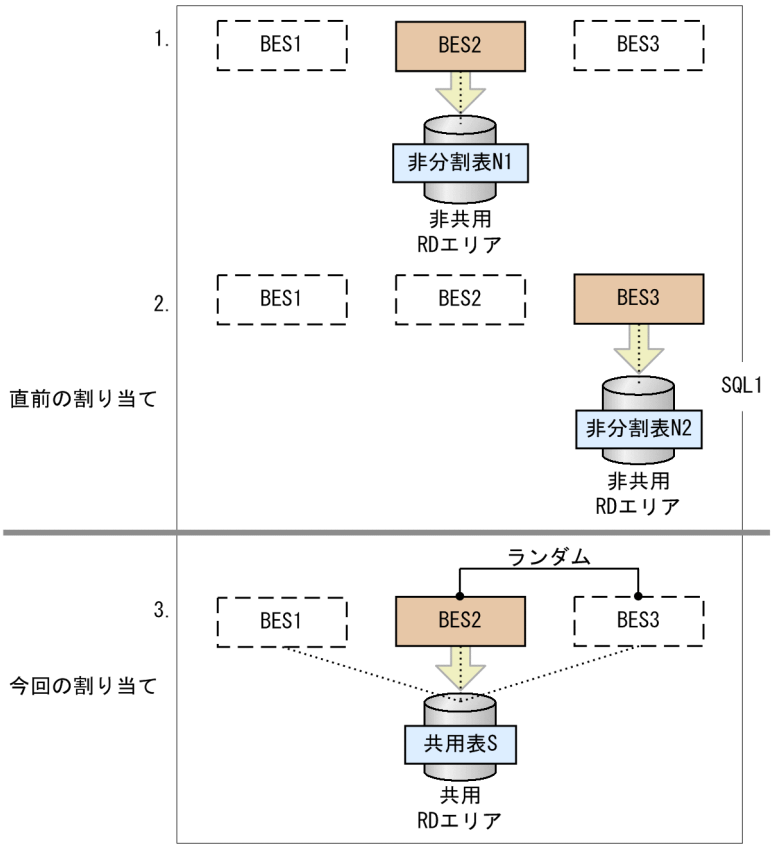


〔説明〕

同一 SQL 内で，横分割表 D1，横分割表 D2，共用表 S を検索します。

1. 分割列に 1 バックエンドサーバだけの検索が行われる制限条件を付けて横分割表 D1 を検索します。横分割表 D1 へのアクセスに使用したバックエンドサーバを BES2 とします。
2. 分割列に 1 バックエンドサーバだけの検索が行われる制限条件を付けて横分割表 D2 を検索します。横分割表 D2 へのアクセスに使用したバックエンドサーバを BES3 とします。
3. 1.，2.の直後に共用表 S にアクセスする場合，横分割表 D1，及び横分割表 D2 へのアクセスに使用したバックエンドサーバのうち，ランダムに選んだ BES2 を割り当てます（ランダムに割り当てるため，BES3 となる場合もあります）。

(c) 例 9

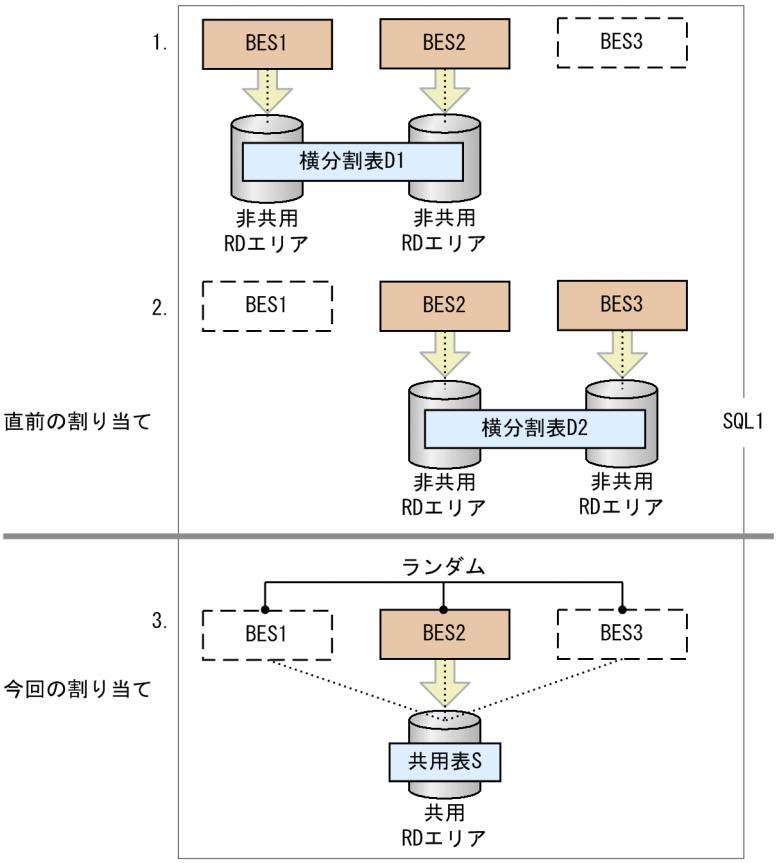


〔説明〕

同一 SQL 内で，非分割表 N1，非分割表 N2，共用表 S を検索します。

1. 非分割表 N1 へのアクセスに使用したバックエンドサーバを BES2 とします。
2. 非分割表 N2 へのアクセスに使用したバックエンドサーバを BES3 とします。
3. 1.，2.の直後に共用表 S にアクセスする場合，非分割表 N1 と非分割表 N2 へのアクセスに使用したバックエンドサーバのうち，ランダムに選んだ BES2 を割り当てます（ランダムに割り当てるため，BES3 となる場合もあります）。

(d) 例 10



〔説明〕

同一 SQL 内で，横分割表 D1，横分割表 D2，共用表 S を検索します。

1. 横分割表 D1 へのアクセスに使用したバックエンドサーバを BES1 及び BES2 とします。
2. 横分割表 D2 へのアクセスに使用したバックエンドサーバを BES2 及び BES3 とします。
3. 1.，2.の直後に共用表 S にアクセスする場合，横分割表 D1，横分割表 D2 へのアクセスに使用したバックエンドサーバのうち，ランダムに選んだ BES2 を割り当てます（ランダムに割り当てるため，BES1 又は BES3 となる場合もあります）。

11.18.6 定義系 SQL，ユティリティ，及び運用コマンド実行時の注意

定義系 SQL，ユティリティ，及び運用コマンドで共用表又は共用インデクスを対象とする場合，HiRDB が内部的に LOCK 文を発行し，全バックエンドサーバで表と処理対象の RD エリアに対して IN EXCLUSIVE MODE 指定で排他を掛けることがあります。このため，該当する RD エリア内の表やインデクスにアクセス中の業務があると，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おそれがあります。

HiRDB が内部的に LOCK 文を発行する定義系 SQL を次に示します。

- 共用表に対する CREATE TABLE，DROP TABLE，PURGE TABLE

- 共用インデクスに対する CREATE INDEX，DROP INDEX
- 共用表を含むスキーマに対する DROP SCHEMA
- 共用表に対する空き領域の再利用機能の変更（ALTER TABLE）
- 共用表に対する主キーの追加及び削除（ALTER TABLE）

HiRDB が内部的に LOCK 文を発行するユティリティを次に示します。

- 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 -k reld，rorg，ixrc，ixmk，ixor）
- 空きページ解放ユティリティ（pdreclaim）
- データベース定義ユティリティ（pddef）
- データディクショナリ搬出入ユティリティ（pdexp）
- データベース構成変更ユティリティ（pdmod -a initialize rdarea）

HiRDB が内部的に LOCK 文を発行する運用コマンドを次に示します。

- pdorend

共用 RD エリアに対して実行できないユティリティや運用コマンドについては，「共用 RD エリア使用上の制限事項」を参照してください。

11.18.7 HiRDB/シングルサーバで共用表を使用する場合

HiRDB/パラレルサーバの場合との相違点について説明します。

注意事項について

HiRDB/シングルサーバで共用表を使用する場合の注意（共用表の操作時の注意事項，共用表の制限事項，及び定義系 SQL，ユティリティ，及び運用コマンド実行時の注意）は，基本的に HiRDB/パラレルサーバと同じです。ただし，HiRDB/シングルサーバはサーバが一つだけなので，サーバ間のデッドロックは発生しません。また，HiRDB/パラレルサーバの場合の運用コマンド実行時の注意は，HiRDB/シングルサーバの場合は該当しません。

共用表及び共用インデクスの格納 RD エリアについて

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。同じユーザ用 RD エリアに混在していると，デッドロックが発生するおそれがあります（共用表を更新中は，共用表及び共用インデクスが格納されている RD エリアにも排他が掛かるため，該当する RD エリアの表又はインデクスにアクセスする業務があると排他待ちになります）。

ローカルバッファの使用について

HiRDB/シングルサーバでローカルバッファを使用して共用表及び共用インデクスを更新する場合，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しても，アボートコード Phb3008 を出力して HiRDB が異常終了することはありません。

HiRDB/シングルサーバから HiRDB/パラレルサーバへの移行について

移行する場合，HiRDB/シングルサーバのシステム内に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）を使用して HiRDB/パラレルサーバに移行しないでください。移行手順を次に示します。

1. HiRDB/シングルサーバで定義している共用表及び共用インデクスがあるかどうかを確認します。

   次に示す SQL 文（システム内に定義されている共用表の名称を確認するため，データディクショナリ表（SQL_TABLES）を検索する）を実行します。表名が出力されなければ，共用表は定義されていません。表名が出力されたら，共用表が定義されています。

   ```
   SELECT TABLE_NAME
   FROM MASTER.SQL_TABLES
   WHERE SHARED=’S’
   WITHOUT LOCK NOWAIT
   ```

2. HiRDB/シングルサーバで定義している共用表及び共用インデクスがあれば，すべて削除します。
3. データベース構成変更ユティリティ（pdmod）を使用して HiRDB/パラレルサーバに移行します。
4. HiRDB/パラレルサーバで共用 RD エリアを定義し，共用表及び共用インデクスを定義し直し，共用 RD エリアに格納します。

11.19　参照制約

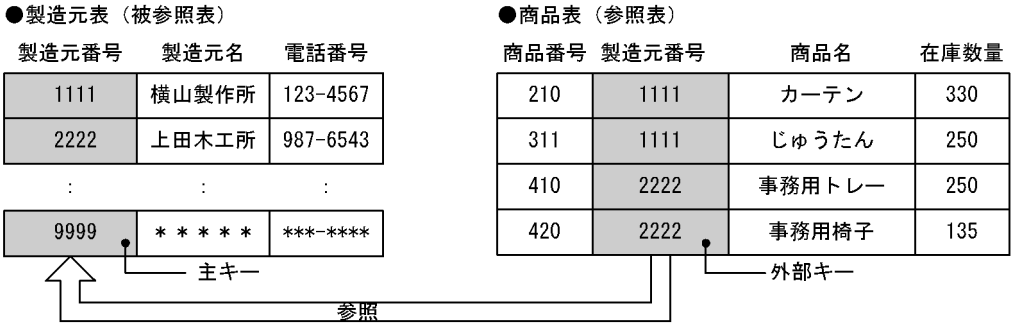
11.19.1　参照制約とは

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。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，整合性を保証できなくなる操作については「データ操作と整合性」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 11–32　被参照表と参照表の例



参照制約を定義する場合，外部キーにインデクスを定義すると処理性能が向上します。ただし，被参照表の主キーを更新しない場合は，外部キー値の更新によるインデクス更新のためのオーバヘッドがあるので，更新時の性能が悪くなることがあります。

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ので，注意してください。次のような場合に，チェックに掛かる処理時間が増加します。

- 更新する列が被参照表の主キーの場合
  被参照表の主キーを参照する参照表の外部キーの数に比例して遅くなります。
- 更新する列が参照表の外部キーの場合
  更新する列を構成列とする外部キーの数に比例して遅くなります。

11.19.2　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。

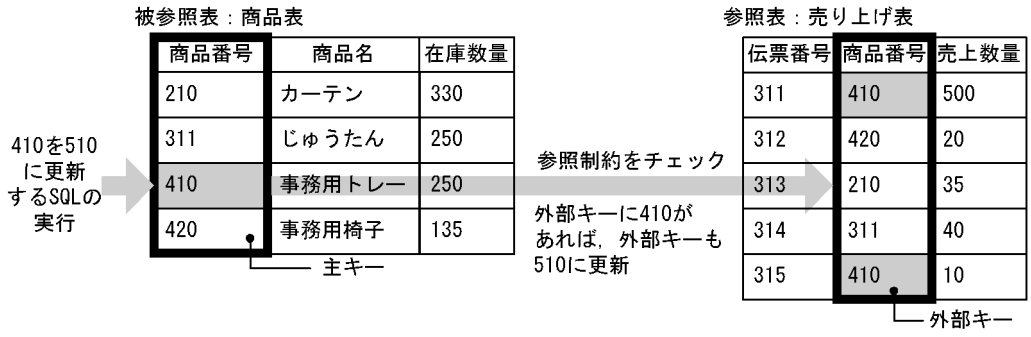
CASCADE，又は RESTRICT を指定した場合の被参照表と参照表の動作について説明します。

(1)　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。

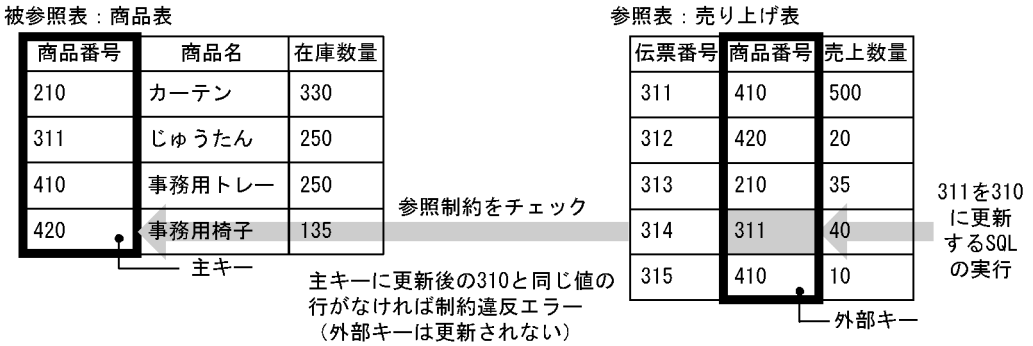
図 11–33　被参照表に更新 SQL を実行するときの動作の例（CASCADE 指定時）



〔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 11-34　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。

CASCADE を指定している場合の主キーに対する操作と参照表の動作と外部キーに対する操作と被参照表の動作を次の表に示します。

表 11-12　主キーに対する操作と参照表の動作（CASCADE 指定時）

主キーに対する操作	被参照表と参照表の行の関係	主キーに対する操作結果	参照表の動作
挿入（INSERT 文）	なし	○	動作しない
更新（UPDATE 文），削除（DELETE 文）	更新前の主キー構成列の値と同じ外部キー構成列の値を持つ行が，参照表にある	○	主キーと同じ値で更新，又は行削除
	更新前の主キー構成列の値と同じ外部キー構成列の値を持つ行が，参照表にない	○	動作しない

（凡例）

○：正常に実行されます。

表 11-13　外部キーに対する操作と被参照表の動作（CASCADE 指定時）

外部キーに対する操作	参照表と被参照表の行の関係		外部キーに対する操作結果	被参照表の動作
挿入（INSERT 文）	挿入する行の外部キー構成列の値と同じ主キー構成列の値を持つ行が被参照表にある		○	動作しない
	挿入する行の外部キー構成列の値と同じ主キー構成列の値を持つ行が被参照表にない	外部キー構成列中にナル値がある	○	
		外部キー構成列中にナル値がない	×	

外部キーに対する操作	参照表と被参照表の行の関係		外部キーに対する操作結果	被参照表の動作
更新（UPDATE 文）	更新後の外部キー構成列の値と同じ主キー構成列の値を持つ行が被参照表にある		○	動作しない
	更新後の外部キー構成列の値と同じ主キー構成列の値を持つ行が被参照表にない	外部キー構成列中にナル値がある	○	
		外部キー構成列中にナル値がない	×	
削除（DELETE 文）	なし		○	動作しない

（凡例）

○：正常に実行されます。

×：制約違反エラーとなります。

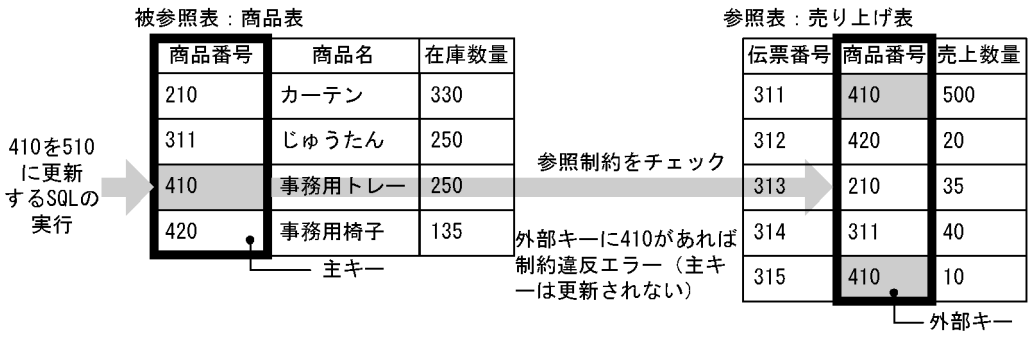
なお，CASCADE を指定すると，主キーの変更を外部キーにも反映するため，表定義時に HiRDB が内部的にトリガを生成します。参照制約動作のトリガ，及びユーザが定義するトリガとの関係については，「参照制約とトリガ」を参照してください。

(2)　RESTRICT を指定している場合

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，被参照表に更新 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 11-35　被参照表に更新 SQL を実行するときの動作の例（RESTRICT 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，参照制約違反エラーになり，主キーへの更新は実行されません。同じ値の行がなければ，被参照表への更新が実行されます。挿入及び削除も同じです。

RESTRICT を指定している場合の主キーに対する操作と被参照表及び参照表の動作を次の表に示します。外部キーに対する操作と被参照表の動作は CASCADE 指定時（表「外部キーに対する操作と被参照表の動作（CASCADE 指定時）」を参照）と同じです。

表 11-14　主キーに対する操作と被参照表及び参照表の動作

主キーに対する操作	被参照表と参照表の行の関係	主キーに対する操作結果	参照表の動作
挿入（INSERT 文）	なし	○	動作しない
更新（UPDATE 文），削除（DELETE 文）	更新前の主キー構成列の値と同じ外部キー構成列の値を持つ行が，参照表にある	×	動作しない
	更新前の主キー構成列の値と同じ外部キー構成列の値を持つ行が，参照表にない	○	

（凡例）

○：正常に実行されます。

×：制約違反エラーとなります。

(3)　被参照表，及び参照表定義時の制限事項

被参照表と参照表の表定義，表定義変更，及び表削除時の制限事項を次に示します。

(a)　表定義（CREATE TABLE）時

- 同じ外部キー構成列（並びが同じでなくてもよい）の外部キーから，同じ被参照表を参照することはできません。
- 次の場合，外部キーは定義できません。
  - WITHOUT ROLLBACK を指定した表，共用表，及び改竄防止表の場合
  - 被参照表が WITHOUT ROLLBACK を指定した表の場合
  - 共用表の場合
  - 改竄防止表の場合
  - 一時表の場合
  - 被参照表が一時表の場合
- 外部キーは，一つの表に 255 個まで定義できます。256 個以上は定義できません。
- 一つの主キーに対して，外部キーは 255 個まで定義できます。256 個以上は定義できません。
- 参照表定義時に参照できる表は，同一スキーマの表だけです。
- 次の条件をすべて満たす場合にだけ，一つの表で，同じ主キーを参照する ON UPDATE CASCADE（更新時の参照制約動作が CASCADE）を指定した参照表を定義できます。
  - 複数の外部キー構成列が重複していない

- 複数の外部キー構成列に関連する検査制約，又は参照制約を定義していない
- 外部キーの文字集合と，その外部キーから参照される表の主キーの文字集合は同じにしてください。

(b) 表定義変更（ALTER TABLE 時）

- 被参照表及び参照表に対して，DROP 句及び RENAME 句を使用した表定義変更はできません。
- 被参照表の主キー構成列，外部キー構成列に対して定義を変更する場合，次の制限があります。
  - CHANGE 句を使用したデータ型やデータ長の変更はできない
  - RENAME 句を使用した列名変更はできない
- WITH PROGRAM を指定した場合，参照表が参照する被参照表を使用する関数，手続き，及びトリガの SQL オブジェクトは無効になります。そのため，ALTER ROUTINE，ALTER PROCEDURE，又は ALTER TRIGGER で再作成する必要があります。

(c) 表削除（DROP TABLE）時

- 外部キーから参照される被参照表は削除できません。

(4) 参照制約を定義する場合の注意事項

- 被参照表と参照表間のデッドロック

  次の条件をすべて満たす場合，被参照表と参照表間でデッドロックが発生することがあります。これらの条件は参照制約動作が RESTRICT でも CASCADE でも同じです。

  - 参照表の行を更新するトランザクションと，被参照表を更新するトランザクションが異なるトランザクションで，かつ同時に実行される
  - 参照表で更新する行の主キー構成列の値と，被参照表で更新する行の外部キー構成列の値が同じである

  被参照表及び参照表を操作する場合，上記条件が重ならないようにしてください。なお，それぞれのトランザクションで，操作対象の表に対して LOCK 文の排他モードで排他制御することでデータの整合性は保証できます。ただし，同時実行性は低下します。

- SQL オブジェクト用バッファ長の容量見積もり

  参照制約動作を指定すると，HiRDB が内部的に制約条件チェックや参照制約動作を実行するトリガを生成するため，SQL オブジェクト用バッファを指定するときにそれらの SQL オブジェクトについても考慮する必要があります。SQL オブジェクト用バッファ長（pd_sql_object_cache_size）の見積もり式については，マニュアル「HiRDB システム定義」を参照してください。

- データディクショナリ LOB 用 RD エリアの容量見積もり

  参照制約動作に CASCADE を指定すると，参照制約動作を実行するトリガを HiRDB が生成します。このトリガのトリガ動作手続きの 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。そのため，参照制約動作に CASCADE を指定する場合は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，「データディクショナリ LOB 用 RD エリアの容量の見積もり」を参照してください。

- バックアップの取得

  バックアップは，被参照表が格納されている全 RD エリア，及び参照表が格納されている全 RD エリアの同期を合わせて取得してください。インナレプリカ機能を使用している場合は全 RD エリアの世代番号を合わせてバックアップを取得してください。

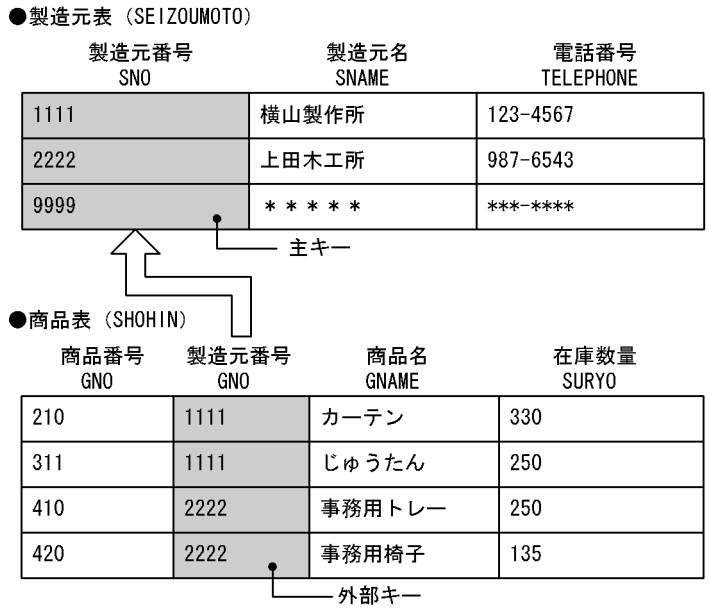
  また，バックアップ取得時点の検査保留状態によって，バックアップの取得範囲が異なります。バックアップの取得時点と取得範囲については，マニュアル「HiRDB システム運用ガイド」の「同時にバックアップを取得する必要がある RD エリア」を参照してください。

(5) 参照制約の定義例

参照制約の定義例を次に示します。

(a) 1 対 1 対応で参照制約を定義する例

被参照表と参照表が 1 対 1 の場合の定義例を次に示します。



参照制約の定義例 1

```
CREATE TABLE SEIZOUMOTO
  (SNO CHAR(4),SNAME NCHAR(6),TELEPHONE CHAR(12))
  PRIMARY KEY(SNO)  …主キーの指定
CREATE TABLE SHOHIN
  (GNO CHAR(4),SNO CHAR(4),GNAME NCHAR(10),SURYO INTEGER)
  CONSTRAINT SHOHIN_FK  …制約名の指定
  FOREIGN KEY(SNO)  …外部キーの指定
  REFERENCES SEIZOUMOTO  …被参照表名の指定
```

参照制約動作の内容

参照制約動作の指定を省略しているため，更新及び削除時は RESTRICT が仮定されます。製造元表の製造元番号（主キー）の更新，削除をする場合，商品表の製造元番号（外部キー）に対応する行があると，参照制約違反エラーとなり，製造元表の製造元番号の更新，削除は抑止されます。

参照制約の定義例２

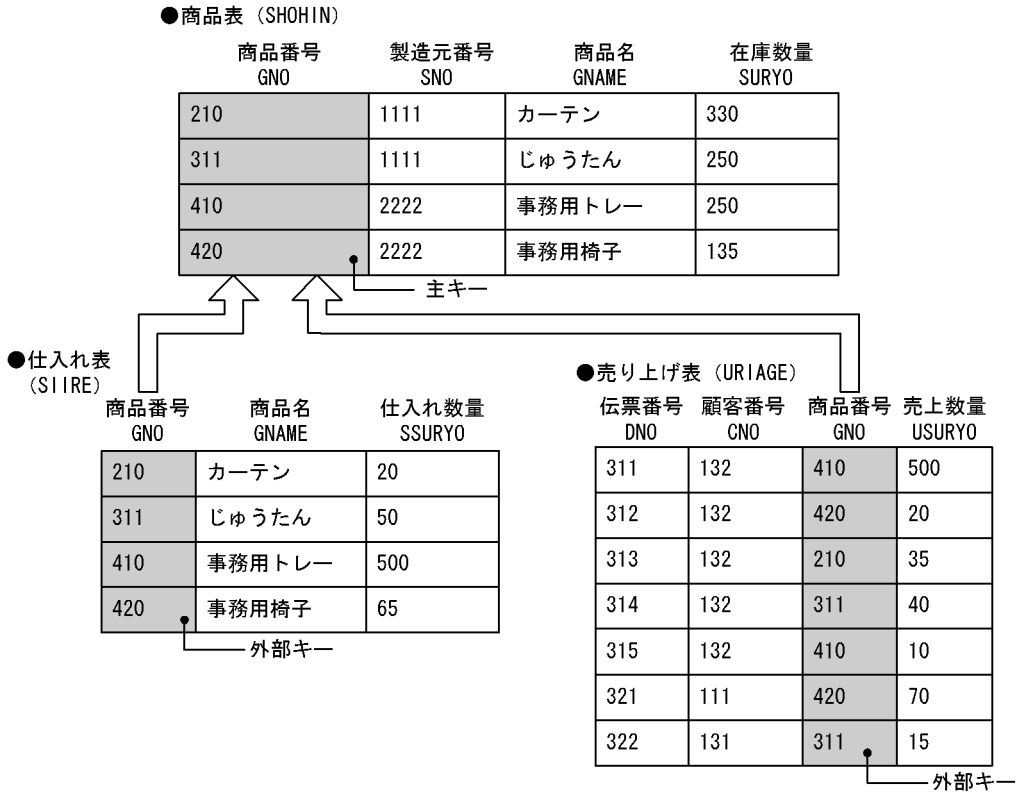
```
CREATE TABLE SEIZOUMOTO
  (SNO CHAR(4),SNAME NCHAR(6),TELEPHONE CHAR(12))
  PRIMARY KEY(SNO)  …主キーの指定
CREATE TABLE SHOHIN
  (GNO CHAR(4),SNO CHAR(4),GNAME NCHAR(10),SURYO INTEGER)
  CONSTRAINT SHOHIN_FK  …制約名の指定
  FOREIGN KEY(SNO)  …外部キーの指定
  REFERENCES SEIZOUMOTO  …被参照表名の指定
  ON UPDATE CASCADE  …更新時の参照制約動作の指定
  ON DELETE CASCADE  …削除時の参照制約動作の指定
```

参照制約動作の内容

製造元表の製造元番号（主キー）を更新した場合，対応する商品表の製造元番号（外部キー）も主キーと同じ値に更新されます。製造元表の行を削除した場合，商品表に対応する行も削除されます。

(b)　1 対 2 対応で参照制約を定義する例

被参照表が 1，参照表が 2 の場合の定義例を次に示します。



参照制約の定義例

```
CREATE TABLE SHOHIN
  (GNO CHAR(4),SNO CHAR(4),GNAME NCHAR(10),SURYO INTEGER)
  PRIMARY KEY(GNO)  …主キーの指定
CREATE TABLE SIIRE
  (GNO CHAR(4),GNAME NCHAR(10),SSURYO INTEGER)
  CONSTRAINT SIIRE_FK  …制約名の指定
  FOREIGN KEY(GNO)  …外部キーの指定
```

```
  REFERENCES SHOHIN  …被参照表名の指定
  ON UPDATE CASCADE  …更新時の参照制約動作の指定
  ON DELETE CASCADE  …削除時の参照制約動作の指定
CREATE TABLE URIAGE
  (DNO CHAR(4),CNO CHAR(4),GNO CHAR(4),USURYO INTEGER)
  CONSTRAINT URIAGE_FK  …制約名の指定
  FOREIGN KEY(GNO)  …外部キーの指定
  REFERENCES SHOHIN  …被参照表名の指定
  ON UPDATE RESTRICT  …更新時の参照制約動作の指定
  ON DELETE RESTRICT  …削除時の参照制約動作の指定
```

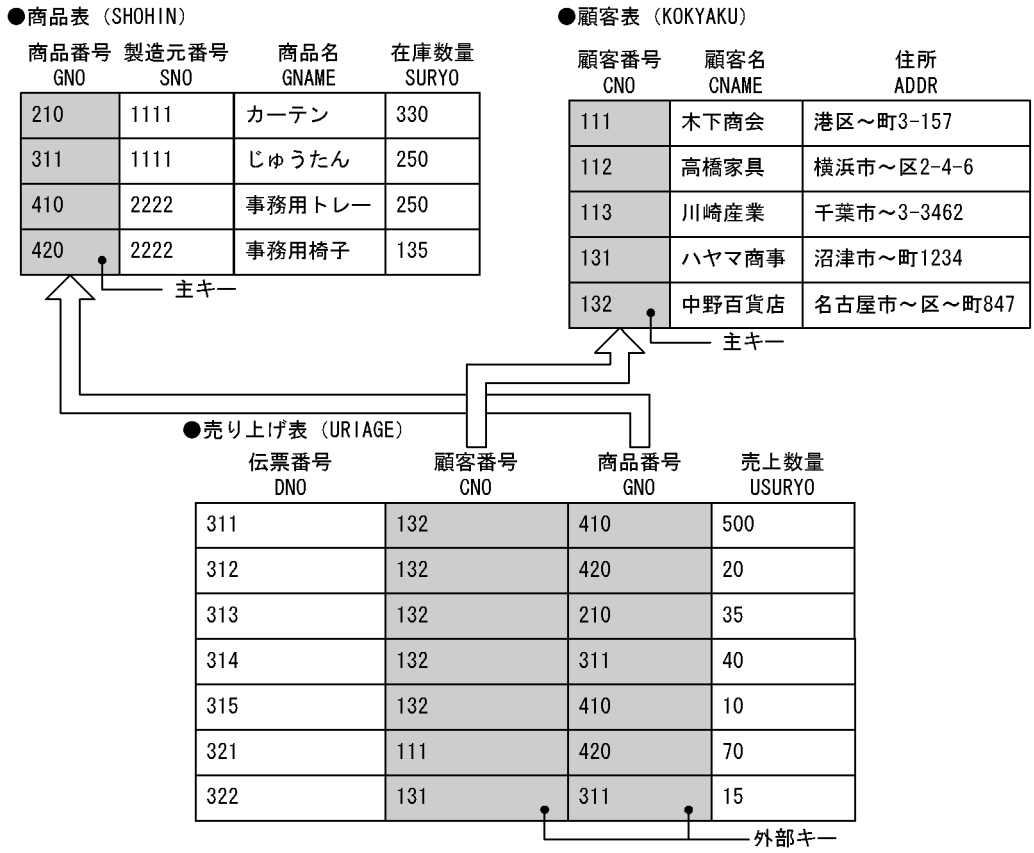
参照制約動作の内容

商品表の商品番号（主キー）を更新する場合，売り上げ表の商品番号（外部キー）に更新前の主キーと同じ値の行があると，参照制約違反エラーとなり，更新は抑止されます。売り上げ表に更新前の主キーと同じ値の行がないときは仕入れ表の対応する商品番号も主キーと同じ値に更新されます。

商品表の行を削除する場合，売り上げ表に更新前の主キーと同じ値の行があると，参照制約違反エラーとなり，削除は抑止されます。売り上げ表で更新前の主キーと同じ値の行がないときは仕入れ表の対応する行も削除されます。

(c) 2対1対応で参照制約を定義する例

被参照表が2，参照表が1の場合の定義例を次に示します。



参照制約の定義例

```
CREATE TABLE SHOHIN
  (GNO CHAR(4),SNO CHAR(4),GNAME NCHAR(10),SURYO INTEGER)
  PRIMARY KEY(GNO)  …主キーの指定
CREATE TABLE KOKYAKU
  (CNO CHAR(4),CNAME NCHAR(8),ADDR NCHAR(24))
  PRIMARY KEY(CNO)  …主キーの指定
CREATE TABLE URIAGE
  (DNO CHAR(4),CNO CHAR(4),GNO CHAR(4),USURYO INTEGER)
  CONSTRAINT URIAGE_SHOHIN_FK  …制約名の指定
  FOREIGN KEY(GNO)  …外部キーの指定
  REFERENCES SHOHIN  …被参照表名の指定
  ON UPDATE CASCADE  …更新時の参照制約動作の指定
  ON DELETE CASCADE  …削除時の参照制約動作の指定
  CONSTRAINT URIAGE_KOKYAKU_FK
  FOREIGN KEY(CNO)  …外部キーの指定
  REFERENCES  KOKYAKU  …被参照表名の指定
  ON UPDATE CASCADE  …更新時の参照制約動作の指定
  ON DELETE CASCADE  …削除時の参照制約動作の指定
```

参照制約動作の内容

商品表の商品番号（主キー）を更新する場合，売り上げ表の商品番号（外部キー）も同じ値に更新されます。商品表の行を削除する場合，売り上げ表の対応する行も削除されます。

顧客表の顧客番号（主キー）を更新する場合，売り上げ表の顧客番号（外部キー）も同じ値に更新されます。顧客表の行を削除する場合，売り上げ表の対応する行も削除されます。

11.19.3　検査保留状態

SQL やユティリティの実行などで表間の参照整合性を保証できなくなった場合，HiRDB は参照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設定してから，整合性を確認してください。

整合性が保証できなくなる操作については「データ操作と整合性」を，整合性の確認手順は「表の整合性確認手順」を参照してください。

(1)　検査保留状態の設定又は解除

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するかどうかを決めたり，又は検査保留状態を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）
- 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。

(2) 検査保留状態の管理

検査保留状態は**ディクショナリ表**と，表が格納された **RD エリアの表情報**で管理しています。ディクショナリ表では表単位，及び制約単位に検査保留状態を管理し，表情報では分割表の場合は RD エリア単位に，分割表ではない場合は表単位に検査保留状態を管理します。

検査保留状態の情報が格納されている場所と内容を次の表に示します。

表 11–15 検査保留状態の情報が格納されている場所と内容（参照制約）

格納場所			格納されている情報
ディクショナリ表	SQL_TABLES 表	CHECK_PEND 列	表単位の参照制約の検査保留状態
	SQL_REFERENTIAL_CONSTRAINTS 表	CHECK_PEND 列	制約単位の参照制約の検査保留状態
RD エリアの表情報		分割していない表の場合	表単位の参照制約又は検査制約の検査保留状態
		分割表の場合	RD エリア単位の参照制約又は検査制約の検査保留状態

(3) 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できなくなる操作を次の表に示します。なお，トリガ動作によって操作対象表にアクセスする場合，トリガ SQL 文に指定した SQL の操作可否に依存します。また，操作対象表がビュー表の場合，ビュー表の基になる実表の操作可否に依存します。

表 11–16 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
操作系 SQL	SELECT 文	対象表の検索	△※1
		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	

検査保留状態の表に対する操作			操作可否
	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	
ユティリティ	リバランスユティリティ（pdrbal）		×
	データベース再編成ユティリティ（pdrorg）	再編成	△※2

（凡例）

△：場合によっては操作できません。

×：操作できません。

注※1

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

注※2

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)」の「規則及び注意事項」を参照してください。

(4) 検査保留状態の表と参照関係がある表に対して制限される操作

次のような参照関係がある表を例に説明します。この場合，検査保留状態になるのは表 T2 と表 T3 だけです。

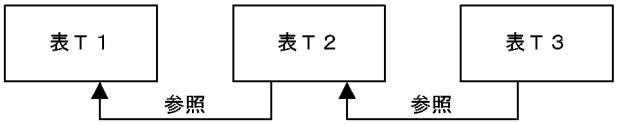


表 T2 と表 T3 が検査保留状態の場合，各表に対して制限される操作について次に示します。

(a) 表 T2 だけが検査保留状態の場合

表 T2 だけが検査保留状態の場合，各表に対して制限される操作を次の表に示します。

表 11–17 表 T2 だけが検査保留状態の場合，各表に対して制限される操作

操作対象表	制限される操作	制限の内容
表 T1	UPDATE（対象表の更新）	表 T2 に定義されている参照制約動作指定によって異なります。 • CASCADE の場合 参照制約動作の操作対象となる RD エリアの表情報が検査保留状態のときは操作できません。ただし，同値更新のときは操作できます。
	DELETE（対象表からの行削除）	

操作対象表	制限される操作	制限の内容
		• RESTRICT の場合 操作できます。参照表 T2 を参照し，整合性チェックを行います。
表 T2	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）	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
	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）	
	リバランスユティリティ（pdrbal）	操作できません。
	データベース再編成ユティリティ（pdrorg）による再編成	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。
表 T3	制限される操作はありません。INSERT 及び DELETE の場合，被参照表 T2 を参照し，整合性チェックを行います。	

(b) 表 T3 だけが検査保留状態の場合

表 T3 だけが検査保留状態の場合，各表に対して制限される操作を次の表に示します。

表 11–18 表 T3 だけが検査保留状態の場合，各表に対して制限される操作

操作対象表	制限される操作	制限の内容
表 T1	UPDATE（対象表の更新）	表 T2 及び表 T3 に定義されている参照制約動作が CASCADE で，参照制約動作の操作対象となる RD エリアの表情報が検査保留状態のときは操作できません。ただし，同値更新のときは操作できます。
	DELETE（対象表からの行削除）	
表 T2	UPDATE（対象表の更新）	表 T2 及び表 T3 に定義されている参照制約動作指定によって異なります。 • CASCADE の場合 参照制約動作の操作対象となる RD エリアの表情報が検査保留状態のときは操作できません。ただし，同値更新のときは操作できます。 • RESTRICT の場合 操作できます。参照表 T3 を参照し，整合性チェックを行います。
	DELETE（対象表からの行削除）	
表 T3	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）	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
	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）	

操作対象表	制限される操作	制限の内容
	ASSIGN LIST 文（対象表からのリスト作成）	
	リバランスユティリティ（pdrbal）	操作できません。
	データベース再編成ユティリティ（pdrorg）による再編成	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。

(c) 表 T2 及び表 T3 が検査保留状態の場合

表 T2 及び表 T3 が検査保留状態の場合，各表に対して制限される操作を次の表に示します。

表 11–19 表 T2 及び表 T3 が検査保留状態の場合，各表に対して制限される操作

操作対象表	制限される操作	制限の内容
表 T1	UPDATE（対象表の更新）	表 T2 及び表 T3 に定義されている参照制約動作が CASCADE で，参照制約動作の操作対象となる RD エリアの表情報が検査保留状態のときは操作できません。ただし，同値更新のときは操作できます。 表 T2 及び表 T3 に定義されている参照制約動作指定が RESTRICT の場合，操作できます。参照表 T2 を参照し，整合性チェックを行います。
	DELETE（対象表からの行削除）	
表 T2	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）	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
	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）	
	リバランスユティリティ（pdrbal）	操作できません。
	データベース再編成ユティリティ（pdrorg）による再編成	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。
表 T3	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）	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
	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）	

操作対象表	制限される操作	制限の内容
	リバランスユティリティ（pdrbal）	操作できません。
	データベース再編成ユティリティ（pdrorg）による再編成	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。

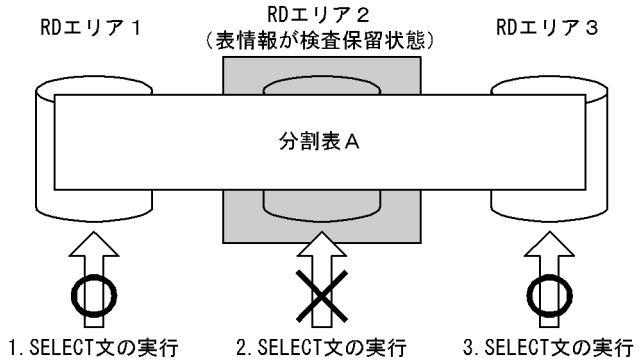
(5) 分割表又はインナレプリカ機能を使用している場合

RD エリア単位に検査保留状態を管理しているため，分割表やインナレプリカ機能を使用していて，実際に操作する RD エリアの表情報が検査保留状態の場合，その分割表や世代に対する操作は制限されることがあります。それぞれについて説明します。

(a) 分割表の場合

分割表で，データを格納している一部の RD エリアが検査保留状態の場合の例を次の図に示します。

図 11-36 分割表で，RD エリア単位に検査保留状態を管理する場合のデータ操作可否



［説明］

分割表 A に対して SELECT 文を実行する場合，実際に操作するデータが RD エリア 2（表情報が検査保留状態）にあると，SELECT 文はエラーになります。RD エリア 1 及び 3 にあるデータに対する操作の場合，SELECT 文は正常に実行できます。

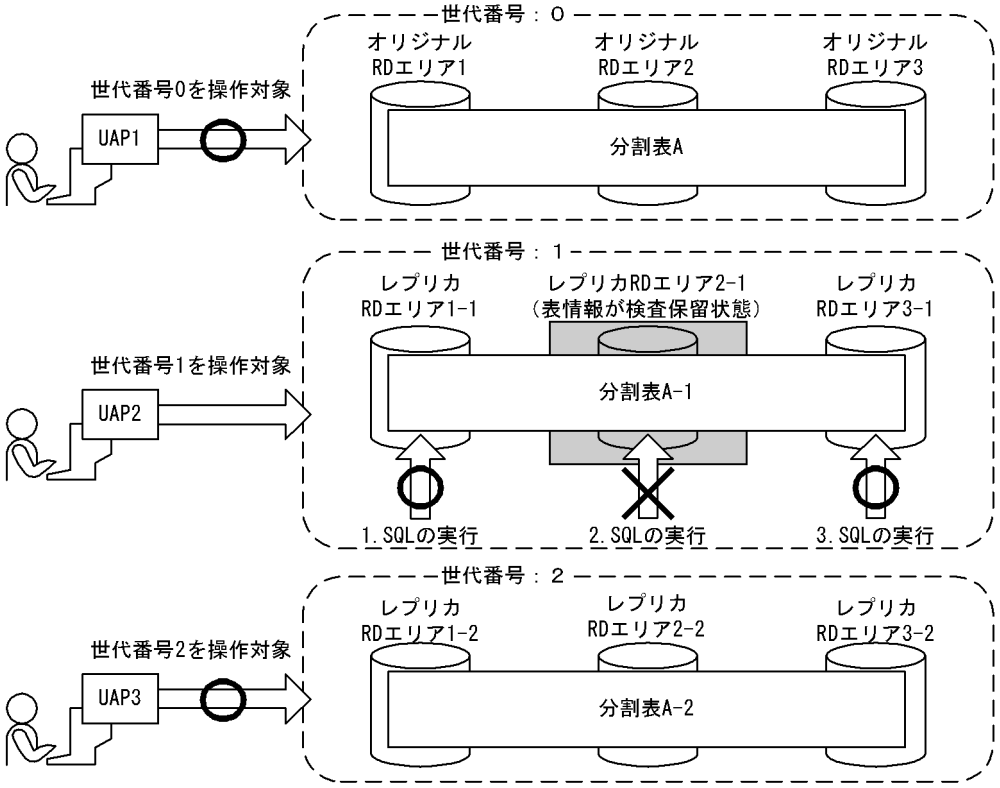
分割表の場合の注意事項

pd_check_pending オペランドに USE を指定していて，参照表のデータを分割格納している RD エリアを再初期化する場合，再初期化後に整合性チェックユティリティで表単位の整合性チェックを実行してください。

(b) インナレプリカ機能を使用している場合

インナレプリカ機能を使用していて，ある世代の一部の RD エリアが検査保留状態の場合の例を次の図に示します。

図 11-37　インナレプリカ機能を使用していて，RD エリア単位に検査保留状態を管理する場合のデータ操作可否



［説明］

世代番号 1（RD エリアの表情報が検査保留状態の RD エリアを含む世代）に対する操作をする場合で，実際に操作するデータがレプリカ RD エリア 2-1 にあるとき，その SQL はエラーになります。

(6)　検査保留状態を使用する場合の注意

- pd_check_pending オペランドの指定値を NOUSE から USE に変更した場合，整合性チェックユティリティを使用して，参照表の整合性を確認する必要があります。確認手順については，「表の整合性確認手順」を参照してください。
- pd_check_pending オペランドに USE を指定していて，参照整合性を保証できなくなる操作をした場合でも，RD エリアの状態によっては検査保留状態を設定できないことがあります。このため，pd_check_pending オペランドの指定値を NOUSE から USE に変更すると，検査保留状態を使用していない場合は正常だった操作がエラーになることがあります。PURGE TABLE 文，又は ALTER TABLE（CHANGE RDAREA）実行時，検査保留状態を設定できる RD エリアの状態を次に示します。

  オープン契機が INITIAL の場合

  - RD エリアが閉塞なし，オープン状態のとき
  - RD エリアが更新可能バックアップ閉塞，かつオープン状態のとき
  - RD エリアがオンライン再編成閉塞，かつオープン状態のとき
  - RD エリアが同期化閉塞，かつオープン状態のときは，閉塞解除後，設定可能

オープン契機が DEFER 又は SCHEDULE の場合

- RD エリアが閉塞なしのとき
- RD エリアが更新可能バックアップ閉塞のとき
- RD エリアがオンライン再編成閉塞のとき
- RD エリアが同期化閉塞のときは，閉塞解除後，設定可能

ユティリティ実行時，検査保留状態を設定できる RD エリアについては，マニュアル「HiRDB コマンドリファレンス」の「コマンド実行時の RD エリアの状態」の「検査保留状態の設定可否」を参照してください。

- pd_check_pending オペランドに USE を指定する場合，検査保留状態に設定される参照表及び RD エリアに対して排他が掛かるため，ユティリティ及び SQL 実行時の排他資源が検査保留状態を使用しない場合とは異なります。

11.19.4 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，次の表に示す操作をした場合，整合性を保証できなくなることがあります。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。

表 11-20 整合性を保証できなくなる場合の，被参照表に対する操作とデータ不整合の発生条件

表又は RD エリアに対する操作		データ不整合の発生条件
データベース作成ユティリティ（pdload）	作成モード（-d オプション指定）のデータロード	データロードした主キー構成列中に，参照表の外部キー構成列と同じ値を持つ行を含まない場合
データベース再編成ユティリティ（pdrorg）	リロード（-k reld 指定）	リロードした主キー構成列中に，参照表の外部キー構成列と同じ値を持つ行を含まない場合
	再編成（-k rorg 指定）	UOC を使用して，参照表の外部キー構成列の値と同じ値の行を削除した場合
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（initialize rdarea）	参照表が，再初期化した RD エリアと異なる RD エリアに格納されている場合
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）		参照関係がある表に対して，更新可能なオンライン再編成運用中※に，カレントデータベースのレプリカ RD エリアとオリジナル RD エリアで次に示す（1）～（4）のどれかの操作をした場合（追い付き反映処理後にデータ不整合になる） （1）次の順序で操作した場合 1.レプリカ RD エリアの参照表にデータを挿入する 2.オリジナル RD エリアの披参照表で，1.で挿入した外部キーと同じ値の行を削除する

表又は RD エリアに対する操作	データ不整合の発生条件
	（2）次の順序で操作した場合 1.レプリカ RD エリアの参照表の外部キーのデータを更新する 2.オリジナル RD エリアの披参照表で，1.で更新した外部キーと同じ値の行を削除する （3）次の順序で操作した場合 1.レプリカ RD エリアの披参照表のデータを削除する 2.オリジナル RD エリアの参照表に，1.で削除した主キーと同じ値の行を挿入する （4）次の順序で操作した場合 1.レプリカ RD エリアの披参照表のデータを更新する 2.オリジナル RD エリアの参照表に 1.で更新する前の主キーと同じ値の行を挿入する
PURGE TABLE 文	参照表にデータが存在する場合
ALTER TABLE による表の分割格納条件の変更	RD エリアの分割や統合の結果，参照表の外部キー構成列と同じ値を持つ行を含まない場合

注※

更新可能なオンライン再編成の運用については，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。

表 11-21　整合性を保証できなくなる場合の，参照表に対する操作とデータ不整合の発生条件

表又は RD エリアに対する操作		データ不整合の発生条件
データベース作成ユティリティ（pdload）	データロード	データロードした外部キー構成列中に，被参照表の主キー構成列と同じ値を持つ行がない場合
データベース再編成ユティリティ（pdrorg）	リロード（-k reld 指定）	リロードした外部キー構成列中に，被参照表の主キー構成列と同じ値を持つ行がない場合
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）		参照関係がある表に対して，更新可能なオンライン再編成運用中※に，カレントデータベースのレプリカ RD エリアとオリジナル RD エリアで次に示す（1）～（5）のどれかの操作をした場合（追い付き反映処理後にデータ不整合になる） （1）次の順序で操作した場合 1.レプリカ RD エリアの参照表にデータを挿入する 2.オリジナル RD エリアの披参照表で，1.で挿入した外部キーと同じ値の行を削除する （2）次の順序で操作した場合 1.レプリカ RD エリアの参照表の外部キーのデータを更新する 2.オリジナル RD エリアの披参照表で，1.で更新した外部キーと同じ値の行を削除する

表又は RD エリアに対する操作	データ不整合の発生条件
	（3）次の順序で操作した場合 1.レプリカ RD エリアの披参照表のデータを削除する 2.オリジナル RD エリアの参照表に，1.で削除した主キーと同じ値の行を挿入する （4）次の順序で操作した場合 1.レプリカ RD エリアの披参照表のデータを更新する 2.オリジナル RD エリアの参照表に 1.で更新する前の主キーと同じ値の行を挿入する （5）次の操作をした場合 レプリカ RD エリアの参照表に，データベース作成ユティリティ（pdload）で，データ不整合となる操作をしたとき

注※

更新可能なオンライン再編成の運用については，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。

(1) 操作対象の表が分割表の場合

操作対象の表が分割表で，表中に不整合データがある場合，ユティリティの実行によって，不整合データが，格納されている RD エリアを移動することがあります。例えば，RD エリア 1，2，及び 3 に分割格納されている表で，不整合データが RD エリア 1 にあるとき，ユティリティの実行によって，不整合データが RD エリア 3 に移動することがあります。不整合データの RD エリアの移動が発生する条件を次の表に示します。

表 11–22 操作対象が分割表の場合，表中の不整合データが RD エリアを移動する条件

表又は RD エリアに対する操作		表中の不整合データが RD エリアを移動する条件
データベース再編成ユティリティ（pdrorg）	再編成（-k rorg 指定）	フレキシブルハッシュ分割表，又は第 2 次元分割列がフレキシブルハッシュ分割のマトリクス分割表に対して次の手順で操作する場合 1. RD エリア単位でデータロードをする 2. HiRDB/シングルサーバの場合，表単位に再編成を実行する※ HiRDB/パラレルサーバの場合，-g オプションを指定して表単位に再編成を実行する※
リバランスユティリティ（pdrbal）		不整合データがある表に対して，RD エリアを追加してリバランスユティリティ（pdrbal）を実行する場合※

注※

pd_check_pending オペランドに USE を指定していて，操作対象表が検査保留状態の場合は実行できません。

(2)　データ不整合が発生する可能性があるそのほかの条件

次の条件をすべて満たす場合も，データ不整合が発生することがあるため，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，「表の整合性確認手順」を参照してください。これらの条件は参照制約動作が RESTRICT でも CASCADE でも同じです。

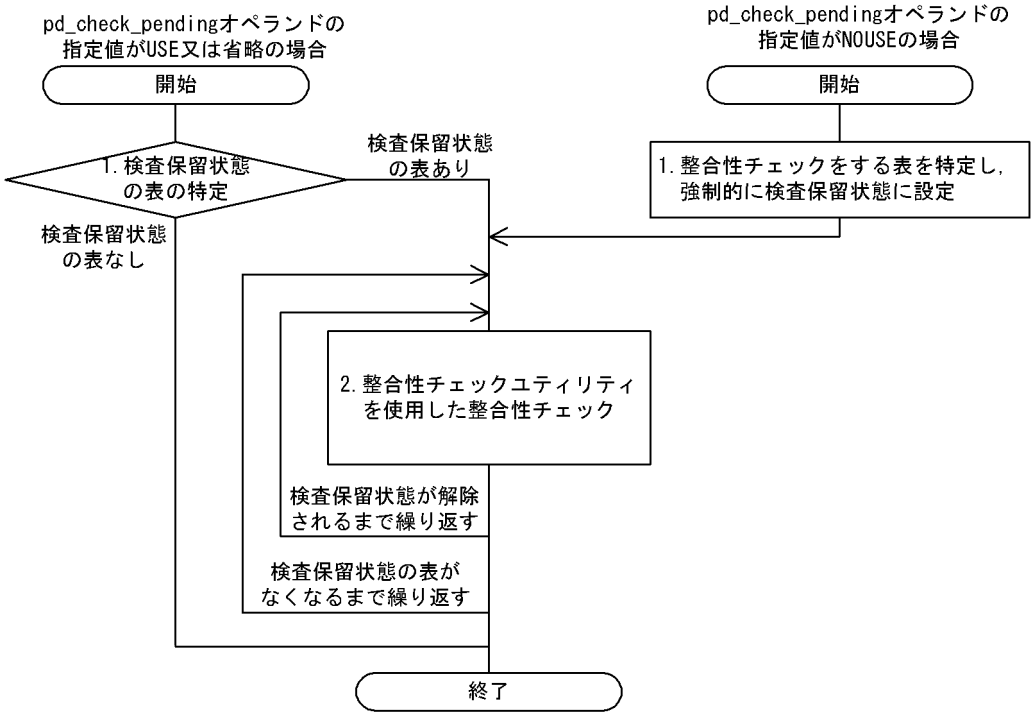
- 参照表の行を削除するトランザクションと，被参照表を更新又は削除するトランザクションが異なるトランザクションで，かつ同時に実行される
- 参照表で削除する行の主キー構成列の値と，被参照表で更新又は削除する行の外部キー構成列の値が同じである
- 被参照表の行を更新又は削除するトランザクションをコミットし，参照表の行を削除するトランザクションをロールバックする

被参照表及び参照表を操作する場合，上記条件が重ならないようにしてください。なお，それぞれのトランザクションで，操作対象の表に対して LOCK 文の共用モード又は排他モードで排他制御することでデータの整合性は保証できます。ただし，同時実行性は低下します。

11.19.5　表の整合性確認手順

データの整合性確認手順の概要を次の図に示します。

図 11-38　データの整合性確認手順の概要（参照制約）



pd_check_pending オペランドの指定値が USE 又は省略の場合

1. 検査保留状態の表の特定

ディクショナリ表の SQL_TABLES 表を検索して，検査保留状態の表名を検出します。

```
SELECT TABLE_SCHEMA, TABLE_NAME FROM MASTER.SQL_TABLES
    WHERE CHECK_PEND = 'C' OR CHECK_PEND2 = 'C'
```

検索結果には，検査保留状態の表の所有者と検査保留状態の表名が返されます。検索結果が 0 行の場合，検査保留状態の表はありません。

なお，インナレプリカ機能使用時に，各世代の表が検査保留状態かどうかを確認するためには，状態解析ユティリティ（pddbst）を使用します。

2. 整合性チェックユティリティを使用した整合性チェック

整合性チェックユティリティで表単位の整合性チェックを実行し，制約違反データがあれば修正します。検査保留状態の表がなくなるまで整合性チェックを繰り返し，なくなれば終了です。整合性チェックユティリティを使用した整合性確認手順については，「検査保留状態を使用する場合の整合性確認手順（参照制約）」を参照してください。

pd_check_pending オペランドの指定値が NOUSE の場合

1. 整合性チェックをする表を特定し，強制的に検査保留状態に設定

整合性チェックをする表を特定するために，次のことを確認します。

- 参照整合性を保証できなくなる操作をした表を被参照表とする参照表が存在するかどうか
- 参照整合性を保証できなくなる操作をした表に参照制約が定義されているかどうか

これらを確認する SQL の実行例を次に示します。

```
SELECT N_PARENTS, N_CHILDREN FROM MASTER.SQL_TABLES
    WHERE TABLE_SCHEMA = '対象表の所有者名' AND TABLE_NAME = '対象表の表名'
```

次の検索結果が返されます。

- 対象表に定義した外部キーの数
- 対象表に定義した主キーを参照する外部キーの数

N_PARENTS がナル値の場合，対象表に参照制約は定義されていません。

N_CHILDREN がナル値の場合，対象表を被参照表とする参照表は存在しません。

N_CHILDREN がナル値以外の場合，次に示す SQL を実行し，対象表を参照する参照表の表名を確認してください。

```
SELECT TABLE_SCHEMA, TABLE_NAME, CONSTRAINT_NAME
    FROM MASTER.SQL_REFERENTIAL_CONSTRAINTS
    WHERE R_OWNER = '対象表の所有者名' AND R_TABLE_NAME = '対象表の表名'
```

検索結果には，対象表を被参照表とする参照表の所有者名，表名及び参照制約の制約名が返されます。検索結果が 0 行の場合，対象表を被参照表とする参照表はありません。

表を特定したら，整合性チェックユティリティを使用して，その表を強制的に検査保留状態に設定します（検査保留状態でない表は，整合性チェックユティリティでチェックできません）。

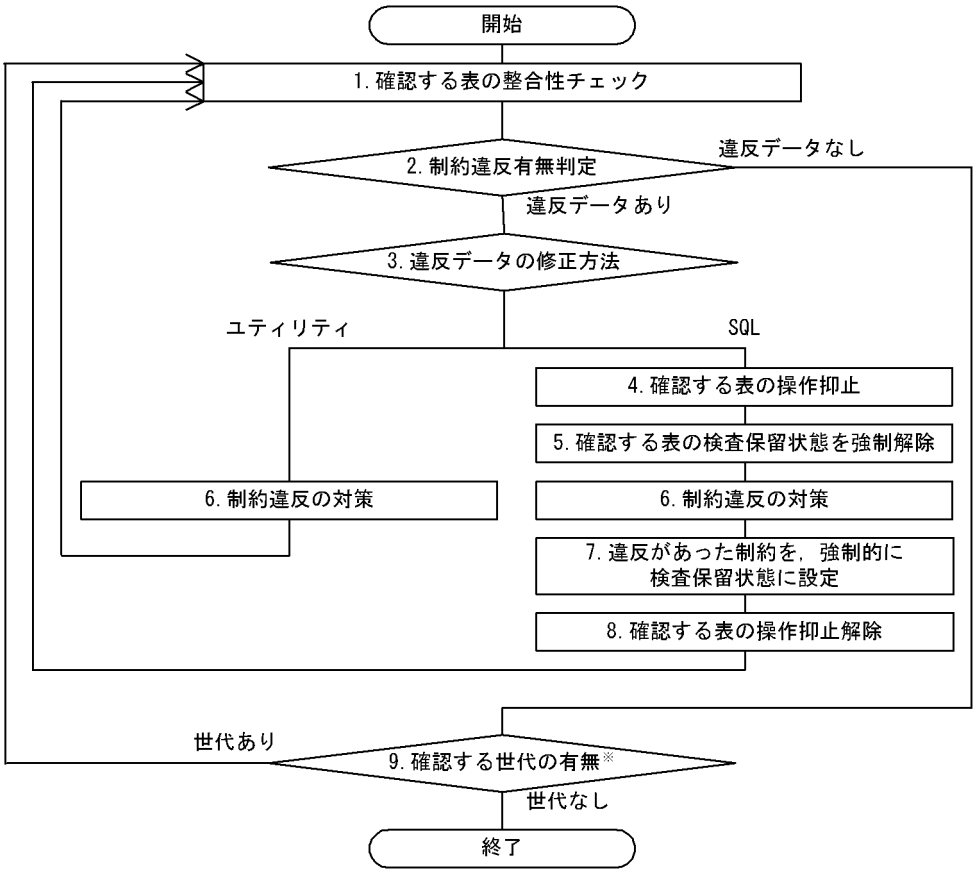
2. 整合性チェックユティリティを使用した整合性チェック

pd_check_pending オペランドの指定値が USE 又は省略の場合の手順 2.と同じです。整合性チェックユティリティを使用した整合性確認手順については，「検査保留状態を使用しない場合の整合性確認手順」を参照してください。

(1) 検査保留状態を使用する場合の整合性確認手順（参照制約）

pd_check_pending オペランドの指定値が USE 又は省略の場合の，整合性チェックユティリティを使用した整合性確認手順を次の図に示します。

図 11-39 検査保留状態を使用する場合の整合性確認手順（参照制約）



注※ 次の場合は不要です。
・インナレプリカ機能を使用していない
・インナレプリカ機能を使用していて，全世代に整合性チェックを実行する場合

1. 確認する表の整合性チェック

表単位，又は制約単位に整合性チェックをします。

インナレプリカ機能を使用している場合，確認する表の世代番号を指定します。インナレプリカ機能を使用していない場合，又は全世代に整合性チェックを実行する場合，世代番号の指定は不要です。

2. 制約違反有無判定

手順 1.の整合性チェック結果で，制約違反データの有無を判定します。

3. 違反データの修正方法

違反データの修正をユティリティで行うか，SQL で行うかを選択します。ユティリティで行う場合，手順 6.へ進んでください。

4. 確認する表の操作抑止

整合性が保証できない表を使用する業務の運用を停止します。

5. 確認する表の検査保留状態を強制解除

制約違反の対策をするため，検査保留状態を強制解除します。

6. 制約違反の対策

ユティリティで修正する場合

対策方法を次に示します。対策後，手順 1.に戻り，整合性チェックを実行し，違反データがないことを確認し，終了します。

条件	対策方法
主キーに必要なデータがない場合	データベース作成ユティリティ（pdload）の追加モードで正しいデータをロードします。
外部キーに制約違反データがある場合	• データベース作成ユティリティ（pdload）の作成モードで正しいデータをロードします。 • データベース再編成ユティリティ（pdrorg）の UOC を使用して不要なデータを削除します。

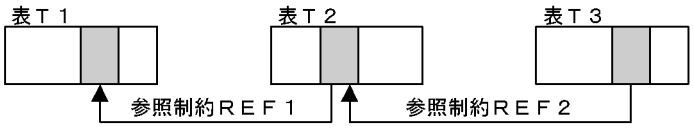
SQL で修正する場合

対策方法を次に示します。対策後，手順 7.に進みます。

条件	対策方法
主キーに必要なデータがない場合	主キーに必要なデータを INSERT 文で挿入します※1。又は，被参照表の既存のデータを UPDATE 文で更新します※2。
外部キーに制約違反データがある場合	外部キーの制約違反データを DELETE 文で削除，又は，UPDATE 文で正しい値に更新します※1。

注※1

外部キーが主キーでもあり，対策を行う表を被参照表とする参照表が存在する場合，修正順序に注意が必要です。例えば，次のような参照関係があるとします。



●REF1 の制約違反の対策をする場合の注意

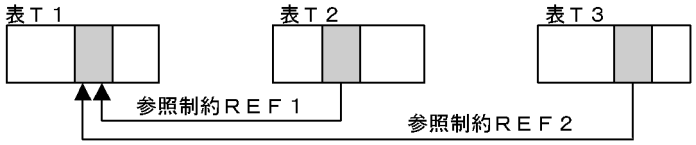
表 T2 のデータを DELETE 文で修正する場合，REF2 で ON DELETE RESTRICT を指定しているときは，対応する表 T3 のデータを先に削除後，表 T2 のデータを削除してください。また，UPDATE 文で修正する場合，REF2 で ON UPDATE RESTRICT を指定しているときは，更新前のデータに対応する表 T3 のデータを削除後，表 T2 のデータを更新してください。

●REF2 の制約違反の対策をする場合の注意

表 T2 のデータを INSERT 文で修正する場合，表 T1 に挿入対象のデータが存在するかどうか確認します。存在しないときは，先に表 T1 にデータを挿入後，表 T2 にデータを挿入してください。また，UPDATE 文で修正する場合，更新後のデータが表 T1 に存在するかどうか確認します。存在しないときは，先に表 T1 にデータを挿入後，表 T2 のデータを更新してください。

注※ 2

対策をする制約とは別の制約で，その表を被参照表とする参照表が存在する場合，修正順序に注意が必要です。例えば，次のような参照関係があるとします。



●REF1 の制約違反の対策をする場合の注意

T1 のデータを UPDATE 文で修正する場合，REF2 で ON UPDATE RESTRICT を指定しているときは，更新前のデータに対応する T3 のデータを削除後，T2 のデータを更新してください。

7. 違反があった制約を，強制的に検査保留状態に設定

整合性チェックユティリティを制約単位で実行し，対策をした制約を強制的に検査保留状態に設定します。

8. 確認する表の操作抑止解除

運用を停止していた業務を再開します。手順 1.に戻り，整合性チェックを実行し，違反データがないことを確認します。

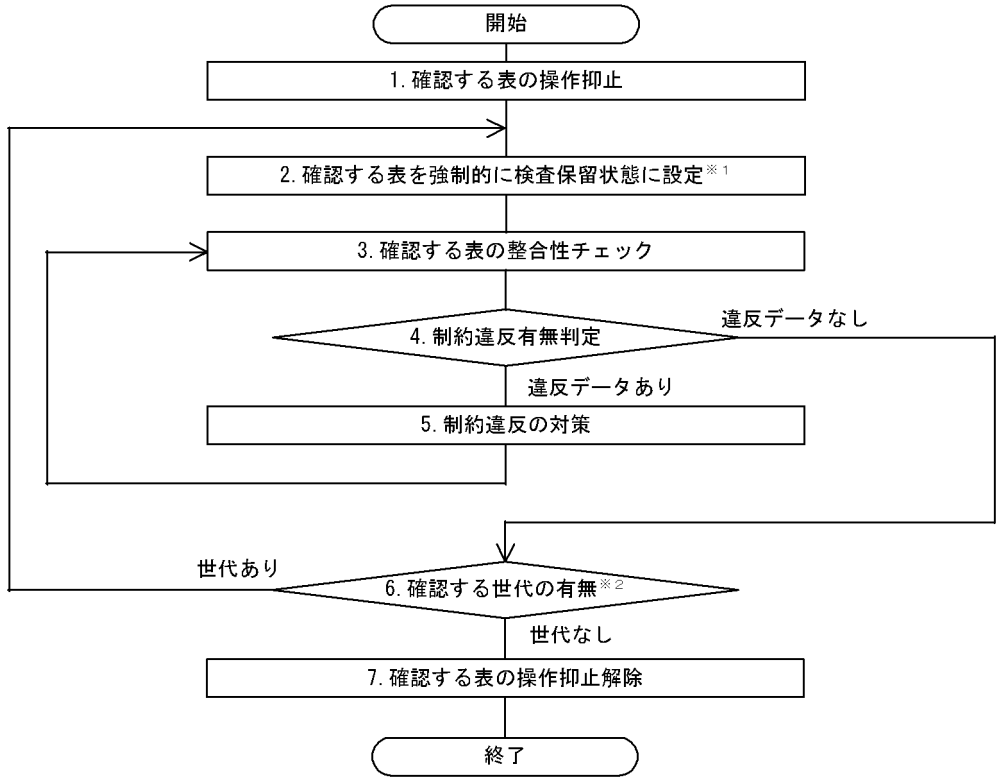
9. 確認する世代の有無

複数世代のレプリカ RD エリアを作成している場合，又は世代単位の整合性チェックを実行した場合，手順 1.に戻り，各世代で整合性チェックを実行します。

(2) 検査保留状態を使用しない場合の整合性確認手順

pd_check_pending オペランドの指定値が NOUSE の場合の，整合性チェックユティリティを使用した整合性確認手順を次の図に示します。

図 11-40　検査保留状態を使用しない場合の整合性確認手順



注※1　制約名単位の整合性検査を実行する場合は不要です。
注※2　インナレプリカ機能を使用していない場合は不要です。また，インナレプリカ機能を使用している場合でも，全世代を対象とする整合性検査を実行する場合は不要です。

1. 確認する表の操作抑止

整合性が保証できない表を使用する業務の運用を停止します。

2. 確認する表を強制的に検査保留状態に設定

確認する表を強制的に検査保留状態に設定します。なお，手順 3.で制約単位の整合性チェックをする場合は，この操作は不要です。

3. 確認する表の整合性チェック

表単位，又は制約単位に整合性チェックをします。

インナレプリカ機能を使用している場合，確認する表の世代番号を指定します。インナレプリカ機能を使用していない場合，又は全世代に整合性チェックを実行する場合，世代番号の指定は不要です。

4. 制約違反有無判定

手順 3.の整合性チェック結果で，制約違反データの有無を判定します。

5. 制約違反の対策

「検査保留状態を使用する場合の整合性確認手順（参照制約）」の手順 6.を参照して，同様に制約違反データを修正してください。

6. 確認する世代の有無

複数世代のレプリカ RD エリアを作成している場合，又は世代単位の整合性チェックを実行した場合，手順 1.に戻り，各世代で整合性チェックを実行してください。

7. 確認する表の操作抑止解除

運用を停止していた業務を再開します。

11.19.6 参照制約とトリガ

(1) 参照制約動作のトリガ

参照制約動作に CASCADE を指定すると，HiRDB が内部的に参照表を更新するトリガを被参照表に対して生成します。HiRDB が内部的に生成するトリガは，次の場合に無効になるため，再作成する必要があります。ただし，HiRDB が生成したトリガだけを再作成することはできません。ALTER ROUTINE を使用して無効になったトリガすべてを再作成してください。また，インデクスの定義，又は削除でインデクス情報が無効になった場合は，ALTER ROUTINE に ALL 指定をしてトリガを再作成してください。

- 更新の場合
  - 参照表の表定義を変更した場合
  - 参照表にインデクスを定義した場合
  - 参照表のインデクスを削除した場合
  - 参照表に，トリガ契機が UPDATE のトリガを生成する場合
  - 参照表のトリガ契機が UPDATE のトリガを削除する場合
  - 参照表が参照する被参照表の主キー構成列の表定義を変更した場合
- 削除の場合
  - 参照表の表定義を変更した場合
  - 参照表にインデクスを定義した場合
  - 参照表のインデクスを削除した場合
  - 参照表に，トリガ契機が DELETE のトリガを生成する場合
  - 参照表のトリガ契機が DELETE のトリガを削除する場合

また，HiRDB が内部的に生成するトリガは，参照表削除（DROP TABLE，又は DROP SCHEMA）時に削除されます。

(2) 参照制約とユーザが定義したトリガの関係

トリガや参照制約が定義されている表に，更新系 SQL（INSERT 文，UPDATE 文，又は DELETE 文）を実行する場合の，トリガ，参照制約の整合性チェック，及び参照制約動作（HiRDB が参照制約定義時に

内部的に生成するトリガ）の動作順序について説明します。これらの動作順序は，条件によって二つのパターンがあります。

パターン 1 の条件：

更新対象が被参照で参照制約動作の指定が RESTRICT だけの場合と，更新対象が参照表の場合

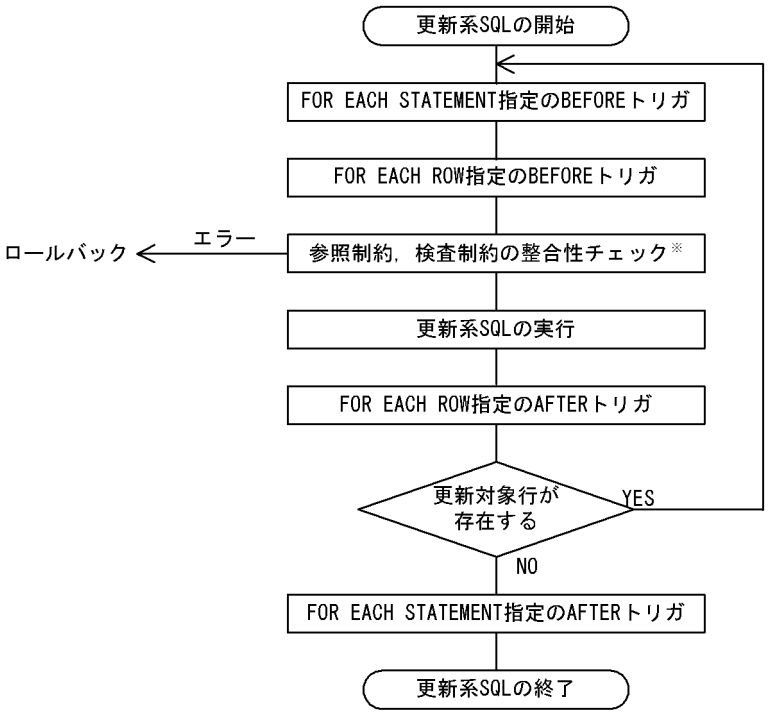
パターン 2 の条件：

更新対象が被参照で参照制約動作の指定に RESTRICT 以外がある場合

なお，更新対象の表が参照表であり，かつ被参照表でもある場合は，被参照表の条件が優先されます。

二つのパターンの場合の動作順序をそれぞれ次に示します。

パターン 1

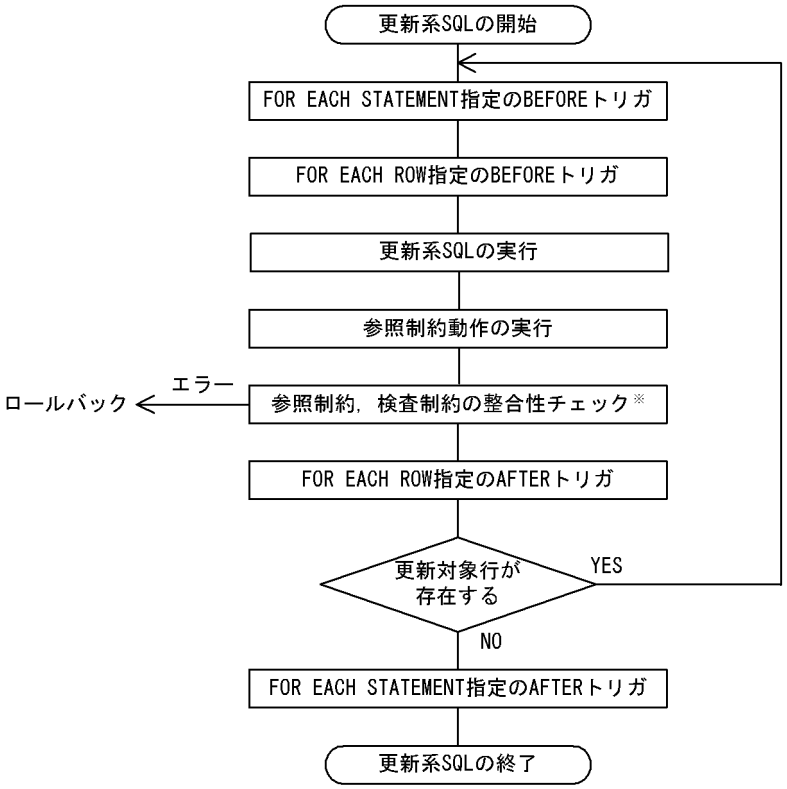


注※

参照制約の整合性はすべてこの時点でチェックされます。チェック内容を次に示します。

1. 更新対象が参照表の場合
   更新（INSERT，UPDATE）データが被参照表に含まれているかどうか
2. 更新対象が被参照表の場合
   更新（UPDATE，DELETE）データが参照表に含まれているかどうか
3. 更新対象が参照表で，かつ被参照表の場合
   上記 1，2 のチェック内容

パターン 2



注※

参照制約の整合性はすべてこの時点でチェックされます。チェック内容はパターン 1 と同じです。

11.19.7 関連製品との連携時の注意

関連製品との連携時の注意事項を次に示します。

- インナレプリカ機能を使用する場合
  - 被参照表又は参照表を格納している RD エリアのインナレプリカを作成するとき，参照関係にある表データを格納するすべての RD エリアの世代番号を合わせてください。また，被参照表や参照表にインデクスが定義されているとき，インデクス格納用 RD エリア及び LOB 用 RD エリアも，表格納用 RD エリアと世代番号を合わせてください。
  - オリジナル RD エリアにある参照表が検査保留状態のとき，レプリカ RD エリアの実体は作成しないでください。オリジナル RD エリアにある参照表の検査保留状態を解除してからレプリカ RD エリアの実体を作成してください。
  - 全世代を対象とした検査保留状態の設定又は解除をするとき，コマンド閉塞かつクローズ状態の世代はレプリカ RD エリアの実体がない世代として扱うため，検査保留状態の設定又は解除の対象から除外されます。レプリカ RD エリアの実体があるのに，設定又は解除の対象から除外された RD エリアがあるとき，RD エリアの閉塞解除後，整合性チェックユティリティを実行して RD エリア中の表情報を更新してください。

- 次に示す操作を実行後，整合性チェックユティリティで全世代指定の表単位の整合性チェックを実行してください。
  ・PURGE TABLE 文
  ・RD エリアの再初期化
  ・レプリカ RD エリアの削除
  ・インナレプリカグループの統合

- **更新可能なオンライン再編成をする運用の場合**

  更新可能なオンライン再編成及びデータベースの一括更新をする場合，追い付き反映時に HiRDB は整合性をチェックしないため，更新可能なオンライン再編成及びデータベースの一括更新の完了後，整合性が保証されません。そのため，pd_check_pending オペランドに USE を指定しているときは参照表が検査保留状態になっていることがあります。整合性チェックユティリティで検査保留状態を解除してください。pd_check_pending オペランドに NOUSE を指定しているときは，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設定してから整合性を確認してください。データの整合性確認手順については，「表の整合性確認手順」を参照してください。

- **HiRDB Datareplicator を使用する場合**

  反映側の表に参照制約の定義がある場合は，条件によってデータ連動できないときがあります。条件の詳細は，マニュアル「HiRDB データ連動機能 HiRDB Datareplicator」を参照してください。

- **分割格納条件を変更する場合**

  被参照表の分割格納条件を変更する場合，及び既存データを削除するような RD エリアの統合又は分割をした場合，分割格納条件変更の完了後の整合性は保証されないため，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，「表の整合性確認手順」を参照してください。

11.20　検査制約

11.20.1　検査制約とは

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制約が**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，整合性を保証できなくなる操作については「データ操作と整合性」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。

11.20.2　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。

(1)　検査制約を定義する表の制限事項

検査制約を定義する表の表定義，及び表定義変更時の制限事項を次に示します。

(a)　表定義（CREATE TABLE）時

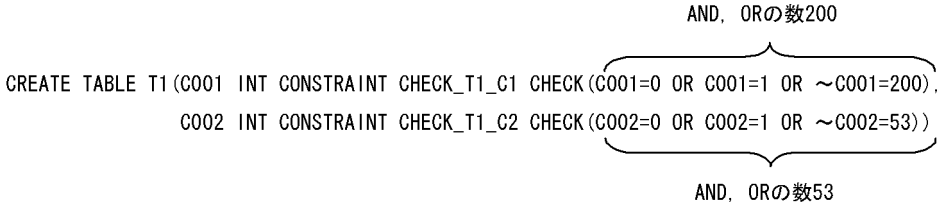
- 検査制約は改竄防止表には定義できません。
- 検査制約は一つの表に 254 個まで定義できます。255 個以上は定義できません。例を次に示します。

```
CREATE TABLE T1(C001 INT CONSTRAINT CHECK_T1_C001 CHECK(C001>0),
                C002 INT CONSTRAINT CHECK_T1_C002 CHECK(C002>0),
                                  :
                C254 INT CONSTRAINT CHECK_T1_C254 CHECK(C254>0))
                C255 INT CONSTRAINT CHECK_T1_C255 CHECK(C255>0))
```

255個

この場合，検査制約数が 254 個より多いため，定義できません。表定義時にエラーとなります。

- 検査制約，及び各検査制約の探索条件中の AND，OR の個数の和は一つの表に 254 個まで定義できます（ただし，CASE 式中の探索条件，及びその探索条件中の AND，OR の個数は除きます）。例を次に示します。



この場合，検査制約数は 2 個ですが，制約名 CHECK_T1_C1 の探索条件中の AND の数が 200，制約名 CHECK_T1_C2 の探索条件中の AND の数 53 で，検査制約数と各検査制約の探索条件中の AND，OR の和が 255（2 + 200 + 53）となり，254 個より多いため定義できません。表定義時にエラーとなります。

なお，表に定義されている検査制約数と各検査制約の探索条件中の AND，OR の和はディクショナリ表の SQL_TABLE 表の，N_CHECK_LIMIT 列に格納されています。

(b) 表定義変更（ALTER TABLE）時

- 検査制約表に対して，DROP 句及び RENAME 句を使用した表定義変更はできません。
- 検査制約表に対して，CHANGE 句を使用した次の変更はできません。
  - データ型やデータ長の変更
  - SPLIT の変更
  - 既定値の設定，解除
  - WITH DEFAULT の設定
- 検査制約表に対して，RENAME 句を使用した列名変更はできません。

(2) 検査制約を定義する場合の注意事項

- SQL オブジェクト用バッファ長の容量見積もり

  検査制約表に対して操作する場合，HiRDB が制約条件をチェックするトリガを生成します。そのため，SQL オブジェクト用バッファを指定するときに HiRDB が生成する制約条件の SQL オブジェクトについても考慮する必要があります。SQL オブジェクト用バッファ長（pd_sql_object_cache_size）の見積もり式については，マニュアル「HiRDB システム定義」を参照してください。
- バックアップの取得

  バックアップ取得時点の検査保留状態によって，バックアップの取得範囲が異なります。バックアップの取得時点と取得範囲については，マニュアル「HiRDB システム運用ガイド」の「同時にバックアップを取得する必要がある RD エリア」を参照してください。
- データディクショナリ用 RD エリアの再編成

検査制約表の定義と削除を繰り返すと，データディクショナリ用 RD エリアの格納効率が低下します。このような場合，データベース状態解析ユティリティ（pddbst）でデータディクショナリ用 RD エリアの格納効率を確認し，必要に応じて再編成してください。

11.20.3　検査保留状態

ユティリティの実行などでデータの整合性を保証できなくなった場合，HiRDB 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。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。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設定することもできます。

pd_check_pending オペランドに NOUSE を指定していると，データの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設定してから，整合性を確認してください。

整合性が保証できなくなる操作については「データ操作と整合性」を，整合性の確認手順は「表の整合性確認手順」を参照してください。

(1)　検査保留状態の管理

検査保留状態は**ディクショナリ表**と，表が格納された **RD エリアの表情報**で管理しています。ディクショナリ表では表単位，及び制約単位に検査保留状態を管理し，表情報では分割表の場合は RD エリア単位に，分割表ではない場合は表単位に検査保留状態を管理します。

検査保留状態の情報が格納されている場所と内容を次の表に示します。

表 11–23　検査保留状態の情報が格納されている場所と内容（検査制約）

格納場所			格納されている情報
ディクショナリ表	SQL_TABLES 表	CHECK_PEND2 列	表単位の検査制約の検査保留状態
	SQL_CHECKS 表	CHECK_PEND2 列	制約単位の検査制約の検査保留状態
RD エリアの表情報		分割していない表の場合	表単位の参照制約又は検査制約の検査保留状態
		分割表の場合	RD エリア単位の参照制約又は検査制約の検査保留状態

(2)　検査保留状態の表に対して制限される操作

参照制約の場合と同じです。「検査保留状態の表に対して制限される操作」を参照してください。

(3)　分割表又はインナレプリカ機能を使用している場合

参照制約の場合と同じです。「分割表又はインナレプリカ機能を使用している場合」を参照してください。ただし，「参照表」を「検査制約表」に読み替えてください。

(4)　検査保留状態を使用する場合の注意

- pd_check_pending オペランドの指定値を NOUSE から USE に変更した場合，整合性チェックユティリティを使用して，検査制約表の整合性を確認する必要があります。確認手順については，「表の整合性確認手順」を参照してください。
- pd_check_pending オペランドに USE を指定する場合，検査保留状態に設定される参照表及び RD エリアに対して排他が掛かるため，ユティリティ及び SQL 実行時の排他資源が検査保留状態を使用しない場合とは異なります。

11.20.4　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，次の表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。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。

表 11–24　整合性を保証できなくなる場合の，検査制約表に対する操作とデータ不整合の発生条件

表又は RD エリアに対する操作		データ不整合の発生条件
データベース作成ユティリティ（pdload）	データロード	検査制約定義で指定した探索条件を満たさないデータをデータロードした場合
データベース再編成ユティリティ（pdrorg）	リロード（-k reld 指定）	検査制約定義で指定した探索条件を満たさないデータをリロードした場合
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）		更新可能なオンライン再編成運用中※に，カレントデータベースのレプリカ RD エリアで，レプリカ RD エリアの検査制約表にデータベース作成ユティリティ（pdload）で，データ不整合となる操作をした場合（追い付き反映処理後にデータ不整合になる）

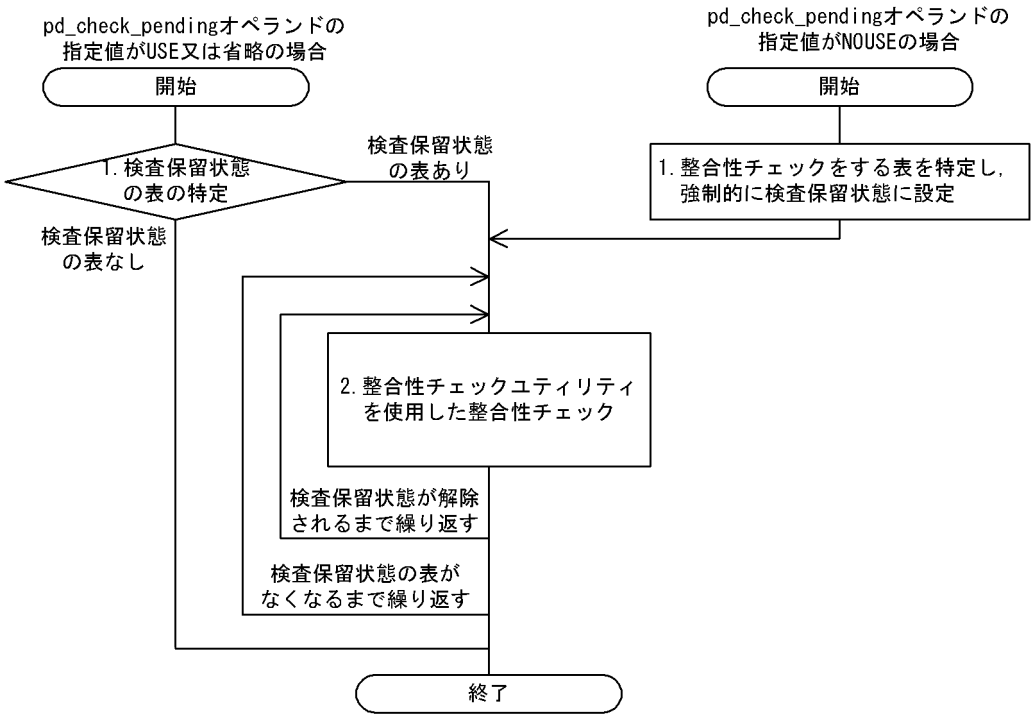
注※

更新可能なオンライン再編成の運用については，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。

11.20.5　表の整合性確認手順

データの整合性確認手順の概要を次の図に示します。

図 11-41　データの整合性確認手順の概要（検査制約）



pd_check_pending オペランドの指定値が USE 又は省略の場合

1. 検査保留状態の表の特定

ディクショナリ表の SQL_TABLES 表を検索して，検査保留状態の表名を検出します。

```
SELECT TABLE_SCHEMA, TABLE_NAME FROM MASTER.SQL_TABLES
    WHERE CHECK_PEND = 'C' OR CHECK_PEND2 = 'C'
```

検索結果には，検査保留状態の表の所有者と検査保留状態の表名が返されます。検索結果が 0 行の場合，検査保留状態の表はありません。

なお，インナレプリカ機能使用時に，各世代の表が検査保留状態かどうかを確認するためには，状態解析ユティリティ（pddbst）を使用します。

2. 整合性チェックユティリティを使用した整合性チェック

整合性チェックユティリティで表単位の整合性チェックを実行し，制約違反データがあれば修正します。検査保留状態の表がなくなるまで整合性チェックを繰り返し，なくなれば終了です。整合性チェックユティリティを使用した整合性確認手順については，「検査保留状態を使用する場合の整合性確認手順（検査制約）」を参照してください。

pd_check_pending オペランドの指定値が NOUSE の場合

1. 整合性チェックをする表を特定し，強制的に検査保留状態に設定

整合性チェックをする表を特定するために，整合性を保証できなくなる操作をした表に検査照制約が定義されているかどうかを確認します。これを確認する SQL の実行例を次に示します。

```
SELECT N_CHECK FROM MASTER.SQL_TABLES
    WHERE TABLE_SCHEMA = '対象表の所有者名' AND TABLE_NAME = '対象表の表名'
```

次の検索結果が返されます。

- 検査制約の定義数

N_CHECK がナル値の場合，対象表に検査制約は定義されていません。

表を特定したら，整合性チェックユティリティを使用して，その表を強制的に検査保留状態に設定します（検査保留状態でない表は，整合性チェックユティリティでチェックできません）。

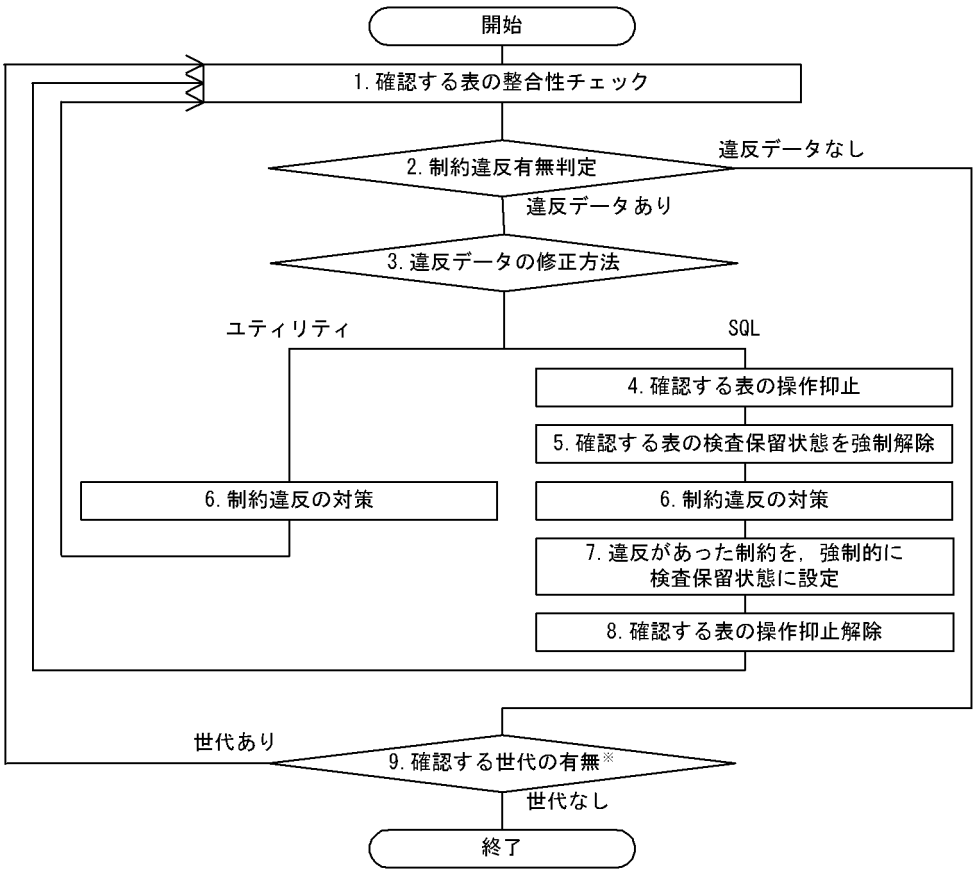
2. 整合性チェックユティリティを使用した整合性チェック

pd_check_pending オペランドの指定値が USE 又は省略の場合の手順 2.と同じです。整合性チェックユティリティを使用した整合性確認手順は，参照制約の場合と同じのため，「検査保留状態を使用しない場合の整合性確認手順」を参照してください。

(1) 検査保留状態を使用する場合の整合性確認手順（検査制約）

pd_check_pending オペランドの指定値が USE 又は省略の場合の，整合性チェックユティリティを使用した整合性確認手順を次の図に示します。

図 11-42 検査保留状態を使用する場合の整合性確認手順（検査制約）



注※ 次の場合は不要です。
・インナレプリカ機能を使用していない
・インナレプリカ機能を使用していて，全世代に整合性チェックを実行する場合

1. 確認する表の整合性チェック

表単位，又は制約単位に整合性チェックをします。

インナレプリカ機能を使用している場合，確認する表の世代番号を指定します。インナレプリカ機能を使用していない場合，又は全世代に整合性チェックを実行する場合，世代番号の指定は不要です。

2. 制約違反有無判定

手順 1.の整合性チェック結果で，制約違反データの有無を判定します。

3. 違反データの修正方法

違反データの修正をユティリティで行うか，SQL で行うかを選択します。ユティリティで行う場合，手順 6.へ進んでください。

4. 確認する表の操作抑止

整合性が保証できない表を使用する業務の運用を停止します。

5. 確認する表の検査保留状態を強制解除

制約違反の対策をするため，検査保留状態を強制解除します。

6. 制約違反の対策

ユティリティで修正する場合

対策方法を次に示します。対策後，手順 1.に戻り，整合性チェックを実行し，違反データがないことを確認し，終了します。

条件	対策方法
検査制約の定義で指定した探索条件を修正する場合	手順を次に示します。 1. 表のデータをすべてアンロードします。 2. DROP TABLE で表の定義を削除します。 3. CREATE TABLE で表を再定義します。このとき，検査制約の定義に正しい探索条件を指定します。 4. 1.でアンロードしたデータをロードします。
表に制約違反データがある場合	• データベース作成ユティリティ（pdload）の作成モードで正しいデータをロードします。 • データベース再編成ユティリティ（pdrorg）の UOC を使用した削除で不要なデータを削除します。

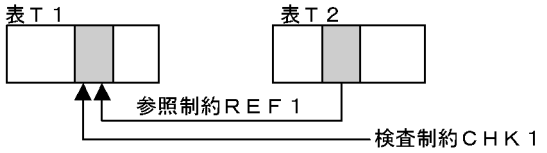
SQL で修正する場合

対策方法を次に示します。対策後，手順 7.に進みます。

条件	対策方法
検査制約の定義で指定した探索条件を修正する場合	ユティリティで修正する場合と同じです。
表に制約違反データがある場合	制約違反データを DELETE 文で削除，又は，UPDATE 文で正しい値に更新します※。

注※

対策する表を被参照表とする参照表が存在する場合，修正順序に注意が必要です。例えば，次のような関係があるとします。



●CHK1 の制約違反の対策をする場合の注意

表 T1 のデータを DELETE 文で修正する場合，REF1 で ON DELETE RESTRICT を指定しているときは，対応する表 T2 のデータを先に削除後，表 T1 のデータを削除してください。また，UPDATE 文で修正する場合，REF1 で ON UPDATE RESTRICT を指定しているときは，更新前のデータに対応する表 T2 のデータを削除後，表 T1 のデータを更新してください。

7. 違反があった制約を強制的に検査保留状態に設定

整合性チェックユティリティを制約単位で実行し，対策をした制約を強制的に検査保留状態に設定します。

8. 確認する表の操作抑止解除

運用を停止していた業務を再開します。手順 1.に戻り，整合性チェックを実行し，違反データがないことを確認します。

9. 確認する世代の有無

複数世代のレプリカ RD エリアを作成している場合，又は世代単位の整合性チェックを実行した場合，手順 1.に戻り，各世代で整合性チェックを実行します。

11.20.6　関連製品との連携時の注意

関連製品との連携時の注意事項を次に示します。

- インナレプリカ機能を使用する場合
  - オリジナル RD エリアにある参照表が検査保留状態のとき，レプリカ RD エリアの実体は作成しないでください。オリジナル RD エリアにある参照表の検査保留状態を解除してからレプリカ RD エリアの実体を作成してください。
  - 全世代を対象とした検査保留状態の設定又は解除をするとき，コマンド閉塞かつクローズ状態の世代はレプリカ RD エリアの実体がない世代として扱うため，検査保留状態の設定又は解除の対象から除外されます。レプリカ RD エリアの実体があるのに，設定又は解除の対象から除外された RD エリアがあるとき，RD エリアの閉塞解除後，整合性チェックユティリティを実行して RD エリア中の表情報を更新してください。
  - 次に示す操作を実行後，整合性チェックユティリティで全世代指定の表単位の整合性チェックを実行してください。

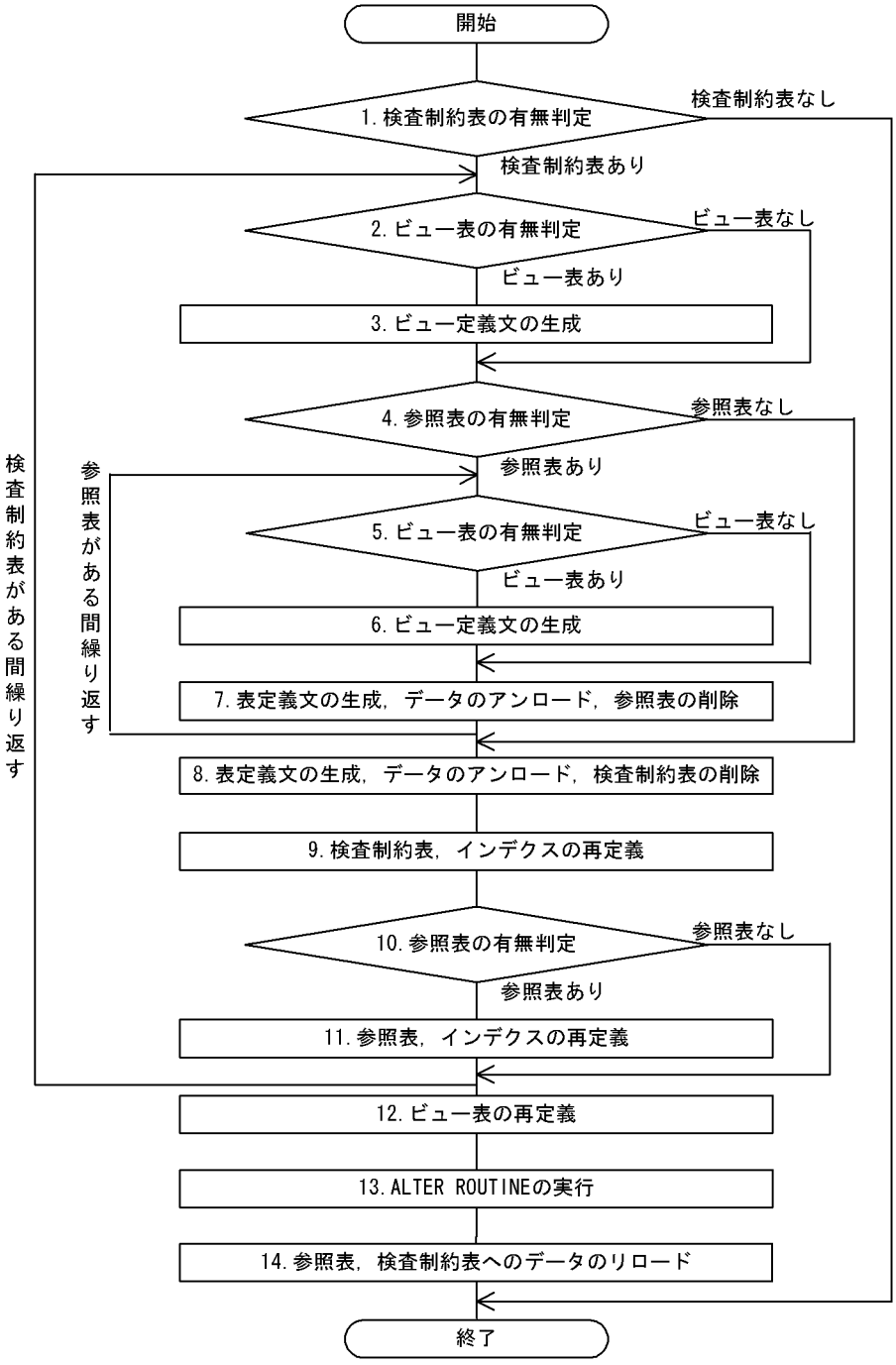
・PURGE TABLE 文
・RD エリアの再初期化
・レプリカ RD エリアの削除
・インナレプリカグループの統合

- **更新可能なオンライン再編成をする運用の場合**
  - 更新可能なオンライン再編成及びデータベースの一括更新をする場合，追い付き反映時に HiRDB は整合性をチェックしないため，更新可能なオンライン再編成及びデータベースの一括更新の完了後，整合性が保証されません。そのため，pd_check_pending オペランドに USE を指定しているときは検査制約表が検査保留状態になっていることがあります。整合性チェックユティリティで検査保留状態を解除してください。pd_check_pending オペランドに NOUSE を指定しているときは，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設定してから整合性を確認してください。データの整合性確認手順については，「表の整合性確認手順」を参照してください。
- **HiRDB Datareplicator を使用する場合**
  - 整合性があるデータが反映されるので，反映側の表に検査制約の定義は不要です。

11.20.7　検査制約表の 64 ビットモードへの移行

HiRDB を 32 ビットモードから 64 ビットモードへ移行する場合，32 ビットモードで定義した検査制約表に挿入や更新をするとエラーになります。32 ビットモードで定義した検査制約表を 64 ビットモードで挿入や更新できるようにするには，64 ビットモードで HiRDB を再開始後，検査制約表を再定義してください。検査制約表の 64 ビットモードへの移行の基本的な流れを次の図に示します。

図 11-43　検査制約表の 64 ビットモードへの移行の基本的な流れ



1. 検査制約表の有無判定

検査制約表の有無を確認するため，次の SQL を実行します。

```
SELECT TABLE_SCHEMA,TABLE_NAME FROM MASTER.SQL_TABLES WHERE N_CHECK > 0
```

結果行数が 1 以上であれば，検査制約表があります。また，検索結果の TABLE_SCHEMA が検査制約表の所有者，TABLE_NAME が検査制約表の表名になります。

2. ビュー表の有無判定

検査制約表を削除すると，検査制約表を使用したビュー表も削除されます。そのため，検査制約表を使用したビュー表があるかどうかを確認する必要があります。検査制約表を使用したビュー表の有無を確認するため，次の SQL を実行します。

```
SELECT VIEW_SCHEMA,VIEW_NAME FROM MASTER.SQL_VIEW_TABLE_USAGE
    WHERE BASE_OWNER=検査制約の表の所有者 AND TABLE_NAME=検査制約の表名
```

結果行数が 1 以上であれば，検査制約表を使用したビュー表があります。また，検索結果の VIEW_SCHEMA がビュー表の所有者，VIEW_NAME がビュー表名になります。

3. ビュー定義文の生成

pddefrev コマンド（定義系 SQL の生成）でビュー定義文を生成してください。

4. 参照表の有無判定

検査制約表を削除する場合，主キーが定義されていて，かつ主キーを参照する参照表が定義されていると，その検査制約表は削除できません。そのため，削除する表の主キーを参照する参照表を削除する必要があります。検査制約表の主キーを参照する参照表の有無を確認するため，次の SQL を実行します。

```
SELECT CONSTRAINT_SCHEMA,TABLE_NAME
    FROM MASTER.SQL_REFERENTIAL_CONSTRAINTS
      WHERE R_OWNER= 検査制約の表の所有者 AND R_TABLE_NAME=検査制約の表名
```

結果行数が 1 以上であれば，参照表があります。また，検索結果の CONSTRAINT_SCHEMA が参照表の所有者，TABLE_NAME が参照表の表名になります。

5. ビュー表の有無判定

参照表を削除すると，参照表を使用したビュー表も削除されます。そのため，参照表を使用したビュー表があるかどうかを確認する必要があります。参照表を使用したビュー表の有無を確認するため，次の SQL を実行します。

```
SELECT VIEW_SCHEMA,VIEW_NAME FROM MASTER.SQL_VIEW_TABLE_USAGE
    WHERE BASE_OWNER=参照表の所有者 AND TABLE_NAME=参照表の表名
```

結果行数が 1 以上であれば，参照表を使用したビュー表があります。また，検索結果の VIEW_SCHEMA がビュー表の所有者，VIEW_NAME がビュー表名になります。

6. ビュー定義文の生成

pddefrev コマンド（定義系 SQL の生成）で検査制約表を参照する参照表を使用したビュー定義文を生成してください。

7. 表定義文の生成，データのアンロード，参照表の削除

pddefrev コマンド（定義系 SQL の生成）で参照表の表定義文を生成してください。表定義文を生成後，削除する参照表のデータをアンロードして，参照表を削除してください。

8. 表定義文の生成，データのアンロード，検査制約表の削除

pddefrev コマンド（定義系 SQL の生成）で検査制約表の表定義文を生成してください。表定義文を生成後，削除する検査制約表のデータをアンロードして，検査制約表を削除してください。

9. 検査制約表，インデクスの再定義

8.で生成した表の表定義文に検査制約表，及びインデクスを再定義してください。

10. 参照表の有無判定

9.で再定義した検査制約表を参照する参照表の有無を 4.と同様にして確認してください。

11. 参照表，インデクスの再定義

9.で再定義した検査制約表を参照する参照表がある場合，7.で生成した表の表定義文に参照表，及びインデクスを再定義してください。

12. ビュー表の再定義

検査制約表を使用したビュー表，又は参照表を使用したビュー表をがある場合，3.，6.で生成したビュー表の定義文にビュー表を再定義してください。

13. ALTER ROUTINE の実行

表，ビュー表などの削除によって関数が無効状態になっている可能性があるため，定義系 SQL の ALTER ROUTINE を実行してください。

14. 参照表，検査制約表へのデータのリロード

再定義した表にデータをリロードしてください。

11.21　圧縮表

11.21.1　データ圧縮機能

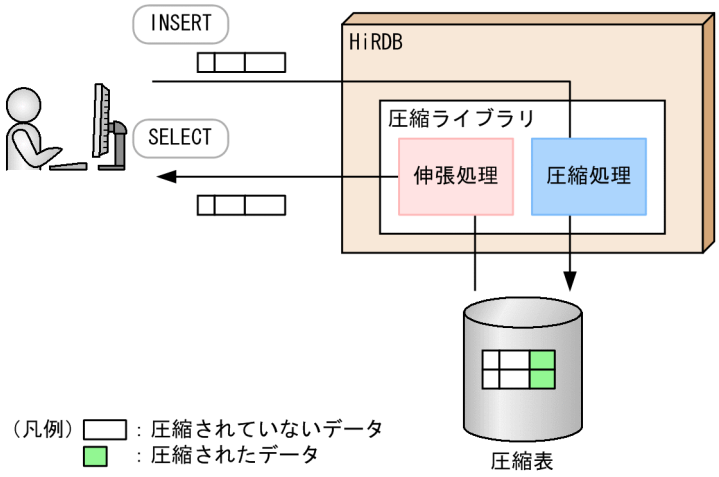
HiRDB が表にデータを格納するとき，データを圧縮して格納できます。これを**データ圧縮機能**といいます。データの圧縮は列単位に指定でき，圧縮したデータを格納する列を**圧縮列**といい，圧縮列がある表を**圧縮表**といいます。

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 11-44　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)　データの圧縮効率の目安

圧縮前と比べてどれだけ格納領域を節約できるかは，圧縮効率で表します。圧縮効率は，次の計算式で求めます。

```
圧縮効率（%）＝
｛（圧縮前のデータ長-圧縮後のデータ長）÷圧縮前のデータ長｝×100
```

また，圧縮率と圧縮効率の関係を次に示します。

```
圧縮率＋圧縮効率＝100
```

圧縮率の測定方法については，「データの圧縮率の測定方法」を参照してください。

圧縮効率の目安を次の表に示します。なお，データの圧縮率及び圧縮効率は，圧縮対象となるデータの実態によって異なるため，ここで示す圧縮効率はあくまで目安値です。

表 11-25　圧縮効率の目安

データ種別	圧縮効率（単位：%）
全文字が同一の BINARY データ	98.51
完全にランダムな BINARY データ	-0.36※
テキストデータ（.txt）	58.50
画像データ（.bmp）	75.42
音声データ（.wav）	9.46

注※

圧縮処理で付与されるヘッダ領域が増加したため，圧縮効率がマイナスになります。圧縮処理については，「データ圧縮の仕組み」を参照してください。

(3)　圧縮表に対する操作で HiRDB が出力するファイル

圧縮表に対する操作によって，HiRDB が出力するファイルのデータの状態を次の表に示します。

表 11-26　HiRDB が出力するファイルのデータの状態

処理内容	ファイル	データの状態
作業表用ファイルが必要な SQL の実行	作業表用ファイル	伸張されたデータ
データベースの更新	システムログファイル	圧縮されたデータ
システムログのアンロード	アンロードログファイル	
データベースのバックアップ（pdcopy コマンド）	バックアップファイル	
表の再編成 （pdrorg -k rorg コマンド）	アンロードデータファイル	圧縮されたデータ※
表のアンロード （pdrorg -k unld コマンド）		伸張されたデータ

注※　UOC を利用して再編成を行う場合は，伸張されたデータが格納されます。

(4)　障害発生時の対処

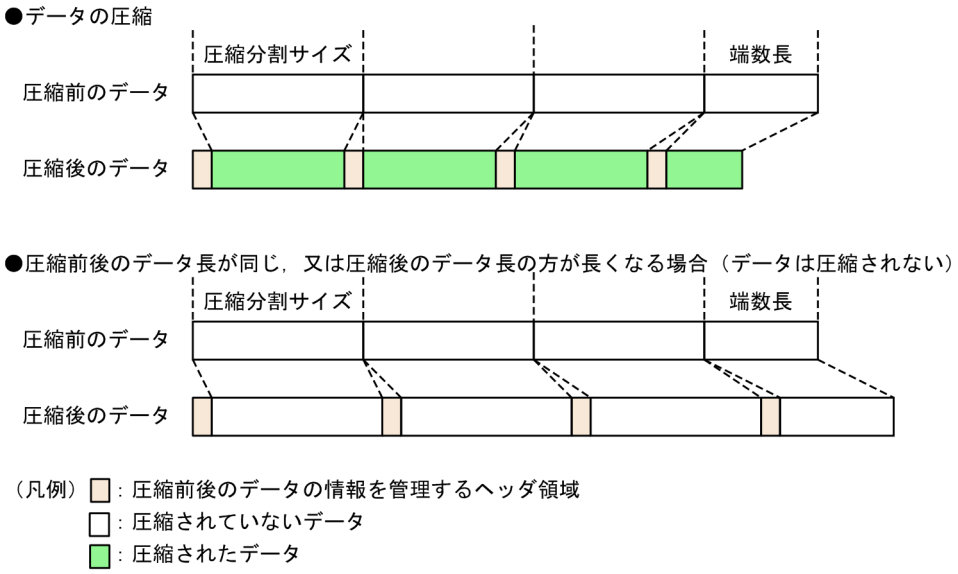
圧縮表のデータやインデクスが格納されている RD エリアに障害が発生した場合，通常のデータベースの回復と同様にデータベース回復ユティリティ（pdrstr）で回復できます。

11.21.2　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 11–45　圧縮前後のデータ



11.21.3　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。

11.21.4　既存の表を圧縮表に変更する方法

(1)　既存の表の列を圧縮列に変更する場合

定義系 SQL の ALTER TABLE の列属性変更定義（CHANGE 列名）で圧縮列に変更することはできません。そのため，既存の表の列を圧縮列に変更する場合は，次の手順で行ってください。

〈手順〉

1. 既存の表のアンロード
   既存の表をアンロードします。
2. 既存の表の削除
   DROP TABLE で既存の表をいったん削除します。
3. 表の再定義
   CREATE TABLE で，圧縮列に変更する列に圧縮指定をして，表を再定義します。圧縮指定以外は変更しないでください。
4. 表のリロード
   3.で再定義した表に，1.でアンロードしたアンロードデータファイルをリロードします。

データのアンロード及びリロードについては，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。

参考

圧縮列に変更する列が表の最後の列の場合，上記手順の 2.及び 3.は次の手順でも圧縮列に変更できます。

- 既存の列の削除，及び圧縮列の追加
  PURGE TABLE で表のデータを 0 件にし，ALTER TABLE の列削除定義（DROP 列名）で圧縮列に変更する列をいったん削除します。次に，ALTER TABLE の列追加定義（ADD 列名）で，削除した列に圧縮指定をして再定義（追加）します。

(2)　既存の表の最後に新規の圧縮列を追加する場合

定義系 SQL の ALTER TABLE の列追加定義（ADD 列名）で圧縮指定をして圧縮列を追加します。その後，圧縮列にデータロードすると，データが圧縮して格納されます。

> **参考**
>
> ALTER TABLE の列追加定義では，表の最後に新しい列を追加します。そのため，圧縮列は表の最後の列にしか追加できません。

11.21.5　圧縮列の定義を変更（圧縮指定を解除）する方法

圧縮列の定義は，定義系 SQL の ALTER TABLE の列属性変更定義（CHANGE 列名）で変更できません。圧縮指定を解除したり，圧縮分割サイズを変更するなど，圧縮列の定義を変更する場合，次の手順で変更してください。

〈手順〉

1. 圧縮表のアンロード

   定義を変更する圧縮表をアンロードします。

2. 表の再定義

   次のどちらかの方法で，圧縮指定を変更，又は解除した表を再定義します。

   - 表の再定義

     DROP TABLE で圧縮表をいったん削除し，CREATE TABLE で圧縮指定を変更又は解除した表を再定義します。

   - 列の削除・追加

     PURGE TABLE で表のデータを 0 件にし，ALTER TABLE の列削除定義（DROP 列名）で圧縮列をいったん削除します。次に，ALTER TABLE の列追加定義（ADD 列名）で圧縮指定を変更又は解除した列を追加します。

     ただし，ALTER TABLE の列追加定義では，表の最後に新しい列を追加するため，圧縮指定を変更又は解除できる列は最後の列だけです。

3. 表のリロード

   2.で定義した表に，1.でアンロードしたアンロードデータファイルをリロードします。

データのアンロード及びリロードについては，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。

11.21.6　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。BINARY 型のデータの圧縮処理や伸張処理に掛かった時間は，統計解析ユティリティ（pdstedit）の UAP に関する統計情報で確認できます。なお，抽象データ型（XML 型）の処理時間については確認できません。
- 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※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。メモリ不足を防ぐため，圧縮分割サイズはシステム内の空きメモリや，pd_max_access_tables オペランドの指定値を考慮した値を指定してください。増加するプロセス固有メモリ所要量については，HiRDB/シングルサーバの場合は「圧縮列に対して操作系 SQL を実行する場合に必要なメモリ所要量の求め方」を，HiRDB/パラレルサーバの場合は「圧縮列に対して操作系 SQL を実行する場合に必要なメモリ所要量の求め方」を参照してください。
- 表の再編成時にエラーが発生し，その対処として圧縮されたデータを含むアンロードデータファイル（pdrorg -k rorg でアンロードされたファイル）をリロードする場合，アンロード元とリロード先の表の圧縮指定（圧縮指定の有無，及び圧縮分割サイズの指定値）が同じである必要があります。圧縮指定が異なる列がある場合，pdrorg はエラー終了します。
- 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。

注※　次のような SQL です。

- SUBSTR 関数を使用している
- POSITION 関数を使用している
- 後方削除更新をしている

11.21.7　データの圧縮率の測定方法

データの圧縮率の測定方法について説明します。実際にデータを圧縮・格納する前にどれくらい圧縮されるのか知りたい場合や，データの格納後にどれくらい圧縮されたのか確認したい場合に測定してください。ただし，抽象データ型（XML 型）の列の場合，pddbst で確認できないため，圧縮・格納後の測定はできません。

(1)　データを圧縮・格納する前の測定方法

データの圧縮率は，圧縮対象となるデータの実態によって大きく異なります。正確な圧縮率は実際にデータを圧縮・格納した後でないと測定できませんが，gzip※を使用して算出した圧縮後のデータ長の概算値を用いることで，圧縮率の概算値を計算できます。計算式を次に示します。

注※

HiRDB が圧縮に使用するアルゴリズム（Deflate）と同等の圧縮アルゴリズムを使用しています。

計算式

```
圧縮率（%）=
｛（gzipによる圧縮後のデータ長×1.05※）÷圧縮前のデータ長｝×100
```

注※

zlib と gzip では，圧縮時に付与される圧縮情報を管理するヘッダの形式が異なるため，圧縮後のデータサイズに 5%の余裕値を加えます。

(2) データを圧縮・格納した後の測定方法

実際にデータを圧縮・格納した後で圧縮率の概算値を算出する計算式を次に示します。

計算式

```
圧縮率（%）=
（圧縮後のデータ長※1の合計÷圧縮前のデータ長※2の合計）※3×100
```

注※1　算出手順を次に示します。

1. -d オプション指定で，RD エリア単位又は表単位にデータベース状態解析ユティリティ（pddbst）を実行します。
2. 出力結果の<BINARY segment>の情報から，次の計算式で圧縮後のデータ長を求めます。

$$\sum_{i=1}^{10} n_i \times a \times b$$

n_i：バイナリ専用セグメントの Used Page Ratio（使用中ページの比率別ページ数）が示す各比率の最大値（例えば，Used Page Ratio が 1～10%の場合は 10%で 0.1，11～20%の場合は 20%で 0.2 となります）

a：n_i に対応した Page の値

b：バイナリ専用セグメントのページサイズ

3. 1.で対象とした RD エリア，又は表に BINARY 型の圧縮列と非圧縮列が混在している場合，2.の算出結果から非圧縮列のデータ長を減算します。非圧縮列のデータ長は次の SQL を実行して算出します。

```
select sum（length（非圧縮列名））　from 表識別子　〔in RDエリア名〕
```

注※2

圧縮前のデータ長は，非圧縮列のデータ長の算出方法（注※1 の 3.を参照）と同じ SQL を実行して算出します。

注※3

1.0 以上の場合，圧縮によってデータ長が増加した，又は圧縮の効果が小さいことを示します。この場合は，圧縮列の定義を変更して，圧縮指定を解除することをお勧めします。圧縮指定の解除については，「圧縮列の定義を変更（圧縮指定を解除）する方法」を参照してください。

11.22　一時表

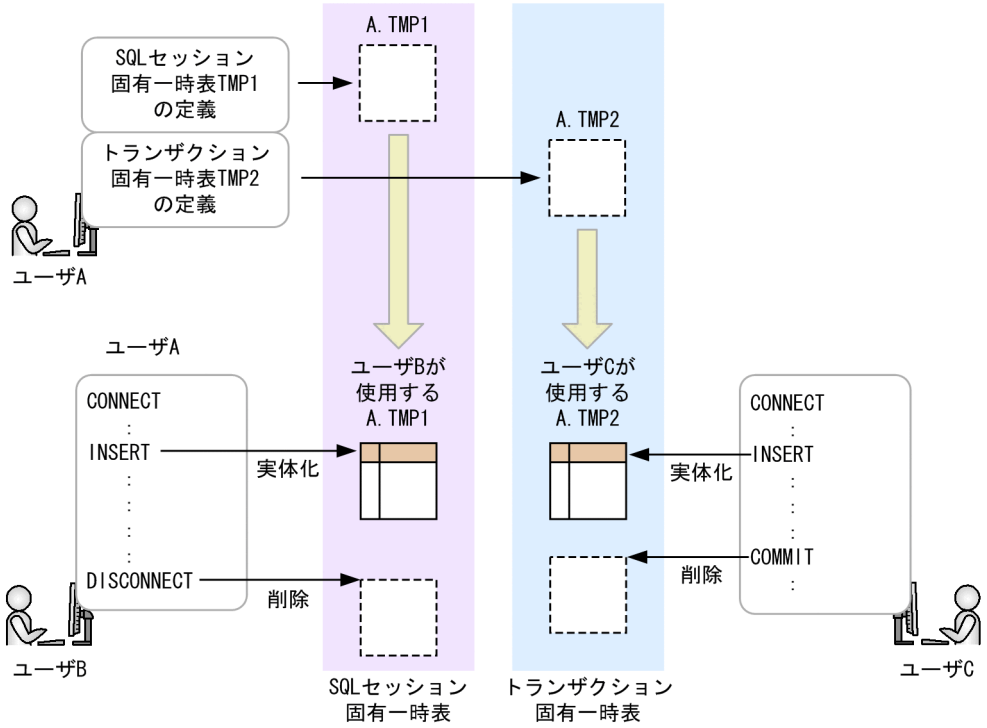
一時表は，トランザクション又は SQL セッションの期間中だけ存在する実表です。トランザクションの期間中だけ存在する一時表を**トランザクション固有一時表**，SQL セッションの期間中だけ存在する一時表を**SQL セッション固有一時表**といいます。

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。一時表用 RD エリアについては，「一時表用 RD エリア」を参照してください。

一時表の概要を次の図に示します。

図 11-46　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，更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。

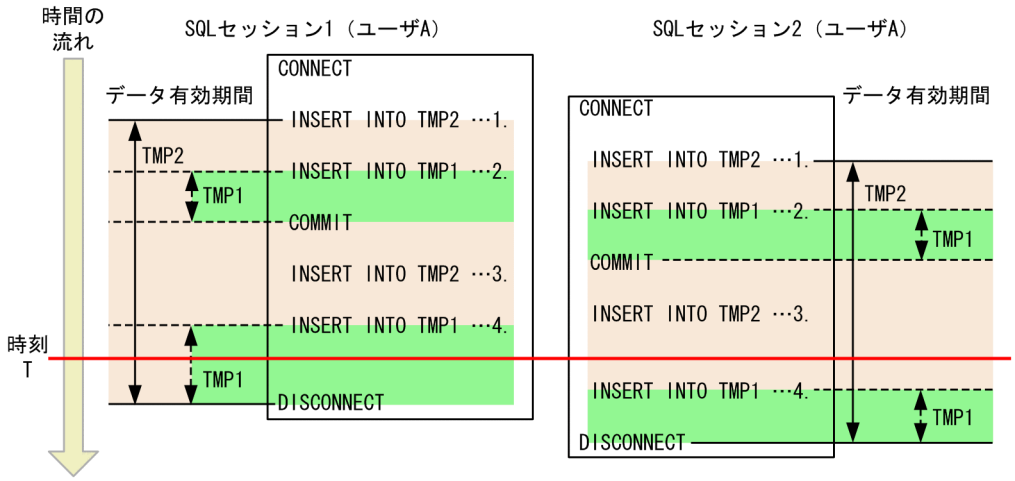
11.22.1　一時表のデータ有効期間

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，データ有効期間とある時点で保持されているデータの例を次の図に示します。

表 11-27　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

図 11-47　一時表のデータ有効期間と，ある時点で保持されているデータの例（その 1）

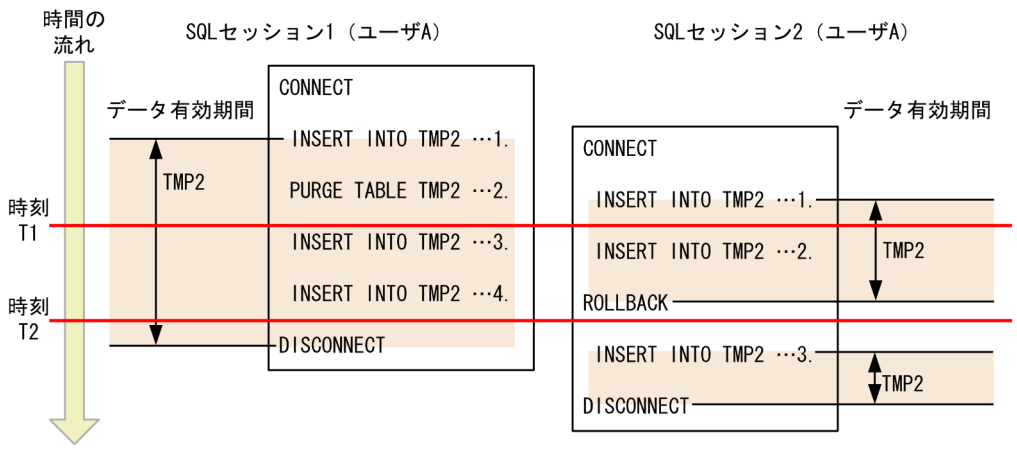


［説明］

時刻 T の時点で，SQL セッション 1 及び 2 が使用する一時表 TMP1 及び TMP2 に保持されているデータを次に示します。

SQL セッション	一時表	保持されているデータ
SQL セッション 1	TMP1	4.で挿入したデータ。
	TMP2	1.及び 3.で挿入したデータ。
SQL セッション 2	TMP1	この時点では，データはありません。
	TMP2	1.及び 3.で挿入したデータ。

図 11-48　一時表のデータ有効期間と，ある時点で保持されているデータの例（その 2）



〔説明〕

時刻 T1 及び T2 の時点で，SQL セッション 1 及び 2 が使用する一時表 TMP2 に保持されているデータを次に示します。

時刻	SQL セッション	保持されているデータ
T1	SQL セッション 1	2.で表のデータを削除しているため，この時点では，データはありません。ただし，TMP2 の実体は存在します。
	SQL セッション 2	1.で挿入したデータ。
T2	SQL セッション 1	3.及び 4.で挿入したデータ。
	SQL セッション 2	この時点では，データはありません。TMP2 を実体化する前の同期点までロールバックしているため，TMP2 の実体は存在しません。

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。
- HiRDB/パラレルサーバで，SQL セッション固有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が異常終了したり，系切り替えが発生したりすると，データ有効期間が終了します。そのため，SQL セッションが終了するまで，該当する一時表に対するデータ操作は SQL エラーになります。

11.22.2　一時表及び一時インデクスの定義方法

(1)　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。

(2)　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。

(3)　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。

11.22.3　格納先 RD エリアの決定規則

一時表用 RD エリアが複数ある場合や，クライアント環境定義 PDTMPTBLRDAREA の指定を省略している場合，HiRDB が，データを格納する一時表用 RD エリアを決定します。HiRDB は，次の順序で格納先 RD エリアを決定します。

(1)　格納先バックエンドサーバの決定（HiRDB/パラレルサーバの場合だけ）

HiRDB/パラレルサーバの場合，まず，データを格納するバックエンドサーバを決定します。このとき，次の規則で格納先候補バックエンドサーバを絞り，その中から，INSERT 文で指定した実表の中で一時表以外の実表にアクセスするバックエンドサーバを優先して使用します。

- クライアント環境定義 PDTMPTBLRDAREA に RD エリアを指定している場合

  指定した RD エリアがあるバックエンドサーバを格納先候補とします。

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 エリアがあるバックエンドサーバを優先して使用します。

- クライアント環境定義 PDTMPTBLRDAREA に RD エリアを指定していない場合

  SQL セッション間共有属性の一時表用 RD エリアがあるバックエンドサーバを格納先候補とします。

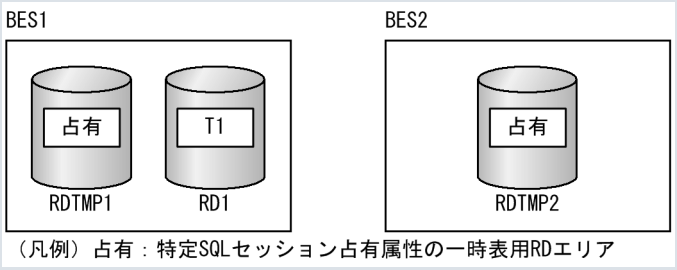
ポイント

HiRDB が格納先バックエンドサーバを決定するとき，INSERT 文で指定した実表の中で，一時表以外の実表にアクセスするバックエンドサーバを優先するため，（例）のように INSERT SELECT を使用すると，バックエンドサーバ間のデータ転送を回避できます。

（例） INSERT INTO TMP1 SELECT C1,C2,C3 FROM T1

一時表 TMP1 に，表 T1 から列 C1，C2，及び C3 を挿入する SQL です。

この SQL を実行する場合の構成を次の図に示します。PDTMPTBLRDAREA には，RDTMP1 と RDTMP2 を指定しています。



このとき，HiRDB は，表 T1 がある BES1 を格納先バックエンドサーバに決定します。これによって，BES1 と BES2 間でデータ転送することなく，SQL を実行できます。

(2) 格納先候補 RD エリアの決定

クライアント環境定義 PDTMPTBLRDAREA の指定によって，格納先候補の RD エリアを決定します。PDTMPTBLRDAREA の指定については，「データを格納する一時表用 RD エリアを指定する場合」を参照してください。

(3) 条件に合致する RD エリアの決定

格納先候補 RD エリアから，次に示すすべての条件に合致する RD エリアを決定します。

- 一時表の操作に関する排他を取得できる状態である。

  一時表の排他については，「一時表の排他制御」を参照してください。
- UAP がアクセスできる状態である。
- 格納する一時表が FIX 表の場合，一時表の行長が RD エリアに格納できるサイズを超えていない。

  詳細は，マニュアル「HiRDB SQL リファレンス」の「CREATE TABLE」の FIX オペランドの規則 3.を参照してください。

- 格納する一時インデクスを構成する列の長さの合計が，RD エリアに格納できるサイズを超えていない。
  詳細は，マニュアル「HiRDB SQL リファレンス」の「CREATE INDEX」の共通規則 5.を参照してください。
- RD エリア内の一時表の利用回数が 500 未満である。
- RD エリア内の一時インデクスの利用回数が 500 未満である。
- 未使用セグメントが存在する。
- 一時表及び一時インデクス数が pd_max_temporary_object_no オペランドの指定値を超えていない。
- pd_tmp_table_initialize_timing オペランドに ACCESS を指定している場合，初期化されていない一時表用 RD エリアである。
  一時表用 RD エリアの初期化については，「一時表用 RD エリアの初期化」を参照してください。

(4) 格納先一時表用 RD エリアの決定

条件に合致した RD エリアから，次の一時表用 RD エリアを優先して使用します。

- 最も多く未使用セグメントが存在する一時表用 RD エリア
- pd_tmp_table_initialize_timing オペランドに ACCESS を指定している場合，初期化されていない一時表用 RD エリア
  一時表用 RD エリアの初期化については，「一時表用 RD エリアの初期化」を参照してください。

11.22.4 一時表用 RD エリアがない場合の対処

HiRDB が一時表にデータを格納する時に，使用できる一時表用 RD エリアがない場合，KFPA19704-E メッセージを出力し，トランザクションを無効にします。このとき，KFPA19704-E メッセージで出力されるエラー要因は，最初に格納先候補になった RD エリアのものだけです。メッセージに表示された RD エリアの対処をしても，頻繁に同じメッセージが出力される場合，次に示す方法で，ほかの一時表用 RD エリアの状態を確認し，対処してください。

〈対処方法〉

pddbls -T コマンドを実行します。

実行結果から，RDAREA_FOR_TEMPORARY_TABLE に OCCUPIED 又は SHARED が表示されている RD エリアがあるかどうかを確認します。

- **OCCUPIED 又は SHARED が表示されている RD エリアがない場合**
  使用できる一時表用 RD エリアがありません。特定 SQL セッション占有属性の一時表用 RD エリアを追加し，必要に応じて追加した RD エリアをクライアント環境定義 PDTMPTBLRDAREA に指定してください。
- **OCCUPIED 又は SHARED が表示されている RD エリアがある場合**

一時表用 RD エリアはありますが，格納先の条件に合致する RD エリアがありません（格納先の条件は，「条件に合致する RD エリアの決定」を参照）。どの条件に合致しないのかを調査するため，一時表用 RD エリアに対して pddbls -a -T，及び pddbst -k -phys を実行します。実行結果から次の表に示す項目を確認し，条件を満たさない場合はそれぞれ対処してください。

表 11–28　確認項目と対処方法

項番	コマンド	確認項目	内容	対処
1	pddbls -a -T	STATUS	RD エリアの状態	RD エリアが次の状態の場合，対処が必要です。 • クローズ状態 • 閉塞中 • pdhold コマンド受け付け状態 RD エリアをオープンしたり，閉塞を解除したりして UAP がアクセスできる状態にしてください。障害閉塞のときは，pdmod コマンドで一時表用 RD エリアを再初期化（initialize rdarea 文）してください。
2		SEGMENT	RD エリア内の未使用セグメント数	未使用セグメントがない場合，pdmod コマンドで次のどれかの対処をしてください。 • 一時表用 RD エリアを追加する（create rdarea 文） • 作成済みの一時表用 RD エリアを再初期化する（initialize rdarea 文） • 作成済みの一時表用 RD エリアを拡張する（expand rdarea 文） • 作成済みの一時表用 RD エリアの属性変更（alter rdarea）で，自動増分を適用する
3	pddbst -k -phys	Page Size	RD エリアのページ長	ページ長についての条件を満たしていない場合，次のどちらかの対処をしてください。 • ページ長についての条件を満たす一時表用 RD エリアを追加する • 条件を満たすように作成済みの一時表用 RD エリアのページ長を変更する
4		Unused Segment	RD エリア内の未使用セグメント数	項番 2 と同じです。

11.22.5　一時表の排他制御

一時表はトランザクション又はSQLセッションごとに固有のデータを保持し，ほかのユーザからアクセスされない表のため，実体化以外では，表の操作に対する排他は基本的に取得しません。一時表の排他制御について説明します。

(1)　一時表の実体化時に取得する排他

一時表が実体化する時に，次に示す排他を取得します。

表 11-29　一時表の実体化時に取得する排他

資源種別名	内容	排他解除時期	
		トランザクション固有一時表	SQL セッション固有一時表
RDAR	RD エリア（表及びインデクス※1格納 RD エリア）	トランザクション決着時	DISCONNECT 文完了時
TABL	表		
RATM	ユーザディレクトリ表情報		
RAIM	ユーザディレクトリインデクス情報※1		
SBMB	ユーザディレクトリセグメント情報		
TEMP	一時表実体化管理情報		一時表の実体化後※2
TPID	ユーザ固有 ID 管理（一時表）		DISCONNECT 文完了時
RDAS	RD エリア状態管理		一時表の実体化後※2
RRAM, TRAL, TRAI	RD エリア管理情報		
PTBL	前処理表		DISCONNECT 文完了時

注※1

一時インデクスがある場合だけです。

注※2

この資源に対する排他は実体化時に一時的に取得する排他です。

(2)　一時表に対する操作で取得する排他

次の SQL 文を実行すると，前処理表（PTBL）に対してだけ排他を取得します。

- LOCK 文
- CREATE INDEX 文

- DROP INDEX 文
- DROP TABLE 文

なお，一時表に対して次の SQL 文を実行しても，ページ，行，キー値の排他は取得しません。

- SELECT 文
- INSERT 文
- UPDATE 文
- DELETE 文
- PURGE TABLE 文

(3) 排他待ち又は実行エラーになる操作

一時表を操作している場合，次に示す操作のどれかを同時に実行すると，排他待ち又は実行エラーになるおそれがあります。

- pdclose（一時表用 RD エリアのクローズ）
- pdhold（一時表用 RD エリアの閉塞）
- pdopen（一時表用 RD エリアのオープン）
- pdrels（一時表用 RD エリアの閉塞解除）
- pdmod コマンドの create rdarea 文（一時表用 RD エリアの追加）
- pdmod コマンドの initialize rdarea 文（一時表用 RD エリアの再初期化）
- pdmod コマンドの remove rdarea 文（一時表用 RD エリアの削除）
- CREATE INDEX 文（一時インデクスの定義）
- DROP INDEX 文（一時インデクスの削除）
- DROP SCHEMA 文（スキーマの削除）
- DROP TABLE 文（一時表の削除）

11.22.6 一時表使用時の制限事項

(1) 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
- 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）

(2) 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
  DROP INDEX
  DROP SCHEMA
  DROP TABLE
  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

12 インデクスの設計

この章では，B-tree 構造のインデクス及びプラグインインデクスを設計する上で検討する項目について説明します。

12.1　インデクスを設計するときの検討項目

表に対する処理性能を向上させるには，インデクスを作成します。ただし，効果的に作成しないと，逆に性能を劣化させることもあります。このため，より効果的なインデクスの作成方法を検討する必要があります。また，インデクスのユーザ用 RD エリアへの格納の仕方によって，表に対する処理性能や操作性が異なります。これらの点を考慮してインデクスを設計する必要があります。

インデクスを設計するときの検討項目を次の表に示します。

表 12-1　インデクスを設計するときの検討項目

記載箇所	設計作業ごとの検討項目	説明
「インデクス」を参照してください。	インデクスの作成	SQL 文に適した B-tree 構造のインデクスを検討するときに，参照してください。
「インデクスの横分割」を参照してください。	インデクスの横分割	B-tree 構造のインデクスの RD エリア構成について検討するときに，参照してください。
「プラグインインデクス」を参照してください。	プラグインインデクスの作成	プラグインインデクスを作成するときに，参照してください。
「プラグインインデクスの横分割」を参照してください。	プラグインインデクスの横分割	プラグインインデクスの RD エリア構成について検討するときに，参照してください。
「インデクスの優先順位」を参照してください。	インデクスの優先順位	検索する表に複数のインデクスが定義されている場合，HiRDB が使用するインデクスの優先順位について説明しています。SQL 文に適したインデクスを検討するときに，参照してください。

12.2　インデクス

ここでは，B-tree 構造のインデクスの設計について説明します。

12.2.1　インデクスの作成

(1)　インデクスの効果

性能の向上

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。詳細は，マニュアル「HiRDB 解説」の「インデクスの基本構造」を参照してください。

(2)　適用基準

(a)　インデクス作成に適している列

インデクス作成に適している列を次に示します。

- データを絞り込むための条件に使用する列
  探索条件に使用する列にインデクスを作成すると，条件を満たすデータを効率良く取り出せます。
- 表の結合処理の条件として使用する列
  表を結合する場合は，結合列にインデクスを作成することで，効率の良い結合処理ができます。特に，結合列に指定することが多い外部キーには，インデクスを作成してください。
- ORDER BY，GROUP BY に指定する列
  探索条件に使用する列に加えて，ORDER BY，GROUP BY に指定する列をインデクス構成列に含めると，HiRDB が実行するソート処理を省略できることがあるため，効率良く処理ができます。ソート処理を省略できる条件については，マニュアル「HiRDB コマンドリファレンス」の「アクセスパス表示ユティリティ（pdvwopt)」の「ORDER BY，GROUP BY を指定した検索に使用する表のインデクス定義」を参照してください。

インデクスを構成する列の組み合わせや順序については，「インデクス構成列の検討」を参照してください。

(b)　インデクス作成に適さない列

次に示す列にインデクスを作成すると性能が低下するため，インデクス作成には適していません。

- 更新頻度の高い列
- データの重複度が高い列（データの種類が少ない列）

それぞれの理由について，次に説明します。

- 更新頻度の高い列

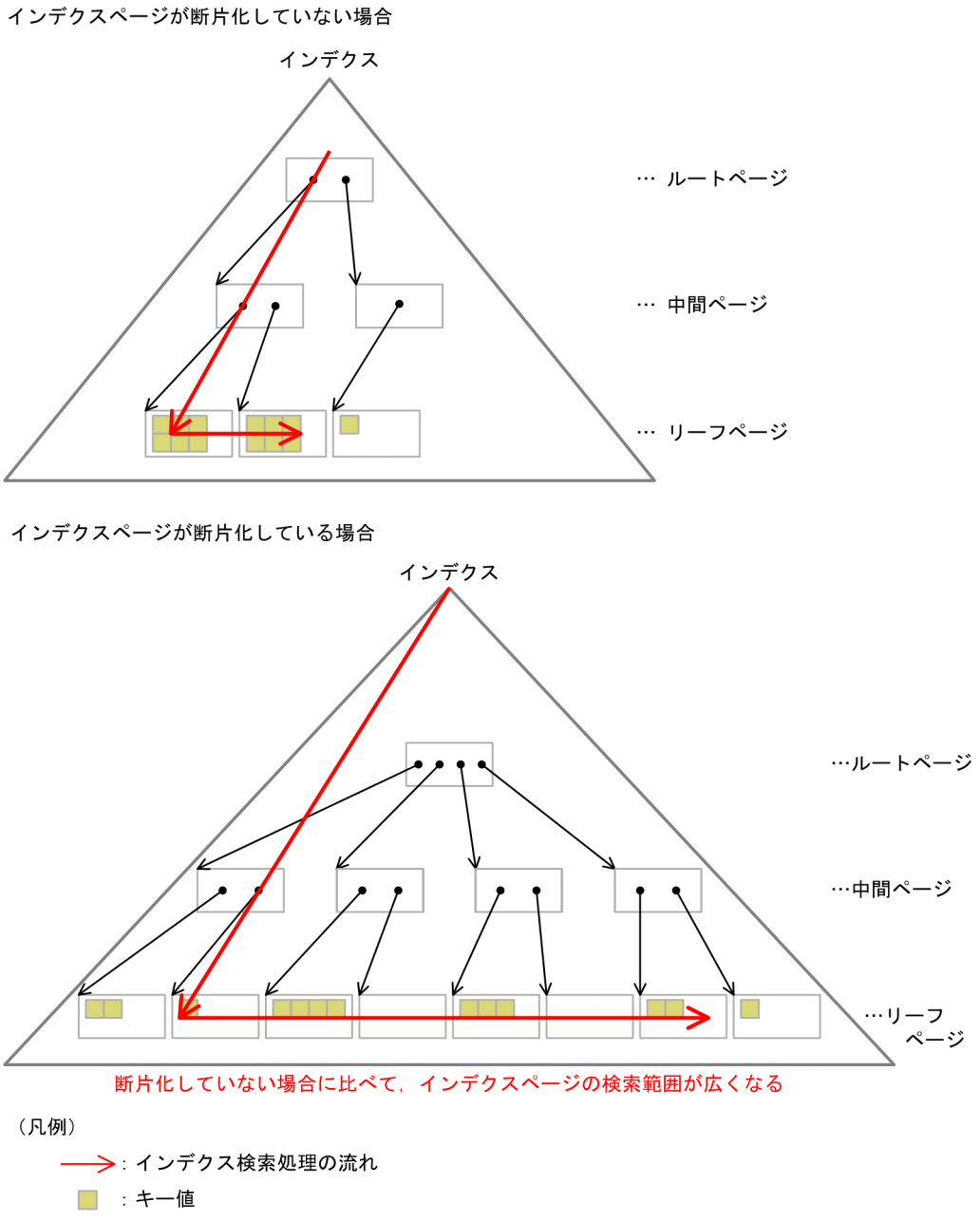
- 更新 SQL の性能への影響

  インデクス構成列の値を更新すると，キー値も更新されます。このインデクスメンテナンスは，更新 SQL の延長で実行されるため，更新 SQL の実行頻度が高い場合は，性能が低下します。

- インデクス格納ページの断片化

  インデクスのキー値はキー値順に並んで格納されているため，キー値が更新されると格納位置が変わり，インデクスのページには断片化した空き領域が発生します。更新 SQL の実行頻度が高い場合は，断片化の進行が早いため，次のインデクス再編成を実行するまでに，インデクスを使用した検索性能が低下します。

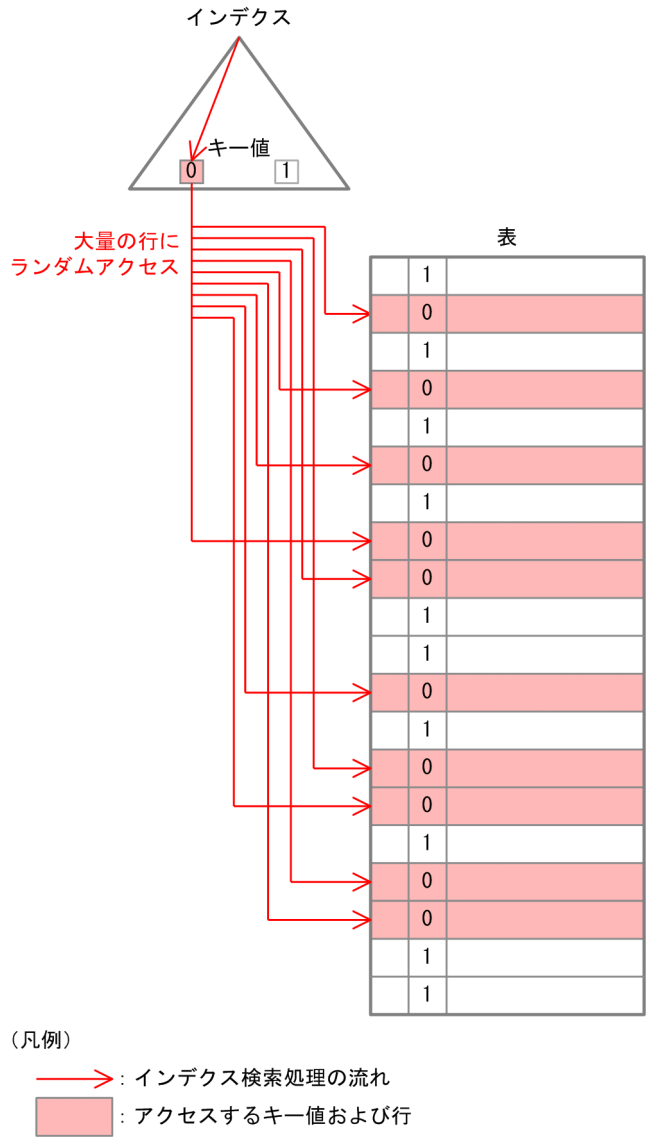
図 12-1　インデクスページの断片化による検索性能への影響



- データの重複度が高い列（データの種類が少ない列）

データの重複度が高い列とは，例えば，0 と 1 の 2 種類の値しか持たないフラグやステータス情報を設定する列が該当します。このような列に対して単一列インデクスを作成しても，行を絞り込めないため，大量の行にアクセスします。インデクスを使った検索は，行の格納順序は意識しないランダムなアクセスです。したがって，大量の行にアクセスすると，インデクスを使用しない場合よりも時間が掛かることがあります。

図 12–2　フラグ列のインデクスを使用した検索処理の例



データの重複度が高い列に対して探索条件を指定する場合は，ほかの探索条件に指定した列にインデクスを作成して，絞り込んでください。絞り込めない場合は，データの重複度が高い列も組み合わせて，複数列インデクスを作成してください。この場合，インデクス構成列の順序は，データの重複度が高い列を一番後ろにしてください。

(3)　作成方法

表にインデクスを作成するには，定義系 SQL の CREATE INDEX を実行します。

(4) 共通規則

1. 一つの表に最大 255 個のインデクスを定義できます。
2. ナル値を含む列又は行のない列に対してもインデクスを定義できます。
3. ビュー表にはインデクスを作成できません。

(5) インデクスを定義できないデータ型

次に示すデータ型の列にはインデクスを定義できません。

- BLOB
- BINARY
- 抽象データ型

(6) インデクスのキー長の上限

インデクスのキー長は，次に示す条件を満たす必要があります。この条件を満たさないとインデクスを定義できません。

```
インデクスのキー長（バイト）
≦MIN｛（インデクス格納RDエリアのページサイズ÷2）-1242，4036｝
```

インデクス格納 RD エリアのページサイズが 4096 バイトの場合は，キー長が最大 806 バイトのインデクスが定義できます。インデクスのキー長については，表「インデクスのキー長一覧」を参照してください。

なお，複数列インデクスの場合は，複数列インデクスを構成する各列のキー長の合計がインデクスのキー長となります。

(7) 注意事項

一つの表に同じインデクスを二つ以上作成できません。異なるインデクス名であっても同じインデクスと扱われる場合の例を次に示します。

●単一列インデクスの場合

```
CREATE INDEX インデクス1 ON 表1（列1 ASC）
CREATE INDEX インデクス2 ON 表1（列1 DESC）
```

この場合，インデクス 2 はインデクス 1 と同じインデクスとして扱われます。このため，先に定義したインデクス 1 が有効になります。

●複数列インデクスの場合

```
CREATE INDEX インデクス1 ON 表1（列1 ASC,列2 ASC）
CREATE INDEX インデクス2 ON 表1（列1 DESC,列2 DESC）
```

又は

```
CREATE INDEX インデクス1 ON 表1（列1 ASC,列2 DESC）
CREATE INDEX インデクス2 ON 表1（列1 DESC,列2 ASC）
```

この場合，インデクス 1 とインデクス 2 は同じインデクスとして扱われます。このため，先に定義したインデクス 1 が有効になります。なお，次に示す場合は，互いに異なるインデクスになります。

```
CREATE INDEX インデクス1 ON 表1（列1 DESC,列2 DESC）
CREATE INDEX インデクス2 ON 表1（列1 ASC,列2 DESC）
```

12.2.2 インデクス構成列の検討

(1) 探索条件を満たすデータだけに絞り込む場合

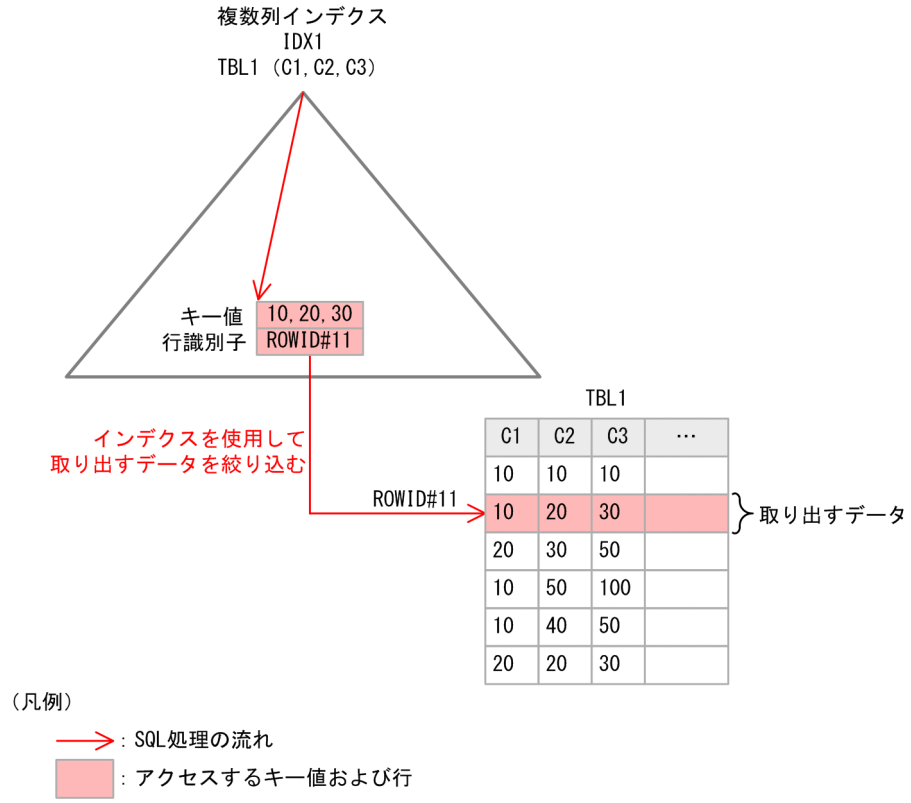
(a) インデクス構成列の組み合わせ

探索条件を満たすデータだけに絞り込む場合は，探索条件に指定する列にインデクスを作成してください。AND 演算子を使用して，複数の探索条件を満たすデータだけに絞り込む場合は，探索条件に指定する複数の列に対して，一つのインデクスを作成してください。すべての探索条件を一つのインデクスで絞り込む方が，効率が良いです。複数の探索条件を一つのインデクスで絞り込む例を次に示します。

（例）

- SQL 文
  SELECT * FROM TBL1 WHERE C1 = 10 AND C2 = 20 AND C3 = 30
- インデクス定義
  CREATE INDEX IDX1 ON TBL1(C1, C2, C3)

図 12-3　複数の探索条件を一つのインデクスで絞り込む例



[注意事項]

AND 演算子を使用して，複数の探索条件を満たすデータだけに絞り込む場合に，探索条件のそれぞれの列に単一列インデクスを作成しても，どれか一つのインデクスしか使用しません※。インデクスに含まれない列の条件は，行データを参照して評価します。次に例を示します。

(例)

- SQL 文

  SELECT * FROM TBL1 WHERE C1 = 10 AND C2 = 20 AND C3 = 30

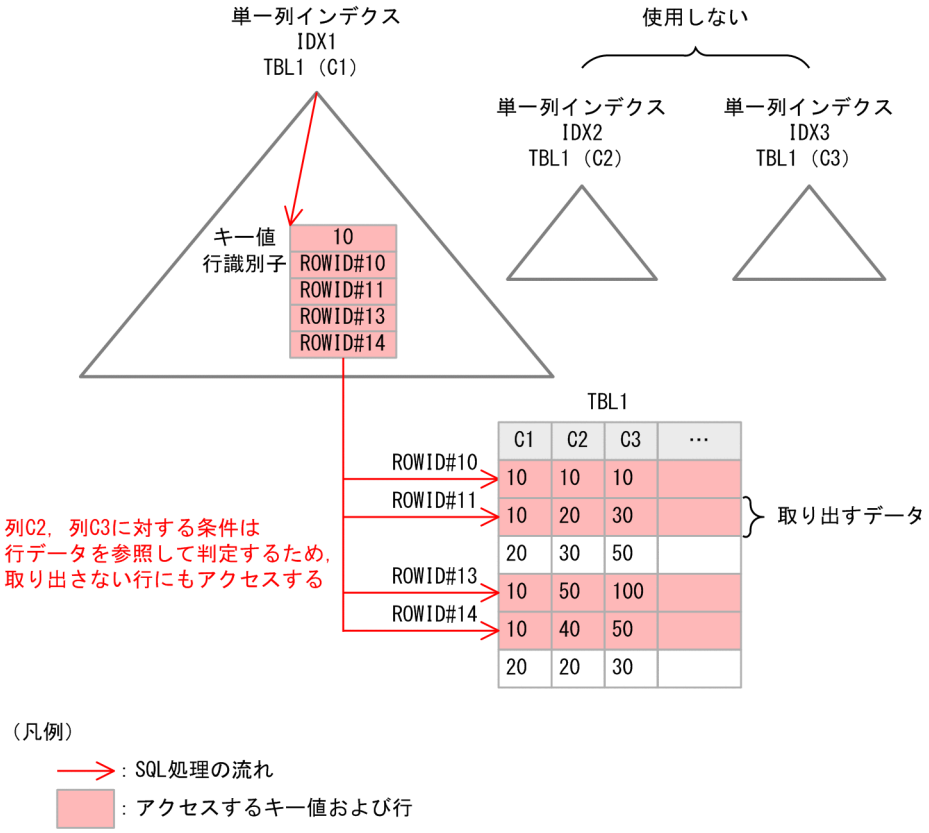
- インデクス定義

  CREATE INDEX IDX1 ON TBL1(C1)

  CREATE INDEX IDX2 ON TBL1(C2)

  CREATE INDEX IDX3 ON TBL1(C3)

図 12-4　探索条件のそれぞれの列に単一列インデクスを作成した場合



注※

SQL 最適化オプションの指定内容によっては，インデクスを複数使用することがあります。ただし，一つの複数列インデクスを使用する方が効率が良いです。

(b)　インデクス構成列の順序

複数列インデクスの場合，第 1 構成列は，必ず「=」条件を指定する列にします。また，「=」条件を指定する場合が多い列ほど先に指定します。これによって，インデクス内の検索範囲を小さくできるため，インデクス内の検索時間を短縮できます。

インデクス構成列順序による検索範囲の違いについて，次に例を示します。

（例 1）

「=」条件を指定した列が第 1 構成列である場合

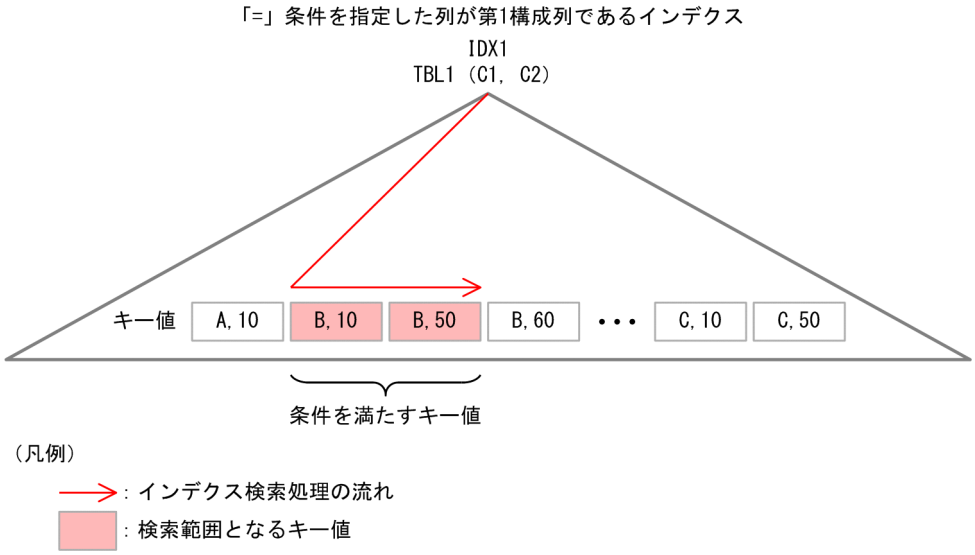
- SQL 文

  SELECT * FROM TBL1 WHERE C1 = 'B' AND C2 BETWEEN 10 AND 50

- インデクス定義

  CREATE INDEX IDX1 ON TBL1(C1,C2)

図 12-5 「=」条件を指定した列が第 1 構成列である場合



（例 2）

「=」条件を指定した列が第 1 構成列ではない場合

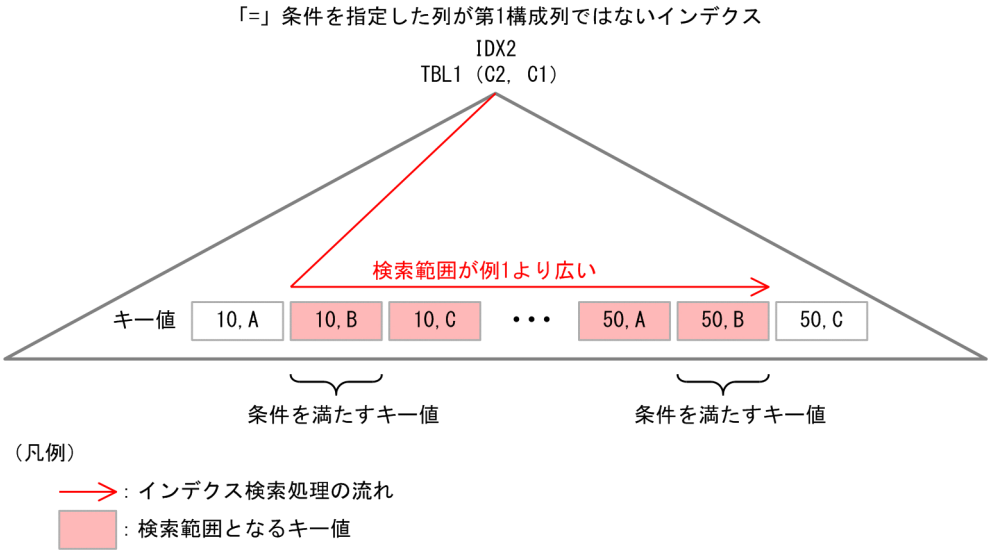
- SQL 文

  SELECT * FROM TBL1 WHERE C1 = 'B' AND C2 BETWEEN 10 AND 50

- インデクス定義

  CREATE INDEX IDX2 ON TBL1(C2,C1)

図 12-6 「=」条件を指定した列が第 1 構成列ではない場合



「=」条件を指定した列については，インデクス構成列の順序を探索条件の指定順序と一致させる必要はありません。次に例を示します。

（例）

- SQL 文

SELECT * FROM TBL1 WHERE C3 = 10 AND C1 = 20 AND C2 = 30

- インデクス定義

  CREATE INDEX IDX1 ON TBL1(C2, C3, C1)

［説明］

インデクス構成列の順序が(C3, C1, C2)である必要はありません。

［注意事項］

「=」条件を指定した場合でも，フラグやステータス情報を設定する列のようにデータの重複度が高い列は，インデクス構成列には適しません。詳細は，「インデクス作成に適さない列」を参照してください。

(2) 表を結合する場合

(a) インデクス構成列の組み合わせ

表を結合する場合は，結合列にインデクスを作成してください。結合列にインデクスを作成することで，表の結合方法を効率の良い NESTED LOOPS JOIN にできます。

- 結合列が複数ある場合は，すべての結合列を含んだ複数列インデクスを作成してください。すべての結合列がインデクス構成列に含まれていないと，NESTED LOOPS JOIN を効率良く処理できません。詳細は，マニュアル「HiRDB パフォーマンスガイド」の「効率の悪い NESTED LOOPS JOIN の対策」を参照してください。
- NESTED LOOPS JOIN では，内表の結合列に作成したインデクスを利用します。内表に探索条件が指定されている場合は，結合列に加えて，探索条件に指定した列をインデクス構成列に含めてください。

結合列と探索条件に指定した列を含んだインデクスの例を次に示します。

(例)

- SQL 文

  SELECT * FROM TBL1 INNER JOIN BY NEST TBL2
  ON TBL1.C1 = TBL2.C1 AND TBL1.C2 = TBL2.C2
  WHERE TBL1.C1=10 AND TBL2.C3 BETWEEN 100 AND 500

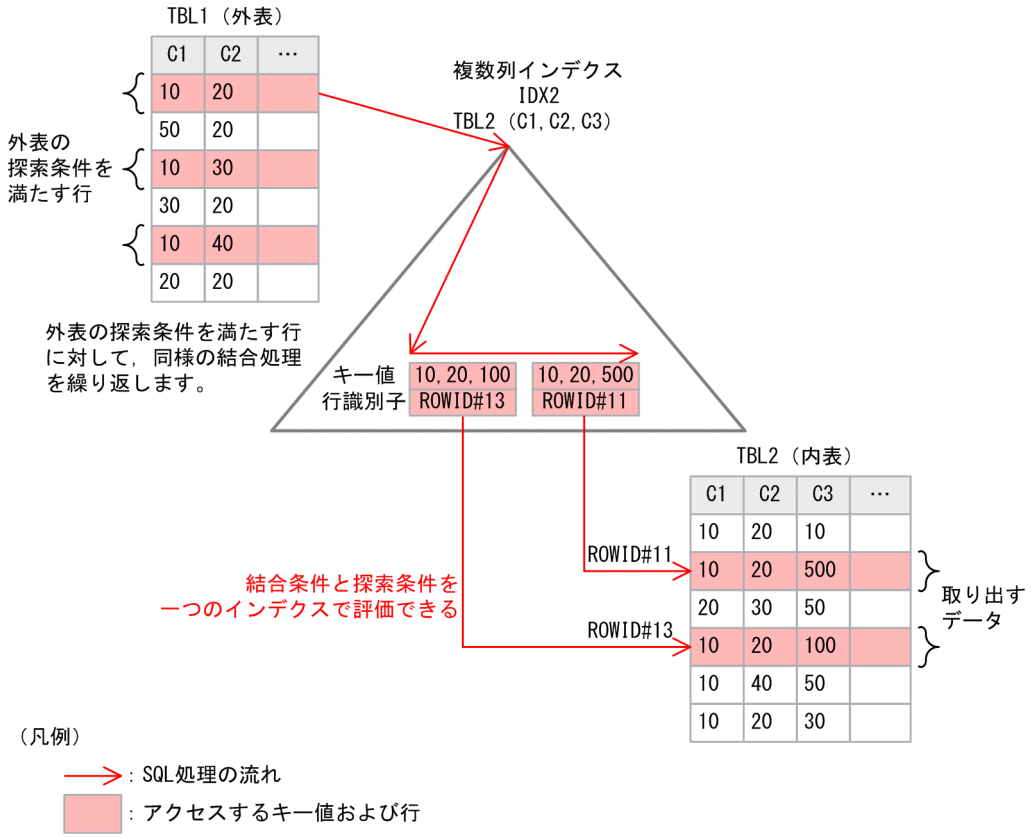
- 内表 TBL2 のインデクス定義

  CREATE INDEX IDX2 ON TBL2(C1,C2,C3)

- 外表 TBL1 のインデクス定義

  CREATE INDEX IDX1 ON TBL1(C1)

図 12-7　結合列と探索条件に指定した列を含んだインデクスの場合



探索条件に指定した列がインデクスに含まれていない場合，探索条件は行データを参照して評価することになります。次に例を示します。

（例）

- SQL 文

  SELECT * FROM TBL1 INNER JOIN BY NEST TBL2
  　ON TBL1.C1 = TBL2.C1 AND TBL1.C2 = TBL2.C2
  　WHERE TBL1.C1=10 AND TBL2.C3 BETWEEN 100 AND 500

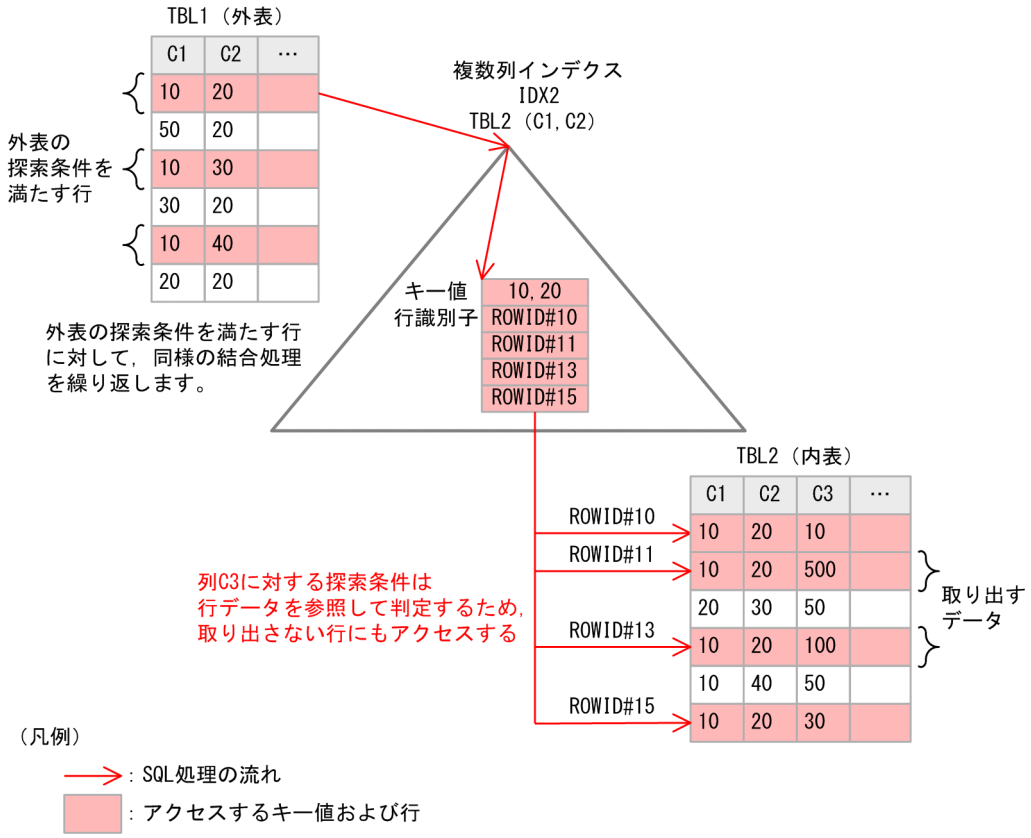
- 内表 TBL2 のインデクス定義

  CREATE INDEX IDX2 ON TBL2(C1,C2)

- 外表 TBL1 のインデクス定義

  CREATE INDEX IDX1 ON TBL1(C1)

図 12-8　探索条件に指定した列がインデクスに含まれていない場合



(b)　インデクス構成列の順序

インデクス構成列の順序は，結合条件及び探索条件の指定方法によって決定してください。詳細は「探索条件を満たすデータだけに絞り込む場合」の「インデクス構成列の順序」を参照してください。

(3)　探索条件を指定し，かつ ORDER BY，GROUP BY を指定している場合

(a)　インデクス構成列の組み合わせ

探索条件に使用する列に加えて，ORDER BY，GROUP BY に指定する列をインデクス構成列に含めると，HiRDB が実行するソート処理を省略できることがあるため，効率良く処理ができます。ソート処理を省略できる条件については，マニュアル「HiRDB コマンドリファレンス」の「アクセスパス表示ユティリティ（pdvwopt)」の「ORDER BY，GROUP BY を指定した検索に使用する表のインデクス定義」を参照してください。

(b)　インデクス構成列の順序

探索条件として指定する列，グループ分け又はソートする列の順で構成する複数列インデクスを作成します。

次に例を示します。

（例）

- SQL 文

SELECT * FROM TBL1 WHERE C3 = 10 AND C1 = 20
ORDER BY C4 DESC,C2 ASC

- インデクス定義
  CREATE INDEX IDX1 ON TBL1(C3 ASC, C1 ASC, C4 DESC, C2 ASC)
  注
  インデクス構成列ごとの ASC，DESC の指定は，ORDER BY の指定と合わせてください。

(4) 一つの表に作成した複数のインデクスの構成列が，重複している場合

SQL ごとにインデクスを検討した結果，一つの表に対して複数のインデクスを作成することがあります。しかし，インデクスの数が多いと性能に影響を与えることがあります。詳細は，「インデクスの数が性能に与える影響」を参照してください。

そこで，構成列が重複しているインデクスが複数ある場合は，一つのインデクスに統合できるか検討してください。インデクスを統合する場合，性能を優先したい SQL の条件に指定した列が，インデクスの第 1 構成列から連続する順序になるようにしてください。次に例を示します。

(例)

- SQL 文 1（性能を優先したい SQL）
  SELECT * FROM TBL1 WHERE C3 = 10 AND C1 = 20
- SQL 文 2
  SELECT * FROM TBL1 WHERE C3 = 10 AND C2 = 30
- インデクス定義
  CREATE INDEX IDX1 ON TBL1(C3, C1, C2)

図 12-9 SQL 文 1 を検索する場合

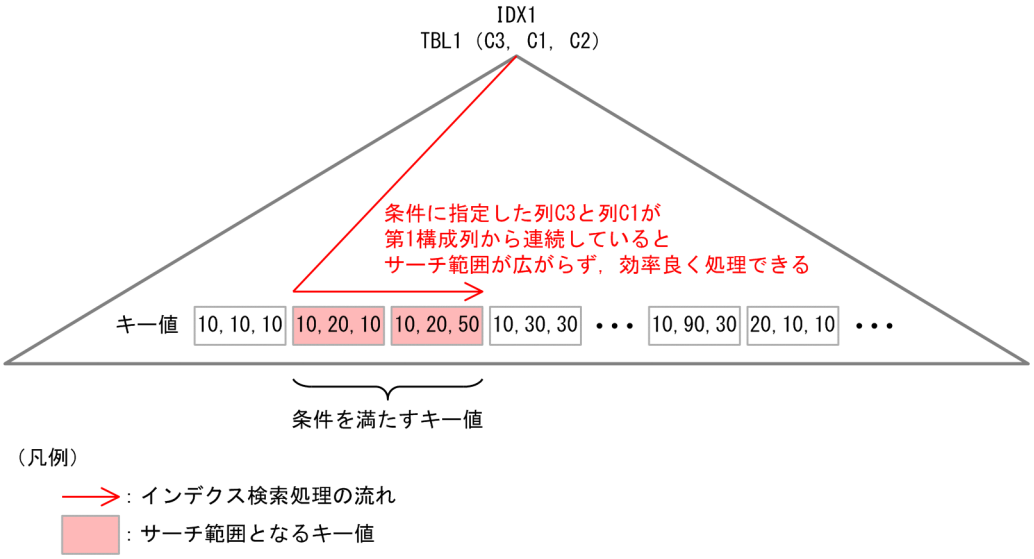
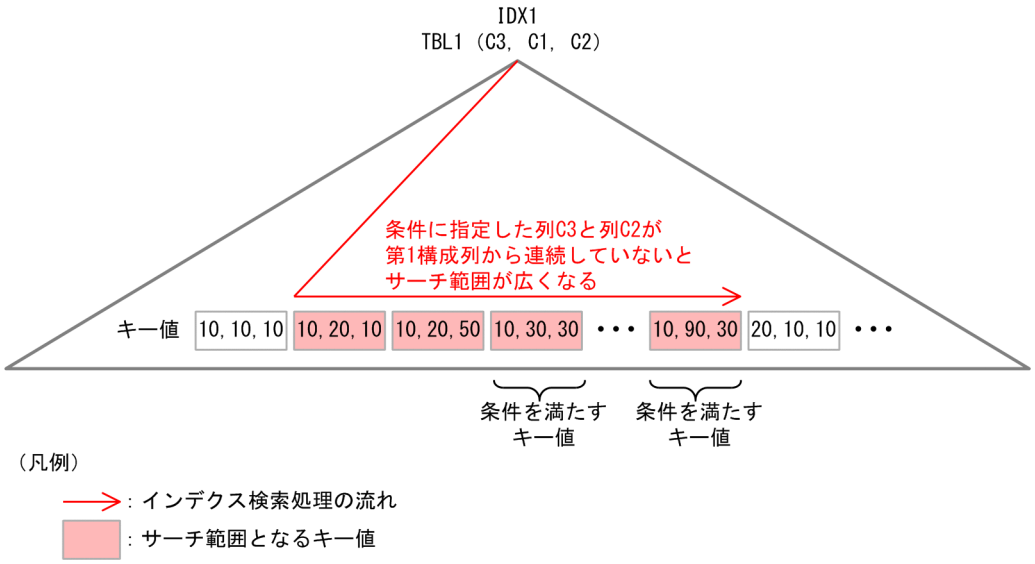


図 12-10　SQL 文 2 を検索する場合



どちらの SQL の性能も優先したい場合は，それぞれの SQL に合わせたインデクスを作成してください。次に例を示します。

（例）

- SQL 文 1
  SELECT * FROM TBL1 WHERE C3 = 10 AND C1 = 20
- SQL 文 2
  SELECT * FROM TBL1 WHERE C3 = 10 AND C2 = 30
- インデクス定義
  CREATE INDEX IDX1 ON TBL1(C3, C1)
  CREATE INDEX IDX2 ON TBL1(C3, C2)

[注意事項]

インデクスの追加や，構成列を変更すると，SQL のアクセスパスが変わることがありますので，その表を操作する SQL のアクセスパスを再確認してください。

12.2.3　インデクスを複数使用する場合

表には複数のインデクスを作成できます。一つのインデクス（単一列インデクス又は複数列インデクス）を使用するよりも複数のインデクスを使用した方が，より行を絞り込める場合に有効です。

12.2.4　除外キー値を設定したインデクスの使用

インデクスを定義した列のデータは，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。ところが，インデクス中にはナル値のような余分なキー値もあります。このように，すべての構成列の値がナル値から成る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。

(1)　インデクスに除外キー値を設定した場合の効果

インデクスに除外キー値を設定することで，次に示す効果が期待できます。

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，VALUE，又は CASE 式を指定した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

(2)　設定方法

除外キー値を設定するには，定義系 SQL の CREATE INDEX に EXCEPT VALUES オプションを指定します。

(3)　注意

- 除外キー値を指定できるのはナル値だけで構成されたキー値です。
- 非ナル値制約の列を含むインデクスには除外キー値を指定できません。
- クラスタキー指定を指定したインデクスには除外キー値を指定できません。
- インデクス順にアンロードをする場合，除外キー値を持つインデクスは指定できません。

12.2.5　インデクスの数が性能に与える影響

行を追加又は削除するとき，該当する表に作成しているすべてのインデクスが更新されます。このため，作成するインデクスの数が増えると，インデクスの更新処理のオーバヘッドが増加します。そのため，次に示す点を考慮してインデクスを作成します。

- 更新の多い列にはインデクスを定義しないようにします。
- 複数列インデクスを作成して，インデクスの数を減らします。

- HiRDB/パラレルサーバの場合に，主に全件検索をするときは，並列処理の効果を上げるため，最低限の数だけインデクスを作成するようにします。

12.3　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できます。

12.3.1　分割キーインデクスと非分割キーインデクス

横分割インデクスを設計する前に，**分割キーインデクス**と**非分割キーインデクス**について理解する必要があります。

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，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。

注

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。

(1)　単一列分割の場合

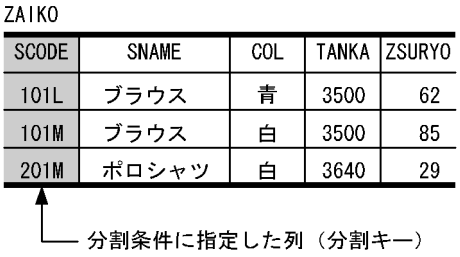
次に示すどちらかの条件を満たす場合，そのインデクスは分割キーインデクスになります。

〈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

次に示す 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 12-11　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。

(2) 複数列分割の場合

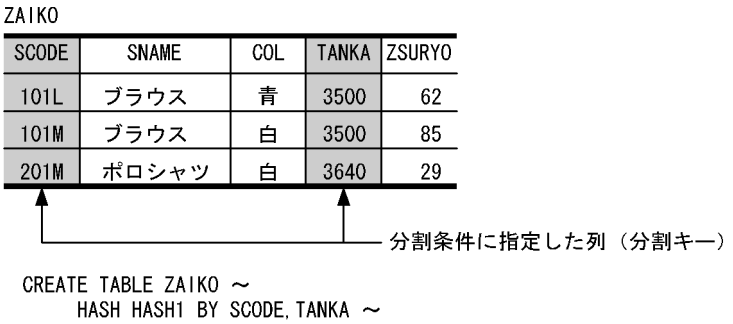
次に示す条件を満たす場合，そのインデクスは分割キーインデクスになります。

〈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

次に示す 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 12-12 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。

12.3.2 インデクスの分割指針

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によって，次の表に示すとおりインデクスの分割指針が異なります。

表 12–2 インデクスの分割指針

インデクスの種類	HiRDB/シングルサーバの場合	HiRDB/パラレルサーバの場合	
		表をサーバ内横分割する場合	表をサーバ間横分割する場合
インデクスが分割キーインデクスの場合	横分割表に対応させてインデクスも横分割します。	横分割表に対応させてインデクスも横分割します。	横分割表に対応させてインデクスも横分割します。
インデクスが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※。	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※。	

注※

非分割キーインデクスは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。具体的には，次に示すアクセスパスを使用した検索ができなくなるため，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。

- KEY SCAN MERGE JOIN
- LIST SCAN MERGE JOIN
- L-KEY R-LIST MERGE JOIN
- L-KEY R-SORT MERGE JOIN
- L-LIST R-KEY MERGE JOIN
- L-LIST R-SORT MERGE JOIN
- L-SORT R-KEY MERGE JOIN
- L-SORT R-LIST MERGE JOIN

これらのアクセスパスについては，マニュアル「HiRDB コマンドリファレンス」のアクセスパス表示ユティリティ（pdvwopt コマンド）を参照してください。

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。

なお，マトリクス分割表にインデクスを定義する場合，非分割キーインデクスであっても分割キーと同様に横分割する必要があります。

12.3.3　設計上の考慮点

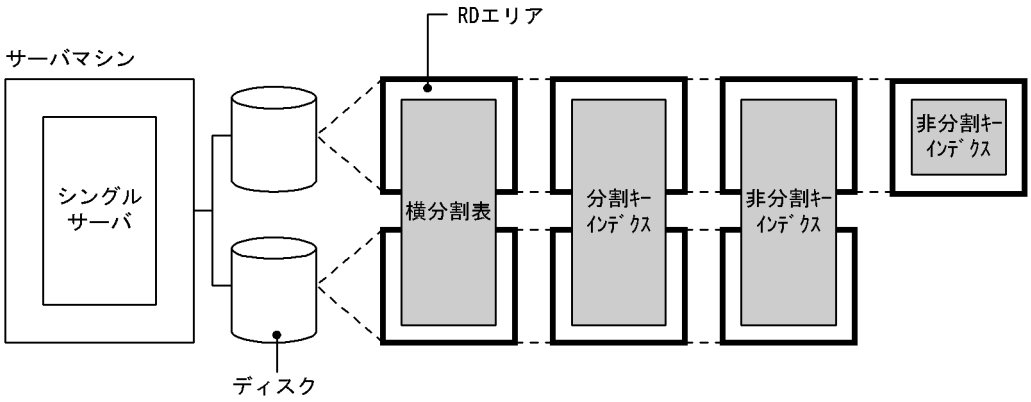
- 横分割表を格納するユーザ用 RD エリアと，横分割表のそれぞれに対応するインデクスを格納するユーザ用 RD エリアを分けます。これによって，それぞれのユーザ用 RD エリアの使用効率が良くなります。
- 表の中に一意にしたいキーがあるときは，このキーに UNIQUE を指定した分割キーインデクスを定義します。又は，分割キーに対してクラスタキーを指定します。また，非分割キーインデクスの場合でも，次のどちらかであれば UNIQUE を指定できます。
  - 非分割インデクス
  - 分割キーを任意の構成列に含む分割インデクス

  ただし，フレキシブルハッシュ分割した表では，インデクスに UNIQUE を指定できません。詳細については，マニュアル「HiRDB SQL リファレンス」の「CREATE INDEX」の「表を横分割する場合の UNIQUE 指定可否」を参照してください。

12.3.4　インデクスの横分割の例（HiRDB/シングルサーバの場合）

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 12–13　インデクスの横分割の例（HiRDB/シングルサーバの場合）



〔説明〕

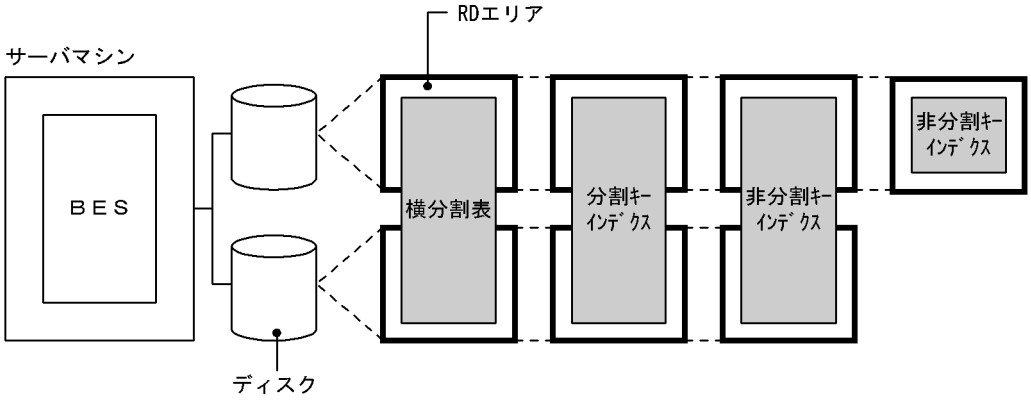
- ディスクのアクセス競合を避けるために，分割した表及びインデクスを格納する RD エリアを異なるディスク上に配置してください。
- 分割キーインデクスは横分割してください。
- 性能を重視する場合は，非分割キーインデクスを横分割しないでください。
- 操作性を重視する場合は，非分割キーインデクスを横分割してください。

12.3.5　インデクスの横分割の例（HiRDB/パラレルサーバの場合）

(1)　表をサーバ内横分割する場合

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 12–14　インデクスの横分割の例（サーバ内横分割の場合）



（凡例）ＢＥＳ：バックエンドサーバ

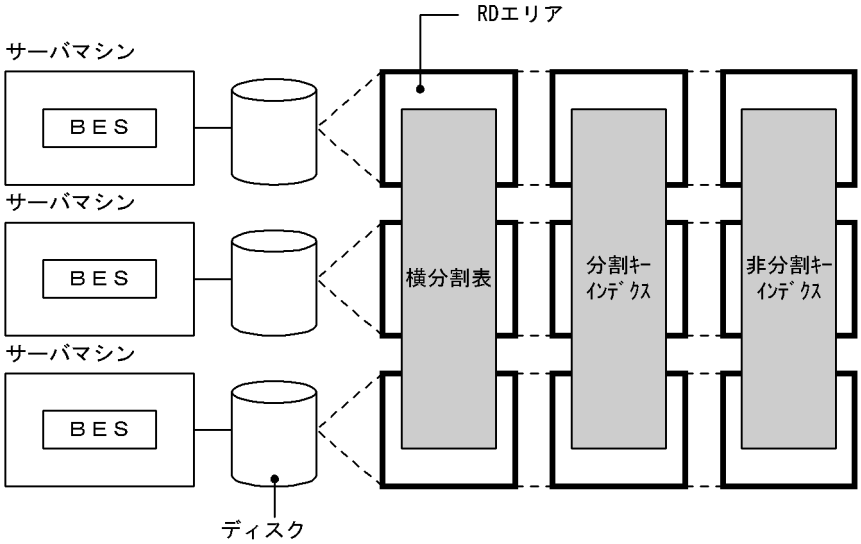
［説明］

- 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)　表をサーバ間横分割する場合

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 12–15　インデクスの横分割の例（サーバ間横分割の場合）



（凡例）ＢＥＳ：バックエンドサーバ

［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。

12.4.1　プラグインインデクスの効果と作成方法

ここでは，プラグインインデクスについて説明します。

(1)　プラグインインデクスの効果

性能の向上

プラグインを使用している場合に，プラグインインデクスを作成すると，表の検索性能を向上できます。プラグインで提供されるインデクス型を使用すると，複雑な検索を高速にできます。

(2)　作成方法

表にプラグインインデクスを作成するには，定義系 SQL の CREATE INDEX を実行します。

(3)　注意

プラグインによっては，あらかじめ指定されたプラグインによるプラグインインデクスの定義を前提としていることがあります。その場合，プラグインインデクスを定義しないで，プラグインインデクスを利用する関数を指定すると，実行時にエラーが返されることがあります。

(4)　プラグインインデクスの一括作成

プラグインインデクスは，データベース作成ユティリティ（pdload）による一括作成ができます。プラグインインデクスの一括作成については，「プラグインが提供する抽象データ型を定義した表の作成」を参照してください。

12.5　プラグイン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，プラグインインデクスも複数のユーザ LOB 用 RD エリアにわたって横分割する必要があります。

12.5.1　プラグインインデクスの横分割の効果

操作性の向上

プラグインインデクスの一括作成をするときに，ユーザ LOB 用 RD エリアごとに独立した運用ができます。

12.5.2　定義方法

プラグインインデクスの横分割の定義方法については，「プラグインが提供する抽象データ型を定義した表の作成」を参照してください。

12.5.3　プラグインインデクスの横分割の形態

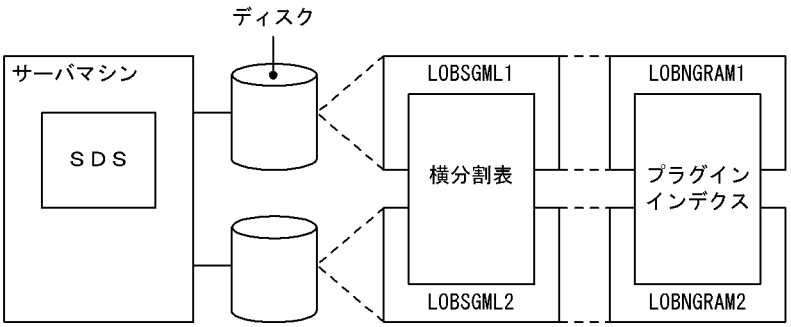
HiRDB/シングルサーバと HiRDB/パラレルサーバの，それぞれの場合でのプラグインインデクスの横分割の形態を次に示します。

(1)　HiRDB/シングルサーバの場合

HiRDB/シングルサーバの場合は，横分割した表に対応させて，複数のディスク上のユーザ LOB 用 RD エリアにわたってプラグインインデクスを横分割できます。

プラグインインデクスの横分割の形態および横分割の例を次の図に示します。

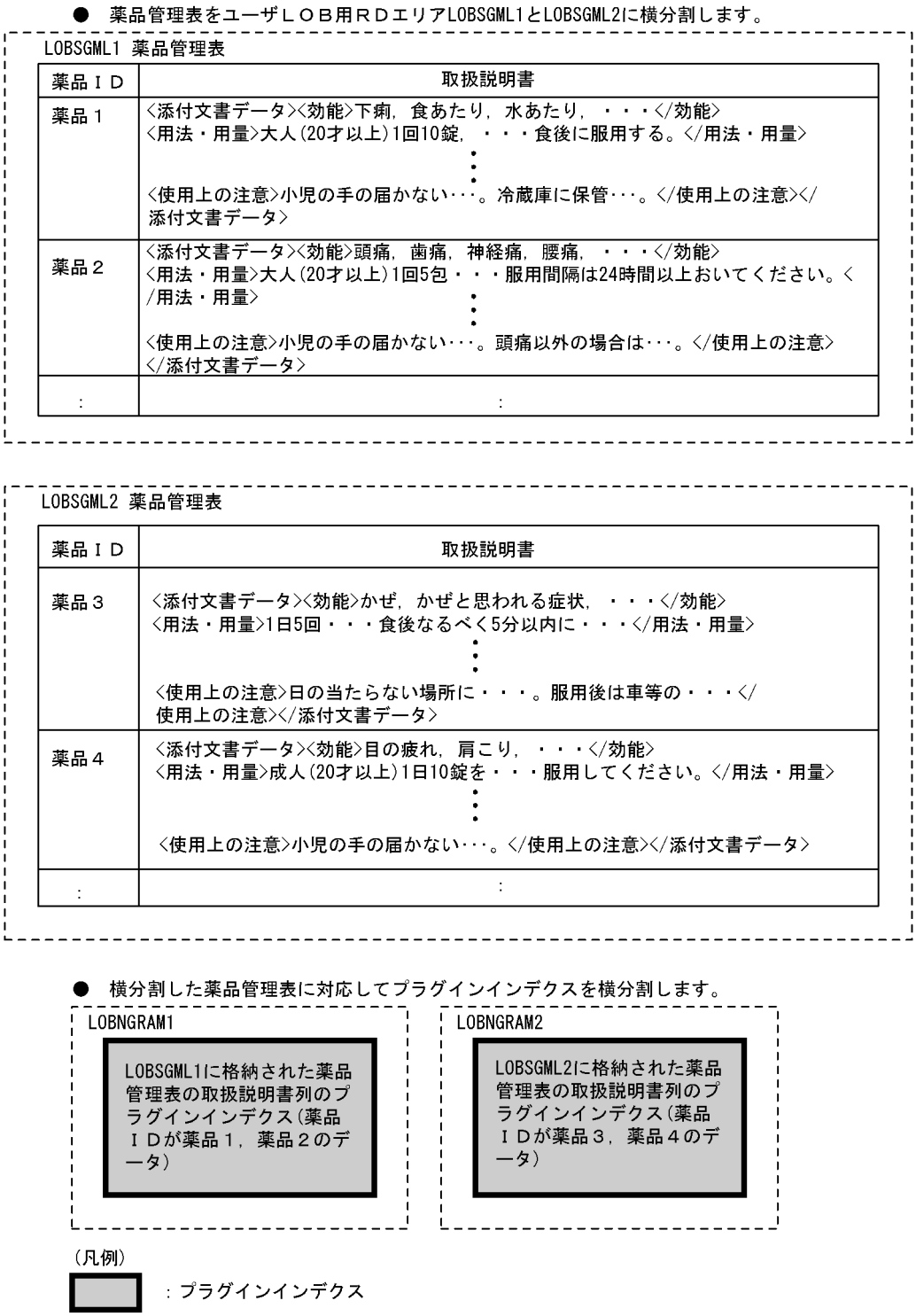
図 12–16　プラグインインデクスの横分割の形態（HiRDB/シングルサーバの場合）



（凡例）
ＳＤＳ：シングルサーバ
LOBSGML1～LOBSGML2及び
LOBNGRAM1～LOBNGRAM2：ユーザＬＯＢ用ＲＤエリア

注　表とプラグインインデクスの対応は１対１になります。

図 12-17　プラグインインデクスの横分割（キーレンジ分割）の例（HiRDB/シングルサーバの場合）



〔説明〕

取扱説明書列にプラグインインデクスが設定してあるとします。

薬品管理表を薬品 ID を条件として，ユーザ LOB 用 RD エリア LOBSGML1，LOBSGML2 にわたって横分割しています。これに対応して，プラグインインデクスを LOBNGRAM1，LOBNGRAM2 に格納しています。

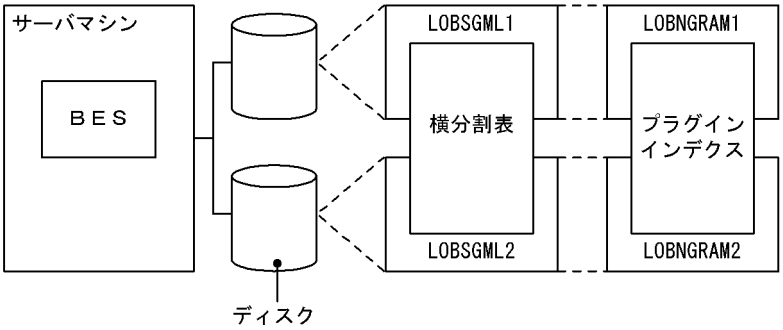
(2) HiRDB/パラレルサーバの場合

HiRDB/パラレルサーバの場合は，横分割した表に対応して，複数のサーバマシン又はバックエンドサーバに配置されたユーザ LOB 用 RD エリアにわたってプラグインインデクスを横分割できます。

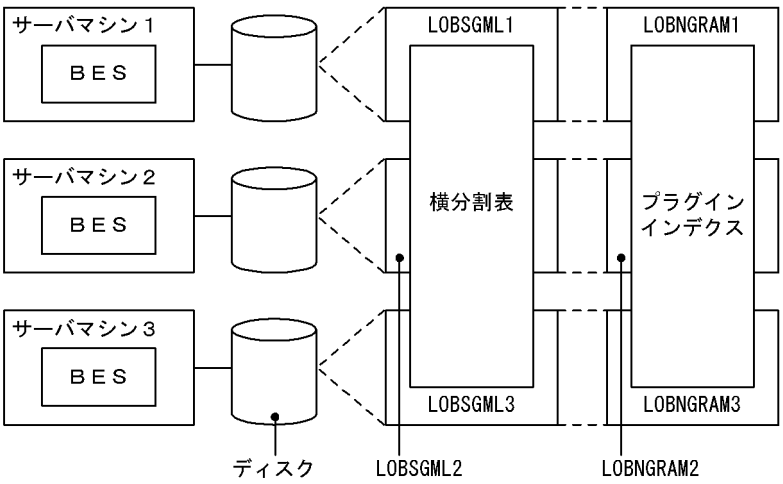
プラグインインデクスの横分割の形態および横分割の例を次の図に示します。

図 12-18 プラグインインデクスの横分割の形態（HiRDB/パラレルサーバの場合）

● バックエンドサーバ内の横分割



● バックエンドサーバ間の横分割



（凡例）
ＢＥＳ：バックエンドサーバ
LOBSGML1～LOBSGML3及び
LOBNGRAM1～LOBNGRAM3：ユーザＬＯＢ用ＲＤエリア

注　表とプラグインインデクスの対応は１対１になります。

図 12-19　プラグインインデクスの横分割（キーレンジ分割）の例（HiRDB/パラレルサーバの場合）

● 薬品管理表をユーザＬＯＢ用ＲＤエリアLOBSGML1～LOBSGML3に横分割します。

LOBSGML1 薬品管理表

薬品ＩＤ	取扱説明書
薬品１	<添付文書データ><効能>下痢，食あたり，水あたり，・・・</効能> <用法・用量>大人(20才以上)1回10錠，・・・食後に服用する。</用法・用量> : <使用上の注意>小児の手の届かない･･･。冷蔵庫に保管･･･。</使用上の注意></添付文書データ>
薬品２	<添付文書データ><効能>頭痛，歯痛，・・・</効能><用法・用量>大人(20才以上)1回5包・・・服用間隔は24時間以上おいてください。</用法・用量> : <使用上の注意>小児の手の届かない･･･。頭痛以外の場合は･･･。</使用上の注意> </添付文書データ>
:	:

LOBSGML2 薬品管理表

薬品ＩＤ	取扱説明書
薬品３	<添付文書データ><効能>かぜ，かぜと思われる症状，・・・</効能> <用法・用量>1日5回・・・食後なるべく5分以内に・・・</用法・用量> : <使用上の注意>日の当たらない場所に・・・。服用後は車等の・・・</使用上の注意></添付文書データ>
薬品４	<添付文書データ><効能>目の疲れ，肩こり，・・・</効能> <用法・用量>成人(20才以上)1日10錠を・・・服用してください。</用法・用量> : <使用上の注意>小児の手の届かない･･･。</使用上の注意></添付文書データ>
:	:

LOBSGML3 薬品管理表

薬品ＩＤ	取扱説明書
薬品５	<添付文書データ><効能>打撲，ねんざ，筋肉痛，・・・</効能> <用法・用量>適当な大きさに切り，・・・１日１～２回・・・</用法・用量> : <使用上の注意>定められた使用方法を厳守してください。・・・</使用上の注意> </添付文書データ>
薬品６	<添付文書データ><効能>すり傷，切り傷，・・・</効能> <用法・用量>成人(20才以上)1日１回患部に噴霧・・・浸して塗布。</用法・用量> : <使用上の注意>万一目に入った場合には･･･。</使用上の注意></添付文書データ>
:	:

● 横分割した薬品管理表に対応してプラグインインデクスを横分割します。

（凡例）

：プラグインインデクス

〔説明〕

取扱説明書列にプラグインインデクスが設定してあるとします。

薬品管理表を薬品 ID を条件として，ユーザ LOB 用 RD エリア LOBSGML1～LOBSGML3 にわたって横分割しています。これに対応して，プラグインインデクスを LOBNGRAM1，LOBNGRAM2，LOBNGRAM3 に格納しています。

12.5.4　設計上の考慮点

横分割表を格納するユーザ LOB 用 RD エリアと，横分割表のそれぞれに対応するプラグインインデクスを格納するユーザ LOB 用 RD エリアを分ける必要があります。

12.5.5　注意

横分割すると RD エリアの数が増えるため，RD エリア指定のデータベースのバックアップ時には，表とインデクスの対応が 1 対 1 であることに注意してください。

12.6 インデクスの優先順位

検索する表に複数のインデクスが定義されている場合，HiRDB が使用するインデクスの優先順位について，次の表に示します。SQL で使用したいインデクスの優先順位を確認してください。使用したいインデクスよりも優先順位の高い別のインデクスがある場合は，使用インデクスの SQL 最適化指定で，使用したいインデクス名を指定してください。詳細については，マニュアル「HiRDB SQL リファレンス」の「使用インデクスの SQL 最適化指定」を参照してください。

表 12-3 HiRDB が使用するインデクスの優先順位

優先順位	HiRDB が優先して使用するインデクスの内容	インデクスがある列（C1）に対する条件の指定例
1 （必ず使用する※1）	インデクス型プラグイン専用関数の条件が「IS TRUE」であり，この関数の第一引数の列に指定したプラグインインデクス	contains(C1,'…') IS TRUE
	構造化繰返し述語の探索条件に含まれるすべての列をインデクス構成列に含むインデクス	ARRAY(C1,C2)[ANY] (C1='ABC' and C2=10) C1，C2 の複数列インデクスを定義しています。
2	プラグイン提供関数の条件が「IS TRUE」であり，この関数の第一引数の列に指定したプラグインインデクス	within(C1,'…') IS TRUE
3	「=」の制限条件の対象になる列にある，UNIQUE を指定したインデクス	C1=100
4	「=」の制限条件の対象になる列にあるインデクス	C1=100
5	「IS NULL」の制限条件の対象になる列にあるインデクス※2	C1 IS NULL
6	LIKE 述語，又は SIMILAR 述語のパターン文字列に定数で「%」による前方一致比較を指定した列にあるインデクス	C1 LIKE 'ABC%' C1 SIMILAR TO 'ABC%'
7	LIKE 述語，又は SIMILAR 述語のパターン文字列に定数で上記以外の前方一致比較を指定した列にあるインデクス	C1 LIKE 'ABC_' C1 SIMILAR TO 'ABC_'
8	IN 述語の制限条件の対象になる列にあるインデクス	C1 IN(10,20,30)
9	BETWEEN 述語の制限条件の対象になる列にあるインデクス	C1 BETWEEN 20 AND 40
	範囲条件を指定した列にあるインデクス	20<=C1 AND C1<=40
10	外への参照がない副問合せを使用した IN 述語の制限条件の対象になる列にある単一列インデクス	C1 IN(SELECT C1 FROM T2)
	外への参照がない副問合せを使用した限定述語「=ANY」又は「=SOME」の制限条件の対象になる列にある単一列インデクス	C1=ANY(SELECT C1 FROM T2) C1=SOME(SELECT C1 FROM T2)
11	>，>=，<及び<=の制限条件の対象になる列にあるインデクス	C1>50 C1<=200

優先順位	HiRDB が優先して使用するインデクスの内容	インデクスがある列（C1）に対する条件の指定例
12※3	スカラ演算（システム定義スカラ関数，IS_USER_CONTAINED_IN_HDS_GROUP は除く）を指定した列にあるインデクス※2	length(C1)=10
13	NOT BETWEEN 述語の制限条件の対象になる列にあるインデクス	C1 NOT BETWEEN 10 AND 30
14	XLIKE 述語及び上記以外の LIKE 述語，又は SIMILAR 述語の制限条件の対象になる列にあるインデクス	C1 XLIKE '%ABC%' C1 LIKE '%ABC%' C1 SIMILAR TO '%ABC%'
15	集合関数（MIN 又は MAX）の引数に指定した列にあるインデクス※4	MIN(C1) MAX(C1)
16	結合条件列及びグループ分け又はソートの対象になる列にあるインデクス	ORDER BY C1
－	否定（NOT BETWEEN を除く）の制限条件の対象になる列にあるインデクス	C1 NOT LIKE '%ABC%' C1 IS NOT NULL
	上記以外の限定述語「ANY」又は「SOME」の制限条件の対象になる列にあるインデクス	C1>=ANY(SELECT C1 FROM T2) C1>SOME(SELECT C1 FROM T2)
	限定述語「ALL」の制限条件の対象になる列にあるインデクス	C1>ALL(SELECT C1 FROM T2)
	プラグイン提供関数の条件が「IS FALSE」又は「IS UNKNOWN」であり，この関数の第一引数の列に指定したプラグインインデクス	within(C1,'…') IS FALSE

（凡例）－：使用しないインデクスを示します。

注

1. 関数呼出し「contains」は，HiRDB Text Search Plug-in が提供するプラグインの関数です。
2. 関数呼出し「within」は，HiRDB Spatial Search Plug-in が提供するプラグインの関数です。
3. 外への参照がある副問合せを含む制限条件の対象になる列にあるインデクスは使用できません。
4. OR 演算子で両側の条件式にインデクスを利用できる場合は，優先度がその条件式中の述語によって変わります。
5. 制限条件とは，結合条件以外の探索条件のことです。
6. 定義したインデクスが有効に利用できないと HiRDB が判断した場合は，該当するインデクスが使用されないことがあります。

注※1

このインデクスを定義していないと，実行できないでエラーとなります。必ず定義してください。

注※2

次の列に対して，ナル値を除外キーとするインデクスは使用しません。

- 「IS NULL」の制限条件を指定した列
- 制限条件中の「VALUE」及び「CASE 式」に指定した列
- 制限条件中の「BIT_AND_TEST」に対して「IS UNKNOWN」，「IS NOT TRUE」，又は「IS NOT FALSE」を指定し，この「BIT_AND_TEST」に指定した列

ただし，上記以外の制限条件を指定しているインデクスは使用します。その場合のナル値を除外キーとするインデクスの使用有無を表「ナル値を除外キーとするインデクスの使用有無」に示します。

注※3

SQL 最適化オプションに「スカラ演算を含むキー条件の適用」を指定した場合だけ，インデクスを優先して使用します。SQL 最適化オプションの詳細は，マニュアル「HiRDB UAP 開発ガイド」を参照してください。また，述語の種類によっては優先順位が下がることもあります。否定が含まれない場合の優先順位は 13～15 の範囲になります。否定が含まれる場合の優先順位は 13～－の範囲になります。

注※4

表の指定が一つで，GROUP BY を指定していない SQL 文の場合は，集合関数（MIN 又は MAX）だけを指定して，次に示すどれかの条件を満たすときに引数に指定した列のインデクスを使用します。

- 集合関数の引数に指定した列が，単一列インデクスの構成列の場合
- 集合関数の引数に指定した列が，除外キーを持たない複数列インデクスの第 n 構成列の場合で，第 1 構成列から第（n－1）構成列までに「=」又は「IS NULL」を指定したとき
- 集合関数の引数に指定した列が，除外キーを持つ複数列インデクスの第 n 構成列の場合で，第 1 構成列から第（n－1）構成列までに「=」を指定したとき

表 12-4　ナル値を除外キーとするインデクスの使用有無

構成列に指定する制限条件		使用有無
IS NULL，VALUE，CASE 式，BIT_AND_TEST	IS NULL，VALUE，CASE 式，BIT_AND_TEST 以外※1	
指定あり	指定あり	使用します
指定あり	指定なし	使用しません
指定なし	指定あり	使用します※2
指定なし	指定なし	使用しません※3

注※1

「HiRDB が使用するインデクスの優先順位」に示す優先順位の 4～15 の制限条件の場合です。

注※2

インデクスが有効に利用できない場合などは，HiRDB が判断してインデクスを使用しないことがあります。

注※3

次に示すすべての条件を満たす検索では使用します。

- 選択式が，インデクス構成列を引数とする集合関数だけ

- FROM 句には，一つの表だけ指定している
- WHERE 句の指定がない

13 RD エリアの設計

この章では，RD エリアを構成するセグメント及びページを設計する上で検討する項目について説明します。

13.1　RD エリアを設計するときの検討項目

RD エリアを構成するセグメントやページの大きさによって，ディスク所要量が異なります。この点を考慮して RD エリアを設計する必要があります。RD エリアを設計するときの検討項目と RD エリアに関する最大値・最小値を次の表に示します。

表 13–1　RD エリアを設計するときの検討項目

設計作業ごとの検討項目		長　所	短　所	記載箇所
セグメントサイズ	大きくした場合	行長が変わるような更新をする場合や，クラスタキーを指定した表に行を追加する場合，行を格納した特定のページに近接する未使用のページを確保できるため，データの入出力時間を削減できます。	セグメント数が少なくなるため，一つのユーザ用 RD エリアに格納できる表とインデクスの数が少なくなります。	「セグメントサイズの決定」を参照してください。
	小さくした場合	一つのユーザ用 RD エリアにデータ量の少ない表を多く格納する場合は，余分な未使用ページを削減できます。	• ユーザ用 RD エリアに大量のデータを追加すると，セグメントの割り当て回数が増加するため，オーバヘッドが大きくなります。 • セグメントの数が多くなるため，表を削除したり，表の全行削除をしたりする場合，排他制御の資源が多くなります。	
セグメント内の空きページ比率	設定した場合	表にデータを追加する場合，クラスタキーを指定している表に対しては，クラスタキーの値に近い場所のページにデータを格納できるため，データの入出力回数を削減できます。	設定値を大きくする分，ディスク所要量が増加します。	「セグメント内の空きページ比率の設定」を参照してください。
	0 にした場合	ディスク所要量を削減できます。	クラスタキーを指定している表の場合，データの追加時にクラスタキーの値に近い場所にデータを格納できなくなるため，格納状態が乱れ，データの入出力回数の削減効果がなくなります。	
ページ長	ページ内の未使用領域の比率を設定した場合	• UPDATE 文で行長が元の行よりも長くなる更新をしても，連続した空き領域が更新後の行長よりも大きい場合は，該当する行がそのページに収まるようになります。 • INSERT 文で行を繰り返し追加するときに，クラスタキーの値に近い場所のページに，1 ページ内が一杯になるまで行を追加できるようになります。	FIX 属性の表の場合は格納効率が悪くなります。	「ページ内の未使用領域の比率の設定」を参照してください。
	ページ内の未使用領域の比率を 0 にした場合	FIX 属性の表の場合，データが昇順になるようなときは，格納効率が良くなります。	行長が元の行よりも長くなる更新をすると，行が複数ページにわたるた	

設計作業ごとの検討項目		長　所	短　所	記載箇所
			め，行にアクセスするときにオーバヘッドが掛かります。	
空き領域の再利用	使用した場合	• 使用中セグメント内の空き領域を有効活用できます。 • RD エリア満杯後の空き領域サーチの性能が向上します。	再利用するのに十分空き領域がない場合，空き領域サーチのオーバヘッドが増えます。	「空き領域の再利用機能」を参照してください。
	使用しない場合	空き領域が十分ある状態であれば，挿入処理が高速にできます。	RD エリアの格納効率が低下します。また，RD エリア満杯後の空き領域サーチの性能が低下します。	
共用 RD エリア	使用した場合	アクセスが多いが，分割が難しい表を共用 RD エリアに格納すると，全バックエンドサーバから参照できるため，並列処理の効率が上がります。	共用表を更新する場合，更新する共用表がある共用 RD エリアに排他を掛けるので，共用 RD エリアのほかの表にアクセスする業務があると，デッドロックが発生することがあります。	「共用 RD エリア (HiRDB/パラレルサーバ限定)」を参照してください。
	使用しない場合	共用 RD エリアが原因となるデッドロックや，サーバ間のグローバルデッドロックが発生しません。	結合処理などの複雑な検索処理の場合，複数のバックエンドサーバ接続とデータ転送のオーバヘッドが増えます。	
一時表用 RD エリア	使用した場合	一時表を使用して，複雑なデータ処理をしたり，ほかのユーザの影響を受けないでトランザクションや SQL セッションを実行できます。また，一時表の場合後処理が不要です。	HiRDB 開始時，又は一時表に最初に INSERT 文が実行された時に，一時表用 RD エリアを初期化するオーバヘッドが発生します。	「一時表用 RD エリア」を参照してください。
	使用しない場合	HiRDB 開始時に，一時表用 RD エリアを初期化するオーバヘッドが発生しません。	複雑な処理で中間処理結果を表に格納する場合，処理完了後にデータを削除するなどの後処理が必要になります。	

表 13–2　RD エリアに関する最大値・最小値

項目	最大値・最小値
総 RD エリア数	3～8,388,592
マスタディレクトリ用 RD エリア数	1
データディレクトリ用 RD エリア数	1
データディクショナリ用 RD エリア数	1～41
ユーザ用 RD エリア数	1～8,388,589
データディクショナリ LOB 用 RD エリア数	1～2
ユーザ LOB 用 RD エリア数	0～8,388,325
レジストリ用 RD エリア数	0～1

項目	最大値・最小値
レジストリ LOB 用 RD エリア数	0～1
リスト用 RD エリア数	0～8,388,588
1RD エリア中の HiRDB ファイル数	1～16
1RD エリア中の実表数	0～500
1RD エリア中のインデクス数	0～500
1RD エリア中のリスト数	0～50,000
総 HiRDB ファイル数	1～134,217,728

●インデクス格納 RD エリアの容量見積もりについて

インデクス格納 RD エリアの容量見積もりについては，「ユーザ用 RD エリアの容量の見積もり」を参照してください。ここでは，容量見積もり時の注意事項を説明します。

1. データベース作成ユティリティ又はデータベース再編成ユティリティでインデクスを一括作成した直後は，データはきれいに格納されています。その後のデータ挿入時にすべてのキーを昇順に挿入しないかぎり，インデクスページスプリットが発生するため，インデクスの一括作成時よりインデクスの容量が大きくなります。
2. インデクスページは基本的に使用中空きページを再使用しません。したがって，キー値を変更するような更新及び削除を行った場合，変更又は削除前のキーが格納されていたページを再使用できません。このように再使用されないむだな使用中空きページができてしまいます。ただし，使用中空きページを再利用する運用もできます。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。
3. キー値の重複がある場合とない場合では基本的にインデクス構造が異なります。このため，正しく重複数を求めないとインデクスの容量が大きく異なってしまいます。インデクスレコード件数が少ない場合は，インデクスの容量に比べてこの誤差の占める割合が大きくなります。

13.2　セグメント

セグメントには次の表に示す状態があります。

表 13–3　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。

13.2.1　セグメントサイズの決定

通常，RD エリアのセグメントサイズは，RD エリア格納ページ数の 1/10 程度にすることをお勧めします。ただし，セグメントサイズの最大は 16,000 ページであるため，RD エリアの容量が大きい場合は，1/10 以下になります。

セグメントサイズの大小による効果と注意を次に示します。

(1)　セグメントサイズを大きくした場合

性能の向上

- 行長が変わるような更新をする場合や，クラスタキーを指定した表に行を追加する場合，行を格納した特定のページに近接する未使用のページを確保できるため，データの入出力時間を削減できます。
- 同じ表のデータが連続したページに格納されるため，**プリフェッチ機能**による一括入力の効果が得られます。プリフェッチ機能を使用する場合，セグメントサイズは，システム共通定義の pdbuffer オペランドの-p オプションで指定する一括入力最大ページ数に合わせることをお勧めします。

注意事項

- セグメント数が少なくなるため，一つのユーザ用 RD エリアに格納できる表とインデクスの数が少なくなります。

(2) セグメントサイズを小さくした場合

ディスク所要量の削減

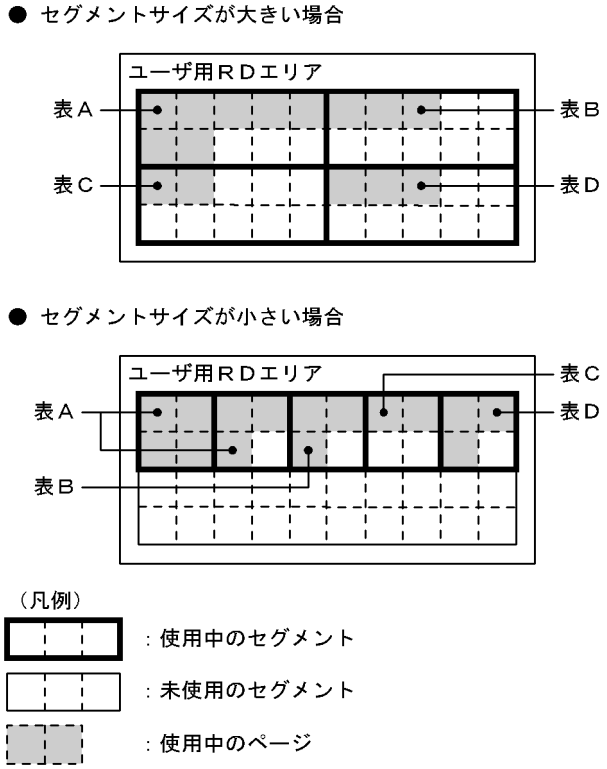
- 一つのユーザ用 RD エリアにデータ量の少ない表を多く格納する場合は，余分な未使用ページを削減できます。

注意事項

- セグメントサイズを小さくしたユーザ用 RD エリアに大量のデータを追加すると，セグメントの割り当て回数が増加するため，オーバヘッドが大きくなります。
- セグメントの数が多くなるため，表を削除したり，表の全行削除をしたりする場合，排他制御の資源が多くなります。

セグメントサイズの大小によるユーザ用 RD エリアの概要を次の図に示します。

図 13–1　セグメントサイズの大小によるユーザ用 RD エリアの概要



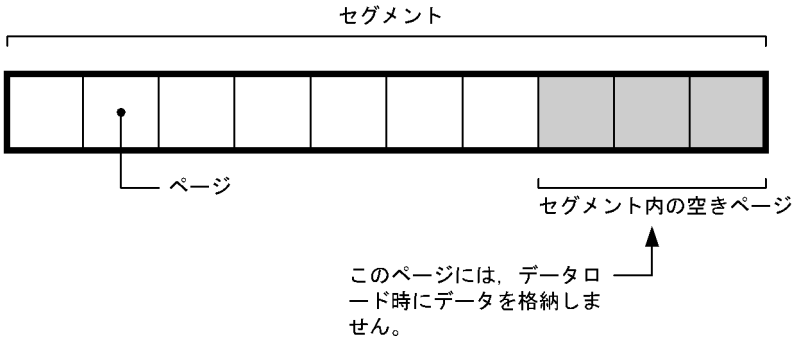
(3) 設定方法

セグメントサイズは，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）の create rdarea 文で設定します。

13.2.2　セグメント内の空きページ比率の設定

表を定義するときにセグメント内に割り当てる空きページの割合を**セグメント内の空きページ比率**といいます。ここでいう空きページとは，未使用ページのことです。セグメント内の空きページの概要を次の図に示します。

図 13–2　セグメント内の空きページの概要



(1)　設定の効果

セグメント内の空きページ比率の設定の効果を次に示します。

性能の向上

- 表にデータを追加する場合，クラスタキーを指定している表に対しては，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ます。このため，データの入出力回数を削減できます。

(2)　適用基準

- クラスタキーを指定した表に，データベース作成ユティリティ（pdload）などでデータを格納した後，データの追加が大量に発生するような場合は，セグメント内の空きページ比率を設定するようにします。
- データの追加又は更新がほとんど発生しない表の場合は，セグメント内の空きページ比率を 0 にします。

(3)　設定方法

セグメント内の空きページ比率は，定義系 SQL の CREATE TABLE の PCTFREE オプションで指定します。

(4)　注意

セグメント内の空きページ比率に 0 を指定した場合，クラスタキーを指定している表にデータを追加するときは，クラスタキーの値に近い場所にデータを格納できなくなります。このため，データの格納状態が乱れ，データの入出力回数の削減効果がなくなります。

13.2.3　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場合はセグメントを解放します。

- 表の定義を削除した場合
- 表の全行削除をした場合
- 表の所有者（スキーマ）を削除した場合
- インデクスの定義を削除した場合
- 表の主キーを削除した場合
- RD エリアの再初期化をした場合
- データロードを作成モード（-d オプション指定）で実行した場合
- 表の再編成をした結果，空きセグメントができた場合

また，次に示す場合では，そのトランザクションで確保したセグメントを解放します。

- データベース作成ユティリティのデータロード処理又はインデクス一括作成処理でロールバックが発生した場合
- データベース再編成ユティリティのリロード処理中にロールバックが発生した場合

13.3　ページ

ページには次の表に示す状態があります。

表 13–4　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。

13.3.1　ページ長の決定

(1)　ページ長を決定するときの考慮点

ページ長を決定するときの考慮点を次に示します。

1. 全件検索及び更新，大量検索及び更新をする業務で，次に示す条件を満たす表，インデクスを格納するページ長は大きくします。
   - インデクスを付けない表を格納する RD エリア
   - クラスタキーを指定した表及びそのインデクスを格納する RD エリア
   - 大量検索，大量更新の範囲条件となるインデクスを格納する RD エリア
2. ページ長は，RD エリアに格納する表の行長を基に，できるだけ無効領域が作られない長さにします。

   無効領域＝ MAX（mod（(ページ長－48)，(行長＋ 2)），ページ長－48－（行長＋ 2）×255)
3. ページ内の未使用領域の比率を指定する場合は，次に示す計算式を目安にします。

   (ページ長×ページ内の未使用領域の比率）÷100－行長×ページ内の未使用領域に格納できる行数

   この場合，ページ内の未使用領域に 1 行も格納できなくなるような無意味なページ内の未使用領域の比率は指定しないようにします。

4. インデクスを格納するページの場合は，一般的に 4096～8192 バイト程度が，入出力効率の面から適しています。
5. 列のデータ型が VARCHAR，NVARCHAR 及び MVARCHAR で，定義長が 256 バイト以上の場合は，該当する列のデータが別のページに分岐されます。また，256 バイト以上の可変長文字列データがある場合，その平均長以上で最も小さいページ長にします。
6. 列のデータ型が VARCHAR，NVARCHAR 及び MVARCHAR の場合，INSERT 文で行をナル値として挿入すると，UPDATE 文で該当する列のデータを実データに更新するときに，更新したデータの長さによっては，該当する列のデータが別のページに分岐されることがあります。可変長文字列データの初期値をナル値として，後から実データに更新することが多い場合は，更新後の行長を見込んでページ長を決定します。
7. HiRDB では，ページ又は行単位に排他制御ができます。行レベルの排他制御をする場合には，1 ページに格納できる行数ができるだけ多くなるように，行長に応じてページ長を設定します。このとき，次に示す点を考慮します。
   - ページ内の未使用領域の比率を最低限にします。
   - ページの入出力要求に対するグローバルバッファの排他待ちが少なくなるようにします。更新が多い表の場合，排他待ちの回数が増える可能性があるため，ページ長を小さくします。
   - ページの入出力要求に対するページの入出力待ちが少なくなるようにします。ランダムアクセスが中心の業務で，ページ長が大き過ぎると，アクセス単位である行長に対する実際の入出力単位が大きくなって不要なデータ転送が発生するため，ページ長を小さくします。

     ただし，UPDATE 文で列のデータ型が VARCHAR，NVARCHAR 及び MVARCHAR のデータに対して行長が変更されるような更新を頻繁にする場合は，表を定義するときにページ内の未使用領域の比率を高めに設定します。ページ内の未使用領域の比率の設定については，「ページ内の未使用領域の比率の設定」を参照してください。

(2) 設定方法

ページ長は，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）の create rdarea 文で設定します。

(3) ページ長を決定するときの注意

表に行を追加する場合，実際の行の長さ（データ型が VARCHAR，NVARCHAR 及び MVARCHAR の列を除いた長さ）がページ長を超えると，エラーになります。実際の行の長さは，「RD エリアの容量の見積もり」に記載されているディスク所要量の見積もり式を参照して求めます。求めた行の長さが，使用するユーザ用 RD エリアのページ長より大きい場合は，ユーザ用 RD エリアを再初期化して，ページ長を再設定する必要があります。RD エリアの再初期化は，データベース構成変更ユティリティ（pdmod）で実行します。RD エリアの再初期化については，マニュアル「HiRDB システム運用ガイド」を参照してください。

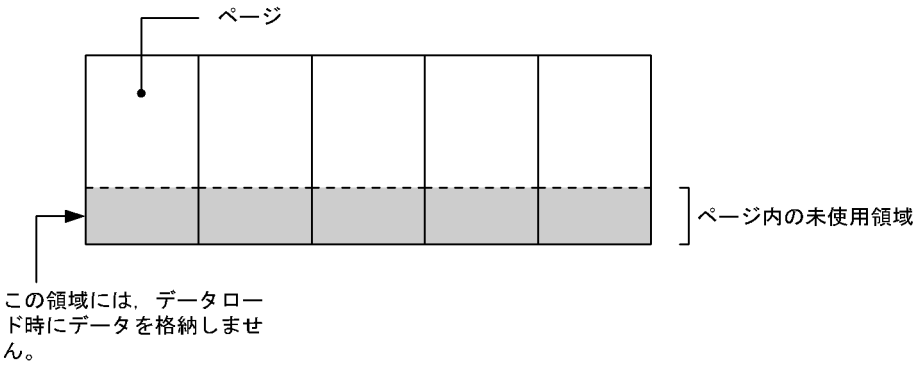
13.3.2　ページ内の未使用領域の比率の設定

表とインデクスを定義するときに，ページ内に割り当てる未使用領域の割合を**ページ内の未使用領域の比率**といいます。未使用領域を設定しておくと，データベース作成ユティリティ（pdload）又はデータベース再編成ユティリティ（pdrorg）でデータを格納するとき，未使用領域にはデータは格納されません。

ただし，データベース作成ユティリティ（pdload）で-y オプション又は option 文を指定して実行し，新しいページが確保できなかった場合は，未使用領域にもデータが格納されます。

ページ内の未使用領域の概要を次の図に示します。

図 13–3　ページ内の未使用領域の概要



(1)　設定の効果

- UPDATE 文で行長が元の行より長くなるような更新をしても，連続した空き領域が更新後の行長よりも大きい場合は，該当する行がそのページに収まるようになります。
- INSERT 文で行を繰り返し追加するときに，クラスタキーの値に近い場所のページに，1 ページ内が一杯になるまで行を追加できるようになります。

(2)　適用基準

1. クラスタキーを指定した表に行を追加する場合に設定します。
2. FIX 属性の表の場合，データが昇順になるようなときは，ページ内の未使用領域の比率を 0 にした方が，格納効率が良くなります。
3. 行長が元の行よりも長くなるような更新をする場合に設定します。
4. 行長が元の行長より長くなるのは，次に示す更新をしたときです。
   - ナル値から実データに更新したとき
   - データ型が VARCHAR，NVARCHAR，MVARCHAR，及び BINARY の列を長い値になるように更新したとき

(3) 設定方法

ページ内の未使用領域の比率は，定義系 SQL の CREATE TABLE，CREATE INDEX，又は ALTER TABLE の PCTFREE オプションで指定します。

(4) 注意

設定した未使用領域に余裕がない場合は，行長を元の行長より長くなるような更新をしたときに，1 行が複数ページにわたるため，入出力回数が増えます。

(5) ページ内の未使用領域の比率の求め方

- 最初に格納した行の行長が L1 で，最終的に L2 になる場合は，次に示す計算式で求めた値を未使用領域の比率とするのが一般的です。

  ページ内の未使用領域の比率=（(L2-L1）÷L2）×100（%）

- 表にクラスタキーを指定している場合は，次のように求めます。

  1. データベース作成ユティリティ（pdload）でこの表にデータを格納する件数をページ当たりで求めます。これを m 件とします。
  2. その後，更に何件のデータを格納するかを求めます。これを n 件とします。
  3. 1.と 2.で求めた m と n から，次に示す計算式でページ内の未使用領域の比率を求めます。

     ページ内の未使用領域の比率=（n÷（m + n））×100（%）

13.3.3 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。

注意事項

1. 非 FIX 表で行長が変わるデータ更新を行った場合は，行長が減った分の領域を再使用できません。
2. 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。
3. 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すると，ページを解放します。インデクスのページは解放されません。
- PURGE TABLE 文を実行すると，表及びインデクスのページとセグメントが解放されます。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。

13.4　リスト用 RD エリアの設計

13.4.1　リスト用 RD エリアの必要数

1 リスト用 RD エリアに作成できるリストの最大数は，次に示すオペランドで指定します。

- データベース初期設定ユティリティ（pdinit）の create rdarea 文の max entries オペランド
- データベース構成変更ユティリティ（pdmod）の create rdarea 文の max entries オペランド
- データベース構成変更ユティリティ（pdmod）の initialize rdarea 文の max entries オペランド

指定できる最大数は 500～50000 です。

13.4.2　ページ長，セグメントサイズの求め方

リストにはリストの基表の行識別子が格納されます。表のように直接データを格納しないので，1 ページ内に比較的大量の行を格納できます。したがって，ページ長及びセグメントサイズをリストに格納する行数に比べて大きく設定した場合，RD エリア内に余分な空き領域が発生するので注意してください。

リスト用 RD エリアのページ長，セグメントサイズを決定する場合，あらかじめサーバ内に作成するリストの平均行数をおおよそ見積もり，次に示すどれかのケースに基づいてページ長及びセグメントサイズを設定してください。

条件	ページ長	セグメントサイズ
サーバ内に作成するリストの平均行数が 3000 行未満の場合	4096	1
サーバ内に作成するリストの平均行数が 3000 行～6000 行の場合	4096	2
サーバ内に作成するリストの平均行数が 6000 行を超える場合	「リストの平均行数が 6000 行を超える場合のページ長の求め方」参照	「リストの平均行数が 6000 行を超える場合のセグメントサイズの求め方」参照

(1)　リストの平均行数が 6000 行を超える場合のページ長の求め方

ページ長は，通常 4096～8192 バイトの範囲にしてください。ただし，リストの入出力回数を減らしてリストの入出力時間を短縮したい場合は，ページ長を大きくしてもかまいません。ただし，ページ長を大きくすると，グローバルバッファの必要量も大きくなるため，共用メモリが大量に必要となるので注意してください。

なお，ページ長は次に示す計算式を満たす値にしてください。

計算式

```
リストの1ページに格納できる行数≦ サーバ内に作成するリストの平均行数÷2
```

リストの 1 ページに格納できる行数は，次に示す計算式から求めます。

リストの 1 ページに格納できる行数＝↓ {ページ長−70−（a×8)} ÷4↓

a：サーバ内のリストの基表を格納している RD エリアの HiRDB ファイルの総数

(2) リストの平均行数が 6000 行を超える場合のセグメントサイズの求め方

セグメントサイズは 1 リストへの RD エリア内領域の割り当て単位です。したがって，1 セグメントが 1 リストへの最低割り当てサイズとなります。セグメントサイズの目安を次に示します。

- セグメント割り当てのオーバヘッドを削減したい場合，セグメントサイズを大きくしてください。
- リスト用 RD エリアのグローバルバッファにプリフェッチ機能を指定する場合は，セグメントサイズに 2 以上の値を指定してください。指定しないと，プリフェッチ機能が動作しないので注意してください。
- セグメントサイズを大きくすると，セグメント内に余分な未使用ページが発生する可能性が高くなります。余分な未使用ページを減らしたい場合は，セグメントサイズを小さくしてください。
- セグメントサイズは次に示す計算式を満たす値にしてください。

  リストの 1 セグメントに格納できる行数
  ≦サーバ内に作成するリストの平均行数÷2

  リストの 1 セグメントに格納できる行数は，次に示す計算式から求めます。

  リストの 1 セグメントに格納できる行数
  ＝リストの 1 ページに格納できる行数×セグメントサイズ

13.4.3 セグメント数の求め方

リスト用 RD エリアに必要なセグメント数は，次に示す計算式から求めます。

計算式

```
リスト用RDエリアに必要なセグメント数＝↑ {↑a÷b↑×（c＋0.5）} ↑
```

a：サーバ内のリストの数

b：サーバ内のリスト用 RD エリアの数

c：1 リストが使用するセグメント数の平均値

次に示す計算式から求めます。

↑サーバ内のリストの平均行数÷リストの 1 セグメントに格納できる行数↑

セグメント数が不足すると，リストが作成できなくなるため，実際には上記の計算式で見積もったセグメント数に余裕を持たせてください。

13.5　空き領域の再利用機能

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。ここでは，次の項目について説明します。

- データ格納時のサーチ方式
- 空き領域の再利用機能とは
- 効果と適用基準
- 使用前の考慮点
- 環境設定
- 実行状態の確認
- 注意事項

13.5.1　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。

  未使用セグメントがある状態のときは，格納効率は良くはなりませんが，高速に処理が行えます。ただし，未使用セグメントがなくなると性能が大きく低下します。

- **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。

  未使用セグメントがある場合でも，空き領域をサーチしてデータを格納するため，格納効率は良くなりますが，その分オーバヘッドが掛かります。

13.5.2　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。

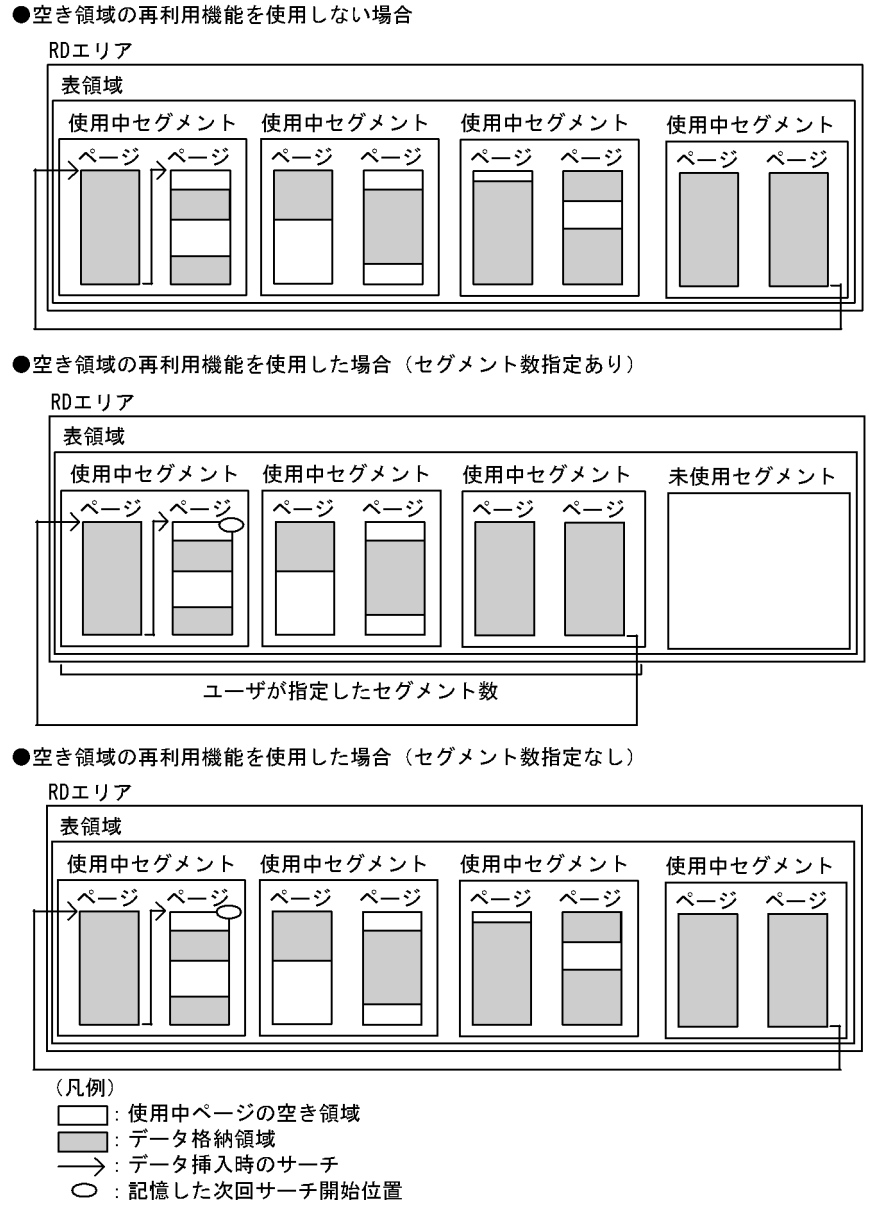
セグメント数を指定しない場合は，RD エリア中に未使用ページがなくなったときに空きページ再利用モードに切り替わります。

RD エリア中に未使用ページがなくなった場合，空き領域の再利用機能による空きページ再利用モードの方が，この機能を使用しないときに比べて，サーチ効率が良くなります。空きページ再利用モードの場合は，次回サーチ位置を記憶していてそれ以降をサーチしますが，この機能を使用しない場合は，常に先頭からサーチします。

なお，空き領域の再利用機能を使用しない場合，常に新規ページ追加モードで動作します。この場合，未使用セグメントがなくなると，性能が大きく低下してしまうため，その前に表の再編成や，使用中空きページ及び使用中空きセグメントの解放などを行って，未使用セグメントを増やしてください。

空き領域の再利用機能の概要を次の図に示します。

図 13–4　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。

13.5.3　効果と適用基準

(1)　効果

この機能を使用すると，次の効果が期待できます。

- 空き領域の有効活用

  使用中ページの空き領域を再利用するため，最小限の RD エリア容量で運用でき，データベースの再編成回数を減らせます。また，1RD エリアに複数の表及びインデクスを格納する場合に，ある表に対して集中して挿入及び削除処理が実行された場合の領域占有を回避できます。

- 可変長列及び BINARY 型の列のページ不足エラーの回避

  通常，ノースプリットオプションを指定しないで 256 バイト以上の可変長文字列を挿入したり，1 ページに入らない BINARY 型の列を挿入したりすると，未使用ページが確保されます。使用中空きページがあっても，未使用ページが確保できなければエラーになりますが，空き領域の再利用機能を使用すると使用中空きページを未使用ページの代わりに確保するのでページ不足エラーを回避できます。

(2)　適用基準

- 空き領域を再利用する処理はオーバヘッドが掛かるため，性能よりも格納効率を優先する場合に，空き領域の再利用機能を使用してください。
- 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。この機能の使用をお勧めする場合の業務特性と効果を次に示します。
  - 削除（更新），挿入を含み，データ量の増加がない場合

空き領域の再利用機能で格納データの最大サイズを指定しておけば，その後は削除されるデータの領域を優先して再利用するので，新規領域を追加しないで業務を継続でき，再編成が不要になります。

（例）行政電子窓口

電子窓口で受け付け業務を 24 時間するシステムで，申請受け付け時に申請データを挿入し，保管期限経過後に削除します。最大保管期限内に受け付ける申請データ数のセグメントサイズを指定すれば，削除されるデータの領域を再利用するので，新規領域を追加しないで業務を継続できます。そのため，再編成が不要となり，業務を停止することなく 24 時間サービスができます。

- 削除（更新），挿入を含み，データ量が徐々に増加する場合

  徐々に増加するデータを新規領域だけでなく，削除した領域にも格納するため，格納効率が向上します。

  （例）顧客管理

  新規顧客のデータを挿入し，不要になった顧客登録データを削除します。初期顧客データ登録終了後，追加，削除業務開始前のセグメントサイズを指定すれば，その後は削除された顧客データの領域を再利用しながら追加されます。

- 挿入処理は新規に未使用ページや未使用セグメントにデータを格納する方が性能が良くなります。そのため，短い周期でデータベース再編成ユティリティ（pdrorg）を実行できる場合は，空き領域の再利用機能を適用しないでデータベースを再編成する方が性能が良くなります。

13.5.4 使用前の考慮点

1. 空き領域の再利用機能が有効になるのは，削除処理によって空き領域が十分にできる場合です。空き領域が十分にないときや全くないときに空き領域のサーチをするなど，むだに空き領域サーチをする場合，セグメント内のページ数の指定を大きくしたり，この機能を中止する必要があります。セグメント内のページ数指定を変更するためには，RD エリアを再作成（削除又は追加）する必要があるため，最初の設計時に指定するセグメント数，セグメントサイズは十分に考慮してください。

   - 次の場合，SEGMENT REUSE オプションでセグメント数は省略してかまいません。

     RD エリア内に表が一つで，インデクス混在なし，自動増分指定なし

   - 次の場合，SEGMENT REUSE オプションでセグメント数を指定する必要があります。

     ・RD エリア内に表が一つで，インデクスは混在なし，自動増分指定あり

     ・RD エリア内に表が一つで，インデクスは混在あり

     ・RD エリア内に表が複数あり

     データ件数が増加する場合，セグメント数を指定し，1 セグメントが満杯になるまでに十分削除がされるようなセグメントサイズにしてください。データ件数が増加しない場合，表が使用する総セグメント数を見積もって，セグメント数を指定すればセグメントサイズは考慮不要です。ただし，同一 RD エリア内の表で，再利用するセグメント数の合計（インデクスが混在している場合，同一 RD エリア内の表で再利用するセグメント数とインデクスの見積もりセグメント数の合計）は RD エリアの総セグメント数以下にしてください。

なお，自動増分指定された表に空き領域の再利用機能が使用された場合，領域の増分を優先し，増分した領域が指定されたセグメント数になると，空き領域の再利用を実行します。

2. 空き領域の再利用機能が有効になると，使用中セグメント内の空き領域をサーチするため，次の機能の効果が低下します。
   - クラスタ表でクラスタキー順にデータを格納する
   - UPDATE 文の実行で行長が長くなる場合に，元のデータの近くにデータを格納する（CREATE TABLE 文の PCTFREE 指定，データロード時又は表の再編成時の tblfree 指定）

   そのため，空き領域の再利用機能を適用する場合は，これらの機能の使用を推奨しません。

13.5.5 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。

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ユーザ表の表数を指定します。

   分割表の場合は 1 分割を 1 表として計算します。インナレプリカ機能を使用している場合，レプリカ RD エリアに格納する表も 1 表として計算します。HiRDB/パラレルサーバの場合，バックエンドサーバごとに計算し，その最大数をこのオペランドに登録します。

   なお，CREATE TABLE で定義又は ALTER TABLE で定義変更した表は，pd_assurance_table_no オペランドの指定数（予約数）まで空き領域の再利用機能を使用できます。予約数を超えた表に挿入が実行された場合，KFPH22030-W メッセージが出力され，空き領域の再利用機能は適用されません。この場合，pd_assurance_table_no オペランドの指定値を増やすと定義したすべての表に空き領域の再利用機能が適用されます。ALTER TABLE の ADD RDAREA 指定で表格納 RD エリアを追加して定義数が予約数を超えた場合や，HiRDB/パラレルサーバで定義数が予約数を超えた場合は，空き領域の再利用を定義した分割表で RD エリアごとに空き領域の再利用が適用されたり，されなかったりする場合があります。

2. 空き領域を再利用するセグメント数を見積もり（表の総データ数から総セグメント数を見積もります。「ユーザ用 RD エリアの容量の見積もり」を参照してください），見積もったセグメント数を定義系 SQL の CREATE TABLE の SEGMENT REUSE オプションで指定します。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。ここで指定したセグメント数はすべての RD エリアに適用されます。

   また，さらに格納効率を向上させたい場合は，SEGMENT REUSE の OPTION で再利用オプション値を指定します。再利用オプション値を指定すると，次の機能が使えるようになります。

各機能の再利用オプション値	機能	格納効率の向上が期待できるケース
1	UPDATE 対応	• 固定長データ，ADT 列の NULL 値からの UPDATE がある • BINARY 列の UPDATE がある • VARCHAR 列，NVARCHAR 列，MVARCHAR 列（NO SPLIT 指定）の UPDATE がある

各機能の再利用オプション値	機能	格納効率の向上が期待できるケース
		• 確保済みセグメント数に比べてサーチモードの切り替え回数が少ない（pddbstコマンドで確認できます）
2	分岐行が多発する表の格納効率向上	分岐行を作成する表を運用するケース
3	UPDATE 対応及び分岐行が多発する表の格納効率向上	• 固定長データ，ADT 列の NULL 値からの UPDATE がある • BINARY 列の UPDATE がある • VARCHAR 列，NVARCHAR 列，MVARCHAR 列（NO SPLIT 指定）の UPDATE がある • 確保済みセグメント数に比べてサーチモードの切り替え回数が少ない（pddbstコマンドで確認できます） • 分岐行を作成する表を運用するケース

なお，OPTION 1（UPDATE 対応）を適用すると，UPDATE 時のページ確保を，新規ページ追加モード時は最終セグメントから，空きページ再利用モード時は前回記憶したサーチ開始位置のあるセグメントから行います。このとき，データロードや表の再編成で作成する空き領域を優先的に利用することがなくなります。そのため，OPTION 1 を指定する表は CREATE TABLE 文の PCTFREE に（0, 0）を指定することを推奨します。

3. 一度定義したセグメント数を変更する場合，ALTER TABLE の SEGMENT REUSE オプションで再度セグメント数を指定します。ページサーチモードとセグメント数の指定値によって，HiRDB は次のように処理します。

   • 新規ページ追加モード時

   指定されたセグメント数が使用中セグメント数より少ない場合，最後に確保したセグメント内に空き領域がなくなった後で空き領域の再利用を実行します。

   • 空きページ再利用モード時

   使用中セグメント数以下のセグメント数が指定された場合，そのまま続行します。使用中セグメント数より多いセグメント数が指定された場合，空き領域をすべて使用した後で空き領域の再利用をいったん中止し，新規に未使用セグメントを確保します。

4. バッチ処理などで一時的に大量追加をする場合など，空き領域の再利用機能を一時中止したい場合，ALTER TABLE で SEGMENT REUSE NO を指定します。実行するとすぐに空き領域の再利用機能は中止され，新規に未使用セグメントが確保されます。

5. 空き領域の再利用機能を使用している表がセグメント確保時に出力する，RD エリアのセグメント使用率通知メッセージ（KFPH00211-I，又は KFPA12300-I）を抑止したい場合，pd_rdarea_warning_point_msgout オペランドに N を指定します。

   削除（更新），挿入を含み，かつデータ量の増加がない場合，空き領域の再利用機能を使用すると，表の再編成や RD エリアの拡張をする必要がなくなります。そのため，ユーザは RD エリアのセグメント使用率通知メッセージの出力を監視する必要もなくなります。削除（更新），挿入を含み，データ量の

増加がなく，かつ次のすべての条件に該当するときは，RD エリアのセグメント使用率通知メッセージの出力を抑止できます。

- 格納 RD エリアに，空き領域の再利用機能を使用している表だけを定義している。
- FIX 属性の表である，又は可変長の列を含まない表である（データ長が長くなるような更新をしない）。

ただし，次に示す場合は空き領域の再利用機能が動作しないおそれがあるため，RD エリアのセグメント使用率通知メッセージを出力し，監視を行ってください。RD エリアの使用状況に応じてユーザが対処する必要があります。

- 空き領域の再利用機能を定義している表数が，pd_assurance_table_no オペランドで指定した予約数より多い。
- 格納 RD エリアに対して空き領域の再利用機能を使用している表を複数定義していて，表定義の SEGMENT REUSE のセグメント数に，最大データ量サイズ以上を指定していない。

13.5.6 実行状態の確認

空き領域の再利用機能が有効かどうかを，データベース状態解析ユティリティ，統計解析ユティリティ，及び UAP 統計レポート機能で表示される項目から確認できます。また，空回りした場合，表（分割表の場合は分割 RD エリア）ごとの一回目にメッセージログに KFPH22031-W が出力されます。表示される項目とその説明を次に示します。

項目	説明	対処
ページサーチモードの切り替え回数	新規ページ追加モードから空きページ再利用モードへ，又は空きページ再利用モードから新規ページ追加モードへのサーチモード切り替え回数です。再利用と未使用セグメントの確保が頻繁に切り替わっているということは，削除でできる空き領域より追加する領域が大きく，セグメントサイズ（ページ数）が小さいということになります。	セグメントサイズ変更や削除実行のタイミングを見直してください。
空き領域の再利用機能のページサーチ空回り回数	使用中セグメント数が指定されたセグメント数になり，空きページ再利用モードに切り替わってサーチしても，空き領域がない場合です。このようなとき，空回り回数がカウントアップされます。空き領域がないのにサーチしているため，むだにサーチ処理をしていることになります。 また，空き領域の再利用のページサーチ空回り回数とページサーチモードの切り替え回数が共に増加している場合は，全く空き領域がない状態で空き領域の再利用が実行されていることになります。	指定するセグメント数やセグメントサイズの見直し，又は空き領域の再利用機能の使用中止を検討してください。
使用中セグメント数	表が使用しているセグメントに空き領域がなくなると未使用セグメントが確保されるため，表の使用中セグメント数が増加します。空き領域の再利用のページサーチ空回り回数と同じ数だけ増加している場合，	−

項目	説明	対処
	空き領域がないのに再利用しようとしてむだにサーチ処理をしていることになります。	

（凡例）－：該当しません。

13.5.7　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
- 空き領域の再利用機能使用時，削除による空き領域が連続しない場合，連続している場合と比較するとページのサーチ処理は遅くなります。この場合，空き領域の再利用機能使用の中止や，データベース再編成ユティリティ（pdrorg）によるデータの再編成を検討してください。
- 非 FIX 表の場合，空き領域の再利用機能を適用していても，セグメント数が増加することがあります。セグメント数の増加は，次のような更新を行うと，データの追加と削除が同量でも発生することがあります。

  ↓（ページ長－48）÷（行長＋ 2）↓件のデータを挿入し，行長が短くなるデータ（NULL 値含む）又は別のページに分岐するようなデータに更新した後，削除する操作を繰り返す場合

  この場合，空きページ解放ユティリティ（pdreclaim）で解放できる満杯ページが発生し，そのページにはデータが格納できなくなります。そのようなページが発生しているかどうかは，データベース状態解析ユティリティ（pddbst）の「Collect Prearranged Full Page」の値で確認できます。

  この満杯ページは，次のどちらかを実行して解放してください。
  - 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）
- 空きページ再利用モードでサーチ実行時でも，同一トランザクションで削除された領域は再利用されません。
- 空き領域の再利用機能を適用していても，非 NULL 値から NULL 値への UPDATE を繰り返すと，実際のデータの容量以上のセグメント数を確保する場合があります。この現象を回避するには，NULL 値への UPDATE ではなく DELETE を使用するような設計及び運用をしてください。

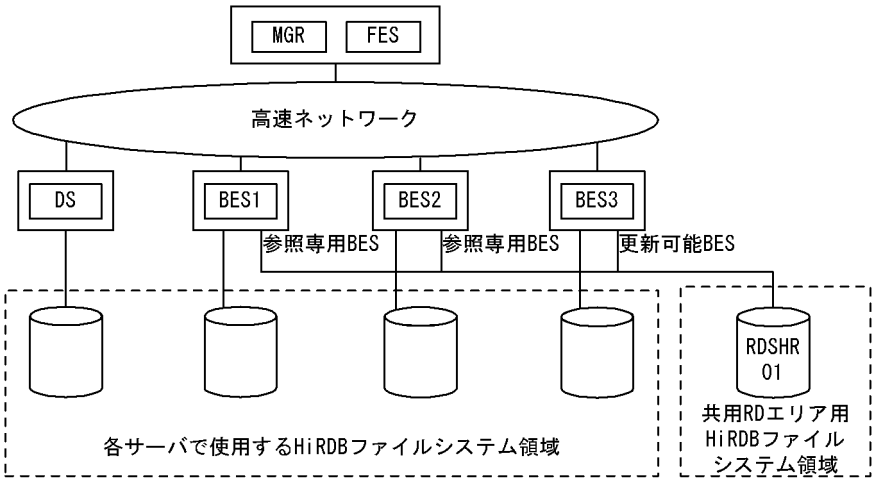
13.6　共用 RD エリア（HiRDB/パラレルサーバ限定）

13.6.1　共用 RD エリアの概要

通常，バックエンドサーバがアクセスできるのは，バックエンドサーバ下の RD エリアだけです。そのため，表の検索や更新時はできるだけ表を分割して格納した方が並列処理ができ，処理速度が向上します。しかし，複数のトランザクションからアクセスが集中し，かつ分割が難しい表などの場合，共用 RD エリアに格納することで並列処理の効率が上がります。**共用 RD エリア**とは，すべてのバックエンドサーバからアクセスできるユーザ用 RD エリアのことです。共用 RD エリアに格納する表を**共用表**，インデクスを**共用インデクス**といい，すべてのバックエンドサーバから参照できます。共用 RD エリアに格納できるのは，共用表及び共用インデクスだけです。共用 RD エリアの概要を次の図に示します。

なお，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。

図 13–5　共用 RD エリアの概要



［説明］

共用 RD エリア RDSHR01 は，BES1～3 すべてのバックエンドサーバから参照できます。ただし，共用表の更新ができるのは更新可能バックエンドサーバ（BES3）だけです。BES1 及び 2 は参照専用バックエンドサーバです。

(1)　効果

全バックエンドサーバが共用 RD エリアにアクセスできるため，並列処理の効率が上がります。

(2)　適用基準

次の場合に共用 RD エリアの使用をお勧めします。

- 複数のトランザクションからアクセスが集中するが，分割が難しい表の場合
- 結合処理のような複雑な検索処理をする場合

(3) 定義方法

共用 RD エリアを使用するには，次のように指定します。

- pd_shared_rdarea_use オペランドに Y を指定します。
- pdfmkfs コマンドの-k オプション（使用目的）に SDB を指定し，キャラクタ型スペシャルファイル名を指定します。また，全バックエンドサーバから同じパス名でアクセスできるようにアクセスパスを設定します。
- 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）の create rdarea 文に shared を指定し，ユーザ用 RD エリアの定義をします。また，更新可能バックエンドサーバを server name オペランドに指定します。server name オペランドに指定しなかったバックエンドサーバはすべて参照専用バックエンドサーバになります。

定義時の注意

- 共用 RD エリアは，pd_max_rdarea_no オペランドで指定した RD エリアの最大数まで定義できます。ただし，共用 RD エリアは全バックエンドサーバの RD エリアの数に加算されます。
- 異なるバックエンドサーバが更新可能バックエンドサーバである共用 RD エリアを，同一 HiRDB ファイルシステム領域に定義してはいけません。
- 共用 RD エリアは，共用 RD エリア用の HiRDB ファイルシステム領域に定義します。pdfmkfs -k コマンドに SDB を指定してください。なお，共用 RD エリア用の HiRDB ファイルシステム領域には共用 RD エリア以外は定義できません。

図「共用 RD エリアの概要」の場合の，データベース構成変更ユティリティ（pdmod）の制御文の例を次に示します。

```
create shared rdarea RDSHR01 globalbuffer buf01 for user used by PUBLIC
  server name BES3   …更新可能バックエンドサーバの指定
  open attribute INITIAL
  page 4096 characters
  storage control segment 20 pages
  file name ”/HiRDB/DATABASE/SHR1/rdshr01_f01”   …ファイル名
          initial 10000 segments ;
```

(4) 共用 RD エリアの更新

共用 RD エリアを更新する場合，LOCK 文で IN EXCLUSIVE MODE を指定し，全バックエンドサーバの共用 RD エリアに排他を掛けなければ実行できません。ただし，インデクスキー値を変更しない UPDATE 文は，LOCK 文を発行しないで実行できます。共用表の更新については，「共用表の操作」を参照してください。なお，共用表及び共用インデクスの更新は COMMIT 文発行時にディスクに書き込まれます。

(5) 共用 RD エリアの閉塞状態の管理

共用 RD エリアへのアクセスは，各バックエンドサーバで個別に管理されます。このため，障害発生時に，バックエンドサーバ間で閉塞状態が異なる場合があります。データベース構成変更ユティリティ（pdmod）

や，データベース回復ユティリティ（pdrstr）を実行する場合は，pdhold コマンドで全バックエンドサーバ下の共用 RD エリアの閉塞状態を一致させる必要があります。なお，pddbls -m コマンドで全バックエンドサーバの共用 RD エリアの状態を表示できます。

(6) 系切り替えの設定

共用 RD エリアを使用する場合には BES が存在するすべてのユニットから共用 RD エリアが格納してある共有ディスクをアクティブにしなければなりません。これによって，更新可能 BES と参照専用 BES が同一ホストに存在しているときに更新可能 BES が系切り替えを行うことで共有ディスクを切り替えると，参照専用 BES から共用 RD エリアを参照できなくなるという問題があります。そのため，BES の存在するユニットの系切り替えを行う場合，系切り替えのシステム構成によって設定方法が異なります。系切り替えのシステム構成については，マニュアル「HiRDB システム運用ガイド」を参照してください。

(a) スタンバイ型系切り替え構成の場合

- 1：1 系切り替え構成の場合

  1：1 系切り替え構成の場合は，更新可能 BES と参照専用 BES とが同一ホストに存在しないので，特別な設定はありません。系切り替えの設定は，マニュアル「HiRDB システム運用ガイド」の「クラスタソフトウェアによる共有ディスクのアクセス制御」を参照してください。

- 相互系切り替え構成/2:1 系切り替え構成の場合

  相互系切り替え構成の場合は，更新可能 BES と参照専用 BES 以外(MGR,FES,DS)の相互系切り替え構成の場合は，特別な設定はありません。系切り替えの設定は，マニュアル「HiRDB システム運用ガイド」の「クラスタソフトウェアによる共有ディスクのアクセス制御」を参照してください。

  更新可能 BES と参照専用 BES の相互系切り替え構成の場合は，共有ディスクの切り替えができません。系切り替えの設定は，マニュアル「HiRDB システム運用ガイド」の「HiRDB による共有ディスクのアクセス制御」を参照してください。

(b) 1：1 スタンバイレス型系切り替えの場合

更新可能 BES で 1：1 スタンバイレス型系切り替え構成の場合は，系切り替え先は参照専用 BES であり，共有ディスクの切り替えができません。そのため，系切り替えの設定は，マニュアル「HiRDB システム運用ガイド」の「HiRDB による共有ディスクのアクセス制御」を参照してください。

(c) 影響分散スタンバイレス型系切り替えの場合

更新可能 BES で影響分散スタンバイレス型系切り替え構成の場合は，系切り替え先は参照専用 BES であり，共有ディスクの切り替えができません。そのため，系切り替えの設定は，マニュアル「HiRDB システム運用ガイド」の「HiRDB による共有ディスクのアクセス制御」を参照してください。

注

参照専用 BES の場合，共用 RD エリアの共有ディスクをクラスタソフトウェアの管理リソースとしないでください。

(7) 共用 RD エリアに対するユティリティ及び運用コマンドの実行

ユティリティ及び運用コマンドで共用 RD エリア，共用表，又は共用インデクスを対象とする場合，HiRDB が内部的に LOCK TABLE 文を発行し，全バックエンドサーバ下の共用 RD エリアに排他を掛けることがあります。このため，共用 RD エリア内の表やインデクスにアクセス中の業務があると，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ことがあります。ユティリティ及び運用コマンド実行時は，対象となる共用 RD エリアをコマンド閉塞しておいてください。

(8) 共用 RD エリア使用上の制限事項

1. 系切り替え機能を使用する場合，更新可能バックエンドサーバがあるユニットは次のように配置してください。
   - 参照専用バックエンドサーバと異なるホストに配置
   - 系を切り替えたときに同一ホスト内で参照バックエンドサーバと混在しないように切り替え先を配置

   参照専用バックエンドサーバは，共用 RD エリアのディスクボリュームをクラスタソフトウェアの管理リソースにしないようにしてください。
2. 共用 RD エリアは全バックエンドサーバに配置されるため，フロータブルサーバは設置できません。
3. レプリケーションの反映先に共用表は指定できません。
4. 共用 RD エリアを更新可能なオンライン再編成の対象とする場合，サーバ単位（-s オプション）の実行はできません。ただし，一部のバックエンドサーバが停止中ですぐに起動できないなどの場合，次の操作は，サーバ単位に実行できます。
   - 強制的に，起動中のバックエンドサーバだけでオンライン再編成閉塞（オンライン再編成のデータベース静止化）を中止する（pdorbegin -u）
   - 強制的に，起動中のバックエンドサーバだけで更新可能なオンライン再編成の追い付き反映処理を中止する（pdorend -u）

   このとき起動していなかったバックエンドサーバは，起動完了後にほかのバックエンドサーバと同じ状態にしてください。
5. ローカルバッファで共用 RD エリアの表やインデクスを更新する場合は，LOCK TABLE 文を発行して更新してください。LOCK TABLE 文を発行しないで更新していると，サーバプロセスが異常終了してトランザクション回復プロセスが回復処理するとき，グローバルバッファで回復対象になる更新ページを保持できないことがあります。更新ページを保持できないと回復できないので，アボートコード Phb3008 を出力してユニットは異常終了します。この場合，HiRDB を再開始してください。

13.7　一時表用 RD エリア

13.7.1　一時表用 RD エリアの概要

一時表用 RD エリアは，一時表や一時インデクスを格納するユーザ用 RD エリアです。

(1)　適用基準

一時表を使用する場合に必要です。

(2)　一時表用 RD エリアの属性

一時表用 RD エリアには，次の二つがあります。

- SQL セッション間共有属性の一時表用 RD エリア

  すべての SQL セッションで使用できる一時表用 RD エリアです。SQL セッション間で共有させることで一時表用 RD エリアの個数を少なくする場合に使用します。

- 特定 SQL セッション占有属性の一時表用 RD エリア

  特定の SQL セッション（クライアント環境定義 PDTMPTBLRDAREA に使用する一時表用 RD エリアを指定している SQL セッション）だけが使用する一時表用 RD エリアです。特定のユーザが膨大なデータを扱い，ほかのユーザが使用する一時表用 RD エリアを圧迫する場合，それを避けるときに使用します。

(3)　作成方法

一時表用 RD エリアの作成方法を次に示します。

1. pd_max_temporary_object_no オペランドに，ある一時点で使用する一時表と一時インデクスの最大数を指定します。0904 互換モードを適用している場合，pd_max_temporary_object_no オペランドの指定に加えて，pd_max_tmp_table_rdarea_no オペランドに一時表用 RD エリアの最大数を指定します。
2. pdfmkfs コマンドの-k オプション（使用目的）に DB を指定し，HiRDB ファイルシステム領域を作成します。
3. 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）の create rdarea 文に次の二つのオペランドを指定して，ユーザ用 RD エリアを作成します。
   - for user used by PUBLIC を指定します。

     一時表用 RD エリアとして使用するには，公用 RD エリアである必要があります。
   - temporary table use を指定します。

     SQL セッション間共有属性の場合は temporary table use shared を，特定 SQL セッション占有属性の場合は temporary table use occupied を指定します。

作成時の注意

- 0904 互換モードを適用している場合，pd_max_tmp_table_rdarea_no オペランドの指定値は，pd_max_rdarea_no オペランドの指定値より小さい値にしてください。一時表用 RD エリアを追加する場合は，追加する一時表用 RD エリアも含めて，pd_max_rdarea_no オペランドで指定する RD エリアの最大数を超えないようにしてください。超える場合，pd_max_rdarea_no オペランドの指定値を変更してください。
- 使用できる一時表用 RD エリアが複数ある場合，使用する一時表用 RD エリアは HiRDB が自動的に決定します。そのため，一つの UAP が使用できる一時表用 RD エリアは，RD エリアの容量，セグメントサイズ，及びページサイズを同じにすることをお勧めします。

  RD エリアの容量，セグメントサイズ，及びページサイズを統一していないと，次の現象が発生します。

  ・一時表への，初回の INSERT 文の実行によって実体化するごとに，ページサイズが異なる RD エリアを格納先として決定する可能性があります。この場合，同じ挿入データであっても，挿入データの行長と格納先 RD エリアのページサイズの大小関係によって，初回の INSERT 文の実行可否が変わります。

  ・格納先 RD エリアは空きセグメント数が多い RD エリアを優先的に選択しますが，格納先 RD エリアの候補となる RD エリアのページサイズやセグメントサイズが異なると，空き容量の小さい RD エリアを格納先 RD エリアとして選択することがあります。
- 一時表用 RD エリアは HiRDB 開始時，又は一時表への最初の INSERT 文実行時に初期化されるため，容量が大き過ぎると初期化のオーバヘッドが大きくなります。一時表用 RD エリアの初期化については，「一時表用 RD エリアの初期化」を参照してください。

作成例

HiRDB/パラレルサーバの BES1 に，SQL セッション間共有属性の一時表用 RD エリア（RDTMP01）を作成する場合のデータベース構成変更ユティリティ（pdmod）の制御文の例を次に示します。

```
create rdarea RDTMP01    …一時表用RDエリア名を指定
  globalbuffer tmpbuf01
  for user used by PUBLIC    …公用RDエリアの指定
  server name BES1
  open attribute INITIAL
  page 4096 characters
  storage control segment 100 pages
  temporary table use shared    …SQLセッション間共有属性を指定
  file name  “/hirdb/db/rdtmp01_f01”
    initial 500 segments ;
```

(4)　一時表用 RD エリアの初期化

HiRDB 開始時に，開始モードに関係なく，HiRDB は前回稼働時に使用した一時表用 RD エリアを初期化します。そのため，初期化する一時表用 RD エリアの容量が多いと，HiRDB の開始に時間が掛かります。高速系切り替え機能やスタンバイレス型系切り替え機能を使用していて，系切り替え時間を短縮する必要がある場合，pd_tmp_table_initialize_timing オペランドに ACCESS を指定することで，HiRDB 開始時に一時表用 RD エリアの初期化をしないようにできます。ただし，ACCESS を指定すると，一時表に最初

に INSERT 文が実行された時点で一時表用 RD エリアを初期化するため，初期化する容量が多いと，INSERT 実行のオーバヘッドが大きくなります。オーバヘッドを小さくするには，一時表用 RD エリアの容量を小さくしてください。

pd_tmp_table_initialize_timing オペランド，及び一時表用 RD エリアの初期化によるオーバヘッド量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。

(5) 一時表用 RD エリアのバックアップ

一時表のデータは，トランザクション又は SQL セッションの期間中だけ保持するデータのため，一時表用 RD エリアのバックアップは取得不要です。一時表用 RD エリアに障害が発生した場合は，データベース構成変更ユティリティ（pdmod）の initialize rdarea 文で RD エリアを再初期化してください。

(6) 一時表用 RD エリア使用上の留意事項

(a) 制限事項

一時表用 RD エリアに対して，次に示す運用コマンド又はユティリティを実行する場合，一部制限があります。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。

- pdhold コマンド
- pdorcheck コマンド
- pdorcreate コマンド
- pdrdrefls コマンド
- データベース複写ユティリティ（pdcopy）
- データベース状態解析ユティリティ（pddbst）
- データベース構成変更ユティリティ（pdmod）
  RD エリアの移動，及び RD エリアのレプリカ定義はできません。
- データベース回復ユティリティ（pdrstr）

(b) 特定 SQL セッション占有属性の一時表用 RD エリアを使用する場合

特定 SQL セッション占有属性の一時表用 RD エリアを使用する SQL セッションは，クライアント環境定義 PDTMPTBLRDAREA に該当する一時表用 RD エリアを指定している SQL セッションだけです。該当する一時表用 RD エリアを，さらに占有する（ほかの SQL セッションで使用できないようにする）には，ローカルバッファを割り当ててください。これによって，該当する一時表用 RD エリアに排他モード（EX）の排他が掛かります。

(c) 非 FIX の一時表を作成する場合

非 FIX の一時表を作成する場合，一時表用 RD エリアは次のどちらかの条件を満たしている必要があります。条件を満たしていない場合，データを格納するときに KFPA11809-I メッセージが出力されることがあります。

- クライアント環境定義の PDTMPTBLRDAREA を指定している場合

  PDTMPTBLRDAREA に指定しているすべての一時表用 RD エリアの定義に，同じページ長※を定義している。

- クライアント環境変数の PDTMPTBLRDAREA を指定していない場合

  すべての SQL セッション間共有属性の一時表用 RD エリアの定義に，同じページ長※を定義している。

  注※

  ページ長は，基本行長より大きい値となります。基本行長については，マニュアル「HiRDB SQL リファレンス」の「既定義型データ長一覧」のデータ長の計算式を基に算出してください。KFPA11809-I メッセージが出力された場合は，KFPA11809-I メッセージの「格納しようとした行長」より大きい値を指定してください。

(d) HiRDB Datareplicator と連携する場合

一時表に対する更新は抽出対象外になります。そのため，HiRDB Datareplicator を使用する場合，一時表について考慮する点は特にありません。

14 HiRDB のメモリ所要量

この章では，HiRDB/シングルサーバ及び HiRDB/パラレルサーバのメモリ所要量の求め方について説明します。

14.1　HiRDB/シングルサーバのメモリ所要量の見積もり

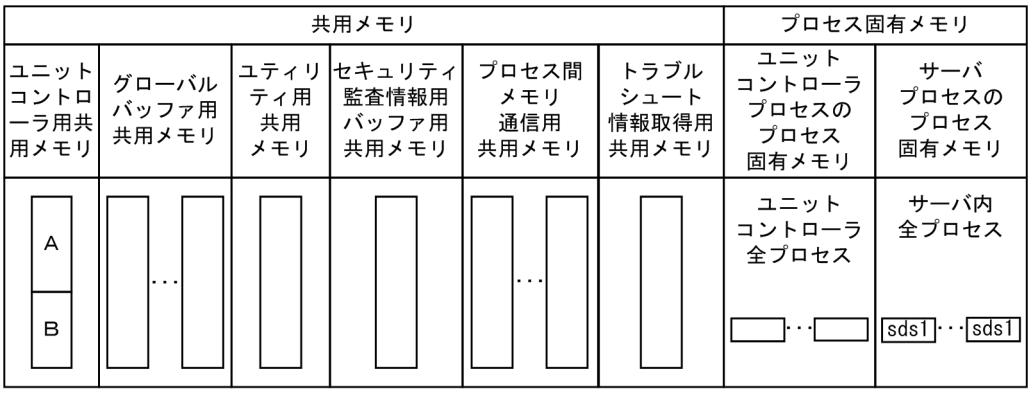
ここでは，HiRDB/シングルサーバのメモリ所要量の見積もり方法について説明します。ここで説明する項目を次に示します。

- メモリ配置
- メモリ所要量の計算式
- ユニットコントローラが使用する共用メモリの計算式
- シングルサーバが使用する共用メモリの計算式
- グローバルバッファが使用する共用メモリの計算式
- SQL 実行時に必要なメモリ所要量の計算式
- SQL 前処理時に必要なメモリ所要量の計算式
- BLOB 型データの検索又は更新時に必要なメモリ所要量の計算式
- ブロック転送又は配列 FETCH で必要なメモリ所要量の計算式

14.1.1　メモリ配置

HiRDB/シングルサーバのメモリ配置を次の図に示します。

図 14–1　HiRDB/シングルサーバのメモリ配置



（凡例）Ａ：ユニットコントローラの各プロセス使用分
Ｂ：シングルサーバのプロセス使用分
sds：シングルサーバ

HiRDB/シングルサーバの共用メモリの詳細を次の表に示します。

表 14–1　HiRDB/シングルサーバの共用メモリの詳細

項目	共用メモリの種類					
	ユニットコントローラ用共用メモリ	グローバルバッファ用共用メモリ	ユティリティ用共用メモリ	セキュリティ監査情報用バッファ用共用メモリ	プロセス間メモリ通信用共用メモリ	トラブルシュート情報取得用共用メモリ
使用目的	システム制御	グローバルバッファ	ユニットコントローラとユティリティとの通信	セキュリティ監査情報用バッファ	クライアント—サーバプロセス間メモリ通信	トラブルシュート情報取得
使用プロセス	全 HiRDB プロセス	シングルサーバ	ユティリティプロセス	シングルサーバ	シングルサーバ，クライアントプロセス	全 HiRDB プロセス
セグメント数	1 個	• グローバルバッファの動的変更機能を使用しない場合 1～512 個 • グローバルバッファの動的変更機能を使用している場合 32 ビットモード：1～1,012 個 64 ビットモード：1～1,512 個	1 個	1 個	環境変数 PDIPC=MEMORY で接続中のクライアント数（0～2000）×2 個	1 個
1 セグメントの上限	表「HiRDB/シングルサーバが使用するメモリ所要量」を参照してください。オペレーティングシステムパラメタの shmmax の値は，計算値以上にしてください。	SHMMAX オペランドの値でセグメントを分割します。オペレーティングシステムパラメタの shmmax の値は，SHMMAX オペランドの値以上にしてください。	表「HiRDB/シングルサーバが使用するメモリ所要量」を参照してください。オペレーティングシステムパラメタの shmmax の値は，計算値以上にしてください。	表「HiRDB/シングルサーバが使用するメモリ所要量」を参照してください。オペレーティングシステムパラメタの shmmax の値は，計算値以上にしてください。	表「HiRDB/シングルサーバが使用するメモリ所要量」を参照してください。オペレーティングシステムパラメタの shmmax の値は，計算値以上にしてください。	10 メガバイト

項目	共用メモリの種類					
	ユニットコントローラ用共用メモリ	グローバルバッファ用共用メモリ	ユティリティ用共用メモリ	セキュリティ監査情報用バッファ用共用メモリ	プロセス間メモリ通信用共用メモリ	トラブルシュート情報取得用共用メモリ
確保条件	なし	グローバルバッファ定義が存在すること	pd_utl_exec_mode=1指定	pd_aud_file_nameオペランドによる監査証跡ファイル用のHiRDBファイルシステム領域名の指定	クライアント環境変数PDIPC=MEMORYで接続中クライアントあり	なし
作成契機	ユニット起動時(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使用時の待機ユニットの起動を含む)	• サーバ起動時（高速系切り替え機能使用時の待機ユニットの起動を含む) • pdbufmod -k {add | upd}実行時	ユティリティ実行時	シングルサーバ起動時	クライアントとサーバの接続時	ユニット起動時（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使用時の待機ユニットの起動を含む)
削除契機	次回ユニット起動時（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使用時の待機ユニットの起動を含む)	• pdbufmod -k del 実行時 • 正常終了又は計画停止の場合：サーバ終了時 • 強制停止,異常終了,又は高速系切り替え機能使用時の待機ユニットの停止の場合：次回ユニット起動時	ユティリティ終了後10分後	シングルサーバ終了時	クライアントとサーバの接続切り離し時	ユニット停止時
pdls -d mem による表示	表示される	表示される	表示される	表示される	表示されない	表示されない
pdls -d mem のSHM-OWNER	MANAGER	サーバ名	UTILITY	AUDDEF	表示なし	表示なし

項目	共用メモリの種類					
	ユニットコントローラ用共用メモリ	グローバルバッファ用共用メモリ	ユティリティ用共用メモリ	セキュリティ監査情報用バッファ用共用メモリ	プロセス間メモリ通信用共用メモリ	トラブルシュート情報取得用共用メモリ
関連オペランド	• pd_shmpool_attribute • pd_sds_shmpool_size	• pd_dbbuff_attribute • pd_dbbuff_modify • pdbuffer • SHMMAX	• pd_utl_exec_mode	• セキュリティ監査機能に関するオペランド※	• PDIPC • PDSENDMEMSIZE • PDRECVMEMSIZE	なし
備考	－	－	pd_utl_exec_mode=1の場合だけ作成されます（pd_utl_exec_mode=0の場合，該当する領域はユニットコントローラ用共用メモリ内に確保されます）。	－	－	－

（凡例）－：該当しません。

注※

詳細はマニュアル「HiRDB システム定義」を参照してください。

14.1.2 メモリ所要量の計算式

HiRDB/シングルサーバが使用するメモリ所要量は，次の表に示すすべての項目を加算した値です。

なお，OS のオペレーティングシステムパラメタの shmmax の指定値については，「OS のオペレーティングシステムパラメタの見積もり」を参照してください。

共用メモリサイズが増加すると，ページフォルトの発生回数が増加し，トランザクション性能に影響を与えるおそれがあります。各オペランドの指定値の目安を参照し，システムに合わせて適切な値を指定することを検討してください。

表 14–2 HiRDB/シングルサーバが使用するメモリ所要量

項目		メモリ所要量（単位：キロバイト）
プロセス固有領域	ユニットコントローラ全プロセスが使用するプロセス固有領域	●32 ビットモードの場合

項目			メモリ所要量（単位：キロバイト）
			E＋↑｛(64＋48×（u＋1)）×(pd_max_server_process の値−b−6）＋（64＋48×(y＋1)）×3＋w＋R｝÷1024↑ ●64 ビットモードの場合 E＋↑｛(64＋64×（u＋1)）×(pd_max_server_process の値−b−6）＋（64＋64×(y＋1)）×3＋w｝÷1024↑＋R ●プラグインを使用する場合，加算します。 ＋1400 ●pd_max_ard_process オペランドが 1 以上の場合，加算します。 ＋r ●リアルタイム SAN レプリケーションを使用する場合，加算します。 ＋↑425×（2×b＋7）÷1024↑ ●pd_process_terminator オペランドに fixed を指定した場合，加算します。 ＋F×（pd_process_terminator_max の値−1） ●インメモリデータ処理を行う場合，加算します。 ＋↑｛K×（pd_max_users の値×2＋7)｝÷1024↑ ●通信トレース格納最大数を変更する場合，加算します。 ＋↑M÷1024↑
	シングルサーバプロセスが使用するプロセス固有領域※1	pd_work_buff_mode=each 指定時	●32 ビットモードの場合 ｛G＋g＋（a＋9）×c＋h＋i＋m＋p＋q＋s｝×（b＋3）＋↑（64＋48×（v＋1））÷1024↑×（b＋3）＋J ●64 ビットモードの場合 ｛G＋g＋（a＋9）×c＋h＋i＋m＋p＋q＋s｝×（b＋3）＋↑（64＋64×（v＋1））÷1024↑×（b＋3）＋J ●インメモリデータ処理を行う場合，加算します。 ＋↑｛K×（b＋3)｝÷1024↑ ●通信トレース格納最大数を変更する場合，加算します。 ＋↑P÷1024↑
		pd_work_buff_mode=pool 指定又は省略時	●32 ビットモードの場合 (G＋g＋a＋↑a÷128×0.1↑＋11＋h＋i＋m＋n＋p＋q＋s）×（b＋3）＋↑（64＋48×（v＋1））÷1024↑×（b＋3）＋J ●64 ビットモードの場合 (G＋g＋a＋↑a÷128×0.1↑＋15＋h＋i＋m＋n＋p＋q＋s）×（b＋3）＋↑（64＋64×（v＋1））÷1024↑×（b＋3）＋J ●インメモリデータ処理を行う場合，加算します。 ＋↑｛K×（b＋3)｝÷1024↑

項目			メモリ所要量（単位：キロバイト）
			●通信トレース格納最大数を変更する場合，加算します。 ＋↑P÷1024↑
		pd_work_buff_mode=pool2 指定	**●32 ビットモードの場合** (G＋g＋↑a÷128×0.1↑＋11＋h＋i＋m＋n＋p＋q＋s)×(b＋3)＋↑(64＋48×(v＋1))÷1024↑×(b＋3)＋J＋(a×S) **●64 ビットモードの場合** (G＋g＋↑a÷128×0.1↑＋15＋h＋i＋m＋n＋p＋q＋s)×(b＋3)＋↑(64＋64×(v＋1))÷1024↑×(b＋3)＋J＋(a×S) **●インメモリデータ処理を行う場合，加算します。** ＋↑{K×(b＋3)}÷1024↑ **●通信トレース格納最大数を変更する場合，加算します。** ＋↑P÷1024↑
共用メモリ	ユニットコントローラ用共用メモリ中，ユニットコントローラが使用する領域		↑d÷1024↑
	ユニットコントローラ用共用メモリ中，シングルサーバが使用する領域		e
	グローバルバッファ用共用メモリ		f
	インメモリデータ処理用共用メモリ		L
	ユティリティ用共用メモリ		t
	セキュリティ監査情報用バッファ用共用メモリ		**●システムによる自動計算の場合** ↑0.3＋MAX{(H＋100)，(H×1.2)}×0.25↑ **●ユーザ指定値（pd_audit_def_buffer_size オペランドを指定）にする場合** pd_audit_def_buffer_size の指定値
	プロセス間メモリ通信用共用メモリ※2		j×k
OS の領域※3	PTE の領域※4		**●共用メモリのページ固定をしていない場合** (↑プロセス固有領域の合計サイズ÷510↑＋pd_max_server_process の値×16＋13)＋{(↑共用メモリの合計サイズ÷510↑＋17)×pd_max_server_process の値} **●共用メモリのページ固定をしている場合** (↑プロセス固有領域の合計サイズ÷510↑＋pd_max_server_process の値×16＋13)＋{(↑共用メモリの合計サイズ÷261120↑＋13)×pd_max_server_process の値}

注※1

プラグインを使用する場合は，1 シングルサーバプロセス当たり 300 を加算してください。

注※2

クライアント環境定義で PDIPC=MEMORY を指定した場合に加算します。プロセス間メモリ通信機能及び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。

なお，HiRDB サーバ又は HiRDB クライアントのどちらかが 32 ビットモードの場合，プロセス間メモリ通信機能で使用する共用メモリは 32 ビット空間内に確保されます。

注※3

OS が使用する領域の詳細は公開されていないため，あくまで HiRDB が確保するメモリ所要量を基にした目安です。実際の値とは大きく異なる場合があるので，正確な値については実際の環境で測定してください。

注※4

PTE については，「システム設計」を参照してください。

a：pd_work_buff_size オペランドの値

b：pd_max_users オペランドの値＋ pd_max_reflect_process_count オペランドの値

c：最大作業表数

SQL 文ごとの作業表数を表「SQL 文ごとの作業表数の求め方」から求めます。表「SQL 文ごとの作業表数の求め方」から求めた作業表数のうちで最大のものを最大作業表数とします。

d：「ユニットコントローラが使用する共用メモリの計算式」で求めた値

e：「シングルサーバが使用する共用メモリの計算式」で求めた値

f：「グローバルバッファが使用する共用メモリの計算式」で求めた値

g：SQL 実行時に必要なメモリ所要量

計算式については，「SQL 実行時に必要なメモリ所要量の計算式」を参照してください。

h：SQL 前処理時に必要なメモリ所要量

計算式については，「SQL 前処理時に必要なメモリ所要量の計算式」を参照してください。

i：LOB バッファ一括入出力ワークメモリ

グローバルバッファ定義に LOB 用グローバルバッファを指定している場合だけ（システム共通定義の pdbuffer オペランドに-b を指定している場合），62 キロバイトを加算してください。

j：プロセス間メモリ通信機能を使用するクライアントの最大同時実行数

分からない場合は，プロセス間メモリ通信機能を使用する全クライアント数，又は pd_max_users オペランドの値を代入してください。

k：プロセス間メモリ通信機能を使用する全クライアントのデータ送受信用メモリサイズ（クライアント環境定義の PDSENDMEMSIZE の値＋ PDRECVMEMSIZE の値）の平均値

m：Java 仮想マシンが使用するメモリ所要量

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，Java 仮想マシンが使用するメモリ所要量を加算します。Java 仮想マシンが使用するメモリ所要量は，Java 仮想マシンのオプション（Hewlett-Packard JRE 1.2.2.04 の場合は-Xms，-Xmx，-Xmn オプション）や Java 仮想マシンのバージョンによって異なります。Java 仮想マシンが使用するメモリ所要量については，Java 仮想マシンのマニュアルを参照してください。

n：作業表用増分メモリサイズ

pd_work_buff_expand_limit オペランドを指定する場合に作業表用増分メモリサイズを加算します。作業表用増分メモリサイズは次に示す計算式から求めます。

作業表用増分メモリサイズ（キロバイト）＝作業表用増分バッファサイズ＋↑（作業表用の増分バッファサイズ÷128）×0.1↑

- 作業表用増分バッファサイズ（キロバイト）＝ MAX（0，ハッシュジョイン，副問合せのハッシュ実行による作業表用増分バッファサイズ）＋ MAX（0，作業表数の増加による作業表用増分バッファサイズ）
- ハッシュジョイン，副問合せのハッシュ実行による作業表用増分バッファサイズ＝ MIN｛(ハッシュジョイン，副問合せのハッシュ実行をするときの作業表用バッファサイズ－pd_work_buff_size オペランドの値)，(pd_work_buff_expand_limit オペランドの値－pd_work_buff_size オペランドの値)｝×ハッシュジョイン，副問合せのハッシュ実行をする同時実行ユーザ数

  ハッシュジョイン，副問合せのハッシュ実行をするときの作業表用バッファサイズの求め方については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。
- 作業表数の増加による作業表用増分バッファサイズ＝ MIN｛(使用作業表数×128－pd_work_buff_size オペランドの値)，(pd_work_buff_expand_limit オペランドの値－pd_work_buff_size オペランドの値)｝×（使用作業表数が pd_work_buff_size オペランドの値÷128 以上になるユーザ数）

  使用作業表数＝ MAX（1SQL 文が使用する作業表用ファイルの数，ASSIGN LIST 文が使用する作業表用ファイルの数）

  1SQL 文が使用する作業表用ファイルの数，及び ASSIGN LIST 文が使用する作業表用ファイルの数の求め方については，「最大ファイル数の見積もり（pdfmkfs -l コマンド）」を参照してください。

p：BLOB 型データ用に必要なメモリ所要量

計算式については，「BLOB 型データの検索又は更新時に必要なメモリ所要量の計算式（HiRDB/シングルサーバの場合）」を参照してください。

q：サーバ側でブロック転送又は配列 FETCH で必要なメモリ所要量

計算式については，「ブロック転送又は配列 FETCH で必要なメモリ所要量の計算式」を参照してください。

r：非同期 READ 用メモリサイズ

pd_max_ard_process オペランドが 1 以上の場合に加算します。次に示す計算式から求めます（単位：キロバイト）。

```
 (90＋
90
```

Σ RDエリア用HiRDBファイルシステム領域管理用メモリ）
i=1
×pd_max_ard_processの値

RD エリア用 HiRDB ファイルシステム領域管理用メモリは計算値の大きい順に 90 領域を計算に使用します。サーバで使用する領域数が 90 領域に満たない場合，その領域まで計算します。

HiRDB ファイルシステム領域管理用メモリはそれぞれの領域の初期設定時のパラメタを使用して，次の計算式から求めます（単位：キロバイト）。

なお，領域の初期設定時のパラメタは pdfstatfs コマンドに-A オプションを指定して実行することで確認できます。

｛（ファイル数※1＋増分数※2）÷64｝×1.5※3

注※1　pdfmkfs -l 指定値，又は pdfstatfs 実行結果の[available file count]に表示される値です。

注※2　pdfmkfs -e 指定値，又は pdfstatfs 実行結果の[available expand count]に表示される値です。

注※3　領域サイズ（pdfmkfs -n 指定値）が 2048 以上の場合に乗算します。

s：HiRDB ファイルシステム用メモリサイズ

次に示す計算式から求めます（単位：キロバイト）。

604＋作業表用HiRDBファイルシステム領域管理用メモリ＋
システムログ用HiRDBファイルシステム領域管理用メモリ＋
90
ΣRDエリア用HiRDBファイルシステム領域管理用メモリ
i=1

作業表用及びシステムログ用 HiRDB ファイルシステム領域管理用メモリは，サーバで使用する HiRDB ファイルシステム領域で計算値が最大になるものを使用します。RD エリアの場合は計算値の大きい順に 90 領域を計算に使用します。サーバで使用する領域数が 90 領域に満たない場合，その領域まで計算します。

HiRDB ファイルシステム領域管理用メモリはそれぞれの領域の初期設定時のパラメタを使用して，次の計算式から求めます（単位：キロバイト）。

なお，領域の初期設定時のパラメタは pdfstatfs コマンドに-A オプションを指定して実行することで確認できます。

｛（ファイル数※1＋増分数※2）÷64｝×1.5※3

注※1　pdfmkfs -l 指定値，又は pdfstatfs 実行結果の[available file count]に表示される値です。

注※2　pdfmkfs -e 指定値，又は pdfstatfs 実行結果の[available expand count]に表示される値です。

注※3　領域サイズ（pdfmkfs -n 指定値）が 2048 以上の場合に乗算します。

t：pd_utl_exec_mode の値が 0 の場合：0

pd_utl_exec_mode の値が 1 の場合：↑｛(b×2000 + 136）÷1024｝↑×1024

u：ユニット制御情報定義として有効な pd_module_trace_max の値

v：シングルサーバ定義として有効な pd_module_trace_max の値

w：HiRDB の再開始用メモリサイズ

このメモリサイズを確保できないと，HiRDB の再開始に失敗します。次の計算式から求めます（単位：バイト）。

```
A＋B

●pd_dbsync_pointオペランドにcommitを指定している場合，加算します。
＋112×（（pd_max_usersの値＋pd_max_reflect_process_countの値）×2＋7）
●pd_inner_replica_controlオペランドに1以上を指定している場合，加算します。
＋C
●キャラクタ型スペシャルファイル上に作成されたRDエリアを格納したHiRDBファイルシステム領域数が1001以上の場合に加算します。
＋D
```

y：システム共通定義，又はユニット制御情報定義に pd_module_trace_max オペランドを指定している場合：pd_module_trace_max の値

それ以外：16383

HiRDB の再開始用メモリサイズを求める計算に使用する変数を次に示します。

変数	値
A	**●32 ビットモードの場合** 246762 ＋ 4×pd_max_rdarea_no の値 ＋ {48×（pd_max_rdarea_no の値＋表数）＋ 304} ×（(pd_max_users の値＋pd_max_reflect_process_count の値）×2 ＋ 7） **●64 ビットモードの場合** 305274 ＋ 8×pd_max_rdarea_no の値 ＋ {64×（pd_max_rdarea_no の値＋表数）＋ 512} ×（(pd_max_users の値＋pd_max_reflect_process_count の値）×2 ＋ 7） **表数**：62 ＋ MAX｛pd_max_access_tables の値，500｝
B	b1×X ＋ b2×Y **b1：サーバ用ステータスファイルのレコード長＜ 4096 の場合** MAX（(↓（3400÷（(↓（((レコード長－40）－308）÷20↓） ＋（↓（レコード長－40）÷20↓）×（MAX（↓4096÷レコード長↓，2）－1)) ＋ 0.7）↓)，1）×MAX(↓4096÷レコード長↓，2）×（レコード長－40） **4096≦サーバ用ステータスファイルのレコード長＜ 12288 の場合** MAX（↓（3400÷（↓（(((レコード長－40）－308）÷20）↓）＋ 0.7）↓，1） ×（レコード長－40） **12288≦サーバ用ステータスファイルのレコード長の場合** MAX（↓（3400÷（↓（(((レコード長－40）－836）÷20）↓）＋ 0.7）↓，1） ×（レコード長－40） X：RD エリア数≦3400 の場合：1 3401≦RD エリア数≦6800 の場合：2 6801≦RD エリア数の場合：3 **b2：サーバ用ステータスファイルのレコード長＜ 4096 の場合** （↓（5662310÷（(↓（((レコード長－40）－308）÷20↓） ＋（↓（レコード長－40）÷20↓）×（MAX（↓4096÷レコード長↓，2）－1))

変数	値
	＋0.7）↓）×MAX（↓4096÷レコード長↓，2）×（レコード長−40） **4096≦サーバ用ステータスファイルのレコード長＜12288 の場合** ↓（5662310÷（↓（((レコード長−40）−308）÷20）↓）＋0.7）↓ ×（レコード長−40） **12288≦サーバ用ステータスファイルのレコード長の場合** ↓（5662310÷（↓（((レコード長−40）−836）÷20）↓）＋0.7）↓ ×（レコード長−40） Y：RD エリア数≦10200 の場合：0 10201≦RD エリア数≦5672510 の場合：1 5672511≦RD エリア数≦11334820 の場合：2 11334821≦RD エリア数の場合：3
C	**●32 ビットモードの場合** （48×pd_max_rdarea_no の値＋ 80）×（(pd_max_users の値＋ pd_max_reflect_process_count の値）×2 ＋ 7） **●64 ビットモードの場合** （64×pd_max_rdarea_no の値＋ 160）×（(pd_max_users の値＋ pd_max_reflect_process_count の値）×2 ＋ 7）
D	**●32 ビットモードの場合** 12012×（↑（キャラクタ型スペシャルファイル上に作成された RD エリアを格納した HiRDB ファイルシステム領域数−1000）÷1000↑） **●64 ビットモードの場合** 16016×（↑（キャラクタ型スペシャルファイル上に作成された RD エリアを格納した HiRDB ファイルシステム領域数−1000）÷1000↑）

E，F，G：固定値

この値は OS によって異なります。OS ごとの値を次に示します（単位：キロバイト）。

OS	E の値	F の値	G の値
AIX	156,900	6,000	12,200
Linux	123,700	1,900	5,800

H：余裕を持って見積もる場合は監査対象イベントの数（CREATE AUDIT の実行回数），詳細に見積もる場合はセキュリティ監査情報用バッファのエントリ数

J：シンクポイント出力同期制御情報取得機能使用時に必要なメモリ（単位：バイト）

pd_dbbuff_trace_level オペランドに 1 を指定し，かつ pd_dfw_awt_process オペランドの指定を省略している場合に，次の値を加算します。

32 ビットモードの場合：

320×シングルサーバに定義したグローバルバッファの数

64 ビットモードの場合：

640×シングルサーバに定義したグローバルバッファの数

K：pd_max_resident_rdarea_no オペランドに 1 以上を指定している場合に，次の値を加算します。

1648 ＋ 16×pd_max_resident_rdarea_no の値＋ 16×pd_max_resident_rdarea_shm_no の値

L：インメモリデータ処理で必要なメモリ所要量

計算式については，「インメモリデータ処理で必要なメモリ所要量」を参照してください。

M：通信トレース処理で必要なメモリ所要量

32 ビットモードの場合：

（16×（N－1024）×2）×（pd_max_server_process の値－pd_max_users の値－3）

64 ビットモードの場合：

（32×（N－1024）×2）×（pd_max_server_process の値－pd_max_users の値－3）

N：ユニット制御情報定義として有効な pd_pth_trace_max の値

オペランドの指定値を 2 のべき乗に切り上げた値になります。

P：通信トレース処理で必要なメモリ所要量

32 ビットモードの場合：

（16×（Q－1024）×2）×（pd_max_users の値＋ 3）

64 ビットモードの場合：

（32×（Q－1024）×2）×（pd_max_users の値＋ 3）

Q：シングルサーバ定義として有効な pd_pth_trace_max の値

オペランドの指定値を 2 のべき乗に切り上げた値になります。

R：シグナルハンドラ用メモリサイズ

0

S：作業表を使用する操作（SQL 及びユティリティ）を行うトランザクションの同時実行数

表 14-3　SQL 文ごとの作業表数の求め方

SQL 文	作業表数の求め方
SELECT 文 INSERT 文（問合せ指定の場合）	1.～8. の指定がない場合：0 1.～8. の指定がある場合：該当する作業表数をすべて加算した値 1. 複数の表を結合して検索する場合 増加作業表数＝（結合表数－1）×2 ＋ 1 2. ORDER BY 句を指定する場合 増加作業表数＝ 2 3. GROUP BY 句を指定する場合 増加作業表数＝ GROUP BY 句指定数 4. DISTINCT 句を指定する場合 増加作業表数＝ DISTINCT 句指定数 5. UNION 句，UNION ALL 句又は EXCEPT[ALL]句を指定する場合 増加作業表数＝（UNION 又は UNION ALL 句指定数）×2 ＋ 1 6. 探索条件中にインデクスを定義した列がある場合

SQL 文	作業表数の求め方
	増加作業表数＝探索条件中のインデクスを定義した列数 7. FOR UPDATE 句又は FOR READ ONLY 句を指定する場合 増加作業表数＝ 1 8. 副問合せ（限定述語）を指定する場合 増加作業表数＝副問合せ指定数
UPDATE 文 DELETE 文	探索条件中のインデクスを定義した列数＋ 1
DROP SCHEMA 文 DROP TABLE 文 DROP INDEX 文 CREATE INDEX 文 REVOKE 文でアクセス権限を取り消す場合	2

14.1.3　ユニットコントローラが使用する共用メモリの計算式

(1)　32 ビットモードの HiRDB の場合

HiRDB/シングルサーバの開始から終了までの間にユニットコントローラが使用する共用メモリは，次に示す HiRDB のプロセスの項目すべてを加算した値です。

なお，ユニットコントローラ全体の共用メモリサイズは 2 ギガバイト以内になるようにしてください。

プロセスの種類	共用メモリの計算式（単位：バイト）
スケジューラ	pd_utl_exec_mode の値が 0 の場合 ｛↑（432 ＋ 304×n）÷1024↑＋ 500 ＋ x ＋↑（134 ＋ pd_trn_rcvmsg_store_buflen の値）÷1024↑｝×1024 pd_utl_exec_mode の値が 1 の場合 ｛↑（432 ＋ 304×n）÷1024↑＋↑（m×2000 ＋ 136）÷1024↑＋ y ＋↑（134 ＋ pd_trn_rcvmsg_store_buflen の値）÷1024↑｝×1024 x：シングルサーバの場合：116 ＋ 5×（m ＋ 3）＋ 14 ユティリティ専用ユニットの場合：0 y：シングルサーバの場合：5×（m ＋ 3）＋ 14 ユティリティ専用ユニットの場合：0 m：pd_max_users の値＋ pd_max_reflect_process_count の値 n：ユニット内のサーバ数＋ユニット内ユティリティサーバ数＋ 1 ユニット内ユティリティサーバ数：27 ＋ α α：ユニット内にシングルサーバがある場合は 12，それ以外は 0
ロックサーバ	ユティリティ専用ユニットでない場合

プロセスの種類	共用メモリの計算式（単位：バイト）
	（320 ＋ 48 ＋ c ＋ d ＋ 48 ＋ 4096 ＋ g ＋ 48 ＋ i ＋ 48 ＋ 4096 ＋ 48 ＋ n ＋ t ＋ u ＋ 16） ×pd_lck_pool_partition の値 c：pd_lck_hash_entry を省略，又は 0 を指定している場合： （↓（8 ＋ 4×MAX（↑（（p ＋ 3）×（pd_max_access_tables の値＋ 4 ＋ 5×2） ＋↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×6）÷10↑ の値を超えない最大の素数，11261））÷16↓＋ 1）×16 pd_lck_hash_entry に 2 以上の素数でない値を指定している場合： （↓（8 ＋ 4×pd_lck_hash_entry の値を超えない最大の素数）÷16↓＋ 1） ×16 pd_lck_hash_entry に 1，又は素数を指定している場合： （↓（8 ＋ 4×pd_lck_hash_entry の値）÷16↓＋ 1）×16 d：（（p ＋ 3）×（pd_max_access_tables の値＋ 4 ＋ 5×2）＋ ↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×6）×96 g：pd_utl_exec_mode の値=1，かつ p ＞ 32 の場合： （（p ＋ 3）×3 ＋ p）×256 pd_utl_exec_mode の値=0，又は p≦32 の場合： （（p ＋ 3）×3 ＋ 32）×256 i：pd_utl_exec_mode の値=1，かつ p ＞ 32 の場合： （（↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×8 ＋（（p ＋ 3） ×（pd_max_access_tables の値＋ 4））×2） ＋ p×（pd_max_rdarea_no の値＋ 1）＋（p ＋ 3）×2×2×5） を偶数に切り上げた値×64 pd_utl_exec_mode の値=0，又は p≦32 の場合： （（↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×8 ＋（（p ＋ 3） ×（pd_max_access_tables の値＋ 4））×2） ＋ 32×（pd_max_rdarea_no の値＋ 1）＋（p ＋ 3）×2×2×5） を偶数に切り上げた値×64 n：pd_utl_exec_mode の値=1，かつ p ＞ 32 の場合： （（p ＋ 3）×3 ＋ p）×48 pd_utl_exec_mode の値=0，又は p≦32 の場合： （（p ＋ 3）×3 ＋ 32）×48 p：pd_max_users の値＋ pd_max_reflect_process_count の値 t：pd_utl_exec_mode の値=1，かつ p ＞ 32 の場合： 48 ＋（（p ＋ 3）×3 ＋ p）×↑pd_max_open_holdable_cursors の値÷16↑×4 pd_utl_exec_mode の値=0，又は p≦32 の場合： 48 ＋（（p ＋ 3）×3 ＋ 32）×↑pd_max_open_holdable_cursors の値÷16↑×4 u：pd_utl_exec_mode の値=1，かつ p ＞ 32 の場合： 48 ＋（（↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×8 ＋（（p ＋ 3）×（pd_max_access_tables の値＋ 4））×2） ＋ p×（pd_max_rdarea_no の値＋ 1）＋（p ＋ 3）×2×2×5） を偶数に切り上げた値×↑pd_max_open_holdable_cursors の値÷16↑×4 pd_utl_exec_mode の値=0，又は p≦32 の場合： 48 ＋（（↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×8 ＋

プロセスの種類	共用メモリの計算式（単位：バイト）
	((p + 3) ×（pd_max_access_tables の値+ 4)) ×2) + 32×（pd_max_rdarea_no の値+ 1）+（p + 3）×2×2×5) を偶数に切り上げた値×↑pd_max_open_holdable_cursors の値÷16↑×4 **ユティリティ専用ユニットの場合** 8416
トランザクションサーバ	288 + 32 + 192×m×2 + 1028 +（420 + 624 + 256 + 384×2 + 128）×（m×2 + 7）+ 256×2 m：pd_max_users の値+ pd_max_reflect_process_count の値
タイマサーバ	32×（pd_max_users の値+ pd_max_reflect_process_count の値+ 3）×（1 +ユニット内ユティリティサーバ数+ 1）+ 1440 ユニット内ユティリティサーバ数：26 + α α：AIX 版で，ユニット内にシングルサーバがある場合は 12，Linux 版で，ユニット内にシングルサーバがある場合は 3
統計ログサーバ	384 + 128×16 + 32 + 288×2 + 1024 + 128×3 + pd_stj_buff_size の値×1024×3 + 64 + 4096 + 8192
プロセスサーバ	192 + 512×a + 96 + 256 +（pd_max_server_process の値+ 50）×（256 + 144）+ 16 + 8×16 + 16 + 16 + 48 + 48×（b + 1） a：シングルサーバの場合：131 ユティリティ専用ユニットの場合：106 b：システム共通定義，又はユニット制御情報定義に pd_module_trace_max オペランドを指定している場合：pd_module_trace_max の値 それ以外：16383
シングルサーバ	992 +（44 + 4）×a×2 +（100 + 4）×（b + 30 + 2）+（100 + 4）×（c + 1）+ 40×b×14 + 256 + 256 + 36×d + 12×e + 8 + 13080 + 272 + 5856×b + f + 16 + 1024 + 272×h a：シングルサーバの定義数 b：ユニット内のシングルサーバ数 c：ユニット数 d：pdunit オペランドの-c オプションの指定数 e：pdcltgrp オペランド指定数 f：2052 + 128×（g + 3） g：シングルサーバの場合：92　ユティリティ専用ユニットの場合：74 h：pd_security_host_group オペランドに指定したホストに対応する IP アドレスの数 pd_security_host_group オペランドを指定していない場合は 0
ネームサーバ	169984
ノードマネジャ	↑（1152 + 432×全ユニット数+ 80×全サーバ数+ 7680 + 1008 + 56×ユニット内のサーバ数+ 240×A + 44×A + 28×A + 16×B + 32） ÷1024↑×1024 A：pd_utl_exec_mode = 0 の場合：1024 pd_utl_exec_mode = 1 で，ユニット内にシングルサーバがある場合：pd_max_users の値×10 + 400 pd_utl_exec_mode = 1 で，ユニット内にシングルサーバがない場合：pd_max_users の値×7

プロセスの種類	共用メモリの計算式（単位：バイト）
	なお，A の値が 1024 を超えない場合は，A を 1024 に置き換えて計算してください。 B：pdcltgrp オペランドを指定しない場合：0 pdcltgrp オペランドを指定する場合：pdcltgrp オペランドの指定数＋ 1
I/O サーバ	↑（28 ＋（↑（32 ＋ A）÷32↑×32））÷128↑×128 A：196940 ＋ pd_max_file_no 値×1024 ＋ pd_max_utl_ios_file_no 値×296 ユティリティ専用ユニットの場合は 34268 ＋ pd_max_utl_ios_file_no 値×296 バイトになります。
ログサーバ	32 ＋ 48 ＋ 128×19 ＋ 384 ＋ 128×7 ＋ 1024 ＋ 512 ＋↑（128 ＋ 256 ＋ 160 ＋ 8 ＋ 64）÷pd_log_rec_leng の値↑×pd_log_rec_leng の値 ＋ 64 ＋ 4096×2 ＋（736 ＋ 512）×B ＋ 128×pd_log_write_buff_count の値 ＋（pd_log_write_buff_count の値＋ A） ×↑ {pd_log_max_data_size の値＋（68 ＋ 44 ＋ 96 ＋ 160)} ÷4096↑×4096 ＋ C ＋↑ {(B ＋ 1）÷12} ↑×8320 pd_max_reflect_process_count オペランドを指定する場合に加算します。 128 ＋ 704 A：16 B：pdlogadfg -d sys オペランドの指定数 C：0
シンクポイントダンプサーバ	↑（368 ＋ 1456×2）÷1024↑×1024 ＋↑ {(96 ＋ 80 ＋ 208 ＋ 208）＋ 192×（pdlogadfg -d spd の指定数） ＋ 416×（pdlogadpf -d spd の指定数)} ÷1024↑×1024
ユニット共通	a ＋ b ＋ 64 ＋（m ＋ 3）×c ＋ 64 ＋ 48 ＋ d ＋ e ＋（pd_max_server_process の値×2 ＋ 100）×（48 ＋ 16）＋ 32 ＋（pd_max_server_process の値×2 ＋ 100 ＋ 384）×32 ＋ 32 ＋ f ＋ g ＋ h ＋ i ＋ p ＋ q ＋（pd_max_server_process の値＋ 127 ＋ r）×32 ＋ 32 ＋ t a：24896 b：2988 c：2716 d：32×32 e：64 ＋ 64×｛(m ＋ 3）×2 ＋ MAX（5, ↓ [m ＋ 3] ÷10↓）＋ 7} f：512×（13 ＋ 3）×2 g：{↑（96 ＋ pd_lck_until_disconnect_cnt の値×112）÷4096↑} ×4096×2 h：↑（pd_registered_port で指定したポート番号の数×16 ＋ 32） ÷1024↑×1024 pd_registered_port を指定していない場合は 0 i：k が 2 以上の場合は 32 ＋（8 ＋ 8×k）×n

プロセスの種類	共用メモリの計算式（単位：バイト）
	それ以外の場合は 0 k：pd_lck_pool_partition の値 m：pd_max_users の値＋ pd_max_reflect_process_count の値 n：（m ＋ 3）×2 ＋ MAX｛5, ↓（m ＋ 3）÷10↓｝＋ 7 p：pd_dbbuff_modify に Y を指定している場合：2064 ＋ pd_max_add_dbbuff_shm_no の値×4 上記以外：2064 q：144 r：pd_utl_exec_mode の値=1，かつ m ＞ 32 の場合：（m ＋ 3）×3 ＋ m pd_utl_exec_mode の値=0，又は m≦32 の場合：（m ＋ 3）×3 ＋ 32 t：352
トランザクションログサーバ	1024 ＋ 512×A ＋｛ 128×B ＋ 128 ＋［F ＋↑（128 ＋ 256 ＋ 8 ＋ 224）÷pd_log_rec_leng の値↑×pd_log_rec_leng の値 ＋↑（pd_log_max_data_size の値＋ 68 ＋ 44 ＋ 96 ＋ 160） ÷pd_log_rec_leng の値↑×pd_log_rec_leng の値］×D ＋ E ＋（48 ＋ 8）×B×2 ｝×ユニット内のサーバ数 ＋ 584×B ＋ 128×B ＋ 64×B×C ＋ 128 ＋｛ F ＋↑（128 ＋ 256 ＋ 8 ＋ 224）÷pd_log_rec_leng の値↑×pd_log_rec_leng の値 ＋↑（pd_log_max_data_size の値＋ 68 ＋ 44 ＋ 96 ＋ 160） ÷pd_log_rec_leng の値↑×pd_log_rec_leng の値 ｝ ＋ E ＋（48 ＋ 8）×（B×2 ＋ 2） A：2 B：7 ＋（pd_max_users の値＋ pd_max_reflect_process_count の値）×2 C：1 D：シングルサーバの場合，pd_log_rollback_buff_count の値が 0 のときは 8， それ以外のときは pd_log_rollback_buff_count の値 ユティリティ専用ユニットの場合は 0 E：0 F：60
ステータスサーバ	↑64÷32↑×32
監査証跡管理サーバ	↑A÷1024↑×1024 A：pd_aud_file_name オペランドの指定がない場合は 640 pd_aud_file_name オペランドの指定がある場合は 640 ＋（304×200）＋ B ＋ C B：pd_aud_async_buff_size オペランドの値が 0 の場合は 0 pd_aud_async_buff_size オペランドの値が 4096 以上の場合は次の計算値 Linux の場合 （160×pd_aud_async_buff_count オペランドの値）

プロセスの種類	共用メモリの計算式（単位：バイト）
	＋｛(↑pd_aud_async_buff_size オペランドの値÷4096↑×4096) ×pd_aud_async_buff_count オペランドの値｝＋4096 Linux 以外の場合 (160×pd_aud_async_buff_count オペランドの値) ＋｛(↑pd_aud_async_buff_size オペランドの値÷4096↑×4096) ×pd_aud_async_buff_count オペランドの値｝ C：pd_aud_auto_loading オペランドに N を指定している場合は 0 pd_aud_auto_loading オペランドに Y を指定している場合は 256×2＋240

(2) 64 ビットモードの HiRDB の場合

HiRDB/シングルサーバの開始から終了までの間にユニットコントローラが使用する共用メモリは，次に示す HiRDB のプロセスの項目すべてを加算した値です。

プロセスの種類	共用メモリの計算式（単位：バイト）
スケジューラ	**pd_utl_exec_mode の値が 0 の場合** ｛↑(432＋304×n)÷1024↑＋500＋x＋↑(134＋pd_trn_rcvmsg_store_buflen の値)÷1024↑｝×1024 **pd_utl_exec_mode の値が 1 の場合** ｛↑(432＋304×n)÷1024↑＋↑(m×2000＋136)÷1024↑＋y＋↑(134＋pd_trn_rcvmsg_store_buflen の値)÷1024↑｝×1024 x：シングルサーバの場合：116＋5×(m＋3)＋14 ユティリティ専用ユニットの場合：0 y：シングルサーバの場合：5×(m＋3)＋14 ユティリティ専用ユニットの場合：0 m：pd_max_users の値＋ pd_max_reflect_process_count の値 n：ユニット内のサーバ数＋ユニット内ユティリティサーバ数＋ 1 ユニット内ユティリティサーバ数：27＋ α α：ユニット内にシングルサーバがある場合は 12，それ以外は 0
ロックサーバ	**ユティリティ専用ユニットでない場合** (496＋80＋c＋d＋64＋8192＋g＋80＋i＋64＋8192＋64＋n＋t＋u＋16) ×pd_lck_pool_partition の値 c：pd_lck_hash_entry を省略，又は 0 を指定している場合： (↓(8＋8×MAX(↑((p＋3)×(pd_max_access_tables の値＋4＋5×2) ＋↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×4)÷10↑ の値を超えない最大の素数，11261))÷16↓＋1)×16 pd_lck_hash_entry に 2 以上の素数でない値を指定している場合： (↓(8＋8×pd_lck_hash_entry の値を超えない最大の素数)÷16↓＋1) ×16 pd_lck_hash_entry に 1，又は素数を指定している場合： (↓(8＋8×pd_lck_hash_entry の値)÷16↓＋1)×16

プロセスの種類	共用メモリの計算式（単位：バイト）
	d：((p ＋ 3) ×（pd_max_access_tables の値＋ 4 ＋ 5×2）＋ ↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×4）×128 g：pd_utl_exec_mode の値=1，かつ p ＞ 32 の場合： ((p ＋ 3) ×3 ＋ p) ×320 pd_utl_exec_mode の値=0，又は p≦32 の場合： ((p ＋ 3) ×3 ＋ 32) ×320 i：pd_utl_exec_mode の値=1，かつ p ＞ 32 の場合： ((↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×5 ＋（(p ＋ 3) ×（pd_max_access_tables の値＋ 4)）×2) ＋↓（↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓）÷3↓ ＋ p×（pd_max_rdarea_no の値＋ 1）＋（p ＋ 3）×2×2×5) を偶数に切り上げた値×112 pd_utl_exec_mode の値=0，又は p≦32 の場合： ((↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×5 ＋（(p ＋ 3) ×（pd_max_access_tables の値＋ 4)）×2) ＋↓（↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓）÷3↓ ＋ 32×（pd_max_rdarea_no 値＋ 1）＋（p ＋ 3）×2×2×5) を偶数に切り上げた値×112 n：pd_utl_exec_mode の値=1，かつ p ＞ 32 の場合： ((p ＋ 3) ×3 ＋ p) ×80 pd_utl_exec_mode の値=0，又は p≦32 の場合： ((p ＋ 3) ×3 ＋ 32) ×80 p：pd_max_users の値＋ pd_max_reflect_process_count の値 t：pd_utl_exec_mode の値=1，かつ p ＞ 32 の場合： 48 ＋（(p ＋ 3) ×3 ＋ p) ×↑pd_max_open_holdable_cursors の値÷16↑×4 pd_utl_exec_mode の値=0，又は p≦32 の場合： 48 ＋（(p ＋ 3) ×3 ＋ 32) ×↑pd_max_open_holdable_cursors の値÷16↑×4 u：pd_utl_exec_mode の値=1，かつ p ＞ 32 の場合： 48 ＋（((↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×5 ＋（(p ＋ 3) ×（pd_max_access_tables の値＋ 4)）×2) ＋↓↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓÷3↓ ＋ p×（pd_max_rdarea_no の値＋ 1）＋（p ＋ 3）×2×2×5) を偶数に切り上げた値×↑pd_max_open_holdable_cursors の値÷16↑×4 pd_utl_exec_mode の値=0，又は p≦32 の場合： 48 ＋（((↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×5 ＋（(p ＋ 3) ×（pd_max_access_tables の値＋ 4)）×2) ＋↓↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓÷3↓ ＋ 32×（pd_max_rdarea_no の値＋ 1）＋（p ＋ 3）×2×2×5) を偶数に切り上げた値×↑pd_max_open_holdable_cursors の値÷16↑×4 **ユティリティ専用ユニットの場合** 16704

プロセスの種類	共用メモリの計算式（単位：バイト）
トランザクションサーバ	304 ＋ 32 ＋ 192×m×2 ＋ 1048 ＋（416 ＋ 800 ＋ 256 ＋ 392×2 ＋ 128）×（m×2 ＋ 7）＋ 256×2 m：pd_max_users の値＋ pd_max_reflect_process_count の値
タイマサーバ	32×（pd_max_users の値＋ pd_max_reflect_process_count の値＋ 3）×（1 ＋ユニット内ユティリティサーバ数＋ 1） ＋ 1440 ＋（48－32）×2 ユニット内ユティリティサーバ数は 26 ＋ α α：AIX 版で，ユニット内にシングルサーバがある場合は 12，Linux 版で，ユニット内にシングルサーバがある場合は 3
統計ログサーバ	424 ＋ 128×16 ＋ 32 ＋ 288×2 ＋ 1168 ＋ 144×3 ＋ pd_stj_buff_size の値×1024×3 ＋ 64 ＋ 4096 ＋ 8192
プロセスサーバ	208 ＋ 528×a ＋ 96 ＋ 256 ＋（pd_max_server_process の値＋ 50）×（256 ＋ 160）＋ 16 ＋ 8×16 ＋ 16 ＋ 16 ＋ 64 ＋ 64×（b ＋ 1） a：シングルサーバの場合：131 ユティリティ専用ユニットの場合：106 b：システム共通定義，又はユニット制御情報定義に pd_module_trace_max オペランドを指定している場合：pd_module_trace_max の値 それ以外：16383
シングルサーバ	1024 ＋（48 ＋ 8）×a×2 ＋（112 ＋ 8）×（b ＋ 30 ＋ 2）＋（104 ＋ 8）×（c ＋ 1）＋ 40×b×14 ＋ 256 ＋ 256 ＋ 40×d ＋ 16×e ＋ 8 ＋ 13096 ＋ 320 ＋ 5856×b ＋ f ＋ 16 ＋ 1024 ＋ 272×h a：シングルサーバの定義数 b：ユニット内のシングルサーバ数 c：ユニット数 d：pdunit オペランドの-c オプションの指定数 e：pdcltgrp オペランド指定数 f：2056 ＋ 128×（g ＋ 3） g：シングルサーバの場合：92　ユティリティ専用ユニットの場合：74 h：pd_security_host_group オペランドに指定したホストに対応する IP アドレスの数 pd_security_host_group オペランドを指定していない場合は 0
ネームサーバ	169984
ノードマネジャ	↑（1312 ＋ 464×HiRDB システムの全ユニット数 ＋ 96×HiRDB システムの全サーバ数＋ 10240 ＋ 1200 ＋ 72×ユニット内 HiRDB サーバ数＋ 240×A ＋ 44×A ＋ 28×A ＋ 16×B ＋ 48） ÷1024↑×1024 A：pd_utl_exec_mode ＝ 0 の場合：1024 pd_utl_exec_mode ＝ 1 で，ユニット内にシングルサーバがある場合：（pd_max_users の値＋ pd_max_reflect_process_count の値）×10 ＋ 400 pd_utl_exec_mode ＝ 1 で，ユニット内にシングルサーバがない場合：（pd_max_users の値＋ pd_max_reflect_process_count の値）×7 なお，A の値が 1024 を超えない場合は，A を 1024 に置き換えて計算してください。

プロセスの種類	共用メモリの計算式（単位：バイト）
	B：pdcltgrp オペランドを指定しない場合：0 pdcltgrp オペランドを指定する場合：pdcltgrp オペランドの指定数＋ 1
I/O サーバ	↑（56 ＋（↑（56 ＋ A）÷32↑×32））÷128↑×128 A：196988 ＋ pd_max_file_no 値×1024 ＋ pd_max_utl_ios_file_no 値×296 ユティリティ専用ユニットの場合は 34316 ＋ pd_max_utl_ios_file_no 値×296 バイトになります。
ログサーバ	32 ＋ 48 ＋ 128×19 ＋ 432 ＋ 128×7 ＋ 1168 ＋ 512 ＋↑（128 ＋ 256 ＋ 160 ＋ 8 ＋ 64）÷pd_log_rec_leng の値↑×pd_log_rec_leng の値 ＋ 64 ＋ 4096×2 ＋（768 ＋ 512）×B ＋ 144×pd_log_write_buff_count の値 ＋（pd_log_write_buff_count の値＋ A） ×↑ {pd_log_max_data_size の値＋（68 ＋ 44 ＋ 96 ＋ 160)} ÷4096↑×4096 ＋ C ＋↑ {(B ＋ 1）÷12} ↑×8320 pd_max_reflect_process_count オペランドを指定する場合に加算します。 128 ＋ 704 A：16 B：pdlogadfg -d sys オペランドの指定数 C：0
シンクポイントダンプサーバ	↑（384 ＋ 1536×2）÷1024↑×1024 ＋↑ {(128 ＋ 80 ＋ 240 ＋ 240）＋ 192×（pdlogadfg -d spd の指定数） ＋ 416×（pdlogadpf -d spd の指定数)} ÷1024↑×1024
ユニット共通	a ＋ b ＋ 80 ＋（m ＋ 3）×c ＋ 64 ＋ 48 ＋ d ＋ e ＋（pd_max_server_process の値×2 ＋ 100）×（64 ＋ 16）＋ 32 ＋（pd_max_server_process の値×2 ＋ 100 ＋ 384）×32 ＋ 32 ＋ f ＋ g ＋ h ＋ i ＋ p ＋ q ＋（pd_max_server_process の値＋ 127 ＋ r）×48 ＋ 32 ＋ t a：34304 b：3480 c：3640 d：48×32 e：80 ＋ 96×｛(m ＋ 3）×2 ＋ MAX（5, ↓［m ＋ 3］÷10↓）＋ 7｝ f：512×（13 ＋ 3）×2 g：{↑（128 ＋ pd_lck_until_disconnect_cnt の値×112）÷4096↑} ×4096×2 h：↑（pd_registered_port で指定したポート番号の数×16 ＋ 32） ÷1024↑×1024 pd_registered_port を指定していない場合は 0 i：k が 2 以上の場合は 32 ＋（8 ＋ 8×k）×n それ以外の場合は 0

プロセスの種類	共用メモリの計算式（単位：バイト）
	k：pd_lck_pool_partition の値 m：pd_max_users の値＋ pd_max_reflect_process_count の値 n：（m ＋ 3）×2 ＋ MAX｛5，↓（m ＋ 3）÷10↓｝＋ 7 p：pd_dbbuff_modify に Y を指定している場合：2064 ＋（pd_max_add_dbbuff_shm_no の値＋ pd_max_resident_rdarea_shm_no の値）×4 上記以外：2064 ＋ pd_max_resident_rdarea_shm_no の値×4 q：144 r：pd_utl_exec_mode の値=1，かつ m ＞ 32 の場合：（m ＋ 3）×3 ＋ m pd_utl_exec_mode の値=0，又は m≦32 の場合：（m ＋ 3）×3 ＋ 32 t：352
トランザクションログサーバ	1168 ＋ 688×A ＋｛ 128×B ＋ 144 ＋［G ＋↑（128 ＋ 256 ＋ 8 ＋ 224）÷pd_log_rec_leng の値↑×pd_log_rec_leng の値 ＋↑（pd_log_max_data_size の値＋ 68 ＋ 44 ＋ 96 ＋ 160） ÷pd_log_rec_leng の値↑×pd_log_rec_leng の値］×D ＋ E ＋（48 ＋ 8）×B×2 ｝×ユニット内のサーバ数 ＋ 600×B ＋ 128×B ＋ 64×B×C ＋ 144 ＋｛ G ＋↑（128 ＋ 256 ＋ 8 ＋ 224）÷pd_log_rec_leng の値↑×pd_log_rec_leng の値 ＋↑（pd_log_max_data_size の値＋ 68 ＋ 44 ＋ 96 ＋ 160） ÷pd_log_rec_leng の値↑×pd_log_rec_leng の値 ｝ ＋ E ＋（48 ＋ 8）×（B×2 ＋ 2） A：2 B：7 ＋（pd_max_users の値＋ pd_max_reflect_process_count の値）×2 C：1 D：シングルサーバの場合，pd_log_rollback_buff_count の値が 0 のときは 8， それ以外のときは pd_log_rollback_buff_count の値 ユティリティ専用ユニットの場合は 0 E：Linux 版の場合：4096 Linux 版以外の場合：0 G：64
ステータスサーバ	↑64÷32↑×32
監査証跡管理サーバ	↑A÷1024↑×1024 A：pd_aud_file_name オペランドの指定がない場合は 704 pd_aud_file_name オペランドの指定がある場合は 704 ＋（320×200）＋ B ＋ C B：pd_aud_async_buff_size オペランドの値が 0 の場合は 0 pd_aud_async_buff_size オペランドの値が 4096 以上の場合は次の計算値 Linux 版の場合： （176×pd_aud_async_buff_count オペランドの値）

プロセスの種類	共用メモリの計算式（単位：バイト）
	＋｛(↑pd_aud_async_buff_size オペランドの値÷4096↑×4096) ×pd_aud_async_buff_count オペランドの値｝＋4096 Linux 版以外の場合： (176×pd_aud_async_buff_count オペランドの値) ＋｛(↑pd_aud_async_buff_size オペランドの値÷4096↑×4096) ×pd_aud_async_buff_count オペランドの値｝ C：pd_aud_auto_loading オペランドに N を指定している場合は 0 pd_aud_auto_loading オペランドに Y を指定している場合は 256×2＋256

14.1.4　シングルサーバが使用する共用メモリの計算式

シングルサーバが使用する共用メモリの計算式を次に示します。

32 ビットモードの場合（単位：キロバイト）

計算式 1＋↑｛(↑(40＋（計算式 2～計算式 7，計算式 9 を加算した値）＋（6.5×1024×1024))÷512↑×512)｝÷1024↑

64 ビットモードの場合（単位：キロバイト）

計算式 1＋↑｛(↑(72＋（計算式 2～計算式 9 を加算した値）＋（6.5×1024×1024))÷512↑×512)｝÷1024↑

計算式 1～10 についての注意事項

- pd_rdarea_open_attribute_use 又は pd_lv_mirror_use オペランドに Y を指定する場合に計算式 3 を加算します。
- pd_dbsync_point 又は pd_system_dbsync_point オペランドのどちらかの指定が commit の場合に計算式 4 を加算します。pd_system_dbsync_point オペランドの省略値は commit です。
  上記以外の場合，計算式 7 を加算します。
- pd_inner_replica_control オペランドを指定する場合に計算式 5 を加算します。
- pd_dfw_awt_process オペランドを指定する場合に計算式 6 を加算します。
- pd_sds_shmpool_size オペランドを省略すると，次の値が設定されます。
  32 ビットモードの場合：
  ↑｛(↑(40＋（計算式 2～計算式 7，計算式 9，10 の合計値))÷512↑×512)｝÷1024↑
  64 ビットモードの場合：
  ↑｛(↑(72＋（計算式 2～計算式 9，10 の合計値))÷512↑×512)｝÷1024↑
- pd_max_resident_rdarea_no オペランドを指定する場合に計算式 8 を加算します。
- pd_max_temporary_object_no の値が 1 以上の場合に計算式 9 を加算します。
- pd_max_tmp_table_rdarea_no の値が 1 以上の場合に計算式 10 を加算します。

計算式 1～10 を次に示します。

計算式の種類	共用メモリの計算式
計算式 1 （単位：キロバイト）	**●32 ビットモードの場合** b×1.3 ＋ c ＋ d ＋ f ＋ 1.6 ＋ q ＋ r ＋ 4 ＋ {[(a ＋ 12) ÷13] ×1.1 ＋ [(a ＋ 62) ÷63] ＋ 3.7} × (e ＋ 3) ＋ 3.5 ＋ { ↑ (↑ (b÷64) ↑) × (8÷16) ↑×4×4 ＋ 12× {(b÷3) ＋ 1－mod (b÷3, 2)} ＋ 8×a× {(e ＋ 3) ×2 ＋ 1 ＋ MAX (e÷10, 5)} ＋ 32 ＋ 20000 ＋↑ {(c÷8) ＋ 7} ÷64↑×8 ＋↑ {(f÷8) ＋ 7} ÷64↑×8 ＋ MAX {a× (e ＋ 3), c÷8} ×104 ＋ MAX {a× (e ＋ 3), f÷8} ×24 ＋↑ {(q÷4) ＋ 7} ÷64↑×8 ＋↑ {[(r－ (s×592 ＋ t×916 ＋ u×172)) ÷2]＋ 7} ÷64↑×8 ＋ MAX {13× (e ＋ 3), q÷4} ×88 ＋ MAX {21× (e ＋ 3), [r－ (s×592 ＋ t×916 ＋ u×172)]÷2} ×60 ＋ 44 ＋ 256 ＋ 1024 ＋ 512※1 } ÷1024 ＋ y ＋ 7.5 ＋↑ {248×v×w ＋ 47×v ＋ 72} ÷1024↑ ＋↑ {↑ (28 ＋ (↑ (32 ＋ ((↑g÷127↑＋ 1) ×2048 ＋ 128)) ÷32↑×32)) ÷128↑×128 } ÷1024↑ D ＋ Σ (Ei) i=1 ＋ 1024 pd_def_buf_control_area_assign オペランドに INITIAL を指定するか，又はこのオペランドを省略した場合に加算します。 ＋ {[(a ＋ 12) ÷13] ×1.1 ＋ [(a ＋ 62) ÷63] ＋ 3.7} × (e ＋ 7) **●64 ビットモードの場合** b×1.3 ＋ c ＋ d ＋ f ＋ 1.6 ＋ q ＋ r ＋ 5 ＋ {[(a ＋ 12) ÷13] ×1.2 ＋ [(a ＋ 62) ÷63] ×1.5 ＋ 4.1} × (e ＋ 3) ＋ 3.5 ＋ { ↑ (↑ (b÷64) ↑) × (8÷16) ↑×4×4 ＋ 12 × {(b÷3) ＋ 1－mod (b÷3, 2)} ＋ 8×a× {(e ＋ 3) ×2 ＋ 1 ＋ MAX (e÷10, 5)} ＋ 48 ＋ 20000 ＋↑ {(c÷8) ＋ 7} ÷64↑×8 ＋↑ {(f÷8) ＋ 7} ÷64↑×8 ＋ MAX {a× (e ＋ 3), c÷8} ×104 ＋ MAX {a× (e ＋ 3), f÷8} ×40 ＋↑ {(q÷4) ＋ 7} ÷64↑×8 ＋↑ {[(r－ (s×600 ＋ t×936 ＋ u×184)) ÷2]＋ 7} ÷64↑×8 ＋ MAX {13× (e ＋ 3), q÷4} ×104 ＋ MAX {21× (e ＋ 3), [r－ (s×600 ＋ t×936 ＋ u×184)]÷2} ×72 ＋ 72 ＋ 256 ＋ 1536 ＋ 512※1

計算式の種類	共用メモリの計算式
	｝÷1024＋y＋7.5 ＋↑｛248×v×w＋64×v＋72｝÷1024↑ ＋↑｛↑（56＋（↑（56＋（（↑g÷127↑＋1）×2048＋128））÷32↑×32）） ÷128↑×128 ｝÷1024 D ＋Σ（Ei） i=1 ＋1024 pd_def_buf_control_area_assign オペランドに INITIAL を指定するか，又はこのオペランドを省略した場合に加算します。 ＋｛[(a＋12）÷13]×1.2＋[(a＋62）÷63]×1.5＋4.1｝×（e＋7）
計算式 2 （単位：バイト）	●32 ビットモードの場合 500×1024 ＋5072×（e＋15）＋（↑436×g÷16↑×16）＋48※1×g＋496×h ＋112×（p＋240） ＋96×x＋32×j＋132×｛19＋（e＋3）×3｝ ＋48×n＋48×｛(e＋3）×2＋1＋MAX（5,（e＋3）÷10)｝ ＋68×B＋152×（A＋120）＋80＋32×g＋64※2＋96※3＋368※4 ＋((↓（↑（g÷8）↑＋3）÷4↓）×4）×j ●64 ビットモードの場合 500×1024 ＋9416×（e＋15）＋（↑536×g÷16↑×16） ＋（↑56※1×g÷16↑×16）＋512×h ＋（↑136×（p＋240）÷16↑×16） ＋144×x＋48×j＋240×｛19＋（e＋3）×3｝ ＋64×n＋96×｛(e＋3）×2＋1＋MAX（5,（e＋3）÷10)｝ ＋68×B＋184×（A＋120）＋96＋48×g＋64※2＋128※3＋448※4 ＋((↓（↑（g÷8）↑＋7）÷8↓）×8）×j
計算式 3 （単位：バイト）	●32 ビットモードの場合 ｛[(↑↑g÷8↑÷4↑）×4]＋8｝×｛(e＋3）×2＋12｝ ●64 ビットモードの場合 ｛[(↑↑g÷8↑÷8↑）×8]＋8｝×｛(e＋3）×2＋12｝
計算式 4 （単位：バイト）	●32 ビットモードの場合 (32＋16×x）×（e×2＋7＋1）＋16 ●64 ビットモードの場合 (48＋32×x）×（e×2＋7＋1）＋16
計算式 5 （単位：バイト）	56×z＋16
計算式 6	●32 ビットモードの場合

計算式の種類	共用メモリの計算式
（単位：バイト）	72 ＋ 52×C ＋ 68×x pd_dbbuff_trace_level オペランドに 1 を指定する場合に加算します。 ＋ 320×x ●64 ビットモードの場合 96 ＋ 56×C ＋ 72×x pd_dbbuff_trace_level オペランドに 1 を指定する場合に加算します。 ＋ 640×x
計算式 7 （単位：バイト）	●32 ビットモードの場合 （32 ＋ 16×x）×10 ＋ 16 ●64 ビットモードの場合 （48 ＋ 32×x）×10 ＋ 16
計算式 8 （単位：バイト）	16 ＋ 112 ＋（48 ＋ 48×G）＋（48 ＋ 32×H）
計算式 9 （単位：バイト）	16 ＋ 80×I
計算式 10 （単位：バイト）	↑（112 ＋（28 ＋ J×52））÷8↑×8

a ：pd_max_access_tables オペランドの値

b ：pd_sql_object_cache_size オペランドの値

c ：pd_table_def_cache_size オペランドの値

d ：pd_auth_cache_size オペランドの値

e ：pd_max_users オペランドの値

f ：pd_view_def_cache_size オペランドの値

g ：pd_max_rdarea_no オペランドの値

h ：pd_max_file_no オペランドの値

j ：インデクス用のグローバルバッファプール数
pd_dbbuff_modify オペランドに Y を指定している場合，pdbuffer コマンドの指定数に，pd_max_add_dbbuff_no オペランドの値を加算して計算します。

n ：pd_lck_until_disconnect_cnt オペランドの値

p ：pd_assurance_index_no オペランドの値

q ：pd_type_def_cache_size オペランドの値

r：pd_routine_def_cache_size オペランドの値

s：インストールしたプラグインの数

t：DML で使用するプラグイン関数の総数 ※5

u：DML で使用するプラグイン関数のパラメタ総数 ※5

v：pd_max_list_users オペランドの値

w：pd_max_list_count オペランドの値

x：総グローバルバッファ数（pdbuffer オペランドの指定数）
pd_dbbuff_modify オペランドに Y を指定している場合，pdbuffer コマンドの指定数に，pd_max_add_dbbuff_no オペランドの値を加算して計算します。

y：pd_registry_cache_size オペランドの値

z：pd_inner_replica_control オペランドの値

A：pd_assurance_table_no オペランドの値

B：サーバ内の最大トランザクション数（2×e ＋ 7）

C：pd_dfw_awt_process オペランドの値

D：指定した pdplgprm オペランドの総数

Ei：i 番目の pdplgprm オペランドで指定した共用メモリのサイズ

G：pd_max_resident_rdarea_no オペランドの値

H：pd_max_resident_rdarea_shm_no オペランドの値

I：pd_max_temporary_object_no オペランドの値

J：pd_max_tmp_table_rdarea_no オペランドの値

T：空き領域の再利用機能を使用する表数

注※1
pd_max_list_users 及び pd_max_list_count オペランドの両方とも 0 でない場合に加算します。

注※2
pd_max_ard_process オペランドが 1 以上の場合に加算します。

注※3
pd_max_reflect_process_count オペランドに 1 以上を指定した場合に加算します。

注※4

再編成時期予測機能を使用する場合に加算します。

注※5

DML で使用するプラグイン関数の総数及び DML で使用するプラグイン関数のパラメタの総数は，次に示す SQL で求められます。

```
SELECT COUNT(*),SUM(N_PARAM) FROM MASTER.SQL_PLUGIN_ROUTINES
WHERE PLUGIN_NAME = 'プラグイン名称'
AND (TIMING_DESCRIPTOR = 'ADT_FUNCTION'
    OR TIMING_DESCRIPTOR IS NULL
    OR TIMING_DESCRIPTOR = 'BEFORE_INSERT'
    OR TIMING_DESCRIPTOR = 'AFTER_INSERT'
    OR TIMING_DESCRIPTOR = 'BEFORE_UPDATE'
    OR TIMING_DESCRIPTOR = 'AFTER_UPDATE'
    OR TIMING_DESCRIPTOR = 'BEFORE_DELETE'
    OR TIMING_DESCRIPTOR = 'AFTER_DELETE'
    OR TIMING_DESCRIPTOR = 'BEFORE_PURGE_TABLE'
    OR TIMING_DESCRIPTOR = 'AFTER_PURGE_TABLE'
    OR TIMING_DESCRIPTOR = 'INDEX_SEARCH'
    OR TIMING_DESCRIPTOR = 'INDEX_COUNT'
    OR TIMING_DESCRIPTOR = 'INDEX_INSERT'
    OR TIMING_DESCRIPTOR = 'INDEX_BEFORE_UPDATE'
    OR TIMING_DESCRIPTOR = 'INDEX_AFTER_UPDATE'
    OR TIMING_DESCRIPTOR = 'INDEX_DELETE'
    OR TIMING_DESCRIPTOR = 'PURGE_INDEX'
    OR TIMING_DESCRIPTOR = 'INDEX_MAINTENANCE_DEFERRED'
    OR TIMING_DESCRIPTOR = 'BEFORE_INSERT_DC'
    OR TIMING_DESCRIPTOR = 'BEFORE_UPDATE_DC'
    OR TIMING_DESCRIPTOR = 'BEFORE_DATA_CHECK'
    OR TIMING_DESCRIPTOR = 'AFTER_DATA_CHECK')
```

14.1.5　グローバルバッファが使用する共用メモリの計算式

グローバルバッファが使用する共用メモリサイズは，pdbuffer 文ごとに計算式 1 で求めます。pd_dbbuff_modify オペランドに Y を指定している場合は，計算式 2 を加算します。計算式 1 及び 2 で求めた値を合計した値が，グローバルバッファが使用する共用メモリの所要量です。

pd_dbbuff_attribute オペランドに fixed 又は hugepage を指定している場合，実メモリ上にページ固定されるため，仮想メモリを構成する実メモリがそのサイズ分減少します。また，残り実メモリとスワップ領域で構成される仮想メモリからもそのサイズ分確保されます。

なお，pdbuffer オペランドを省略した場合，HiRDB が共用メモリサイズを自動計算するので見積もりは必要ありません。

計算式の種類	共用メモリの計算式（単位：キロバイト）
計算式 1	●32 ビットモードの場合 n Σ {

計算式の種類	共用メモリの計算式（単位：キロバイト）
	i=1 ↑｛752＋64＋（296＋64※1）×（Pi＋4） ＋（124＋80※2＋96×A×Mi）×Ui｝÷T↑×T ＋Si×｛Pi＋4＋（Ui×Mi×A）｝＋Vi ｝÷1024 **●64 ビットモードの場合** n Σ｛管理領域部＋データ格納部｝÷1024 i=1 管理領域部： ↑｛944＋64＋（480＋112※1）×（Pi＋4） ＋（176＋96※2＋136×A×Mi）×Ui｝÷T↑×T データ格納部： Si×｛Pi＋4＋（Ui×Mi×A）｝＋Vi
計算式 2	**●32 ビットモードの場合** {↑（↑（s×1024÷2）÷8↑＋112）÷2048↑×2048× ↑a÷（s×1024）↑｝÷1024 **●64 ビットモードの場合** {↑（↑（s×1024÷2）÷8↑＋144）÷2048↑×2048× ↑a÷（s×1024）↑｝÷1024

n：グローバルバッファプール数

i：計算対象のグローバルバッファプール定義

P：グローバルバッファ面数

A：

pd_max_ard_process オペランドが 1 以上の場合は 2

pd_max_ard_process オペランドが 0 の場合は 1

M：一括入力最大ページ数

U：同時実行最大プリフェッチ数

T：グローバルバッファのバウンダリ

- Linux の場合

  pd_dbbuff_dev_sector_size オペランドに 512 を指定した場合は 2048，4096 を指定した場合は 4096
- Linux 以外の場合

  4096

S：グローバルバッファに割り当てた RD エリアの最大ページ長

V：

- Linux の場合
  pd_dbbuff_dev_sector_size オペランドに 4096 を指定し，かつ S が 4096 の倍数でない場合は 2048。
  pd_dbbuff_dev_sector_size オペランドに 4096 を指定していない，又は S が 4096 の倍数の場合は 0。
- Linux 以外の場合
  0

s：SHMMAX 指定値

a：計算式 1 の総計

注※1　LOB 用グローバルバッファの場合に加算します。

注※2　pd_max_ard_process オペランドが 1 以上の場合に加算します。

14.1.6　SQL 実行時に必要なメモリ所要量の計算式

(1)　グループ分け高速化機能実行時に必要なメモリ所要量の求め方

クライアント環境定義で PDSQLOPTLVL オペランドを指定するか，HiRDB システム定義で pd_optimize_level オペランドを指定（又は省略）した場合，適用条件を満たす SQL を実行すると，グループ分け高速化機能が働きます。このとき，HiRDB はクライアント環境定義の PDAGGR オペランドの値に基づいてプロセス固有領域を確保します。確保するメモリサイズを次に示します。

計算式

```
e＋↑d÷4↑×4＋↑（17＋4×a＋4×b＋c＋d）÷4↑×4×（N＋1）
                                                （単位：バイト）
```

a：グループ化する列の数

b：集合関数の演算数
COUNT，SUM，MAX，MIN は一つにつき 1 で換算します。
AVG（COUNT），AVG（SUM）は一つにつき 2 で換算します。

c：グループ化する列の列長（表「グループ化するときの列の長さ及び集合関数の演算領域の長さ」を参照して求めてください）

d：集合関数の演算領域長（表「グループ化するときの列の長さ及び集合関数の演算領域の長さ」を参照して求めてください）

e：32 ビットモードの場合：MAX（4×N×24 , 16408）
　64 ビットモードの場合：MAX（8×N×40 , 32808）

N：クライアント環境定義の PDAGGR オペランドの値

表 14-4　グループ化するときの列の長さ及び集合関数の演算領域の長さ

列のデータ型	グループ化する列の列長	集合関数の演算領域長※1
INTEGER	4	6
SMALLINT	2	4※2
DECIMAL(p,s)	↑（p＋1）÷2↑	↑（p＋7）÷2↑※3
FLOAT	8	10
SMALLFLT	4	6
INTERVAL YEAR TO DAY	5	8
INTERVAL HOUR TO SECOND	4	6
CHAR(n)	n	n＋3
VARCHAR(n)	n＋2	n＋5
NCHAR(n)	2×n	2×n＋2
NVARCHAR(n)	2×n＋2	2×n＋4
MCHAR(n)	n	n＋3
MVARCHAR(n)	n＋2	n＋5
DATE	4	6
TIME	3	6
BLOB(n)	－	－
BINARY(n)	n＋2	n＋5

（凡例）－：該当しません。

注※1

　集合関数が COUNT の場合，集合関数演算領域長はデータ型にかかわらず 6 になります。

注※2

　集合関数が AVG，SUM の場合は集合関数演算領域長は 6 になります。

注※3

　集合関数が AVG，SUM の場合，集合関数演算領域長は次の値になります。

　集合関数の値の型が DECIMAL 型で精度が 29 けたのとき：18

　集合関数の値の型が DECIMAL 型で精度が 38 けたのとき：23

集合関数の値のデータ型の規則については，マニュアル「HiRDB SQL リファレンス」の「集合関数」を参照してください。

(2) 列ごとのデータ抑制指定時に必要なメモリ所要量の求め方

列ごとのデータ抑制を指定（CREATE TABLE の列定義に SUPPRESS を指定）した表に対してアクセスするときに使用するメモリサイズは，次に示す計算式で求められます。

計算式

a＋128　　（単位：バイト）

ａ：表中で列ごとのデータ抑制が指定されている列の定義長の合計値

(3) ハッシュジョイン及び副問合せのハッシュ実行時に必要なメモリ所要量の求め方

クライアント環境定義で PDADDITIONALOPTLVL オペランドを指定するか，HiRDB システム定義で pd_additional_optimize_level オペランドを指定すると，SQL 拡張最適化オプションが働きます。この SQL 拡張最適化オプションで，「ハッシュジョイン，副問合せのハッシュ実行の適用（APPLY_HASH_JOIN）」を指定した場合，表の結合又は副問合せの SQL を実行すると，次に示すメモリサイズのプロセス固有領域を確保します。

計算式

●32ビットモードの場合

$$\sum_{i=1}^{a}(13\times1024+6\times1024\times b+c)$$

●64ビットモードの場合

$$\sum_{i=1}^{a}(13\times1024+7\times1024\times b+c)$$

（単位：バイト）

ａ：SELECT 文のハッシュジョイン最大数

SELECT 文のハッシュジョイン最大数については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

ｂ：ハッシュ表行数によって適用されるハッシュジョイン処理を求めて，次に示す表から代入する値を決定してください。

ハッシュ表行数の目安	適用されるハッシュジョイン処理		ｂの値
1500 以内	一括ハッシュジョイン		0.5
1500×（バケット分割数÷3）以内	バケット分割ハッシュジョイン	1 レベルバケット分割	1
1500×（バケット分割数÷3）^{2} 以内		2 レベルバケット分割	2

ハッシュ表行数の目安	適用されるハッシュジョイン処理		bの値
1500×（バケット分割数÷3）^{2} を超える場合		3 レベルバケット分割	3

ハッシュ表行数：ジョインの場合はジョインの内表件数です。副問合せの場合は探索条件中の外への参照列を含む述語を除いた副問合せ探索件数です。

バケット分割数：MIN｛↓（ハッシュ表サイズ÷2）÷ハッシュ表ページ長↓，64｝

ハッシュ表サイズ：HiRDB システム定義の pd_hash_table_size オペランド，又はクライアント環境変数の PDHASHTBLSIZE オペランドで指定した値です。

ハッシュ表ページ長：次に示す表から c（ハッシュ表最大行長）に対応するハッシュ表ページ長を選択してください。

ハッシュ表最大行長	ハッシュ表ページ長（単位：バイト）
0～1012	4096
1013～2036	8192
2037～4084	16384
4085～16360	32768
16361～32720	↑（ハッシュ表最大行長＋ 48）÷2048↑×2048

c：ハッシュ表最大行長

ハッシュ表最大行長については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(4) スナップショット方式指定時に必要なメモリ所要量の求め方

pd_pageaccess_mode オペランドを省略した場合又は SNAPSHOT を指定した場合，スナップショット方式を適用する SQL 文を実行すると，データベース検索時のページアクセス方式にスナップショット方式を使用します。このとき，表又はインデクスの格納 RD エリアのページサイズに基づいて，動的に次に示すメモリサイズのプロセス固有領域を確保します。

計算式

```
a×2　　（単位：バイト）
```

a：検索対象の表又はインデクスが格納されている RD エリア中の最大ページ長

ただし，LOB 用 RD エリアは除きます。

(5) 先頭から n 行の検索結果を取得する機能実行時に必要なメモリ所要量の求め方

先頭から n 行の検索結果を取得する機能を使用すると，検索結果の先頭（又はユーザが指定した先頭からのオフセット行数分読み飛ばした位置）から n 行取得できます。

LIMIT 句に指定した行数が 1 以上で，（オフセット行数＋ LIMIT 句に指定した行数）の値が 32,767 以下の場合，（オフセット行数＋ LIMIT 句に指定した行数）以内に入り得る行をメモリに保持します。確保するプロセス固有領域のメモリサイズは，次に示す計算式で求められます。なお，（オフセット行数＋ LIMIT 句に指定した行数）の値が 32,768 以上になる場合は作業表を作成するため，「作業表用ファイルの容量の見積もり」を参照してください。

計算式

```
{100＋（a＋2）×（オフセット行数＋LIMIT句に指定した行数）} ×b
                                                （単位：バイト）
```

a：行長

行長は 32,720 バイト以下でなければなりません。行長は次の計算式で求められます。

```
 m
Σ（Ai）＋2×m＋4＋c
i=1
                                                （単位：バイト）
```

m：選択式，GROUP BY 句，又は ORDER BY 句に指定した列数

FOR UPDATE 句を指定した場合は 1 を加算してください。ただし，選択式に ROW を指定している場合は表の全列数になります。

Ai：先頭 n 行保持領域に格納する行の i 番目の列データ長

列のデータ長については，表「データ長一覧」を参照し，d に定義長を代入して求めてください。

ただし，BLOB データ，定義長が 256 バイト以上の文字データ（各国・混在文字データも含む），BINARY データのうち，下記に属さない列の場合は 12 バイトになります。

- DISTINCT 句指定の選択式に指定する列
- UNION [ALL]によって集合演算対象となっている問合せ指定中の選択式
- ORDER BY 句に指定した列

また，FOR UPDATE 句を指定した場合に，m に加算した 1 に対応する Ai は 12 バイトとします。

c：8

ただし，次の場合は 0 になります。

- 検索対象の表に EX モードで排他が掛かっている場合
- WITHOUT LOCK を指定した場合
- グループ分け高速化機能を指定した場合
- 複数の表を結合する場合

b：先頭 n 行保持領域数

先頭 n 行の保持領域数は次の計算式で求められます。

```
1＋UNION [ALL]句指定数
```

(6) 探索条件にインデクス型プラグイン専用関数を指定したSQL文実行時に必要なメモリ所要量の求め方

探索条件にインデクス型プラグイン専用関数を指定したSQL文の実行時に確保するプロセス固有領域のメモリサイズは，次に示す計算式で求められます。

計算式

```
a×500＋（20＋6）×800＋16　　（単位：バイト）
```

ａ：行長。行長は次の計算式で求められます。

```
 m
Σ（Ai）＋4×（m＋2）＋12＋4＋8
i=1
                                                （単位：バイト）
```

m：選択式，結合条件，GROUP BY句，又はORDER BY句に指定した列数

FOR UPDATE句を指定した場合は1を加算してください。ただし，選択式にROWを指定している場合は表の全列数になります。

Ai：取り出す行のi番目の列データ長

列のデータ長については，表「データ長一覧」を参照し，dに定義長を代入して求めてください。

ただし，BLOBデータ，又は定義長が256バイト以上の文字データ（各国・混在文字データも含む）で，下記に属さない列の場合は12バイトになります。

- 結合条件中に指定する列（結合列）
- DISTINCT句指定の選択式に指定する列
- 限定述語の副問合せ中の選択式に指定する列
- IN述語の副問合せ中の選択式に指定する列
- UNION [ALL]，又はEXCEPT [ALL]によって集合演算対象となっている問合せ指定中の選択式
- ORDER BY句に指定した列

また，FOR UPDATE句を指定した場合に，mに加算した1に対応するAiは12バイトとします。

(7) 拡張SQLエラー情報出力機能使用時に必要なメモリ所要量の求め方

拡張SQLエラー情報出力機能を使用した場合，次のときにプロセス固有領域を確保します。

(a) OPEN文実行時

計算式

```
●32ビットモードの場合
（16＋16×m）＋a
●64ビットモードの場合
```

(24＋24×m) ＋a

（単位：バイト）

a：？パラメタ又は埋込み変数のデータ長の合計

$$a=\sum_{i=1}^{m}(ai)$$

m：SQL 文中の？パラメタ又は埋込み変数の数

ai：i 番目の？パラメタ又は埋込み変数のデータ長

埋込み変数又は？パラメタのデータ長を次の表に示します。

表 14-5　埋込み変数又は？パラメタのデータ長

データ型	列長（標識変数なし）	列長（標識変数あり，埋込み変数又は？パラメタ）
INTEGER	4	6
SMALLINT	2	4
DECIMAL(p，s)	↑（p＋1）÷2↑	↑（p＋5）÷2↑
FLOAT	8	10
SMALLFLT	4	6
INTERVAL YEAR TO DAY	5	7
INTERVAL HOUR TO SECOND	4	6
CHAR(n)	n	n＋2
VARCHAR(n)	n＋2	n＋4
NCHAR(n)	2×n	2×n＋2
NVARCHAR(n)	2×n＋2	2×n＋4
MCHAR(n)	n	n＋2
MVARCHAR(n)	n＋2	n＋4
DATE	4	6
TIME	3	5
BLOB(n)	n＋4	n＋8
TIMESTAMP(p)	7＋（p÷2）	9＋（p÷2）
BINARY(n)	n＋4	n＋8

(b) 定義系 SQL の PREPARE 文実行時

計算式

SQL文長＋20
（単位：バイト）

(8) 部分構造インデクスの定義，又は部分構造インデクスを定義した表の更新時に必要なメモリ所要量の求め方

(a) 部分構造インデクスの定義時

定義系 SQL の CREATE INDEX で部分構造インデクスを定義する場合に確保するプロセス固有領域は，次に示す計算式で求められます。

計算式

（インデクスキー長※×100＋64）　（単位：バイト）

注※

表に定義する部分構造インデクスの最大定義長です。

(b) 部分構造インデクスを定義した表の更新時

操作系 SQL の INSERT，UPDATE，又は DELETE で部分構造インデクスを定義した表を更新する場合に確保するプロセス固有領域は，次に示す計算式で求められます。

計算式

（インデクスキー長※1×100＋64＋128）＋Σ（インデクスキー長＋128）※2　（単位：バイト）

注※1

表に定義している部分構造インデクスの最大定義長です。

注※2

USING UNIQUE TAG 指定の部分構造インデクス数です。

(9) 圧縮列に対して操作系 SQL を実行する場合に必要なメモリ所要量の求め方

SQL 実行時，データの格納，及び抽出対象に圧縮列が含まれる場合，次に示すメモリサイズのプロセス固有領域を確保します。

計算式

MIN（圧縮分割サイズ，圧縮列の定義長）※×C＋L　（単位：バイト）

C：次に示すどれかの条件に該当する場合は 2。該当しない場合は 1。

- SUBSTR 関数を使用している
- POSITION 関数を使用している
- 後方削除更新をしている

L：SQL の実行対象となる圧縮表が格納されている RD エリアのページ長

複数の RD エリアが対象になる場合は，最大のページ長で計算する。

注※

SQL の実行対象となる全圧縮列の中で最大になる値で計算します。

14.1.7　SQL 前処理時に必要なメモリ所要量の計算式

(1)　ストアドプロシジャを使用しない場合に必要なメモリ所要量の求め方

ストアドプロシジャを使用しない場合，SQL 前処理時に確保するメモリサイズは，次に示す計算式で求められます。

計算式

```
↑ {
    2586＋Si×60＋Pi×20＋Ti×1424＋Ci×Ti×72＋Wi×776＋Ti×Wi×72
    ＋Ki×276＋Ki×Ti×72＋Li×3＋Li×Ti＋Di×Ti×134＋Ari×108
    ＋Gi×44＋Sli×40＋Upi×110＋Fi×90＋ Ti×Cwi×48
    ＋MAX（Pi, Wpi）×60
  } ×1.2 ÷1024↑
                                                    （単位：キロバイト）
```

Si：SQL 文中の検索項目数

Pi：SQL 文中の埋込み変数，？パラメタ又は SQL パラメタの数

Ti：SQL 文中の表名の数

Ci：SQL 文中の列名の数

Wi：SQL 文中の論理演算子（AND 及び OR）に出てくる述語の数

Ki：SQL 文中の定数の数

Li：SQL 文中の定数の長さの合計（単位：バイト）

Di：SQL 文中に定義された格納条件の総数

Ari：SQL 文中の四則演算及び連結演算の数

Gi：SQL 文中の GROUP BY 句に指定した列の数

Ori：SQL 文中の ORDER BY 句に指定した列指定又はソート項目指定番号の数

Fi：SQL 文中の集合関数及びスカラ関数の総数

Sli：SQL 文中の問合せ指定の数

Upi：SQL 文中の更新列数

Cwi：SQL 文中の CASE 式中の WHEN の数

Wpi：SQL 文中の WITH 句に対応する変数の数

注

SELECT_APSL が適用されている場合は，前記の計算式で求めた値を 3 倍してください。

SELECT_APSL が適用され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（pdvwopt）を使用すると分かります。アクセスパス表示ユティリティ（pdvwopt）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。

(2)　ストアドプロシジャを使用する場合に必要なメモリ所要量の求め方

ストアドプロシジャを使用する場合，SQL 前処理時に確保するメモリサイズ（単位：キロバイト）は，「ストアドプロシジャを使用しない場合に必要なメモリ所要量の求め方」の計算式で求めた値に，ストアドプロシジャごとのプロシジャ制御用オブジェクト長を加算します。プロシジャ制御用オブジェクト長の計算式については，システム共通定義の pd_sql_object_cache_size オペランドの 1 ストアドプロシジャのプロシジャ制御用オブジェクト長を参照してください。1 ストアドプロシジャのプロシジャ制御用オブジェクト長については，マニュアル「HiRDB システム定義」の「1 ルーチンのルーチン制御用オブジェクト長の計算式」を参照してください。

14.1.8　BLOB 型データの検索又は更新時に必要なメモリ所要量の計算式（HiRDB/シングルサーバの場合）

BLOB 型データの検索又は更新時に必要なメモリ所要量は次に示す計算式で求められます。

計算式

ａ＋ｂ＋17　　（単位：キロバイト）

ａ：1SQL 文中に指定する BLOB 型入力変数又は出力変数で，実行する SQL 文の中で次に示す計算式の結果が最大となる値です。

$$\uparrow \left\{ \sum_{i=1}^{c} (\text{BLOB型入力変数iの実長}^{※1} + 118) \right\} + d$$

```
Σ（BLOB型出力変数jの定義長※2＋86）
j=1
} ÷1024↑
```

注※1

埋込み変数で UAP から HiRDB サーバに受け渡された BLOB 型データの実際の長さです。

注※2

HiRDB から UAP に返す BLOB 型データを受け取る UAP の埋込み変数の宣言長です。
INSERT－SELECT 文の場合は，SELECT 側で射影する BLOB 列を出力変数とみなします。

b：同時オープン中のカーソルで結合検索を行う SQL 文の組み合わせで，次に示す計算式の結果が最大となる値です。

```
↑ {
 e
Σ {
i=1
 d
Σ（BLOB型出力変数jの定義長＋18）}
j=1
} ÷1024↑
```

c：入力変数の数

d：出力変数の数

e：同時オープン中のカーソル数

14.1.9 ブロック転送又は配列 FETCH で必要なメモリ所要量の計算式

ブロック転送又は配列 FETCH で必要なメモリ所要量は，次の計算式で求められます。

条件		PDBLKBUFFSIZE オペランドの指定値	
		省略又は 0	1 以上
FETCH 文の INTO 句に配列型の埋込み変数を指定する		計算式 1	
FETCH 文の INTO 句に配列型の埋込み変数を指定しない	PDBLKF オペランドを省略又は 1	－	計算式 2
	PDBLKF オペランドが 2 以上	計算式 1	

（凡例）－：該当しません。

計算式 1

```
↑ {864＋16×ａ＋（6×ａ＋2×d＋ｂ）×ｃ} ÷1024↑
                                    （単位：キロバイト）
```

a：SELECT 句で指定する検索項目数

b：FETCH 文で受け取る検索結果中の 1 行のデータ長（各列の最大長の合計。単位はバイト）

c：PDBLKF オペランドの指定値又は配列数

d：SELECT 句で指定する検索項目で，BINARY 型を指定した選択式の数

計算式 2

$MAX\ (X_1,\ X_2)$
（単位：キロバイト）

X_1：↑（864 + 22×a + 2×c + b）÷1024↑

X_2：PDBLKBUFFSIZE オペランドの値

a：SELECT 句で指定する検索項目数

b：FETCH 文で受け取る検索結果中の 1 行のデータ長（実際に取得する各列の長さの合計。単位はバイト）

c：SELECT 句で指定する検索項目で，BINARY 型を指定した選択式の数

14.1.10 インメモリデータ処理で必要なメモリ所要量

インメモリデータ処理で必要なメモリ所要量は次に示す計算式で求められます。

計算式

計算式 1 ＋D×2　　（単位：キロバイト）

計算式 1

$$\sum_{i=1}^{n} \uparrow \{736+32\times A+48+448\times B+2048+C\times B\} \div 1024 \uparrow$$
（単位：キロバイト）

n：インメモリ RD エリアの数

A：インメモリ RD エリアを構成する HiRDB ファイル数

B：インメモリ RD エリアの総ページ数

C：インメモリ RD エリアのページサイズ

D：計算式 2 の値

計算式２（インメモリデータバッファが使用する共用メモリセグメント数）

```
↑計算式１の値÷（SHMMAXオペランドの値×1024）↑
```

計算式２は 1RD エリア当たりの計算式です。インメモリ RD エリアの数だけ計算してください。

計算式２で求めた値は，pd_max_resident_rdarea_shm_no オペランド又は OS のオペレーティングシステムパラメタの見積もりに使います。

14.2　HiRDB/パラレルサーバのメモリ所要量の見積もり

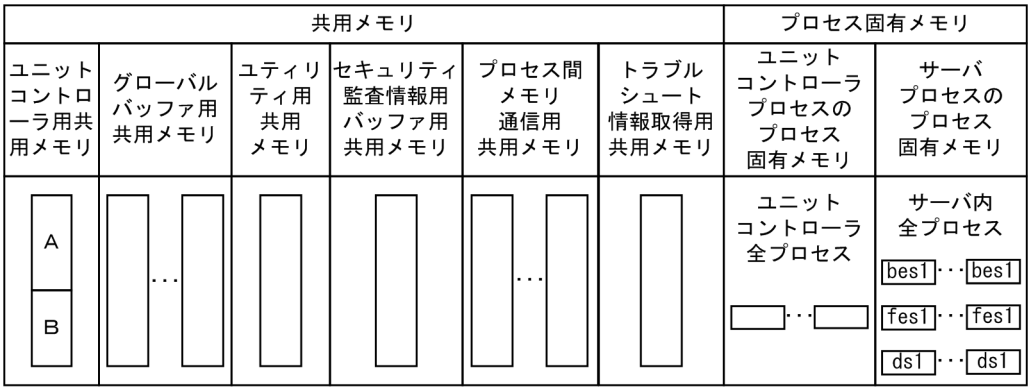
ここでは，HiRDB/パラレルサーバを構成する各ユニットのメモリ所要量の見積もり方法について説明します。ここで説明する項目を次に示します。

- メモリ配置
- メモリ所要量の計算式
- ユニットコントローラが使用する共用メモリの計算式
- 各サーバが使用する共用メモリの計算式
- グローバルバッファが使用する共用メモリの計算式
- SQL 実行時に必要なメモリ所要量の計算式
- SQL 前処理時に必要なメモリ所要量の計算式
- BLOB 型データの検索又は更新時に必要なメモリ所要量の計算式（フロントエンドサーバの場合）
- ブロック転送又は配列 FETCH で必要なメモリ所要量の計算式（フロントエンドサーバの場合）
- BLOB 型データの検索又は更新時に必要なメモリ所要量の計算式（バックエンドサーバ又はディクショナリサーバの場合）

14.2.1　メモリ配置

HiRDB/パラレルサーバの各ユニットのメモリ配置を次の図に示します。

図 14–2　HiRDB/パラレルサーバの各ユニットのメモリ配置



（凡例）A：ユニットコントローラの各プロセス使用分
B：各サーバのプロセス使用分
bes：バックエンドサーバ　fes：フロントエンドサーバ　ds：ディクショナリサーバ

HiRDB/パラレルサーバの各ユニットの共用メモリの詳細を次の表に示します。

表 14-6　HiRDB/パラレルサーバの各ユニットの共用メモリの詳細

項目	共用メモリの種類					
	ユニットコントローラ用共用メモリ	グローバルバッファ用共用メモリ	ユティリティ用共用メモリ	セキュリティ監査情報用バッファ用共用メモリ	プロセス間メモリ通信用共用メモリ	トラブルシュート情報取得用共用メモリ
使用目的	システム制御	グローバルバッファ	ユニットコントローラとユティリティとの通信	セキュリティ監査情報用バッファ	クライアント―サーバプロセス間メモリ通信	トラブルシュート情報取得
使用プロセス	全 HiRDB プロセス	バックエンドサーバ，ディクショナリサーバ	ユティリティプロセス	フロントエンドサーバ	フロントエンドサーバ，クライアントプロセス	全 HiRDB プロセス
セグメント数	1 個	• グローバルバッファの動的変更機能を使用しない場合 1～512 個※1 • グローバルバッファの動的変更機能を使用している場合 32 ビットモード： 1～1,012 個※1 64 ビットモード ：1～1,512 個※1	1 個	1 個	環境変数 PDIPC=MEMORY で接続中のクライアント数（0～2000）×2 個	1 個
1 セグメントの上限	表「HiRDB/パラレルサーバの各ユニットが使用するメモリ所要量」を参照してください。オペレーティングシステムパラメタの shmmax の値は，計算値以上にしてください。	SHMMAX オペランドの値でセグメントを分割します。オペレーティングシステムパラメタの shmmax の値は，SHMMAX オペランドの値以上にしてください。	表「HiRDB/パラレルサーバの各ユニットが使用するメモリ所要量」を参照してください。オペレーティングシステムパラメタの shmmax の値は，計算値以上にしてください。	表「HiRDB/パラレルサーバの各ユニットが使用するメモリ所要量」を参照してください。オペレーティングシステムパラメタの shmmax の値は，計算値以上にしてください。	表「HiRDB/パラレルサーバの各ユニットが使用するメモリ所要量」を参照してください。オペレーティングシステムパラメタの shmmax の値は，計算値以上にしてください。	10 メガバイト

項目	共用メモリの種類					
	ユニットコントローラ用共用メモリ	グローバルバッファ用共用メモリ	ユティリティ用共用メモリ	セキュリティ監査情報用バッファ用共用メモリ	プロセス間メモリ通信用共用メモリ	トラブルシュート情報取得用共用メモリ
確保条件	なし	グローバルバッファ定義が存在すること	pd_utl_exec_mode=1 指定	pd_aud_file_name オペランドによる監査証跡ファイル用の HiRDB ファイルシステム領域名の指定	クライアント環境変数 PDIPC=MEMORY で接続中クライアントあり	なし
作成契機	ユニット起動時(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使用時の待機ユニットの起動を含む)	• サーバ起動時（高速系切り替え機能使用時の待機ユニットの起動を含む） • pdbufmod -k {add | upd}実行時	ユティリティ実行時	フロントエンドサーバ起動時	クライアントとサーバの接続時	ユニット起動時（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使用時の待機ユニットの起動を含む）
削除契機	次回ユニット起動時（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使用時の待機ユニットの起動を含む）	• pdbufmod -k del 実行時 • 正常終了又は計画停止の場合：サーバ終了時 • 強制停止，異常終了，又は高速系切り替え機能使用時の待機ユニットの停止の場合：次回ユニット起動時	ユティリティ終了後 10 分後	フロントエンドサーバ終了時	クライアントとサーバの接続切り離し時	ユニット停止時
pdls -d mem による表示	表示される	表示される	表示される	表示される	表示されない	表示されない
pdls -d mem の SHM-	MANAGER	サーバ名	UTILITY	AUDDEF	表示なし	表示なし

項目	共用メモリの種類					
	ユニットコントローラ用共用メモリ	グローバルバッファ用共用メモリ	ユティリティ用共用メモリ	セキュリティ監査情報用バッファ用共用メモリ	プロセス間メモリ通信用共用メモリ	トラブルシュート情報取得用共用メモリ
OWNER						
関連オペランド	• pd_shmpool_attribute • pd_dic_shmpool_size • pd_bes_shmpool_size	• pd_dbbuff_attribute • pd_dbbuff_modify • pdbuffer • SHMMAX	• pd_utl_exec_mode	• セキュリティ監査機能に関するオペランド※2	• PDIPC • PDSENDMEMSIZE • PDRECVMEMSIZE	なし
備考	−	−	pd_utl_exec_mode=1 の場合だけ作成されます（pd_utl_exec_mode=0 の場合，該当する領域はユニットコントローラ用共用メモリ内に確保されます）。	−	−	−

（凡例）−：該当しません。

注※1

グローバルバッファ割り当てバックエンドサーバ又はディクショナリサーバ当たりの数です。

注※2

詳細はマニュアル「HiRDB システム定義」を参照してください。

14.2.2 メモリ所要量の計算式

HiRDB/パラレルサーバの各ユニットが使用するメモリ所要量は，次の表に示すすべての項目を加算した値です。

なお，OS のオペレーティングシステムパラメタの shmmax の指定値については，「OS のオペレーティングシステムパラメタの見積もり」を参照してください。

共用メモリサイズが増加すると，ページフォルトの発生回数が増加し，トランザクション性能に影響を与えるおそれがあります。各オペランドの指定値の目安を参照し，システムに合わせて適切な値を指定することを検討してください。

表 14–7　HiRDB/パラレルサーバの各ユニットが使用するメモリ所要量

項目				メモリ所要量（単位：キロバイト）
プロセス固有領域	ユニットコントローラ全プロセスが使用するプロセス固有領域			**●32 ビットモードの場合** J＋K×ユニット内 FES 数＋ L×（ユニット内 BES 数＋ユニット内 DS 数）＋↑｛(64 ＋ 48×（v ＋ 1)) ×（pd_max_server_process の値－w－3）＋（64 ＋ 48×（ac ＋ 1)）×3 ＋ z｝÷1024↑ **●64 ビットモードの場合** J＋K×ユニット内 FES 数＋ L×（ユニット内 BES 数＋ユニット内 DS 数）＋↑｛(64 ＋ 64×（v ＋ 1)) ×（pd_max_server_process の値－w－3）＋（64 ＋ 64×（ac ＋ 1)）×3 ＋ z｝÷1024↑＋ ab **●プラグインを使用する場合，加算します。** ＋ 1400 **●pd_max_ard_process オペランドが 1 以上の場合，加算します。** ＋ s **●リアルタイム SAN レプリケーションを使用する場合，加算します。** ＋↑（A ＋ B ＋ C）÷1024↑ **●pd_process_terminator オペランドに fixed を指定した場合，加算します。** ＋ M×（pd_process_terminator_max の値－1） **●インメモリデータ処理を行う場合，加算します。** ＋↑｛T×（pd_max_bes_process の値×2 ＋ 7）× ユニット内 BES 数｝÷1024↑ **●通信トレース格納最大数を変更する場合，加算します。** ＋↑V÷1024↑
	各サーバプロセスが使用するプロセス固有領域※1 ※2	フロントエンドサーバ		(N ＋ h ＋ m ＋ p ＋ q ＋ ad）×b ＋ y **●通信トレース格納最大数を変更する場合，加算します。** ＋↑W÷1024↑
		ディクショナリサーバ	pd_work_buff_mode =each 指定時	{P ＋ i ＋ m ＋（a ＋ 9）×2 ＋ r ＋ t｝×b ＋ y ＋ S **●通信トレース格納最大数を変更する場合，加算します。** ＋↑W÷1024↑
			pd_work_buff_mode =pool 指定又は省略時	**●32 ビットモードの場合** (P ＋ i ＋ m ＋ a ＋↑a÷128×0.1↑＋ 11 ＋ n ＋ r ＋ t）×b ＋ y ＋ S **●64 ビットモードの場合** (P ＋ i ＋ m ＋ a ＋↑a÷128×0.1↑＋ 15 ＋ n ＋ r ＋ t）×b ＋ y ＋ S

項目				メモリ所要量（単位：キロバイト）
				●通信トレース格納最大数を変更する場合，加算します。 ＋↑W÷1024↑
			pd_work_buff_mode=pool2 指定	●32 ビットモードの場合 (P＋i＋m＋↑a÷128×0.1↑＋11＋n＋r＋t)×b＋y＋S＋(a×ae) ●64 ビットモードの場合 (P＋i＋m＋↑a÷128×0.1↑＋15＋n＋r＋t)×b＋y＋S＋(a×ae) ●通信トレース格納最大数を変更する場合，加算します。 ＋↑W÷1024↑
		バックエンドサーバ	pd_work_buff_mode=each 指定時	{Q＋g＋（a＋9）×c＋i＋m＋r＋t} ×（b＋3）＋y＋S ●インメモリデータ処理を行う場合，加算します。 ＋↑｛T×（b＋3)｝÷1024↑ ●通信トレース格納最大数を変更する場合，加算します。 ＋↑W÷1024↑
			pd_work_buff_mode=pool 指定又は省略時	●32 ビットモードの場合 (Q＋g＋a＋↑a÷128×0.1↑＋11＋i＋m＋n＋r＋t）×（b＋3）＋y＋S ●64 ビットモードの場合 (Q＋g＋a＋↑a÷128×0.1↑＋15＋i＋m＋n＋r＋t）×（b＋3）＋y＋S ●インメモリデータ処理を行う場合，加算します。 ＋↑｛T×（b＋3)｝÷1024↑ ●通信トレース格納最大数を変更する場合，加算します。 ＋↑W÷1024↑
			pd_work_buff_mode=pool2 指定	●32 ビットモードの場合 (Q＋g＋↑a÷128×0.1↑＋11＋i＋m＋n＋r＋t）×（b×3）＋y＋S＋(a×ae) ●64 ビットモードの場合 (Q＋g＋↑a÷128×0.1↑＋15＋i＋m＋n＋r＋t）×（b×3）＋y＋S＋(a×ae) ●インメモリデータ処理を行う場合，加算します。 ＋↑｛T×（b＋3)｝÷1024↑ ●通信トレース格納最大数を変更する場合，加算します。 ＋↑W÷1024↑

項目		メモリ所要量（単位：キロバイト）
共用メモリ	ユニットコントローラ用共用メモリ中，ユニットコントローラが使用する領域	↑d÷1024↑
	ユニットコントローラ用共用メモリ中，各サーバが使用する領域※1	e
	グローバルバッファ用共用メモリ	f
	インメモリデータ処理用共用メモリ	U
	ユティリティ用共用メモリ	u
	セキュリティ監査情報用バッファ用共用メモリ	**●システムによる自動計算の場合** ↑0.3 + MAX｛(R + 100)，(R×1.2)｝×0.25↑ **●ユーザ指定値（pd_audit_def_buffer_size オペランドを指定）にする場合** pd_audit_def_buffer_size の指定値
	プロセス間メモリ通信用共用メモリ※3	j×k
OS の領域※4	PTE の領域※5	**●共用メモリのページ固定をしていない場合** （↑プロセス固有領域の合計サイズ÷510↑＋pd_max_server_process の値×16 + 13）＋｛(↑共用メモリの合計サイズ÷510↑＋17）×pd_max_server_process の値｝ **●共用メモリのページ固定をしている場合** （↑プロセス固有領域の合計サイズ÷510↑＋pd_max_server_process の値×16 + 13）＋｛(↑共用メモリの合計サイズ÷261120↑＋13）×pd_max_server_process の値｝

注※1

ユニット内に複数のサーバ（システムマネジャを除きます）がある場合は，サーバごとに計算した値をすべて加算してください。

ただし，影響分散スタンバイレス型系切り替えの対象となるユニットの場合は，次の手順で求めた値を，ユニット内の全バックエンドサーバのメモリ所要量として加算してください。

1. ユニット内のホスト BES 及びゲスト BES として受け入れ可能な BES ごとに，計算式中の最大 BES プロセス数（変数 b，及び，変数 y と変数 W で使用する最大起動プロセス数）を，ユニット内の最大プロセス数（pd_ha_max_server_process）に置き換えて計算した値を求めます。
2. 1 で求めた中で最大の値を加算してください。

注※2

プラグインを使用する場合は，1 サーバプロセス当たり 300 を加算してください。

注※3

クライアント環境定義で PDIPC=MEMORY を指定した場合に加算します。プロセス間メモリ通信機能及び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。

なお，HiRDB サーバ又は HiRDB クライアントのどちらかが 32 ビットモードの場合，プロセス間メモリ通信機能で使用する共用メモリは 32 ビット空間内に確保されます。

注※ 4

OS が使用する領域の詳細は公開されていないため，あくまで HiRDB が確保するメモリ所要量を基にした目安です。実際の値とは大きく異なる場合があるので，正確な値については実際の環境で測定してください。

注※ 5

PTE については，「システム設計」を参照してください。

a：pd_work_buff_size オペランドの値

b：次のうちのどれかの値

- ディクショナリサーバの場合は，（pd_max_dic_process オペランドの値＋ pd_max_reflect_process_count オペランドの値）となります。
- バックエンドサーバの場合は，次の値となります。

  影響分散スタンバイレス型系切り替えの対象となるユニットの場合：

  メモリ所要量を計算するユニットの pd_ha_max_server_process オペランドの値

  上記以外のユニットの場合：

  pd_max_bes_process オペランドの値＋ pd_max_reflect_process_count オペランドの値
- pd_max_dic_process 又は pd_max_bes_process オペランドを省略する場合は，（pd_max_users オペランドの値＋ pd_max_reflect_process_count オペランドの値）となります。

c：最大作業表数

SQL 文ごとの作業表数を表「SQL 文ごとの作業表数の求め方」から求めます。表「SQL 文ごとの作業表数の求め方」から求めた作業表数のうちで最大のものを最大作業表数とします。

d：「ユニットコントローラが使用する共用メモリの計算式」で求めた値

e：「各サーバが使用する共用メモリの計算式」で求めた値

f：「グローバルバッファが使用する共用メモリの計算式」で求めた値

g：SQL 実行時に必要なメモリ所要量

計算式については，「SQL 実行時に必要なメモリ所要量の計算式」を参照してください。

h：SQL 前処理時に必要なメモリ所要量

計算式については，「SQL 前処理時に必要なメモリ所要量の計算式」を参照してください。

i：LOB バッファ一括入出力ワークメモリ

該当するサーバの LOB 用 RD エリアに LOB 用グローバルバッファを指定している場合だけ（システム共通定義の pdbuffer オペランドに-b を指定している場合），62 キロバイトを加算してください。

j：プロセス間メモリ通信機能を使用するクライアントの最大同時実行数

分からない場合は，プロセス間メモリ通信機能を使用する全クライアント数，又は pd_max_users オペランドの値を代入してください。

k：プロセス間メモリ通信機能を使用する全クライアントのデータ送受信用メモリサイズ（クライアント環境定義の PDSENDMEMSIZE の値＋ PDRECVMEMSIZE の値）の平均値

m：Java 仮想マシンが使用するメモリ所要量

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，Java 仮想マシンが使用するメモリ所要量を加算します。Java 仮想マシンが使用するメモリ所要量は，Java 仮想マシンのオプション（Hewlett-Packard JRE 1.2.2.04 の場合は-Xms，-Xmx，-Xmn オプション）や Java 仮想マシンのバージョンによって異なります。Java 仮想マシンが使用するメモリ所要量については，Java 仮想マシンのマニュアルを参照してください。

n：作業表用増分メモリサイズ

pd_work_buff_expand_limit オペランドを指定する場合に作業表用増分メモリサイズを加算します。作業表用増分メモリサイズは次に示す計算式から求めます。

作業表用増分メモリサイズ（キロバイト）＝作業表用増分バッファサイズ＋↑（作業表用の増分バッファサイズ÷128）×0.1↑

- 作業表用増分バッファサイズ（キロバイト）＝ MAX（0，ハッシュジョイン，副問合せのハッシュ実行による作業表用増分バッファサイズ）＋ MAX（0，作業表数の増加による作業表用増分バッファサイズ）
- ハッシュジョイン，副問合せのハッシュ実行による作業表用増分バッファサイズ＝ MIN｛(ハッシュジョイン，副問合せのハッシュ実行をするときの作業表用バッファサイズ－pd_work_buff_size オペランドの値)，(pd_work_buff_expand_limit オペランドの値－pd_work_buff_size オペランドの値)｝×ハッシュジョイン，副問合せのハッシュ実行をする同時実行ユーザ数

  ハッシュジョイン，副問合せのハッシュ実行をするときの作業表用バッファサイズの求め方については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。
- 作業表数の増加による作業表用増分バッファサイズ＝ MIN｛(使用作業表数×128－pd_work_buff_size オペランドの値)，(pd_work_buff_expand_limit オペランドの値－pd_work_buff_size オペランドの値)｝×（使用作業表数が pd_work_buff_size オペランドの値÷128 以上になるユーザ数）

  使用作業表数＝ MAX（1SQL 文が使用する作業表用ファイルの数，ASSIGN LIST 文が使用する作業表用ファイルの数）

  1SQL 文が使用する作業表用ファイルの数，及び ASSIGN LIST 文が使用する作業表用ファイルの数の求め方については，「最大ファイル数の見積もり（pdfmkfs -l コマンド）」を参照してください。

p：BLOB 型データ用に必要なメモリ所要量

計算式については，「BLOB 型データの検索又は更新時に必要なメモリ所要量の計算式（フロントエンドサーバの場合）」を参照してください。

q：サーバ側でブロック転送又は配列 FETCH で必要なメモリ所要量

計算式については，「ブロック転送又は配列 FETCH で必要なメモリ所要量の計算式（フロントエンドサーバの場合）」を参照してください。

r：BLOB 型データ用に必要なメモリ所要量

計算式については，「BLOB 型データの検索又は更新時に必要なメモリ所要量の計算式（バックエンドサーバ又はディクショナリサーバの場合）」を参照してください。

s：非同期 READ 用メモリサイズ

pd_max_ard_process オペランドが 1 以上の場合に加算します。次に示す計算式から求めます（単位：キロバイト）。

```
 (90＋
90
Σ RDエリア用HiRDBファイルシステム領域管理用メモリ）
i=1
×pd_max_ard_processの値
```

RD エリア用 HiRDB ファイルシステム領域管理用メモリは計算値の大きい順に 90 領域を計算に使用します。サーバで使用する領域数が 90 領域に満たない場合，その領域まで計算します。

HiRDB ファイルシステム領域管理用メモリはそれぞれの領域の初期設定時のパラメタを使用して，次の計算式から求めます（単位：キロバイト）。

なお，領域の初期設定時のパラメタは pdfstatfs コマンドに-A オプションを指定して実行することで確認できます。

```
｛（ファイル数※1＋増分数※2）÷64｝×1.5※3
```

注※1　pdfmkfs -l 指定値，又は pdfstatfs 実行結果の[available file count]に表示される値です。

注※2　pdfmkfs -e 指定値，又は pdfstatfs 実行結果の[available expand count]に表示される値です。

注※3　領域サイズ（pdfmkfs -n 指定値）が 2048 以上の場合に乗算します。

t：HiRDB ファイルシステム用メモリサイズ

次に示す計算式から求めます（単位：キロバイト）。

```
604＋作業表用HiRDBファイルシステム領域管理用メモリ＋
システムログ用HiRDBファイルシステム領域管理用メモリ＋
90
ΣRDエリア用HiRDBファイルシステム領域管理用メモリ
i=1
```

作業表用及びシステムログ用 HiRDB ファイルシステム領域管理用メモリは，サーバで使用する HiRDB ファイルシステム領域で計算値が最大になるものを使用します。RD エリアの場合は計算値の大きい順に 90 領域を計算に使用します。サーバで使用する領域数が 90 領域に満たない場合，その領域まで計算します。

HiRDB ファイルシステム領域管理用メモリはそれぞれの領域の初期設定時のパラメタを使用して，次の計算式から求めます（単位：キロバイト）。

なお，領域の初期設定時のパラメタは pdfstatfs コマンドに-A オプションを指定して実行することで確認できます。

{（ファイル数※1＋増分数※2）÷64} ×1.5※3

注※1　pdfmkfs -l 指定値，又は pdfstatfs 実行結果の[available file count]に表示される値です。

注※2　pdfmkfs -e 指定値，又は pdfstatfs 実行結果の[available expand count]に表示される値です。

注※3　領域サイズ（pdfmkfs -n 指定値）が 2048 以上の場合に乗算します。

u：pd_utl_exec_mode の値が 0 の場合：0

pd_utl_exec_mode の値が 1 の場合：↑ {(b×2000 ＋ 136）÷1024} ↑×1024

v：ユニット制御情報定義として有効な pd_module_trace_max の値

w：ユニット内の全サーバプロセスに対して，（最大起動プロセス数＋ 3）を合計した値

最大起動プロセス数については，マニュアル「HiRDB システム定義」を参照してください。

y：ユニット内の各サーバプロセスに対して次の計算式で求めた値の総和

32 ビットモードの場合：

↑ {(64 ＋ 48×（pd_module_trace_max の値＋ 1)）×（最大起動プロセス数＋ 3)} ÷1024↑

64 ビットモードの場合：

↑ {(64 ＋ 64×（pd_module_trace_max の値＋ 1)）×（最大起動プロセス数＋ 3)} ÷1024↑

最大起動プロセス数については，マニュアル「HiRDB システム定義」を参照してください。

なお，影響分散スタンバイレス型系切り替えの対象となるユニットの場合，「最大起動プロセス数」は，メモリ所要量を計算するユニットの「pd_ha_max_server_process オペランドの値」に置き換えてください。

z：HiRDB の再開始用メモリサイズ

このメモリサイズを確保できないと，HiRDB の再開始に失敗します。次の計算式から求めます（単位：バイト）。

（D＋E＋F）×ディクショナリサーバ数＋（D＋E＋F＋G）×バックエンドサーバ数＋ΣH

HiRDB の再開始用メモリサイズを求める計算に使用する変数を次に示します。

変数	値
D	**●32 ビットモードの場合** 246762 ＋ 4×pd_max_rdarea_no の値 ＋ {48×（pd_max_rdarea_no の値＋表数）＋ 304} ×（(pd_max_users の値※＋ pd_max_reflect_process_count の値）×2 ＋ 7） **●64 ビットモードの場合** 305274 ＋ 8×pd_max_rdarea_no の値 ＋ {64×（pd_max_rdarea_no の値＋表数）＋ 512} ×（(pd_max_users の値※＋ pd_max_reflect_process_count の値）×2 ＋ 7） **表数**：62 ＋ MAX｛pd_max_access_tables の値，500｝
E	b1×X ＋ b2×Y **b1：サーバ用ステータスファイルのレコード長＜ 4096 の場合** MAX（(↓（3400÷（(↓（(レコード長－40）－308）÷20↓）

変数	値
	＋（↓（レコード長−40）÷20↓）×（MAX（↓4096÷レコード長↓，2）−1）） ＋0.7）↓），1）×MAX(↓4096÷レコード長↓，2）×（レコード長−40） 4096≦サーバ用ステータスファイルのレコード長＜12288 の場合 MAX（↓（3400÷（↓（（（レコード長−40）−308）÷20）↓）＋0.7）↓，1） ×（レコード長−40） 12288≦サーバ用ステータスファイルのレコード長の場合 MAX（↓（3400÷（↓（（（レコード長−40）−836）÷20）↓）＋0.7）↓，1） ×（レコード長−40） X：サーバ内の RD エリア数≦3400 の場合：1 3401≦サーバ内の RD エリア数≦6800 の場合：2 6801≦サーバ内の RD エリア数の場合：3 b2：サーバ用ステータスファイルのレコード長＜4096 の場合 （↓（5662310÷（（↓（（レコード長−40）−308）÷20↓） ＋（↓（レコード長−40）÷20↓）×（MAX（↓4096÷レコード長↓，2）−1）） ＋0.7）↓）×MAX（↓4096÷レコード長↓，2）×（レコード長−40） 4096≦サーバ用ステータスファイルのレコード長＜12288 の場合 ↓（5662310÷（↓（（（レコード長−40）−308）÷20）↓）＋0.7）↓ ×（レコード長−40） 12288≦サーバ用ステータスファイルのレコード長の場合 ↓（5662310÷（↓（（（レコード長−40）−836）÷20）↓）＋0.7）↓ ×（レコード長−40） Y：サーバ内の RD エリア数≦10200 の場合：0 10201≦サーバ内の RD エリア数≦5672510 の場合：1 5672511≦サーバ内の RD エリア数≦11334820 の場合：2 11334821≦サーバ内の RD エリア数の場合：3
F	pd_dbsync_point オペランドに commit を指定している場合，加算します。 ＋112×（（pd_max_users の値※＋ pd_max_reflect_process_count の値）×2＋7）
G	pd_inner_replica_control オペランドに 1 以上を指定している場合，加算します。 ●32 ビットモードの場合 （48×pd_max_rdarea_no の値＋80）×（（pd_max_users の値※＋ pd_max_reflect_process_count の値）×2＋7） ●64 ビットモードの場合 （64×pd_max_rdarea_no の値＋160）×（（pd_max_users の値※＋ pd_max_reflect_process_count の値）×2＋7）
H	サーバ内のキャラクタ型スペシャルファイル上に作成された RD エリアを格納した HiRDB ファイルシステム領域数が 1001 以上のバックエンドサーバに対して加算します。 ●32 ビットモードの場合 12012×（↑（キャラクタ型スペシャルファイル上に作成された RD エリアを格納した HiRDB ファイルシステム領域数−1000）÷1000↑） ●64 ビットモードの場合 16016×（↑（キャラクタ型スペシャルファイル上に作成された RD エリアを格納した HiRDB ファイルシステム領域数−1000）÷1000↑）

注※

ディクショナリサーバの場合は pd_max_dic_process の値となります。バックエンドサーバの場合は pd_max_bes_process の値となります。ただし，pd_max_dic_process 又は pd_max_bes_process を省略する場合は，pd_max_users の値となります。

A：425×（2×b ＋ 7）×ユニット内 FES 数

B：425×（2×b ＋ 7）×ユニット内 DS 数

C：425×（2×b ＋ 7）×ユニット内 BES 数

J，K，L，M，N，P，Q：固定値

この値は OS によって異なります。OS ごとの値を次に示します（単位：キロバイト）。

OS	J の値	K の値	L の値	M の値	N の値	P の値	Q の値
AIX	136,300	25,000	31,200	6,000	11,900	11,700	13,400
Linux	113,800	13,400	15,800	1,900	5,400	5,900	7,200

R：余裕を持って見積もる場合は監査対象イベントの数（CREATE AUDIT の実行回数），詳細に見積もる場合はセキュリティ監査情報用バッファのエントリ数

S：シンクポイント出力同期制御情報取得機能使用時に必要なメモリ（単位：バイト）

pd_dbbuff_trace_level オペランドに 1 を指定し，かつ pd_dfw_awt_process オペランドの指定を省略している場合に，次の値を加算します。

32 ビットモードの場合：

320×シングルサーバに定義したグローバルバッファの数

64 ビットモードの場合：

640×シングルサーバに定義したグローバルバッファの数

T：pd_max_resident_rdarea_no オペランドに 1 以上を指定している場合に，次の値を加算します。

1648 ＋ 16×pd_max_resident_rdarea_no の値＋ 16×pd_max_resident_rdarea_shm_no の値

U：インメモリデータ処理で必要なメモリ所要量

計算式については，「インメモリデータ処理で必要なメモリ所要量」を参照してください。

V：通信トレース処理で必要なメモリ所要量

32 ビットモードの場合：

（16×（Z－1024）×2）×（pd_max_server_process の値－w）

64 ビットモードの場合：

（32×（Z－1024）×2）×（pd_max_server_process の値－w）

W：通信トレース処理で必要なメモリ所要量

ユニット内の各サーバプロセスについて算出した次の値です。

32 ビットモードの場合：

（16×（aa－1024）×2）×（最大起動プロセス数＋ 3）

64 ビットモードの場合：

（32×（aa－1024）×2）×（最大起動プロセス数＋ 3）

最大起動プロセス数については，マニュアル「HiRDB システム定義」を参照してください。

なお，影響分散スタンバイレス型系切り替えの対象となるユニットの場合，「最大起動プロセス数」は，メモリ所要量を計算するユニットの「pd_ha_max_server_process オペランドの値」に置き換えてください。

Z：ユニット制御情報定義として有効な pd_pth_trace_max の値です。

オペランドの指定値を 2 のべき乗に切り上げた値になります。

ａａ：各サーバ定義として有効な pd_pth_trace_max の値です。

オペランドの指定値を 2 のべき乗に切り上げた値になります。

ａｂ：シグナルハンドラ用メモリサイズ

0

ａｃ：システム共通定義，又はユニット制御情報定義に pd_module_trace_max オペランドを指定している場合：pd_module_trace_max の値

それ以外の場合：16383

ａｄ：フロントエンドサーバで使用する SQL 実行用通信メモリ所要量

4×システム内 BES 数

ａｅ： 作業表を使用する操作（SQL 又はユティリティ）を行うトランザクションの同時実行数

表 14-8　SQL 文ごとの作業表数の求め方

SQL 文	作業表数の求め方
SELECT 文 INSERT（-SELECT）文	1.～8. の指定がない場合：0 1.～8. の指定がある場合：該当する作業表数をすべて加算した値 1. 複数の表を結合して検索する場合 増加作業表数＝（結合表数－1）×2 ＋ 1 2. ORDER BY 句を指定する場合 増加作業表数＝ 2 3. GROUP BY 句を指定する場合 増加作業表数＝ GROUP BY 句指定数 4. DISTINCT 句を指定する場合 増加作業表数＝ DISTINCT 句指定数 5. UNION 句，UNION ALL 句又は EXCEPT[ALL]句を指定する場合 増加作業表数＝（UNION 又は UNION ALL 句指定数）×2 ＋ 1 6. 探索条件中にインデクスを定義した列がある場合 増加作業表数＝探索条件中のインデクスを定義した列数 7. FOR UPDATE 句又は FOR READ ONLY 句を指定する場合

SQL 文	作業表数の求め方
	増加作業表数＝ 1 8. 副問合せ（限定述語）を指定する場合 増加作業表数＝副問合せ指定数
UPDATE 文 DELETE 文	探索条件中のインデクスを定義した列数＋ 1
DROP SCHEMA 文 DROP TABLE 文 DROP INDEX 文 CREATE INDEX 文 REVOKE 文でアクセス権限を取り消す場合	2

14.2.3 ユニットコントローラが使用する共用メモリの計算式

(1) 32 ビットモードの場合

ユニットの開始から終了までの間にユニットコントローラが使用する共用メモリは，次に示す HiRDB のプロセスの項目すべてを加算した値です。

なお，ユニットコントローラ全体の共用メモリサイズは 2 ギガバイト以内になるようにしてください。

プロセスの種類	共用メモリの計算式（単位：バイト）
スケジューラ	pd_utl_exec_mode の値が 0 の場合 ｛↑（432 ＋ 304×n）÷1024↑＋ 494 ＋ x ＋ z ＋↑（134 ＋ pd_trn_rcvmsg_store_buflen の値）÷1024↑｝×1024 pd_utl_exec_mode の値が 1 の場合 ｛↑（432 ＋ 304×n）÷1024↑＋↑（s×2000 ＋ 136）÷1024↑＋ y ＋ z ＋↑（134 ＋ pd_trn_rcvmsg_store_buflen の値）÷1024↑｝×1024 x：ユニット内に MGR がある場合：37 ユニット内に FES がある場合：57 ＋ 1×（s ＋ 3）＋ 14 ユニット内に DS がある場合：102 ＋ 5×（t ＋ 3）＋ 14 ユニット内に BES がある場合：{192 ＋ 9×（u ＋ 3）＋ 14｝×（BES 数＋ β ＋ γ） y：ユニット内に MGR がある場合：0 ユニット内に FES がある場合：1×（s ＋ 3）＋ 14 ユニット内に DS がある場合：5×（t ＋ 3）＋ 14 ユニット内に BES がある場合：{9×（u ＋ 3）＋ 14｝×（BES 数＋ β ＋ γ） z：影響分散スタンバイレス型系切り替えの対象となるユニットの場合： ↑64 ＋ {(ユニット内 BES 数＋受け入れ可能なゲスト BES 数※）×48｝÷1024↑ 上記以外のユニットの場合：0

プロセスの種類	共用メモリの計算式（単位：バイト）
	n：ユニット内のサーバ数＋ β ＋ γ ＋ユニット内ユティリティサーバ数＋ 1 ユニット内ユティリティサーバ数：24 ＋ α α：ユニット内に MGR がある場合：3 ユニット内に FES がある場合：3 ユニット内に DS がある場合：7 ユニット内に BES がある場合：（BES 数＋ β）×15 s：pd_max_users の値＋ pd_max_reflect_process_count の値 t：pd_max_dic_process の値＋ pd_max_reflect_process_count の値 u：pd_max_bes_process の値＋ pd_max_reflect_process_count の値 β：影響分散スタンバイレス型系切り替えの対象となるユニットの場合： 受け入れ可能なゲスト BES 数※ 上記以外のユニットの場合：0 γ：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：代替 BES 数 上記以外のユニットの場合：0 注※　pd_ha_max_guest_act_servers の値
ロックサーバ	**ユニット内にサーバ（FES，BES，又は DS）がある場合** 影響分散スタンバイレス型系切り替え機能の対象となるユニットで，ゲスト BES についての計算をする場合に使用するサーバごとのオペランド（pd_lck_pool_size，pd_lck_pool_partition，pd_lck_hash_entry，pd_max_bes_process など）の値は，そのゲスト BES のオペランドの値ではなく，そのユニット内の全ゲスト BES に指定されている値の最大値を使用します。また，影響分散スタンバイレス型系切り替え機能の対象となるユニットでは，「ユニット内サーバ」は「全ホスト BES ＋全ゲスト BES」です。 1：1 スタンバイレス型系切り替え機能の対象となるユニットでは，「ユニット内サーバ」は「全正規 BES ＋全代替 BES」です。 $\sum_{x=1}^{y} \{ 320 + 48 + c_x + d_x + 48 + 4096 + g_x + 48 + i_x + 48 + 12252 + 48 + n_x + p_x + t_x + u_x + 16 \}$ ×pd_lck_pool_partition の値※ 注※　FES の場合 pd_fes_lck_pool_partition の値 x：ユニット内サーバ通し番号 y：ユニット内サーバ数 c_x：pd_lck_hash_entry を省略，又は 0 を指定している場合： FES で，pd_fes_lck_pool_size が指定されていないとき （↓（8 ＋ 4×MAX（↑（↓↓（p_x ＋ 3）×（pd_max_access_tables の値＋ 4）÷6↓ ÷pd_fes_lck_pool_partition の値↓×6）÷10↑の値を超えない最大の素数，11261））÷16↓＋ 1）×16 FES で，pd_fes_lck_pool_size が指定されているとき （↓（8 ＋ 4×MAX（↑（↓pd_fes_lck_pool_size の値÷pd_fes_lck_pool_partition の値↓×6） ÷10↑の値を超えない最大の素数，11261））÷16↓＋ 1）×16

プロセスの種類	共用メモリの計算式（単位：バイト）
	BES，又は DS のとき （↓（8 ＋ 4×MAX（↑（(p_x ＋ 3）×2×5 ＋↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×6） ÷10↑の値を超えない最大の素数，11261））÷16↓＋ 1）×16 pd_lck_hash_entry に 2 以上の素数でない値を指定している場合： （↓（8 ＋ 4×pd_lck_hash_entry の値を超えない最大の素数）÷16↓＋ 1）×16 pd_lck_hash_entry に 1，又は素数を指定している場合： （↓（8 ＋ 4×pd_lck_hash_entry の値）÷16↓＋ 1）×16 d_x：FES で，pd_fes_lck_pool_size が指定されていない場合： （↓↓（p_x ＋ 3）×（pd_max_access_tables の値＋ 4）÷6↓ ÷pd_fes_lck_pool_partition の値↓×6）×96 FES で，pd_fes_lck_pool_size が指定されている場合： ↓pd_fes_lck_pool_size の値÷pd_fes_lck_pool_partition の値↓×6×96 BES，又は DS の場合： （(p_x ＋ 3）×2×5 ＋↓pd_lck_pool_size の値 ÷pd_lck_pool_partition の値↓×6）×96 g_x：FES の場合： （p ＋ 3）×2×256 BES で，pd_utl_exec_mode の値=1，かつ s ＞ 32 の場合： （（p ＋ 3）×2 ＋ s）×256 BES で，pd_utl_exec_mode の値=0，又は s≦32 の場合： （（p ＋ 3）×2 ＋ 32）×256 DS で，pd_utl_exec_mode の値=1，かつ s ＞ 16 の場合： （（p ＋ 3）×2 ＋ s）×256 DS で，pd_utl_exec_mode の値=0，又は s≦16 の場合： （（p ＋ 3）×2 ＋ 16）×256 i_x：FES で，pd_fes_lck_pool_size が指定されていない場合： （↓↓（p_x ＋ 3）×（pd_max_access_tables の値＋ 4）÷6↓ ÷pd_fes_lck_pool_partition の値↓）×12×64 FES で，pd_fes_lck_pool_size が指定されている場合： （↓pd_fes_lck_pool_size の値÷pd_fes_lck_pool_partition の値↓×8）を偶数に切り上げた値×64 BES で，pd_utl_exec_mode の値=1，かつ s ＞ 32 の場合： （↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×8 ＋（(p_x ＋ 3）×2×2×5 ＋ s×（pd_max_rdarea_no の値＋ 1）） を偶数に切り上げた値×64 BES で，pd_utl_exec_mode の値=0，又は s≦32 の場合： （↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×8 ＋（(p_x ＋ 3）×2×2×5 ＋ 32×（pd_max_rdarea_no の値＋ 1）） を偶数に切り上げた値×64 DS で，pd_utl_exec_mode の値=1，かつ s ＞ 16 の場合：

プロセスの種類	共用メモリの計算式（単位：バイト）
	（↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×8 ＋（p_x＋3）×2×2×5＋s＋4）を偶数に切り上げた値×64 DS で，pd_utl_exec_mode の値=0，又は s≦16 の場合： （↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×8 ＋（p_x＋3）×2×2×5＋20）を偶数に切り上げた値×64 n_x：FES の場合： （p_x＋3）×2×48 BES で，pd_utl_exec_mode の値=1，かつ s＞32 の場合： （（p_x＋3）×2×17＋s）×48 BES で，pd_utl_exec_mode の値=0，又は s≦32 の場合： （（p_x＋3）×2×17＋32）×48 DS で，pd_utl_exec_mode の値=1，かつ s＞16 の場合： （（p_x＋3）×2×17＋s）×48 DS で，pd_utl_exec_mode の値=0，又は s≦16 の場合： （（p_x＋3）×2×17＋16）×48 p_x：FES で，HiRDB システム内の FES 数＞1 の場合：s＋1 FES で，HiRDB システム内の FES 数=1 の場合：s BES で，s＞pd_max_bes_process の値 の場合：s BES で，s≦pd_max_bes_process の値 の場合： pd_max_bes_process オペランドの値 ＋pd_max_reflect_process_count オペランドの値 DS で，s＞pd_max_dic_process の値 の場合：s DS で，s≦pd_max_dic_process の値 の場合： pd_max_dic_process オペランドの値 ＋pd_max_reflect_process_count オペランドの値 s：pd_max_users の値＋pd_max_reflect_process_count の値 t_x：FES の場合： 48＋（p_x＋3）×2×↑pd_max_open_holdable_cursors の値÷16↑×4 BES で，pd_utl_exec_mode の値=1，かつ s＞32 の場合： 48＋（（p_x＋3）×2＋s）×↑pd_max_open_holdable_cursors の値÷16↑×4 BES で，pd_utl_exec_mode の値=0，又は s≦32 の場合： 48＋（（p_x＋3）×2＋32）×↑pd_max_open_holdable_cursors の値÷16↑×4 DS で，pd_utl_exec_mode の値=1，かつ s＞16 の場合： 48＋（（p_x＋3）×2＋s）×↑pd_max_open_holdable_cursors の値÷16↑×4 DS で，pd_utl_exec_mode の値=0，又は s≦16 の場合： 48＋（（p_x＋3）×2＋16）×↑pd_max_open_holdable_cursors の値÷16↑×4 u_x：FES で，pd_fes_lck_pool_size が指定されていない場合： 48＋（↓↓（p_x＋3）×（pd_max_access_tables の値＋4）÷6↓÷pd_fes_lck_pool_partition の値↓）×12 ×↑pd_max_open_holdable_cursors の値÷16↑×4 FES で，pd_fes_lck_pool_size が指定されている場合：

プロセスの種類	共用メモリの計算式（単位：バイト）
	48 ＋ ↓pd_fes_lck_pool_size の値÷pd_fes_lck_pool_partition の値↓ ×8 × ↑pd_max_open_holdable_cursors の値÷16↑ ×4 BES で，pd_utl_exec_mode の値=1，かつ s ＞ 32 の場合： 48 ＋（↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓ ×8 ＋（p_x ＋ 3）×2×2×5 ＋ s×（pd_max_rdarea_no の値＋ 1）） を偶数に切り上げた値× ↑pd_max_open_holdable_cursors の値÷16↑ ×4 BES で，pd_utl_exec_mode の値=0，又は s≦32 の場合： 48 ＋（↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓ ×8 ＋（p_x ＋ 3）×2×2×5 ＋ 32×（pd_max_rdarea_no の値＋ 1）） を偶数に切り上げた値× ↑pd_max_open_holdable_cursors の値÷16↑ ×4 DS で，pd_utl_exec_mode の値=1，かつ s ＞ 16 の場合： 48 ＋（↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓ ×8 ＋（p_x ＋ 3）×2×2×5 ＋ s ＋ 4）を偶数に切り上げた値 × ↑pd_max_open_holdable_cursors の値÷16↑ ×4 DS で，pd_utl_exec_mode の値=0，又は s≦16 の場合： 48 ＋（↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓ ×8 ＋（p_x ＋ 3）×2×2×5 ＋ 20）を偶数に切り上げた値 × ↑pd_max_open_holdable_cursors の値÷16↑ ×4 **●ユニット内にサーバ（FES，BES，又は DS）がない場合** 8416
トランザクション サーバ	288 ＋ 32×B ＋ 192×s×2 **ユニット内に FES がある場合に加算します※** ＋ 1028 ＋（420 ＋ 624 ＋ 256 ＋ 384×2）×（A×2 ＋ 7）＋ 256×2 ＋ 128×（システム内 BES 数×4 ＋システム内 DS 数×2 ＋システム内 FES 数）×（A×2 ＋ 7） ＋ C **ユニット内に BES がある場合に加算します※** ＋ 1028 ＋（420 ＋ 624 ＋ 256 ＋ 384×2） ×（u×2 ＋ 7）＋ 256×2 ＋ 128×（システム内 BES 数×4 ＋システム内 DS 数×2 ＋システム内 FES 数）×（A×2 ＋ 7） ＋ D **ユニット内に DS がある場合に加算します※** ＋ 1028 ＋（420 ＋ 624 ＋ 256 ＋ 384×2） ×（t×2 ＋ 7）＋ 256×2 ＋ 128×（システム内 BES 数×4 ＋システム内 DS 数×2 ＋システム内 FES 数）×（A×2 ＋ 7） ＋ E s ：pd_max_users の値＋ pd_max_reflect_process_count の値 t ：pd_max_dic_process の値＋ pd_max_reflect_process_count の値 u ：pd_max_bes_process の値＋ pd_max_reflect_process_count の値 A ：マルチ FES の場合：s ＋ 1

プロセスの種類	共用メモリの計算式（単位：バイト）
	マルチ FES でない場合：s B：影響分散スタンバイレス型系切り替えの対象となるユニットの場合： ホスト BES 数＋ pd_ha_max_act_guest_servers オペランドの補正値 上記以外のユニットの場合：ユニット内サーバ数 C：≪条件≫に示すどちらかの条件に一致する場合： 128×（システム内 BES 数×4 ＋システム内 DS 数×2 ＋システム内 FES 数）×（A×2 ＋ 7） ≪条件≫に示すどちらの条件にも一致しない場合：0 D：≪条件≫に示すどちらかの条件に一致する場合： 128×（システム内 BES 数×4 ＋システム内 DS 数×2 ＋システム内 FES 数）×（u×2 ＋ 7） ≪条件≫に示すどちらの条件にも一致しない場合：0 E：≪条件≫に示すどちらかの条件に一致する場合： 128×（システム内 BES 数×4 ＋システム内 DS 数×2 ＋システム内 FES 数）×（t×2 ＋ 7） ≪条件≫に示すどちらの条件にも一致しない場合：0 ≪条件≫ • pd_rpl_reflect_mode オペランドに uap を指定している • システム内に，pdstart -k stls オペランドを指定したフロントエンドサーバが存在する 注※　上記の計算式で算出した値を各サーバの数分だけ加算します。 1：1 スタンバイレス型系切り替え適用ユニットの場合： 正規 BES 数＋代替 BES 数 上記以外の場合はユニット内サーバ数
タイマサーバ	32×（pd_max_users の値＋ pd_max_reflect_process_count の値＋ 3） ×（システム内の BES 数＋ 1 ＋ユニット内ユティリティサーバ数＋ 1） ＋ 1440 ユニット内ユティリティサーバ数は 23 ＋ α α：ユニット内に MGR がある場合：2 ユニット内に FES がある場合：3 ユニット内に DS がある場合：7 ユニット内に BES がある場合：BES 数×b b：AIX 版の場合：15 Linux 版の場合：6
統計ログサーバ	384 ＋ 128×16 ＋ 32 ＋ 288×2 ＋ 1024 ＋ 128×3 ＋ pd_stj_buff_size の値×1024×3 ＋ 64 ＋ 4096 ＋ 8192
プロセスサーバ	192 ＋ 512×MAX（c，256）＋ 96 ＋ 256 ＋（pd_max_server_process の値＋ 50）×（256 ＋ 144）＋ 16 ＋ 8×34 ＋ 16 ＋ 16 ＋ 48 ＋ 48×（k ＋ 1） c：↑（51 ＋ d ＋ e ＋ f ＋（g×ユニット内の BES 数※）＋ h ＋ i）÷16↑×16 d：ユニット内に MGR がある場合は 59，ない場合は j e：ユニット内に DS がある場合は 17，ない場合は 0 f：ユニット内に FES がある場合は 11，ない場合は 0

プロセスの種類	共用メモリの計算式（単位：バイト）
	g：ユニット内に BES がある場合は 25，ない場合は 0 h：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 9，使用しない場合は 0 i：影響分散スタンバイレス型系切り替えの対象となるユニットの場合は 1，対象でないユニットの場合は 0 j：pd_mlg_msg_log_unit オペランドに manager を指定しているときは 1，local を指定しているときは 2 k：システム共通定義，又はユニット制御情報定義に pd_module_trace_max オペランドを指定している場合は pd_module_trace_max の値，指定していない場合は 16383 注※ 1：1 スタンバイレス型系切り替えの対象となっている場合は（BES 数×2）となります。影響分散スタンバイレス型系切り替えの対象となるユニットの場合は pd_ha_max_act_guest_servers オペランドの補正値を含みます。
システムマネジャ	992 + (44 + 4) × (g + h + i) + (100 + 4) × {(p + q + 3) + u× (15 + 1)} + (92 + 4) ×c + 40× (k + m + n×o + u) ×14 + 256×m + 128×c + 36×d + 12×e + 96×o + v × (16×35× (k + u) + 15 + 36×z + 15) + w× (48× B + 15 + 4×z + 15 + 4×y + 15) + v× (132 + 15) + 8 + 13080 + 5856×p + s + s×o + 16 + 96×o + 1024 + 272× A c：ユニット数 d：pdunit オペランドの-c オプション指定数 e：pdcltgrp オペランド指定数 g：システム内の FES 数 h：システム内の BES 数 i：システム内の DS 数 j：ユニット内の FES 数 k：ユニット内の BES 数 m：ユニット内の DS 数 n：ユニット内の代替 BES 数 o：ユニットが 1：1 スタンバイレス型系切り替えの対象となっている場合は 1， 対象となっていない場合は 0 p：i + k + m + n q：24 + t + j×3 + k×15 + m×7 r：14× (k + m + u) + p + q + u×15 + 2 + 38 + 5 + p×4 s：272 + 2052 + 128× (r + 3) + v× (40× (k + u) + 72× (k + u)) t：ユニット内に MGR がある場合は 2，ない場合は 0 u：受け入れ可能なゲスト BES 数（pd_ha_max_act_guest_servers オペランドの値） v：ユニットが影響分散スタンバイレス型系切り替えの対象となっている場合は 1， 対象となっていない場合は 0 w：影響分散スタンバイレス型系切り替えの対象となっているユニットがある場合は 1， ない場合は 0 y：HA グループ内のユニット数の総和 z：HA グループ内のサーバ数の総和 A：pd_security_host_group オペランドに指定したホストに対応する IP アドレスの数 pd_security_host_group オペランドを指定していない場合は 0 B：システム共通定義に指定している pdhagroup オペランドの数

プロセスの種類	共用メモリの計算式（単位：バイト）
ネームサーバ	MAX｛65536，（X ＋ Y ＋ Z)｝＋ MAX（16384，L）＋ M X：↑（256 ＋ 16 ＋ 156×ユニット数＋ 16 ＋ 16×126）÷1024↑×1024 Y：8192 Z：↑（264×（Z_2 ＋ Z_3 ＋ j ＋ 32））÷1024↑×1024 Z_2：b ＋ 10 ＋ c ＋ 11×自ユニット内の HiRDB サーバ数＋ d ＋ e Z_3：f ＋ 7 ＋ g ＋ 4×自ユニット内の HiRDB サーバ数＋ h ＋ i L：↑（224×（L_2 ＋ L3 ＋ L4））÷1024↑×1024 L_2：k ＋ 2×システム内のユニット数 L_3：システム内の BES 数＋システム内の FES 数＋システム内の DS 数 L_4：15×システム内の HiRDB サーバ数 M：ユニット内の HiRDB サーバ数＋ z ＋ m×ユニット内システムサーバ数 b：ユニットに MGR がある場合：3 ユニットに MGR がない場合：0 c：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：2 それ以外のユニットの場合：0 d：影響分散スタンバイレス型系切り替えの対象となるユニットの場合： 11×受け入れ可能なゲスト BES 数 それ以外のユニットの場合：0 e：1：1 スタンバイレス型系切り替えの対象となるユニットの場合： 6×自ユニット内の HiRDB サーバ数 それ以外のユニットの場合：0 f：ユニットに MGR がある場合：3 ユニットに MGR がない場合：0 g：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：2 それ以外のユニットの場合：0 h：影響分散スタンバイレス型系切り替えの対象となるユニットの場合： 3×受け入れ可能なゲスト BES 数 それ以外のユニットの場合：0 i：1：1 スタンバイレス型系切り替えの対象となるユニットの場合： 4×自ユニット内の HiRDB サーバ数 それ以外のユニットの場合：0 j：ユニット内のサーバ数＋ β ＋ γ ＋ユニット内ユティリティサーバ数＋ 2 ユニット内ユティリティサーバ数：23 ＋ α k：ユニットに MGR がある場合：3 ユニットに MGR がない場合：0 m：38 ＋ n ＋ o n：ユニットに MGR があり，かつユニット数が 2 以上の場合：5 ユニットに MGR があり，かつユニット数が 1 の場合：4 ユニットに MGR がなく，かつ pd_mlg_msg_log_unit の値が local の場合：4 ユニットに MGR がなく，かつ pd_mlg_msg_log_unit の値が manager の場合：3 o：4×ユニット内の HiRDB サーバ数 z：23 ＋ α

プロセスの種類	共用メモリの計算式（単位：バイト）
	α：ユニット内に MGR がある場合：3 ユニット内に FES がある場合：3 ユニット内に DS がある場合：7 ユニット内に BES がある場合：（BES 数＋ β）×15 β：影響分散スタンバイレス型系切り替えの対象となるユニットの場合： 受け入れ可能なゲスト BES 数※ 上記以外のユニットの場合：0 γ：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：代替 BES 数 上記以外のユニットの場合：0 注※　pd_ha_max_guest_act_servers の値
ノードマネジャ	**ユニット内に MGR がある場合** ↑（1152 ＋ 432×システムの全ユニット数＋ 80×システムの全サーバ数 ＋ 7680 ＋ 1008 ＋ 56×C ＋ 240×A ＋ 44×A ＋ 28×A ＋ 16×B ＋ 16×システムの全 BES 数＋ 8×システムの全ユニット数 ＋ 64×システムの全サーバ数＋ 32×システムの全サーバ数＋ 32） ÷1024↑×1024 **ユニット内に MGR がない場合** ↑（1008 ＋ 56×C ＋（240×A ＋ 44×A ＋ 28×A）×F ＋ 16×B ＋ 16×システムの全 BES 数＋ 8×システムの全ユニット数 ＋ 64×システムの全サーバ数＋ 32×システムの全サーバ数＋ 32） ÷1024↑×1024 A：pd_utl_exec_mode=0 の場合：1024 pd_utl_exec_mode=1 の場合：pd_max_users の値×システムの全 BES 数×3 ユニットに MGR がある場合は次に示す値を加算：pd_max_users の値×4 ＋ 200 ユニットに DS がある場合は次に示す値を加算：pd_max_users の値×3 ＋ 200 ユニットに BES がある場合は次に示す値を加算：pd_max_users の値×D 上記の計算式で算出した A の値が 1024 を超えない場合は，A を 1024 に置き換えてください。 B：pdcltgrp オペランドを指定しない場合：0 pdcltgrp オペランドを指定する場合：pdcltgrp 指定数＋ 1 C：ユニット内サーバ数＋ E D：ユニット内 BES 数＋ E E：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：ユニット内の代替 BES 数 影響分散スタンバイレス型系切り替えの対象となるユニットの場合：受け入れ可能なゲスト BES 数 上記以外のユニットの場合：0 F：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：2 上記以外のユニットの場合：1
I/O サーバ	↑（28 ＋（↑（32 ＋ A）÷32↑×32））÷128↑×128 **影響分散スタンバイレス型系切り替え機能の対象となるユニットではない場合** A：34268 ＋ pd_max_utl_ios_file_no の値×296 ＋ 158064※1 ＋ {(162800 ＋ pd_max_file_no の値×1024）×BES 数}※2 ＋ {162800 ＋ pd_max_file_no の値×1024}※3

プロセスの種類	共用メモリの計算式（単位：バイト）
	注※1　FES がある場合に加算します。 注※2　BES がある場合に加算します。 注※3　DS がある場合に加算します。 なお，1：1 スタンバイレス型系切り替え構成対象ユニットの場合には，上記の式で求めた値を 2 倍にします。 **影響分散スタンバイレス型系切り替え機能の対象となるユニットの場合** A：↑48 ＋ 24×BES 数※4÷16↑×16 ＋↑（177904 ＋ pd_max_file_no の値×1024）÷16↑×16×BES 数※4 ＋ 34016 注※4　pd_ha_max_act_guest_servers の値を含みます。
ログサーバ	32 ＋ 48 ＋ 128×37 ＋ { 384 ＋ 128×7 ＋ 1024 ＋ 512 ＋↑（128 ＋ 256 ＋ 160 ＋ 8 ＋ 64）÷pd_log_rec_leng の値※↑ ×pd_log_rec_leng の値※ ＋ 64 ＋ 4096×2 ＋（736 ＋ 512）×B ＋↑ {(B ＋ 1）÷12} ↑×8320 ＋ 128×pd_log_write_buff_count の値※ ＋（pd_log_write_buff_count の値※＋ A） ×↑ {pd_log_max_data_size の値※＋（68 ＋ 44 ＋ 96 ＋ 160)} ÷4096↑ ×4096 ＋ C } ×ユニット内サーバ数＋ D ＋ 128×ユニット内 FES 数 **pd_max_reflect_process_count オペランドを指定する場合に加算します。** (128 ＋ 704）×（ユニット内 BES 数＋ D） A：16 B：pdlogadfg -d sys オペランド指定数※ C：0 D：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：ユニット内の代替 BES 数 影響分散スタンバイレス型系切り替えの対象となるユニットの場合： pd_ha_max_act_guest_servers オペランドの補正値 注※ ユニット内の全サーバの指定値で，最大値を指します。1：1 スタンバイレス型系切り替えの対象となるユニットの場合はユニット内の全サーバとユニット内で動作する代替 BES の指定値で，最大値を指します。影響分散スタンバイレス型系切り替えの対象となるユニットの場合はユニット内の全サーバと HA グループに含まれる全 BES の指定値で，最大値を指します。
シンクポイントダンプサーバ	{ ↑（368 ＋ 1456×2）÷1024↑×1024 ＋↑ {(96 ＋ 80 ＋ 208 ＋ 208）＋ 192×（pdlogadfg -d spd の指定数※） ＋ 416×（pdlogadpf -d spd の指定数※)} ÷1024↑×1024 } ×（全サーバ数＋ A） A：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：ユニット内の代替 BES 数

プロセスの種類	共用メモリの計算式（単位：バイト）
	影響分散スタンバイレス型系切り替えの対象となるユニットの場合： pd_ha_max_act_guest_servers オペランドの補正値 注※ ユニット内の全サーバの指定値で，最大値を指します。1：1 スタンバイレス型系切り替えの対象となるユニットの場合はユニット内の全サーバとユニット内で動作する代替 BES の指定値で，最大値を指します。影響分散スタンバイレス型系切り替えの対象となるユニットの場合はユニット内の全サーバと HA グループに含まれる全 BES の指定値で，最大値を指します。
ユニット共通	a ＋ {b ＋ 64 ＋（s ＋ 3）×c ＋ 64 ＋ 48 ＋ d ＋ e} ×（ユニット内の FES，BES，DS の合計数＋ i） ＋（g×（ユニット内の BES，DS の合計数＋ i））＋ f ＋（pd_max_server_process の値×2 ＋ 100）×（48 ＋ 16）＋ 32 ＋（pd_max_server_process の値×2 ＋ 100 ＋ 384）×32 ＋ 32 ＋ h ＋ j ＋ v ＋ w ＋（pd_max_server_process の値＋ 127 ＋ y）×32 ＋ 32 ＋ A **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に加算します。** (↑（28 ＋(↑（56 ＋ 72584）÷32↑×32)）÷128↑×128) a：26000 b：2988 c：2716 d：32×32 e：64 ＋ 64×｛(s ＋ 3）×2 ＋ MAX（5，↓［s ＋ 3］÷10↓）＋ 7｝ f：512×（13 ＋ユニット内の FES，BES，DS の合計数×3）×2 g：{↑（96 ＋ pd_lck_until_disconnect_cnt の値×112）÷4096↑} ×4096×2 h：↑（pd_registered_port で指定したポート番号の数×16 ＋ 32） ÷1024↑×1024 pd_registered_port を指定していない場合は 0 i：1：1 スタンバイレス型系切り替えの場合は代替 BES 数 影響分散スタンバイレス型系切り替えの場合は pd_ha_max_guest_servers 補正値 j：p×（ユニット内の FES 数）＋ q×（ユニット内の BES 数＋ i）＋ r×（ユニット内の DS 数） k：FES の pd_fes_lck_pool_partition の値 m：BES の pd_lck_pool_partition の値 n：DS の pd_lck_pool_partition の値 o：(s ＋ 3）×2 ＋ MAX｛5，↓（s ＋ 3）÷10↓｝＋ 7 p：k が 2 以上の場合は 32 ＋（8 ＋ 8×k）×o それ以外の場合は 0 q：m が 2 以上の場合は 32 ＋（8 ＋ 8×m）×o それ以外の場合は 0 r：n が 2 以上の場合は 32 ＋（8 ＋ 8×n）×o それ以外の場合は 0 s：pd_max_users の値＋ pd_max_reflect_process_count の値 t：pd_max_dic_process の値＋ pd_max_reflect_process_count の値 u：pd_max_bes_process の値＋ pd_max_reflect_process_count の値

プロセスの種類	共用メモリの計算式（単位：バイト）
	s は，DS の場合は t，BES の場合は u とします。pd_max_dic_process 又は pd_max_bes_process を省略する場合は s とします。 v：pd_dbbuff_modify に Y を指定し，かつユニット内に BES 又は DS がある場合：69648 ＋（BES 定義又は DS 定義の pd_max_add_dbbuff_shm_no の値のユニット内最大値）×136 pd_dbbuff_modify に N を指定し，かつユニット内に BES 又は DS がある場合：69648 上記以外：2192 w：144 y：次に示す計算式で算出した値を，各サーバの数分だけ加算します。 ユニット内に FES がある場合に加算します。 （z ＋ 3）×2 ユニット内に BES があり※1，pd_utl_exec_mode=1 かつ s ＞ 32 の場合に加算します。 （z ＋ 3）×34 ＋ s ユニット内に BES があり※1，pd_utl_exec_mode=0 又は s≦32 の場合に加算します。 （z ＋ 3）×34 ＋ 32 ユニット内に DS があり，pd_utl_exec_mode=1 かつ s ＞ 16 の場合に加算します。 （z ＋ 3）×34 ＋ s ユニット内に DS があり，pd_utl_exec_mode=0 又は s≦16 の場合に加算します。 （z ＋ 3）×34 ＋ 16 z：FES で，HiRDB システム内の FES 数＞ 1 の場合：s ＋ 1 FES で，HiRDB システム内の FES 数＝ 1 の場合：s BES で，s ＞ pd_max_bes_process の値※2 の場合：s BES で，s≦pd_max_bes_process の値※2 の場合：u DS で，s ＞ pd_max_dic_process の値の場合：s DS で，s≦pd_max_dic_process の値の場合：t A：640 注※1 1：1 スタンバイレス型系切り替えの対象となるユニットの場合，ユニット内の全正規 BES と全代替 BES が対象となります。 影響分散スタンバイレス型系切り替えの対象となるユニットの場合，ユニット内の全ホスト BES と全ゲスト BES が対象となります。 注※2 影響分散スタンバイレス型系切り替えとなるユニットで，ゲスト BES についての計算をする場合，そのゲスト BES のオペランドの値ではなく，そのユニット内の全ゲスト BES に指定されている値の最大値を計算に使用してください。
トランザクションログサーバ	{1024 ＋ 512×A} ×（ユニット内サーバ数＋ H） ＋ { 128×B ＋ 128 ＋[F ＋↑（128 ＋ 256 ＋ 8 ＋ 224）÷pd_log_rec_leng の値※↑×pd_log_rec_leng の値※ ＋↑（pd_log_max_data_size の値※＋ 68 ＋ 44 ＋ 96 ＋ 160） ÷pd_log_rec_leng の値※↑×pd_log_rec_leng の値※] ×D ＋ E ＋（48 ＋ 8）×B×2 } ×（ユニット内の BES 及び DS 数＋ H）

プロセスの種類	共用メモリの計算式（単位：バイト）
	＋｛ 584×B＋128×B＋64×B×C＋128＋F ＋↑（128＋256＋8＋224）÷pd_log_rec_leng の値※↑×pd_log_rec_leng の値※ ＋↑（pd_log_max_data_size の値※＋68＋44＋96＋160） ÷pd_log_rec_leng の値※↑×pd_log_rec_leng の値※ ＋E＋（48＋8）×（B×2＋2） ｝×（ユニット内サーバ数＋H） A：2 B：7＋J×2 C：システム全体の BES 数×4＋システム全体の DS 数×2 ＋システム全体の FES 数 D：pd_log_rollback_buff_count の値が 0 の場合：8 それ以外の場合：pd_log_rollback_buff_count の値 E：0 F：60 H：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：ユニット内の代替 BES 数 影響分散スタンバイレス型系切り替えの対象となるユニットの場合： pd_ha_max_act_guest_servers オペランドの補正値 J：ユニット内サーバ中の，s，t，及び u の最大値 s：pd_max_users の値＋ pd_max_reflect_process_count の値 t：pd_max_dic_process の値＋ pd_max_reflect_process_count の値 u：pd_max_bes_process の値＋ pd_max_reflect_process_count の値 注※ ユニット内の全サーバの指定値で，最大値を指します。1：1 スタンバイレス型系切り替えの対象となるユニットの場合はユニット内の全サーバとユニット内で動作する代替 BES の指定値で，最大値を指します。影響分散スタンバイレス型系切り替えの対象となるユニットの場合はユニット内の全サーバと HA グループに含まれる全 BES の指定値で，最大値を指します。
ステータスサーバ	↑64÷32↑×32×（ユニット内サーバ数＋A） A：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：ユニット内の代替 BES 数 影響分散スタンバイレス型系切り替えの対象となるユニットの場合： pd_ha_max_act_guest_servers オペランドの補正値
監査証跡管理サーバ	↑A÷1024↑×1024 A：pd_aud_file_name オペランドの指定がない場合は 640 pd_aud_file_name オペランドの指定がある場合は 640＋（304×200）＋B＋C B：pd_aud_async_buff_size オペランドの値が 0 の場合は 0 pd_aud_async_buff_size オペランドの値が 4096 以上の場合は次の計算値 Linux の場合 （160×pd_aud_async_buff_count オペランドの値） ＋｛(↑pd_aud_async_buff_size オペランドの値÷4096↑×4096) ×pd_aud_async_buff_count オペランドの値｝＋4096 Linux 以外の場合 （160×pd_aud_async_buff_count オペランドの値）

プロセスの種類	共用メモリの計算式（単位：バイト）
	＋｛(↑pd_aud_async_buff_size オペランドの値÷4096↑×4096) ×pd_aud_async_buff_count オペランドの値｝ C：pd_aud_auto_loading オペランドに Y を指定し，かつ MGR があるユニットの場合は 256×（システム全体のユニット数＋ 1）＋ 240 上記以外の場合は 0 スタンバイレス型系切り替え機能を適用するユニットの場合，自ユニットのメモリ使用量に，系切り替え先となるユニットのセキュリティ監査のメモリ使用量を加算する必要があります。

(2) 64 ビットモードの場合

ユニットの開始から終了までの間にユニットコントローラが使用する共用メモリは，次に示す HiRDB のプロセスの項目すべてを加算した値です。

プロセスの種類	共用メモリの計算式（単位：バイト）
スケジューラ	**pd_utl_exec_mode の値が 0 の場合** ｛↑（432 ＋ 304×n）÷1024↑＋ 494 ＋ x ＋ z ＋↑（134 ＋ pd_trn_rcvmsg_store_buflen の値）÷1024↑｝×1024 **pd_utl_exec_mode の値が 1 の場合** ｛↑（432 ＋ 304×n）÷1024↑＋↑（s×2000 ＋ 136）÷1024↑＋ y ＋ z ＋↑（134 ＋ pd_trn_rcvmsg_store_buflen の値）÷1024↑｝×1024 x：ユニット内に MGR がある場合：37 ユニット内に FES がある場合：57 ＋ 1×（s ＋ 3）＋ 14 ユニット内に DS がある場合：102 ＋ 5×（t ＋ 3）＋ 14 ユニット内に BES がある場合：{192 ＋ 9×（u ＋ 3）＋ 14｝×（BES 数＋ β ＋ γ） y：ユニット内に MGR がある場合：0 ユニット内に FES がある場合：1×（s ＋ 3）＋ 14 ユニット内に DS がある場合：5×（t ＋ 3）＋ 14 ユニット内に BES がある場合：{9×（u ＋ 3）＋ 14｝×（BES 数＋ β ＋ γ） z：影響分散スタンバイレス型系切り替えの対象となるユニットの場合： ↑64 ＋ {(ユニット内 BES 数＋受け入れ可能なゲスト BES 数※）×48｝÷1024↑ 上記以外のユニットの場合：0 n：ユニット内のサーバ数＋ β ＋ γ ＋ユニット内ユティリティサーバ数＋ 1 ユニット内ユティリティサーバ数：24 ＋ α α：ユニット内に MGR がある場合：3 ユニット内に FES がある場合：3 ユニット内に DS がある場合：7 ユニット内に BES がある場合：(BES 数＋ β）×15 s：pd_max_users の値＋ pd_max_reflect_process_count の値 t：pd_max_dic_process の値＋ pd_max_reflect_process_count の値 u：pd_max_bes_process の値＋ pd_max_reflect_process_count の値 β：影響分散スタンバイレス型系切り替えの対象となるユニットの場合： 受け入れ可能なゲスト BES 数※

プロセスの種類	共用メモリの計算式（単位：バイト）
	上記以外のユニットの場合：0 γ：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：代替 BES 数 上記以外のユニットの場合：0 注※　pd_ha_max_guest_act_servers の値
ロックサーバ	**ユニット内にサーバ（FES，BES，又は DS）がある場合** 影響分散スタンバイレス型系切り替え機能の対象となるユニットで，ゲスト BES についての計算をする場合に使用するサーバごとのオペランド（pd_lck_pool_size，pd_lck_pool_partition，pd_lck_hash_entry，pd_max_bes_process など）の値は，そのゲスト BES のオペランドの値ではなく，そのユニット内の全ゲスト BES に指定されている値の最大値を使用します。また，影響分散スタンバイレス型系切り替え機能の対象となるユニットでは，「ユニット内サーバ」は「全ホスト BES ＋全ゲスト BES」です。 1：1 スタンバイレス型系切り替え機能の対象となるユニットでは，「ユニット内サーバ」は「全正規 BES ＋全代替 BES」です。 $\sum_{x=1}^{y}$ ｛ 496 ＋ 80 ＋ c_x ＋ d_x ＋ 64 ＋ 8192 ＋ g_x ＋ 80 ＋ i_x ＋ 64 ＋ 16336 ＋ 64 ＋ n_x ＋ p_x ＋ t_x ＋ u_x ＋ 16 ｝×pd_lck_pool_partition の値※ 注※　FES の場合は pd_fes_lck_pool_partition の値 x：ユニット内サーバ通し番号 y：ユニット内サーバ数 c_x：pd_lck_hash_entry を省略，又は 0 を指定している場合： FES で，pd_fes_lck_pool_size が指定されていないとき （↓（8 ＋ 8×MAX（↑（↓↓（p_x ＋ 3）×（pd_max_access_tables の値＋ 4）÷4↓ ÷pd_fes_lck_pool_partition の値↓×4）÷10↑の値を超えない最大の素数，11261）） ÷16↓＋ 1）×16 FES で，pd_fes_lck_pool_size が指定されているとき （↓（8 ＋ 8×MAX（↑（↓pd_fes_lck_pool_size の値÷pd_fes_lck_pool_partition の値↓×4）÷10↑ の値を超えない最大の素数，11261））÷16↓＋ 1）×16 BES，又は DS のとき （↓（8 ＋ 8×MAX（↑（（p_x ＋ 3）×2×5 ＋↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×4）÷10↑ の値を超えない最大の素数，11261））÷16↓＋ 1）×16 pd_lck_hash_entry に 2 以上の素数でない値を指定している場合： （↓（8 ＋ 8×pd_lck_hash_entry の値を超えない最大の素数）÷16↓＋ 1） ×16 pd_lck_hash_entry に 1，又は素数を指定している場合： （↓（8 ＋ 8×pd_lck_hash_entry の値）÷16↓＋ 1）×16 d_x：FES で，pd_fes_lck_pool_size が指定されていない場合： （↓↓（p_x ＋ 3）×（pd_max_access_tables の値＋ 4）÷4↓

プロセスの種類	共用メモリの計算式（単位：バイト）
	÷pd_fes_lck_pool_partition の値↓×4）×128 FES で，pd_fes_lck_pool_size が指定されている場合： ↓pd_fes_lck_pool_size の値÷pd_fes_lck_pool_partition の値↓×4×128 BES，又は DS の場合： （（p_x ＋ 3）×2×5 ＋↓pd_lck_pool_size の値 ÷pd_lck_pool_partition の値↓×4）×128 g_x：FES の場合： （p ＋ 3）×2×320 BES で，pd_utl_exec_mode の値=1，かつ s ＞ 32 の場合： （（p ＋ 3）×2 ＋ s）×320 BES で，pd_utl_exec_mode の値=0，又は s≦32 の場合： （（p ＋ 3）×2 ＋ 32）×320 DS で，pd_utl_exec_mode の値=1，かつ s ＞ 16 の場合： （（p ＋ 3）×2 ＋ s）×320 DS で，pd_utl_exec_mode の値=0，又は s≦16 の場合： （（p ＋ 3）×2 ＋ 16）×320 i_x：FES で，pd_fes_lck_pool_size が指定されていない場合： （↓↓（p_x ＋ 3）×（pd_max_access_tables の値＋ 4）÷4↓ ÷pd_fes_lck_pool_partition の値↓）×8×112 FES で，pd_fes_lck_pool_size が指定されている場合： （↓pd_fes_lck_pool_size の値÷pd_fes_lck_pool_partition の値↓×5 ＋↓↓pd_fes_lck_pool_size の値÷pd_fes_lck_pool_partition の値↓÷3↓）を偶数に切り上げた値×112 BES で，pd_utl_exec_mode の値=1，かつ s ＞ 32 の場合： （↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×5 ＋↓↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓÷3↓ ＋（p_x ＋ 3）×2×2×5 ＋ s×（pd_max_rdarea_no の値＋ 1）） を偶数に切り上げた値×112 BES で，pd_utl_exec_mode の値=0，又は s≦32 の場合： （↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×5 ＋↓↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓÷3↓ ＋（p_x ＋ 3）×2×2×5 ＋ 32×（pd_max_rdarea_no の値＋ 1）） を偶数に切り上げた値×112 DS で，pd_utl_exec_mode の値=1，かつ s ＞ 16 の場合： （↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×5 ＋↓↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓÷3↓ ＋（p_x ＋ 3）×2×2×5 ＋ s ＋ 4）を偶数に切り上げた値×112 DS で，pd_utl_exec_mode の値=0，又は s≦16 の場合： （↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×5 ＋↓↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓÷3↓ ＋（p_x ＋ 3）×2×2×5 ＋ 20） を偶数に切り上げた値×112

プロセスの種類	共用メモリの計算式（単位：バイト）
	n_x：FES の場合： （p_x ＋ 3）×2×80 BES で，pd_utl_exec_mode の値=1，かつ s ＞ 32 の場合： （(p ＋ 3）×2×17 ＋ s）×80 BES で，pd_utl_exec_mode の値=0，又は s≦32 の場合： （(p ＋ 3）×2×17 ＋ 32）×80 DS で，pd_utl_exec_mode の値=1，かつ s ＞ 16 の場合： （(p ＋ 3）×2×17 ＋ s）×80 DS で，pd_utl_exec_mode の値=0，又は s≦16 の場合： （(p ＋ 3）×2×17 ＋ 16）×80 p_x：FES で，HiRDB システム内の FES 数＞ 1 の場合：s ＋ 1 FES で，HiRDB システム内の FES 数=1 の場合：s BES で，s ＞ pd_max_bes_process の値 の場合：s BES で，s≦pd_max_bes_process の値 の場合： pd_max_bes_process オペランドの値 ＋ pd_max_reflect_process_count オペランドの値 DS で，s ＞ pd_max_dic_process の値 の場合：s DS で，s≦pd_max_dic_process の値 の場合： pd_max_dic_process オペランドの値 ＋ pd_max_reflect_process_count オペランドの値 s ：pd_max_users の値＋ pd_max_reflect_process_count の値 t_x：FES の場合： 48 ＋（p_x ＋ 3）×2×↑pd_max_open_holdable_cursors の値÷16↑×4 BES で，pd_utl_exec_mode の値=1，かつ s ＞ 32 の場合： 48 ＋（(p_x ＋ 3）×2 ＋ s）×↑pd_max_open_holdable_cursors の値÷16↑×4 BES で，pd_utl_exec_mode の値=0，又は s≦32 の場合： 48 ＋（(p_x ＋ 3）×2 ＋ 32）×↑pd_max_open_holdable_cursors の値÷16↑×4 DS で，pd_utl_exec_mode の値=1，かつ s ＞ 16 の場合： 48 ＋（(p_x ＋ 3）×2 ＋ s）×↑pd_max_open_holdable_cursors の値÷16↑×4 DS で，pd_utl_exec_mode の値=0，又は s≦16 の場合： 48 ＋（(p_x ＋ 3）×2 ＋ 16）×↑pd_max_open_holdable_cursors の値÷16↑×4 u_x：FES で，pd_fes_lck_pool_size が指定されていない場合： 48 ＋（↓↓（p_x ＋ 3）×（pd_max_access_tables の値＋ 4）÷4↓÷ pd_fes_lck_pool_partition の値↓）×8 ×↑pd_max_open_holdable_cursors の値÷16↑×4 FES で，pd_fes_lck_pool_size が指定されている場合： 48 ＋（↓pd_fes_lck_pool_size の値÷pd_fes_lck_pool_partition の値↓×5 ＋↓↓pd_fes_lck_pool_size の値÷pd_fes_lck_pool_partition の値↓÷3↓） を偶数に切り上げた値×↑pd_max_open_holdable_cursors の値÷16↑×4 BES で，pd_utl_exec_mode の値=1，かつ s ＞ 32 の場合： 48 ＋（↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×5 ＋↓↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓÷3↓

プロセスの種類	共用メモリの計算式（単位：バイト）
	＋（p_x ＋ 3）×2×2×5 ＋ s×（pd_max_rdarea_no の値＋ 1）） を偶数に切り上げた値×↑pd_max_open_holdable_cursors の値÷16↑×4 BES で，pd_utl_exec_mode の値=0，又は s≦32 の場合： 48 ＋（↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×5 ＋↓↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓÷3↓ ＋（p_x ＋ 3）×2×2×5 ＋ 32×（pd_max_rdarea_no の値＋ 1）） を偶数に切り上げた値×↑pd_max_open_holdable_cursors の値÷16↑×4 DS で，pd_utl_exec_mode の値=1，かつ s ＞ 16 の場合： 48 ＋（↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×5 ＋↓↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓÷3↓ ＋（p_x ＋ 3）×2×2×5 ＋ s ＋ 4）を偶数に切り上げた値 ×↑pd_max_open_holdable_cursors の値÷16↑×4 DS で，pd_utl_exec_mode の値=0，又は s≦16 の場合： 48 ＋（↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓×5 ＋↓↓pd_lck_pool_size の値÷pd_lck_pool_partition の値↓÷3↓ ＋（p_x ＋ 3）×2×2×5 ＋ 20）を偶数に切り上げた値 ×↑pd_max_open_holdable_cursors の値÷16↑×4 **●ユニット内にサーバ（FES，BES，又は DS）がない場合** 16704
トランザクション サーバ	304 ＋ 32×B ＋ 192×s×2 **ユニット内に FES がある場合に加算します※** ＋ 1048 ＋（416 ＋ 800 ＋ 256 ＋ 392×2）×（A×2 ＋ 7）＋ 256×2 ＋ 128×（システム内 BES 数×4 ＋システム内 DS 数×2 ＋システム内 FES 数）×（A×2 ＋ 7） ＋ C **ユニット内に BES がある場合に加算します※** ＋ 1048 ＋（416 ＋ 800 ＋ 256 ＋ 392×2）×（u×2 ＋ 7） ＋ 256×2 ＋ 128×（システム内 BES 数×4 ＋システム内 DS 数×2 ＋システム内 FES 数）×（A×2 ＋ 7） ＋ D **ユニット内に DS がある場合に加算します※** ＋ 1048 ＋（416 ＋ 800 ＋ 256 ＋ 392×2）×（t×2 ＋ 7） ＋ 256×2 ＋ 128×（システム内 BES 数×4 ＋システム内 DS 数×2 ＋システム内 FES 数）×（A×2 ＋ 7） ＋ E s：pd_max_users の値＋ pd_max_reflect_process_count の値 t：pd_max_dic_process の値＋ pd_max_reflect_process_count の値 u：pd_max_bes_process の値＋ pd_max_reflect_process_count の値 A：マルチ FES の場合は s ＋ 1，マルチ FES でない場合は s B：影響分散スタンバイレス型系切り替えの対象となるユニットの場合：

プロセスの種類	共用メモリの計算式（単位：バイト）
	ホスト BES 数＋ pd_ha_max_act_guest_servers オペランドの値 上記以外のユニットの場合：ユニット内サーバ数 C：≪条件≫に示すどちらかの条件に一致する場合： 128×（システム内 BES 数×4 ＋システム内 DS 数×2 ＋システム内 FES 数）×（A×2 ＋ 7） ≪条件≫に示すどちらの条件にも一致しない場合：0 D：≪条件≫に示すどちらかの条件に一致する場合： 128×（システム内 BES 数×4 ＋システム内 DS 数×2 ＋システム内 FES 数）×（u×2 ＋ 7） ≪条件≫に示すどちらの条件にも一致しない場合：0 E：≪条件≫に示すどちらかの条件に一致する場合： 128×（システム内 BES 数×4 ＋システム内 DS 数×2 ＋システム内 FES 数）×（t×2 ＋ 7） ≪条件≫に示すどちらの条件にも一致しない場合：0 ≪条件≫ • pd_rpl_reflect_mode オペランドに uap を指定している • システム内に，pdstart -k stls オペランドを指定したフロントエンドサーバが存在する 注※　上記の計算式で算出した値を各サーバの数分だけ加算します。 1：1 スタンバイレス型系切り替え適用ユニットの場合： 正規 BES 数＋代替 BES 数 上記以外の場合はユニット内サーバ数
タイマサーバ	32×（pd_max_users の値＋ pd_max_reflect_process_count の値＋ 3） ×（システム内の BES 数＋ 1 ＋ユニット内ユティリティサーバ数＋ 1） ＋ 1440 ＋（48－32）×2 ユニット内ユティリティサーバ数は 23 ＋ α α：ユニット内に MGR がある場合：2 ユニット内に FES がある場合：3 ユニット内に DS がある場合：7 ユニット内に BES がある場合：BES 数×b b：AIX 版の場合：15 Linux 版の場合：6
統計ログサーバ	424 ＋ 128×16 ＋ 32 ＋ 288×2 ＋ 1168 ＋ 144×3 ＋ pd_stj_buff_size の値×1024×3 ＋ 64 ＋ 4096 ＋ 8192
プロセスサーバ	208 ＋ 528×MAX（c，256）＋ 96 ＋ 256 ＋（pd_max_server_process の値＋ 50）×（256 ＋ 160）＋ 16 ＋ 8×34 ＋ 16 ＋ 16 ＋ 64 ＋ 64×（k ＋ 1） c：↑（51 ＋ d ＋ e ＋ f ＋（g×ユニット内の BES 数※）＋ h ＋ i）÷16↑×16 d：ユニット内に MGR がある場合は 59，ない場合は j e：ユニット内に DS がある場合は 17，ない場合は 0 f：ユニット内に FES がある場合は 11，ない場合は 0 g：ユニット内に BES がある場合は 25，ない場合は 0 h：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 9，使用しない場合は 0

プロセスの種類	共用メモリの計算式（単位：バイト）
	i：影響分散スタンバイレス型系切り替えの対象となるユニットの場合は 1，対象でないユニットの場合は 0 j：pd_mlg_msg_log_unit オペランドに manager を指定しているときは 1，local を指定しているときは 2 k：システム共通定義，又はユニット制御情報定義に pd_module_trace_max オペランドを指定している場合は pd_module_trace_max の値，指定していない場合は 16383 m：ユニット内に MGR がある場合は 0，ユニット内に MGR がない場合は 1 注※ 1：1 スタンバイレス型系切り替えの対象となっている場合は（BES 数×2）となります。影響分散スタンバイレス型系切り替えの対象となるユニットの場合は pd_ha_max_act_guest_servers オペランドの補正値を含みます。
システムマネジャ	1024 ＋（48 ＋ 8）×（g ＋ h ＋ i）＋（108 ＋ 8）×｛(p ＋ q ＋ 3）＋ u×（15 ＋ 1)｝＋（104 ＋ 8）×c ＋ 40×（k ＋ m ＋ n×o ＋ u）×14 ＋ 256×m ＋ 128×c ＋ 40×d ＋ 16×e ＋ 96×o ＋ v×（16×35×（k ＋ u）＋ 15 ＋ 44×z ＋ 15）＋ w×（48×B ＋ 15 ＋ 4×z ＋ 15 ＋ 4×y ＋ 15）＋ v×（144 ＋ 15）＋ 8 ＋ 13096 ＋ 5856×p ＋ s ＋ s×o ＋ 16 ＋ 96×o ＋ 1024 ＋ 272×A c：ユニット数 d：pdunit オペランドの-c オプション指定数 e：pdcltgrp オペランド指定数 g：システム内の FES 数 h：システム内の BES 数 i：システム内の DS 数 j：ユニット内の FES 数 k：ユニット内の BES 数 m：ユニット内の DS 数 n：ユニット内の代替 BES 数 o：ユニットがスタンバイレス型系切り替えの対象となっている場合は 1， 対象となっていない場合は 0 p：i ＋ k ＋ m ＋ n q：24 ＋ t ＋ j×3 ＋ k×15 ＋ m×7 r：14×（k ＋ m ＋ u）＋ p ＋ q ＋ u×15 ＋ 2 ＋ 38 ＋ 5 ＋ p×4 s：320 ＋ 2052 ＋ 148×（r ＋ 3）＋ v×（40×（k ＋ u）＋ 72×（k ＋ u）） t：ユニット内に MGR がある場合は 2，ない場合は 0 u：受け入れ可能なゲスト BES 数（pd_ha_max_act_guest_servers オペランドの値） v：ユニットが影響分散スタンバイレス型系切り替えの対象となっている場合 1， 対象となっていない場合は 0 w：影響分散スタンバイレス型系切り替えの対象となっているユニットがある場合は 1， ない場合は 0 y：HA グループ内のユニット数の総和 z：HA グループ内のサーバ数の総和 A：pd_security_host_group オペランドに指定したホストに対応する IP アドレスの数 pd_security_host_group オペランドを指定していない場合は 0 B：システム共通定義に指定している pdhagroup オペランドの数
ネームサーバ	MAX｛65536，（X ＋ Y ＋ Z)｝＋ MAX（16384，L）＋ M

プロセスの種類	共用メモリの計算式（単位：バイト）
	X：↑（272 ＋ 16 ＋ 156×ユニット数＋ 16 ＋ 16×126）÷1024↑×1024 Y：8192 Z：↑（264×（Z_2 ＋ Z_3 ＋ j ＋ 32））÷1024↑×1024 Z_2：b ＋ 10 ＋ c ＋ 11×自ユニット内の HiRDB サーバ数＋ d ＋ e Z_3：f ＋ 7 ＋ g ＋ 4×自ユニット内の HiRDB サーバ数＋ h ＋ i L：↑（224×（L_2 ＋ L3 ＋ L4））÷1024↑×1024 L_2：k ＋ 2×システム内のユニット数 L_3：システム内の BES 数＋システム内の FES 数＋システム内の DS 数 L_4：15×システム内の HiRDB サーバ数 M：ユニット内の HiRDB サーバ数＋ z ＋ m×ユニット内システムサーバ数 b：ユニットに MGR がある場合：3 ユニットに MGR がない場合：0 c：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：2 それ以外のユニットの場合：0 d：影響分散スタンバイレス型系切り替えの対象となるユニットの場合： 11×受け入れ可能なゲスト BES 数 それ以外のユニットの場合：0 e：1：1 スタンバイレス型系切り替えの対象となるユニットの場合： 6×自ユニット内の HiRDB サーバ数 それ以外のユニットの場合：0 f：ユニットに MGR がある場合：3 ユニットに MGR がない場合：0 g：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：2 それ以外のユニットの場合：0 h：影響分散スタンバイレス型系切り替えの対象となるユニットの場合： 3×受け入れ可能なゲスト BES 数 それ以外のユニットの場合：0 i：1：1 スタンバイレス型系切り替えの対象となるユニットの場合： 4×自ユニット内の HiRDB サーバ数 それ以外のユニットの場合：0 j：ユニット内のサーバ数＋ β ＋ γ ＋ユニット内ユティリティサーバ数＋ 2 ユニット内ユティリティサーバ数：23 ＋ α k：ユニットに MGR がある場合：3 ユニットに MGR がない場合：0 m：38 ＋ n ＋ o n：ユニットに MGR があり，かつユニット数が 2 以上の場合：5 ユニットに MGR があり，かつユニット数が 1 の場合：4 ユニットに MGR がなく，かつ pd_mlg_msg_log_unit の値が local の場合：4 ユニットに MGR がなく，かつ pd_mlg_msg_log_unit の値が manager の場合：3 o：4×ユニット内の HiRDB サーバ数 z：23 ＋ α α：ユニット内に MGR がある場合：3

プロセスの種類	共用メモリの計算式（単位：バイト）
	ユニット内に FES がある場合：3 ユニット内に DS がある場合：7 ユニット内に BES がある場合：（BES 数＋ β）×15 β：影響分散スタンバイレス型系切り替えの対象となるユニットの場合： 受け入れ可能なゲスト BES 数※ 上記以外のユニットの場合：0 γ：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：代替 BES 数 上記以外のユニットの場合：0 注※　pd_ha_max_guest_act_servers の値
ノードマネジャ	**ユニット内に MGR がある場合** ↑（1312 ＋ 464×システムの全ユニット数＋ 96×システムの全サーバ数 ＋ 10240 ＋ 1200 ＋ 72×C ＋ 240×A ＋ 44×A ＋ 28×A ＋ 16×B ＋ 16×システムの全 BES 数＋ 8×システムの全ユニット数 ＋ 64×システムの全サーバ数＋ 32×システムの全サーバ数＋ 48 ＋ J ＋ 16×K） ÷1024↑×1024 **ユニット内に MGR がない場合** ↑（1200 ＋ 72×C ＋（240×A ＋ 44×A ＋ 28×A）×F ＋ 16×B ＋ 16×システムの全 BES 数＋ 8×システムの全ユニット数 ＋ 64×システムの全サーバ数＋ 32×システムの全サーバ数＋ 48） ÷1024↑×1024 A：pd_utl_exec_mode=0 の場合：1024 pd_utl_exec_mode=1 の場合：（pd_max_users の値＋ pd_max_reflect_process_count の値）×システムの全 BES 数×3 ユニットに MGR がある場合は次に示す値を加算：（pd_max_users の値＋ pd_max_reflect_process_count の値）×4 ＋ 200 ユニットに DS がある場合は次に示す値を加算：（pd_max_users の値＋ pd_max_reflect_process_count の値）×3 ＋ 200 ユニットに BES がある場合は次に示す値を加算：（pd_max_users の値＋ pd_max_reflect_process_count の値）×D 上記の計算式で算出した A の値が 1024 を超えない場合は，A を 1024 に置き換えてください。 B：pdcltgrp オペランドを指定しない場合：0 pdcltgrp オペランドを指定する場合：pdcltgrp 指定数＋ 1 C：ユニット内サーバ数＋ E D：ユニット内 BES 数＋ E E：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：ユニット内の代替 BES 数 影響分散スタンバイレス型系切り替えの対象となるユニットの場合：受け入れ可能なゲスト BES 数 上記以外のユニットの場合：0 F：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：2 上記以外のユニットの場合：1 J：pd_system_expand_unit オペランドを指定した場合：16 pd_system_expand_unit オペランドを省略した場合：0

プロセスの種類	共用メモリの計算式（単位：バイト）
	K：pd_system_expand_unit オペランドを指定した場合：pd_system_expand_unit オペランドに指定したユニット数 pd_system_expand_unit オペランドを省略した場合：0
I/O サーバ	↑（56 ＋（↑（56 ＋ A）÷32↑×32））÷128↑×128 **影響分散スタンバイレス型系切り替え機能の対象となるユニットではない場合** A：34268 ＋ pd_max_utl_ios_file_no の値×296 ＋ 158064※1 ＋ {(162800 ＋ pd_max_file_no の値×1024）×BES 数} ※2 ＋ {162800 ＋ pd_max_file_no の値×1024} ※3 注※1　FES がある場合に加算します。 注※2　BES がある場合に加算します。 注※3　DS がある場合に加算します。 なお，1：1 スタンバイレス型系切り替え構成対象ユニットの場合には，上記の式で求めた値を 2 倍にします。 **影響分散スタンバイレス型系切り替え機能の対象となるユニットの場合** A：↑64 ＋ 24×BES 数※4÷16↑×16 ＋↑（177952 ＋ pd_max_file_no の値×1024）÷16↑×16×BES 数※4 ＋ 34064 注※4　pd_ha_max_act_guest_servers の値を含みます。
ログサーバ	32 ＋ 48 ＋ 128×37 ＋ { 432 ＋ 128×7 ＋ 1168 ＋ 512 ＋↑（128 ＋ 256 ＋ 160 ＋ 8 ＋ 64）÷pd_log_rec_leng の値※↑ ×pd_log_rec_leng の値※ ＋ 64 ＋ 4096×2 ＋（768 ＋ 512）×B ＋↑ {(B ＋ 1）÷12} ↑×8320 ＋ 144×pd_log_write_buff_count の値※ ＋（pd_log_write_buff_count の値※＋ A） ×↑ {pd_log_max_data_size の値※＋（68 ＋ 44 ＋ 96 ＋ 160)} ÷4096↑ ×4096 ＋ C } ×ユニット内サーバ数＋ D ＋ 128×ユニット内 FES 数 **pd_max_reflect_process_count オペランドを指定する場合に加算します。** (128 ＋ 704）×（ユニット内 BES 数＋ D) A：16 B：pdlogadfg -d sys オペランド指定数※ C：0 D：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：ユニット内の代替 BES 数 影響分散スタンバイレス型系切り替えの対象となるユニットの場合： pd_ha_max_act_guest_servers オペランドの補正値 注※ ユニット内の全サーバの指定値で，最大値を指します。1：1 スタンバイレス型系切り替えの対象となるユニットの場合はユニット内の全サーバとユニット内で動作する代替 BES の指定値で，最

プロセスの種類	共用メモリの計算式（単位：バイト）
	大値を指します。影響分散スタンバイレス型系切り替えの対象となるユニットの場合はユニット内の全サーバと HA グループに含まれる全 BES の指定値で，最大値を指します。
シンクポイントダンプサーバ	{ ↑（384 ＋ 1536×2）÷1024↑×1024 ＋↑ {(128 ＋ 80 ＋ 240 ＋ 240）＋ 192×（pdlogadfg -d spd の指定数※） ＋ 416×（pdlogadpf -d spd の指定数※)} ÷1024↑×1024 } ×（全サーバ数＋ A） A：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：ユニット内の代替 BES 数 影響分散スタンバイレス型系切り替えの対象となるユニットの場合： pd_ha_max_act_guest_servers オペランドの補正値 注※ ユニット内の全サーバの指定値で，最大値を指します。1：1 スタンバイレス型系切り替えの対象となるユニットの場合はユニット内の全サーバとユニット内で動作する代替 BES の指定値で，最大値を指します。影響分散スタンバイレス型系切り替えの対象となるユニットの場合はユニット内の全サーバと HA グループに含まれる全 BES の指定値で，最大値を指します。
ユニット共通	a ＋ {b ＋ 80 ＋（s ＋ 3）×c ＋ 64 ＋ 48 ＋ d ＋ e} ×（ユニット内の FES，BES，DS の合計数＋ i） ＋（g×（ユニット内の BES，DS の合計数＋ i））＋ f ＋（pd_max_server_process の値×2 ＋ 100）×（64 ＋ 16）＋ 32 ＋（pd_max_server_process の値×2 ＋ 100 ＋ 384）×32 ＋ 32 ＋ h ＋ j ＋ v ＋ w ＋（pd_max_server_process の値＋ 127 ＋ y）×48 ＋ 32 ＋ A **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に加算します。** (↑（56 ＋(↑（56 ＋ 88560）÷32↑×32)）÷128↑×128) a：35408 b：3480 c：3608 d：48×32 e：80 ＋ 96×｛(s ＋ 3）×2 ＋ MAX（5，↓［s ＋ 3］÷10↓）＋ 7} f：512×（13 ＋（ユニット内の FES，BES，DS の合計数＋ i）×3）×2 g：{↑（128 ＋ pd_lck_until_disconnect_cnt の値×112）÷4096↑} ×4096×2 h：↑（pd_registered_port で指定したポート番号の数×16 ＋ 32） ÷1024↑×1024 pd_registered_port を指定していない場合は 0 i：1：1 スタンバイレス型系切り替えの場合は代替 BES 数 影響分散スタンバイレス型系切り替えの場合は pd_ha_max_guest_servers 補正値 j：p×（ユニット内の FES 数）＋ q×（ユニット内の BES 数＋ i）＋ r×（ユニット内の DS 数） k：FES の pd_fes_lck_pool_partition の値 m：BES の pd_lck_pool_partition の値 n：DS の pd_lck_pool_partition の値 o：(s ＋ 3）×2 ＋ MAX｛5，↓（s ＋ 3）÷10↓｝＋ 7

プロセスの種類	共用メモリの計算式（単位：バイト）
	p：k が 2 以上の場合は 32 ＋（8 ＋ 16×k）×o それ以外の場合は 0 q：m が 2 以上の場合は 32 ＋（8 ＋ 16×m）×o それ以外の場合は 0 r：n が 2 以上の場合は 32 ＋（8 ＋ 16×n）×o それ以外の場合は 0 s：pd_max_users の値＋ pd_max_reflect_process_count の値 t：pd_max_dic_process の値＋ pd_max_reflect_process_count の値 u：pd_max_bes_process の値＋ pd_max_reflect_process_count の値 s は，DS の場合は t，BES の場合は u，pd_max_dic_process 又は pd_max_bes_process を省略する場合は s とします。 v：pd_dbbuff_modify に Y を指定し，かつユニット内に BES 又は DS がある場合：69648 ＋（(BES 定義又は DS 定義の pd_max_add_dbbuff_shm_no の値のユニット内最大値）＋ pd_max_resident_rdarea_shm_no 値）×136 pd_dbbuff_modify に N を指定し，かつユニット内に BES 又は DS がある場合：69648 ＋ pd_max_resident_rdarea_shm_no 値×136 上記以外：2192 w：144 y：次に示す計算式で算出した値を，各サーバの数分だけ加算します。 ユニット内に FES がある場合に加算します。 （z ＋ 3）×2 ユニット内に BES があり※1，pd_utl_exec_mode=1 かつ s ＞ 32 の場合に加算します。 （z ＋ 3）×34 ＋ s ユニット内に BES があり※1，pd_utl_exec_mode=0 又は s≦32 の場合に加算します。 （z ＋ 3）×34 ＋ 32 ユニット内に DS があり，pd_utl_exec_mode=1 かつ s ＞ 16 の場合に加算します。 （z ＋ 3）×34 ＋ s ユニット内に DS があり，pd_utl_exec_mode=0 又は s≦16 の場合に加算します。 （z ＋ 3）×34 ＋ 16 z：FES で，HiRDB システム内の FES 数＞ 1 の場合：s ＋ 1 FES で，HiRDB システム内の FES 数=1 の場合：s BES で，s ＞ pd_max_bes_process の値※2 の場合：s BES で，s≦pd_max_bes_process の値※2 の場合：u DS で，s ＞ pd_max_dic_process の値の場合：s DS で，s≦pd_max_dic_process の値の場合：t A：640 注※1 1：1 スタンバイレス型系切り替えの対象となるユニットの場合，ユニット内の全正規 BES と全代替 BES が対象となります。 影響分散スタンバイレス型系切り替えの対象となるユニットの場合，ユニット内の全ホスト BES と全ゲスト BES が対象となります。

プロセスの種類	共用メモリの計算式（単位：バイト）
	注※2 影響分散スタンバイレス型系切り替えとなるユニットで，ゲスト BES についての計算をする場合，そのゲスト BES のオペランドの値ではなく，そのユニット内の全ゲスト BES に指定されている値の最大値を計算に使用してください。
トランザクションログサーバ	{1168 ＋ 688×A} ×（ユニット内サーバ数＋ H） ＋ { 128×B ＋ 144 ＋[G ＋ ↑（128 ＋ 256 ＋ 8 ＋ 224）÷pd_log_rec_leng の値※ ↑ ×pd_log_rec_leng の値※ ＋ ↑（pd_log_max_data_size の値※＋ 68 ＋ 44 ＋ 96 ＋ 160） ÷pd_log_rec_leng の値※ ↑ ×pd_log_rec_leng の値※] ×D ＋ E ＋（48 ＋ 8）×B×2 } ×（ユニット内の BES 及び DS 数＋ H） ＋ { 600×B ＋ 128×B ＋ 64×B×C ＋ 144 ＋ G ＋ ↑（128 ＋ 256 ＋ 8 ＋ 224）÷pd_log_rec_leng の値※ ↑ ×pd_log_rec_leng の値※ ＋ ↑（pd_log_max_data_size の値※＋ 68 ＋ 44 ＋ 96 ＋ 160） ÷pd_log_rec_leng の値※ ↑ ×pd_log_rec_leng の値※ ＋ E ＋（48 ＋ 8）×（B×2 ＋ 2） } ×（ユニット内サーバ数＋ H） A：2 B：7 ＋ J×2 C：システム全体の BES 数×4 ＋システム全体の DS 数×2 ＋システム全体の FES 数 D：pd_log_rollback_buff_count の値が 0 の場合：8 それ以外の場合：pd_log_rollback_buff_count の値 E：Linux 版の場合：4096 Linux 版以外の場合：0 G：64 H：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：ユニット内の代替 BES 数 影響分散スタンバイレス型系切り替えの対象となるユニットの場合： pd_ha_max_act_guest_servers オペランドの補正値 J：ユニット内サーバ中の，s，t，及び u の最大値 s：pd_max_users の値＋ pd_max_reflect_process_count の値 t：pd_max_dic_process の値＋ pd_max_reflect_process_count の値 u：pd_max_bes_process の値＋ pd_max_reflect_process_count の値 注※ ユニット内の全サーバの指定値で，最大値を指します。1：1 スタンバイレス型系切り替えの対象となるユニットの場合はユニット内の全サーバとユニット内で動作する代替 BES の指定値で，最大値を指します。影響分散スタンバイレス型系切り替えの対象となるユニットの場合はユニット内の全サーバと HA グループに含まれる全 BES の指定値で，最大値を指します。
ステータスサーバ	↑64÷32↑ ×32×（ユニット内サーバ数＋ A） A：1：1 スタンバイレス型系切り替えの対象となるユニットの場合：ユニット内の代替 BES 数

プロセスの種類	共用メモリの計算式（単位：バイト）
	影響分散スタンバイレス型系切り替えの対象となるユニットの場合： pd_ha_max_act_guest_servers オペランドの補正値
監査証跡管理サーバ	↑A÷1024↑×1024 A：pd_aud_file_name オペランドの指定がない場合は 704 pd_aud_file_name オペランドの指定がある場合は 704 ＋（320×200）＋ B ＋ C B：pd_aud_async_buff_size オペランドの値が 0 の場合は 0 pd_aud_async_buff_size オペランドの値が 4096 以上の場合は次の計算値 Linux 版の場合： （176×pd_aud_async_buff_count オペランドの値） ＋｛(↑pd_aud_async_buff_size オペランドの値÷4096↑×4096) ×pd_aud_async_buff_count オペランドの値｝＋ 4096 Linux 版以外の場合： （176×pd_aud_async_buff_count オペランドの値） ＋｛(↑pd_aud_async_buff_size オペランドの値÷4096↑×4096) ×pd_aud_async_buff_count オペランドの値｝ C：pd_aud_auto_loading オペランドに Y を指定し，かつ MGR があるユニットの場合は 256×（システム全体のユニット数＋ 1）＋ 256 上記以外の場合は 0 スタンバイレス型系切り替え機能を適用するユニットの場合，自ユニットのメモリ使用量に，系切り替え先となるユニットのセキュリティ監査のメモリ使用量を加算する必要があります。

14.2.4　各サーバが使用する共用メモリの計算式

(1)　フロントエンドサーバが使用する共用メモリの計算式

フロントエンドサーバが使用する共用メモリの計算式を次に示します。計算式で使用している変数については，「計算式で使用する変数」を参照してください。

●32 ビットモードの場合

```
40＋b×1.3＋c＋d＋k＋1.6＋x＋y＋4
＋ {[ (a＋12) ÷13]×1.1＋[ (a＋62) ÷63]＋3.7} × (e＋3)
＋ {
    ↑↑b÷64↑× (8÷16) ↑×4×4
    ＋12× { (b÷3) ＋1-mod (b÷3, 2) }
    ＋4×a× { (e＋3) ×2＋1＋MAX (e÷10, 5) }
    ＋32＋4＋16＋ {56× (f＋1) ×g} ＋20000
    ＋↑ { (c÷8) ＋7} ÷64↑×8＋↑ { (k÷8) ＋7} ÷64↑×8
    ＋MAX {a× (e＋3) , c÷8} ×104＋MAX {a× (e＋3) , k÷8} ×24
    ＋↑ { (x÷4) ＋7} ÷64↑×8
    ＋↑ {[ (y- (s×592＋t×916＋u×172) ) ÷2]＋7} ÷64↑×8
    ＋MAX {13× (e＋3) , x÷4} ×88
    ＋60×MAX {21× (e＋3) ,  (y- (s×592＋t×916＋u×172) ) ÷2}
    ＋44＋256＋1024
  } ÷1024＋A＋7
```

```
   I
＋Σ（Ji）
  i=1
＋6.5×1024×f
＋1024
●pd_def_buf_control_area_assignオペランドにINITIALを指定するか，又はこのオペランドを省略した場合に加算します。
＋｛［（a＋12）÷13］×1.1＋［（a＋62）÷63］＋3.7｝×（e＋7）
                                                    （単位：キロバイト）
```

●64 ビットモードの場合

```
40＋b×1.3＋c＋d＋k＋1.6＋x＋y＋5
＋｛[（a＋12）÷13]×1.2＋[（a＋62）÷63]×1.5＋4.1｝×（e＋3）
＋｛
    ↑↑b÷64↑×（8÷16）↑×4×4
    ＋12×｛（b÷3）＋1-mod（b÷3，2）｝
    ＋4×a×｛（e＋3）×2＋1＋MAX（e÷10，5）｝
    ＋48＋8＋16＋｛72×（f＋1）×g｝＋20000
    ＋↑｛（c÷8）＋7｝÷64↑×8＋↑｛（k÷8）＋7｝÷64↑×8
    ＋MAX｛a×（e＋3），c÷8｝×104＋MAX｛a×（e＋3），k÷8｝×40
    ＋↑｛（x÷4）＋7｝÷64↑×8
    ＋↑｛[（y-（s×600＋t×936＋u×184））÷2]＋7｝÷64↑×8
    ＋MAX｛13×（e＋3），x÷4｝×104
    ＋72×MAX｛21×（e＋3），（y-（s×600＋t×936＋u×184））÷2｝
    ＋72＋256＋1536
  ｝÷1024＋A＋7
   I
＋Σ（Ji）
  i=1
＋6.5×1024×f
＋1024
●pd_def_buf_control_area_assignオペランドにINITIALを指定するか，又はこのオペランドを省略した場合に加算します。
＋｛［（a＋12）÷13］×1.2＋［（a＋62）÷63］×1.5＋4.1｝×（e＋7）
                                                    （単位：キロバイト）
```

(2)　ディクショナリサーバが使用する共用メモリの計算式

ディクショナリサーバが使用する共用メモリの計算式を次に示します。

32 ビットモードの場合（単位：キロバイト）

計算式 1 ＋↑｛(↑（40 ＋（計算式 2～計算式 5 を加算した値））÷512↑×512)｝÷1024↑

64 ビットモードの場合（単位：キロバイト）

計算式 1 ＋↑｛(↑（72 ＋（計算式 2～計算式 5 を加算した値））÷512↑×512)｝÷1024↑

計算式で使用している変数については，「計算式で使用する変数」を参照してください。

注意事項

- pd_dbsync_point 又は pd_system_dbsync_point オペランドのどちらかの指定が commit の場合に計算式 3 を加算します。pd_system_dbsync_point オペランドの省略値は commit です。

  上記以外の場合，計算式 5 を加算します。

- pd_dfw_awt_process オペランドを指定する場合に計算式 4 を加算します。
- pd_dic_shmpool_size オペランドを省略すると，次の値が設定されます。
  32 ビットモードの場合：
  ↑ {(↑（40 ＋（計算式 2～計算式 5 の合計値））÷512↑×512)} ÷1024↑
  64 ビットモードの場合：
  ↑ {(↑（72 ＋（計算式 2～計算式 5 の合計値））÷512↑×512)} ÷1024↑

条件	共用メモリの計算式

計算式 1
（単位：キロバイト）
	●32 ビットモードの場合
	b×1.3
	＋ {
	↑↑b÷64↑×（8÷16）↑×4×4
	＋12×｛(b÷3）＋1−mod（b÷3，2)｝
	＋8×a×｛(e＋3）×2＋1＋MAX（e÷10，5)｝
	＋512
	｝÷1024
	＋3.5＋↑（224×v×w）÷1024↑＋0.5
	＋↑ {
	↑（28＋（↑（32＋（(↑g÷127↑＋1）×2048＋128））÷32↑×32))
	÷128↑×128
	｝÷1024↑
	K
	＋ Σ（Li）
	i=1
	●64 ビットモードの場合
	b×1.3
	＋ {
	↑↑b÷64↑×（8÷16）↑×4×4
	＋12×｛(b÷3）＋1−mod（b÷3，2)｝
	＋8×a×｛(e＋3）×2＋1＋MAX（e÷10，5)｝
	＋1024
	｝÷1024
	＋3.5＋↑（224×v×w）÷1024↑＋0.5
	＋↑ {
	↑（56＋（↑（56＋（(↑g÷127↑＋1）×2048＋128））÷32↑×32))
	÷128↑×128
	｝÷1024↑
	K
	＋ Σ（Li）
	i=1

計算式 2 （単位：バイト）	**●32 ビットモードの場合** 500×1024 ＋（↑436×g÷16↑×16）＋496×h＋112×240

条件	共用メモリの計算式
	＋5072×（e＋15）＋96×z ＋32×m＋172×｛a×（e＋3）＋（e＋3）×2＋22｝＋16 ＋48×p＋36×｛(e＋3）×2＋1＋MAX（5，[e＋3]÷10)｝ ＋68×G＋152×120＋80＋64※1＋368※2 ＋（(↓（↑（g÷8）↑＋3）÷4↓）×4）×m ●64 ビットモードの場合 500×1024 ＋（↑536×g÷16↑×16）＋512×h ＋（↑136×240÷16↑×16） ＋9424×（e＋15）＋144×z ＋48×m＋336×｛a×（e＋3）＋（e＋3）×2＋22｝＋16 ＋64×p＋72×｛(e＋3）×2＋1＋MAX（5，[e＋3]÷10)｝ ＋68×G＋184×120＋96＋64※1＋448※2 ＋（(↓（↑（g÷8）↑＋7）÷8↓）×8）×m
計算式 3 （単位：バイト）	●32 ビットモードの場合 （32＋16×z）×（G＋1）＋16 ●64 ビットモードの場合 （48＋32×z）×（G＋1）＋16
計算式 4 （単位：バイト）	●32 ビットモードの場合 72＋52×H＋68×z pd_dbbuff_trace_level オペランドに 1 を指定する場合に加算します。 ＋320×z ●64 ビットモードの場合 96＋56×H＋72×z pd_dbbuff_trace_level オペランドに 1 を指定する場合に加算します。 ＋640×z
計算式 5 （単位：バイト）	●32 ビットモードの場合 （32＋16×z）×10＋16 ●64 ビットモードの場合 （48＋32×z）×10＋16

注※1

pd_max_ard_process オペランドが 1 以上の場合に加算します。

注※2

再編成時期予測機能を使用する場合に加算します。

(3) バックエンドサーバが使用する共用メモリの計算式

バックエンドサーバが使用する共用メモリの計算式を次に示します。

32 ビットモードの場合（単位：キロバイト）

計算式 1 ＋↑｛(↑（40 ＋（計算式 2～計算式 7 を加算した値））÷512↑×512)｝÷1024↑

64 ビットモードの場合（単位：キロバイト）

計算式 1 ＋↑｛(↑（72 ＋（計算式 2～計算式 9 を加算した値））÷512↑×512)｝÷1024↑

計算式で使用している変数については，「計算式で使用する変数」を参照してください。

計算式 1～9 についての注意事項

- 次に示すどれかの条件を満たす場合に計算式 3 を加算します。
  - ・pd_rdarea_open_attribute_use オペランドに Y を指定する場合
  - ・pd_lv_mirror_use オペランドに Y を指定する場合
  - ・高速系切り替え機能を使用する場合
- 次に示すどれかの条件を満たす場合に計算式 4 を加算します。
  - ・pd_dbsync_point オペランドに commit を指定する場合
  - ・pd_shared_rdarea_use オペランドに Y を指定する場合
  - ・上記以外の場合，計算式 7 を加算します。
- pd_inner_replica_control オペランドを指定する場合に計算式 5 を加算します。
- pd_dfw_awt_process オペランドを指定する場合に計算式 6 を加算します。
- pd_bes_shmpool_size オペランドを省略すると，次の値が設定されます。
  32 ビットモードの場合：
  ↑｛(↑（40 ＋（計算式 2～計算式 7 の合計値））÷512↑×512)｝÷1024↑
  64 ビットモードの場合：
  ↑｛(↑（72 ＋（計算式 2～計算式 9 の合計値））÷512↑×512)｝÷1024↑
- pd_max_resident_rdarea_no オペランドを指定する場合に計算式 8 を加算します。
- pd_max_temporary_object_no の値が 1 以上の場合に計算式 9 を加算します。

条件	共用メモリの計算式
計算式 1 （単位：キロバイト）	**●32 ビットモードの場合** b×1.3 ＋｛ ↑↑b÷64↑×（8÷16）↑×4×4 ＋12×｛(b÷3）＋1－mod（b÷3，2)｝ ＋8×a×｛(e＋3）×2＋1＋MAX（e÷10，5)｝＋512＋512※1 ｝÷1024 ＋↑｛72＋8×v×（8＋3×w)｝÷1024↑ ＋↑｛(↑g÷127↑＋1）×2048＋128｝÷1024↑ ＋↑｛ ↑（28＋（↑（32＋（(↑g÷127↑＋1）×2048＋128))÷32↑×32)) ÷128↑×128｝

条件	共用メモリの計算式

	} ÷1024↑
	M
	＋ Σ（Ni）
	i=1
	●64 ビットモードの場合
	b×1.3
	＋ {
	↑↑b÷64↑×（8÷16）↑×4×4
	＋ 12×｛(b÷3) ＋ 1－mod（b÷3，2)｝
	＋ 8×a×｛(e ＋ 3) ×2 ＋ 1 ＋ MAX（e÷10，5)｝＋ 1024 ＋ 512※1
	} ÷1024
	＋↑｛72 ＋ 24×v×（2 ＋ w)｝÷1024↑
	＋↑ {
	↑（56 ＋（↑（56 ＋（(↑g÷127↑＋ 1）×2048 ＋ 128)）÷32↑×32))
	÷128↑×128
	} ÷1024↑
	M
	＋ Σ（Ni）
	i=1

計算式 2 （単位：バイト）	**●32 ビットモードの場合** 500×1024 ＋（436 ＋ 48※1）×g ＋ 496×h ＋ 112×r ＋ 5072×（e ＋ 15）＋ 96×z ＋ 32×m ＋ 172×｛a×（e ＋ 3）＋（e ＋ 3）×2 ＋ 22｝＋ 16 ＋ 48×p ＋ 48×｛(e ＋ 3) ×2 ＋ 1 ＋ MAX（5，[e ＋ 3]÷10)｝ ＋ 68×G ＋ 152×F ＋ 80 ＋ 32×g ＋ 64※2＋ 96※3 ＋（(↓（↑（g÷8）↑＋ 3）÷4↓）×4）×m **●64 ビットモードの場合** 500×1024 ＋（536 ＋ 56※1）×g ＋ 512×h ＋ 136×r ＋ 9424×（e ＋ 15）＋ 144×z ＋ 48×m ＋ 336×｛a×（e ＋ 3）＋（e ＋ 3）×2 ＋ 22｝＋ 16 ＋ 64×p ＋ 96×｛(e ＋ 3) ×2 ＋ 1 ＋ MAX（5，[e ＋ 3]÷10)｝ ＋ 68×G ＋ 184×F ＋ 96 ＋ 48×g ＋ 64※2＋ 128※3 ＋（(↓（↑（g÷8）↑＋ 7）÷8↓）×8）×m
計算式 3 （単位：バイト）	**●32 ビットモードの場合** {{[(↑↑g÷8↑÷4↑）×4]＋ 8｝×（a×[e ＋ 3]＋ e ＋ 15)｝ **●64 ビットモードの場合** {{[(↑↑g÷8↑÷8↑）×8]＋ 8｝×（a×[e ＋ 3]＋ e ＋ 15)｝
計算式 4 （単位：バイト）	**●32 ビットモードの場合** (32 ＋ 16×z）×（e×2 ＋ 7 ＋ 1）＋ 16

条件	共用メモリの計算式
	●64 ビットモードの場合 （48 ＋ 32×z）×（e×2 ＋ 7 ＋ 1）＋ 16
計算式 5 （単位：バイト）	56×E ＋ 16
計算式 6 （単位：バイト）	●32 ビットモードの場合 72 ＋ 52×H ＋ 68×z pd_dbbuff_trace_level オペランドに 1 を指定する場合に加算します。 ＋ 320×z ●64 ビットモードの場合 96 ＋ 56×H ＋ 72×z pd_dbbuff_trace_level オペランドに 1 を指定する場合に加算します。 ＋ 640×z
計算式 7 （単位：バイト）	●32 ビットモードの場合 （32 ＋ 16×z）×10 ＋ 16 ●64 ビットモードの場合 （48 ＋ 32×z）×10 ＋ 16
計算式 8 （単位：バイト）	16 ＋ 112 ＋（48 ＋ 48×Q）＋ (48 ＋ 32×R)
計算式 9 （単位：バイト）	16 ＋ 80×S

注※1

pd_max_list_users 及び pd_max_list_count オペランドの両方とも 0 でない場合に加算します。

注※2

pd_max_ard_process オペランドが 1 以上の場合に加算します。

注※3

pd_max_reflect_process_count オペランドに 1 以上を指定した場合に加算します。

(4) 計算式で使用する変数

a：pd_max_access_tables オペランドの値

b：pd_sql_object_cache_size オペランドの値

c：pd_table_def_cache_size オペランドの値

d：pd_auth_cache_size オペランドの値

e：pd_max_users オペランドの値 ※1

f：バックエンドサーバの総数

g：pd_max_rdarea_no オペランドの値

h：pd_max_file_no オペランドの値

k：pd_view_def_cache_size オペランドの値

m：インデクス用のグローバルバッファプール数

pd_dbbuff_modify オペランドに Y を指定している場合，該当するサーバに関連する pdbuffer 文の数に，該当するサーバ定義の pd_max_add_dbbuff_no オペランドの値を加算して計算します。

p：pd_lck_until_disconnect_cnt オペランドの値

q：MIN（e ＋ 3，p）

r：pd_assurance_index_no オペランドの値

s：インストールしたプラグインの数

t：DML で使用するプラグイン関数の総数 ※2

u：DML で使用するプラグイン関数のパラメタ総数 ※2

v：pd_max_list_users オペランドの値

w：pd_max_list_count オペランドの値

x：pd_type_def_cache_size オペランドの値

y：pd_routine_def_cache_size オペランドの値

z：グローバルバッファ総数（pdbuffer オペランドの指定数）

pd_dbbuff_modify オペランドに Y を指定している場合，該当するサーバに関連する pdbuffer 文の数に，該当するサーバ定義の pd_max_add_dbbuff_no オペランドの値を加算して計算します。

A：pd_registry_cache_size オペランドの値

E：pd_inner_replica_control オペランドの値

F：pd_assurance_table_no オペランドの値

G：サーバ内の最大トランザクション数（2×e ＋ 7）

H：pd_dfw_awt_process オペランドの値

I：フロントエンドサーバで指定した pdplgprm オペランドの総数

Ji：フロントエンドサーバで指定した i 番目の pdplgprm オペランドで指定した共用メモリのサイズ

K：ディクショナリサーバで指定した pdplgprm オペランドの総数

Li：ディクショナリサーバで指定した i 番目の pdplgprm オペランドで指定した共用メモリのサイズ

M：バックエンドサーバで指定した pdplgprm オペランドの総数

Ni：バックエンドサーバで指定した i 番目の pdplgprm オペランドで指定した共用メモリのサイズ

Q：pd_max_resident_rdarea_no オペランドの値

R：pd_max_resident_rdarea_shm_no オペランドの値

S：pd_max_temporary_object_no オペランドの値

U：空き領域の再利用機能を使用する表数

注※1

ディクショナリサーバの場合は pd_max_dic_process オペランドの値となります。バックエンドサーバの場合は pd_max_bes_process オペランドの値となります。ただし，pd_max_dic_process 又は pd_max_bes_process オペランドを省略する場合は，pd_max_users オペランドの値となります。

注※2

DML で使用するプラグイン関数の総数及び DML で使用するプラグイン関数のパラメタの総数は，次に示す SQL で求められます。

```
SELECT COUNT(*),SUM(N_PARAM) FROM MASTER.SQL_PLUGIN_ROUTINES
WHERE PLUGIN_NAME = 'プラグイン名称'
AND (TIMING_DESCRIPTOR = 'ADT_FUNCTION'
    OR TIMING_DESCRIPTOR IS NULL
    OR TIMING_DESCRIPTOR = 'BEFORE_INSERT'
    OR TIMING_DESCRIPTOR = 'AFTER_INSERT'
    OR TIMING_DESCRIPTOR = 'BEFORE_UPDATE'
    OR TIMING_DESCRIPTOR = 'AFTER_UPDATE'
    OR TIMING_DESCRIPTOR = 'BEFORE_DELETE'
    OR TIMING_DESCRIPTOR = 'AFTER_DELETE'
    OR TIMING_DESCRIPTOR = 'BEFORE_PURGE_TABLE'
    OR TIMING_DESCRIPTOR = 'AFTER_PURGE_TABLE'
    OR TIMING_DESCRIPTOR = 'INDEX_SEARCH'
    OR TIMING_DESCRIPTOR = 'INDEX_COUNT'
    OR TIMING_DESCRIPTOR = 'INDEX_INSERT'
    OR TIMING_DESCRIPTOR = 'INDEX_BEFORE_UPDATE'
    OR TIMING_DESCRIPTOR = 'INDEX_AFTER_UPDATE'
    OR TIMING_DESCRIPTOR = 'INDEX_DELETE'
    OR TIMING_DESCRIPTOR = 'PURGE_INDEX'
    OR TIMING_DESCRIPTOR = 'INDEX_MAINTENANCE_DEFERRED'
    OR TIMING_DESCRIPTOR = 'BEFORE_INSERT_DC'
    OR TIMING_DESCRIPTOR = 'BEFORE_UPDATE_DC'
    OR TIMING_DESCRIPTOR = 'BEFORE_DATA_CHECK'
    OR TIMING_DESCRIPTOR = 'AFTER_DATA_CHECK')
```

14.2.5　グローバルバッファが使用する共用メモリの計算式

(1)　影響分散スタンバイレス型系切り替え機能を使用しない場合

グローバルバッファが使用する共用メモリサイズは，ディクショナリサーバ又はバックエンドサーバごとに計算式 1 に代入して求めます。サーバマシンごとに算出する場合，pdbuffer 文のオプションの指定によって計算対象になるかどうかが異なります。pdbuffer 文のオプションによる計算条件を次の表に示します。

表 14–9　pdbuffer 文のオプションによる計算条件（影響分散スタンバイレス型系切り替え機能を使用しない場合）

pdbuffer 文のオプション	計算条件
-r	-r で指定した RD エリアがあるサーバの場合，計算対象とします。
-i	-i で指定したインデクスが格納されている RD エリアがあるサーバの場合，計算対象とします。
-b	-b で指定した RD エリアがあるサーバの場合，計算対象とします。
-o	そのサーバにある全 RD エリア中で pdbuffer -r で指定していない RD エリアがある場合，計算対象とします。

pd_dbbuff_modify オペランドに Y を指定している場合は，計算式 2 を加算します。計算式 1 及び 2 で求めた値を合計した値が，そのサーバでグローバルバッファが使用する共用メモリの所要量です。

pd_dbbuff_attribute オペランドに fixed 又は hugepage を指定している場合，実メモリ上にページ固定されるため，仮想メモリを構成する実メモリがそのサイズ分減少します。また，残り実メモリとスワップ領域で構成される仮想メモリからもそのサイズ分確保されます。

なお，pdbuffer オペランドを省略した場合，HiRDB が共用メモリサイズを自動計算するので見積もりは必要ありません。

計算式の種類	共用メモリの計算式（単位：キロバイト）
計算式 1	**●32 ビットモードの場合** $\sum_{i=1}^{n}$ { ↑ {752 + 64 + (296 + 64※1) × (Pi + 4) + (124 + 80※2 + 96×A×Mi) ×Ui} ÷T↑×T + Si× {Pi + 4 + (Ui×Mi×A)} + Vi } ÷1024 **●64 ビットモードの場合** $\sum_{i=1}^{n}$ {管理領域部＋データ格納部} ÷1024

計算式の種類	共用メモリの計算式（単位：キロバイト）
	管理領域部： ↑｛944＋64＋（480＋112※1）×（Pi＋4） ＋（176＋96※2＋136×A×Mi）×Ui｝÷T↑×T データ格納部： Si×｛Pi＋4＋（Ui×Mi×A)｝＋Vi
計算式 2	**●32 ビットモードの場合** {↑（↑（s×1024÷2）÷8↑＋112）÷2048↑×2048× ↑a÷（s×1024）↑｝÷1024 **●64 ビットモードの場合** {↑（↑（s×1024÷2）÷8↑＋144）÷2048↑×2048× ↑a÷（s×1024）↑｝÷1024

n：グローバルバッファプール数

i：計算対象のグローバルバッファプール定義

P：グローバルバッファ面数

A：

pd_max_ard_process オペランドが 1 以上の場合は 2

pd_max_ard_process オペランドが 0 の場合は 1

M：一括入力最大ページ数

U：同時実行最大プリフェッチ数

T：グローバルバッファのバウンダリ

- Linux の場合

  pd_dbbuff_dev_sector_size オペランドに 512 を指定した場合は 2048，4096 を指定した場合は 4096
- Linux 以外の場合

  4096

S：グローバルバッファに割り当てた RD エリアの最大ページ長

V：

- Linux の場合

  pd_dbbuff_dev_sector_size オペランドに 4096 を指定し，かつ S が 4096 の倍数でない場合は 2048。

  pd_dbbuff_dev_sector_size オペランドに 4096 を指定していない，又は S が 4096 の倍数の場合は 0。

- Linux 以外の場合
  0

s：SHMMAX 指定値

a：計算式 1 の総計

注※1　LOB 用グローバルバッファの場合に加算します。

注※2　pd_max_ard_process オペランドが 1 以上の場合に加算します。

(2)　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

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，ユニットごとに計算式 1 に代入して求めます。ユニットごとに算出する場合，pdbuffer 文のオプションの指定によって計算対象になるかどうかが異なります。pdbuffer 文のオプションによる計算条件を次の表に示します。

表 14–10　pdbuffer 文のオプションによる計算条件（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）

pdbuffer 文のオプション	計算条件
-r，-b	-r で指定した RD エリアで，同じ HA グループに属する RD エリアの場合，計算対象とします。
-i	-i で指定したインデクスが格納されている RD エリアが，同じ HA グループに属する RD エリアの場合，計算対象とします。
-o	同じ HA グループ内で，pdbuffer -r オプションで指定していない RD エリアがある場合，計算対象とします。

pd_dbbuff_attribute オペランドに fixed 又は hugepage を指定している場合，実メモリ上にページ固定されるため，仮想メモリを構成する実メモリがそのサイズ分減少します。また，残り実メモリとスワップ領域で構成される仮想メモリからもそのサイズ分確保されます。なお，pdbuffer オペランドを省略した場合，HiRDB が共用メモリサイズを自動計算するので見積もりは必要ありません。

計算式の種類	共用メモリの計算式（単位：キロバイト）
計算式 1	**●32 ビットモードの場合** n Σ { i=1 (96 ＋ ((752×（A ＋ B)) ＋ (288×（F ＋ (8×（A ＋ B)))) ＋ 8×F×（A ＋ B) ＋ 16) ＋ H ＋ D) ＋ V ＋ 2048 ＋ G ＋ (E×F ＋ (8×（A ＋ B))) } ÷1024 **●64 ビットモードの場合** n

計算式の種類	共用メモリの計算式（単位：キロバイト）
	Σ｛管理領域部＋データ格納部｝÷1024 i=1 管理領域部： ((144＋((944×（A＋B)）＋（464×（F＋（8×（A＋B)))) ＋（16×F×（A＋B))))＋16＋H＋D) データ格納部： 2048＋G＋（E×F＋（8×（A＋B)))＋V

n：このユニットに割り当てられるグローバルバッファプール数

i：計算対象のグローバルバッファプール定義

A：ホスト BES 数

B：受け入れ可能なゲスト BES の最大数

C：一括入力ページ数（pdbuffer -p に指定した値）

D：プリフェッチ機能使用時（pdbuffer -m 指定）に加算します。

32 ビットモードの場合

2×(((80×U×C)＋(80×U)＋(124×U)＋(8×U×C))×(A＋B))

64 ビットモードの場合

2×(((112×U×C)＋(96×U)＋(176×U)＋(16×U×C))×(A＋B))

E：pdbuffer 文のオプションの指定によって値が異なります。オプションと計算式を次に示します。

pdbuffer 文のオプション	最大値の計算式
-r，-b	(MAX（バッファサイズ(pdbuffer -l の値)，MAX（指定した RD エリアで同じ HA グループに属する RD エリアのページサイズ)))
-i	(MAX（バッファサイズ（pdbuffer -l の値)，MAX（-i で指定したインデクスが格納されている RD エリアが同じ HA グループに属する RD エリアのページサイズ)))
-o	(MAX（バッファサイズ（pdbuffer -l の値)，MAX（同じ HA グループ内で pdbuffer -r オプションで指定していない RD エリアのページサイズ)))

F：バッファ面数（pdbuffer -n の値）

G：プリフェッチ機能使用時（pdbuffer -m 指定）に加算します。

2×((E×U×C)×(A＋B))

H：LOB 用 RD エリアが指定されている（pdbuffer -b 指定）場合に加算します。

32 ビットモードの場合：64×（F＋（8×（A＋B)))

64 ビットモードの場合：112×（F＋（8×（A＋B)))

U：同時実行最大プリフェッチ数（pdbuffer -m の値）

V：

- Linux の場合

  pd_dbbuff_dev_sector_size オペランドに 4096 を指定し，かつ E が 4096 の倍数でない場合は 2048。

  pd_dbbuff_dev_sector_size オペランドに 4096 を指定していない，又は E が 4096 の倍数の場合は 0。

- Linux 以外の場合

  0

14.2.6　SQL 実行時に必要なメモリ所要量の計算式

(1)　グループ分け高速化機能実行時に必要なメモリ所要量の求め方

クライアント環境定義で PDSQLOPTLVL オペランドを指定するか，HiRDB システム定義で pd_optimize_level オペランドを指定（又は省略）した場合，適用条件を満たす SQL を実行すると，グループ分け高速化機能が働きます。このとき，HiRDB はクライアント環境定義の PDAGGR オペランドの値に基づいてプロセス固有領域を確保します。確保するメモリサイズを次に示します。

グループ分け高速化機能実行時に必要なメモリ所要量は，バックエンドサーバを定義するサーバマシンについてだけ計算してください。

計算式

```
e＋↑d÷4↑×4＋↑（17＋4×a＋4×b＋c＋d）÷4↑×4×（N＋1）
                                                （単位：バイト）
```

a：グループ化する列の数

b：集合関数の演算数

COUNT，SUM，MAX，MIN は一つにつき 1 で換算します。

AVG（COUNT），AVG（SUM）は一つにつき 2 で換算します。

c：グループ化する列の列長（表「グループ化するときの列の長さ及び集合関数の演算領域の長さ」を参照して求めてください）

d：集合関数の演算領域長（表「グループ化するときの列の長さ及び集合関数の演算領域の長さ」を参照して求めてください）

e：32 ビットモードの場合：MAX（4×N×24，16408）

64 ビットモードの場合：MAX（8×N×40，32808）

N：クライアント環境定義の PDAGGR オペランドの値

表 14-11　グループ化するときの列の長さ及び集合関数の演算領域の長さ

列のデータ型	グループ化する列の列長	集合関数の演算領域長※1
INTEGER	4	6
SMALLINT	2	4※2
DECIMAL(p,s)	↑（p＋1）÷2↑	↑（p＋7）÷2↑※3
FLOAT	8	10
SMALLFLT	4	6
INTERVAL YEAR TO DAY	5	8
INTERVAL HOUR TO SECOND	4	6
CHAR(n)	n	n＋3
VARCHAR(n)	n＋2	n＋5
NCHAR(n)	2×n	2×n＋2
NVARCHAR(n)	2×n＋2	2×n＋4
MCHAR(n)	n	n＋3
MVARCHAR(n)	n＋2	n＋5
DATE	4	6
TIME	3	6
BLOB(n)	－	－
BINARY(n)	n＋2	n＋5

（凡例）－：該当しません。

注※1

集合関数が COUNT の場合，集合関数演算領域長はデータ型にかかわらず 6 になります。

注※2

集合関数が AVG，SUM の場合，集合関数演算領域長は 6 になります。

注※3

集合関数が AVG，SUM の場合，集合関数演算領域長は次の値になります。

集合関数の値の型が DECIMAL 型で精度が 29 けたのとき：18

集合関数の値の型が DECIMAL 型で精度が 38 けたのとき：23

集合関数の値のデータ型の規則については，マニュアル「HiRDB SQL リファレンス」の「集合関数」を参照してください。

(2) 列ごとのデータ抑制指定時に必要なメモリ所要量の求め方

列ごとのデータ抑制を指定（CREATE TABLE の列定義に SUPPRESS を指定）した表に対してアクセスするときに使用するメモリサイズは，次に示す計算式で求められます。

計算式

a＋128　　（単位：バイト）

ａ：表中の列ごとのデータ抑制が指定されている列の定義長の合計値

(3) ハッシュジョイン及び副問合せのハッシュ実行時に必要なメモリ所要量の求め方

クライアント環境定義で PDADDITIONALOPTLVL オペランドを指定するか，HiRDB システム定義で pd_additional_optimize_level オペランドを指定すると，SQL 拡張最適化オプションが働きます。この SQL 拡張最適化オプションで，「ハッシュジョイン，副問合せのハッシュ実行の適用（APPLY_HASH_JOIN）」を指定した場合，表の結合又は副問合せの SQL を実行すると，次に示すメモリサイズのプロセス固有領域を確保します。

計算式

●32ビットモードの場合

$$\sum_{i=1}^{a}(13\times1024+6\times1024\times b+c)$$

●64ビットモードの場合

$$\sum_{i=1}^{a}(13\times1024+7\times1024\times b+c)$$

（単位：バイト）

ａ：SELECT 文のハッシュジョイン最大数

SELECT 文のハッシュジョイン最大数については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

ｂ：ハッシュ表行数によって適用されるハッシュジョイン処理を求めて，次に示す表から代入する値を決定してください。

ハッシュ表行数の目安	適用されるハッシュジョイン処理		ｂの値
1500 以内	一括ハッシュジョイン		0.5
1500×（バケット分割数÷3）以内	バケット分割ハッシュジョイン	1 レベルバケット分割	1
1500×（バケット分割数÷3）^{2} 以内		2 レベルバケット分割	2
1500×（バケット分割数÷3）^{2} を超える場合		3 レベルバケット分割	3

ハッシュ表行数：ジョインの場合はジョインの内表件数です。副問合せの場合は探索条件中の外への参照列を含む述語を除いた副問合せ探索件数です。

バケット分割数：MIN｛↓（ハッシュ表サイズ÷2）÷ハッシュ表ページ長↓，64｝

ハッシュ表サイズ：HiRDB システム定義の pd_hash_table_size オペランド，又はクライアント環境変数の PDHASHTBLSIZE オペランドで指定した値です。

ハッシュ表ページ長：次に示す表から c（ハッシュ表最大行長）に対応するハッシュ表ページ長を選択してください。

ハッシュ表最大行長	ハッシュ表ページ長（単位：バイト）
0～1012	4096
1013～2036	8192
2037～4084	16384
4085～16360	32768
16361～32720	↑（ハッシュ表最大行長＋ 48）÷2048↑×2048

c：ハッシュ表最大行長

ハッシュ表最大行長については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(4) スナップショット方式指定時に必要なメモリ所要量の求め方

pd_pageaccess_mode オペランドを省略した場合又は SNAPSHOT を指定した場合，スナップショット方式を適用する SQL 文を実行すると，データベース検索時のページアクセス方式にスナップショット方式を使用します。このとき，表又はインデクスの格納 RD エリアのページサイズに基づいて，動的に次に示すメモリサイズのプロセス固有領域を確保します。

計算式

a×2　　（単位：バイト）

a：検索対象の表又はインデクスが格納されている RD エリア中の最大ページ長。

ただし，LOB 用 RD エリアは除きます。

(5) 先頭から n 行の検索結果を取得する機能実行時に必要なメモリ所要量の求め方

先頭から n 行の検索結果を取得する機能を使用すると，検索結果の先頭（又はユーザが指定した先頭からのオフセット行数分読み飛ばした位置）から n 行取得できます。

LIMIT 句に指定した行数が 1 以上で，（オフセット行数＋ LIMIT 句に指定した行数）の値が 32,767 以下の場合，（オフセット行数＋ LIMIT 句に指定した行数）以内に入り得る行をメモリに保持します。確保するプロセス固有領域のメモリサイズは，次に示す計算式で求められます。なお，（オフセット行数＋ LIMIT

句に指定した行数）の値が 32,768 以上になる場合は作業表を作成するため，「作業表用ファイルの容量の見積もり」を参照してください。

計算式

```
{100＋（a＋2）×（オフセット行数＋LIMIT句に指定した行数）} ×b　　（単位：バイト）
```

a：行長

行長は 32,720 バイト以下でなければなりません。行長は次の計算式で求められます。

$$\sum_{i=1}^{m}(Ai)+2\times m+4+c$$

（単位：バイト）

m：選択式，GROUP BY 句，又は ORDER BY 句に指定した列数

FOR UPDATE 句を指定した場合は 1 を加算してください。ただし，選択式に ROW を指定している場合は表の全列数になります。

Ai：先頭 n 行保持領域に格納する行の i 番目の列データ長

列のデータ長については，表「データ長一覧」を参照し，d に定義長を代入して求めてください。

ただし，BLOB データ，定義長が 256 バイト以上の文字データ（各国・混在文字データも含む），BINARY データのうち，下記に属さない列の場合は 12 バイトになります。

- DISTINCT 句指定の選択式に指定する列
- UNION [ALL]によって集合演算対象となっている問合せ指定中の選択式
- ORDER BY 句に指定した列

また，FOR UPDATE 句を指定した場合に，m に加算した 1 に対応する Ai は 12 バイトとします。

c：8

ただし，次の場合は 0 になります。

- 検索対象の表に EX モードで排他が掛かっている場合
- WITHOUT LOCK を指定した場合
- グループ分け高速化機能を指定した場合
- 複数の表を結合する場合

b：先頭 n 行保持領域数

先頭 n 行保持領域数は次の計算式で求められます。

```
1＋UNION [ALL]句指定数
```

(6) 探索条件にインデクス型プラグイン専用関数を指定したSQL文実行時に必要なメモリ所要量の求め方

探索条件にインデクス型プラグイン専用関数を指定したSQL文の実行時に確保するプロセス固有領域のメモリサイズは，次に示す計算式で求められます。

計算式

```
a×500＋（20＋6）×800＋16　　（単位：バイト）
```

ａ：行長。行長は次の計算式で求められます。

```
 m
Σ（Ai）＋4×（m＋2）＋12＋4＋8
i=1
                                                （単位：バイト）
```

m：選択式，結合条件，GROUP BY句，又はORDER BY句に指定した列数

FOR UPDATE句を指定した場合は1を加算してください。ただし，選択式にROWを指定している場合は表の全列数になります。

Ai：取り出す行のi番目の列データ長

列のデータ長については，表「データ長一覧」を参照し，dに定義長を代入して求めてください。

ただし，BLOBデータ，又は定義長が256バイト以上の文字データ（各国・混在文字データも含む）で，下記に属さない列の場合は12バイトになります。

- 結合条件中に指定する列（結合列）
- DISTINCT句指定の選択式に指定する列
- 限定述語の副問合せ中の選択式に指定する列
- IN述語の副問合せ中の選択式に指定する列
- UNION [ALL]，又はEXCEPT [ALL]によって集合演算対象となっている問合せ指定中の選択式
- ORDER BY句に指定した列

また，FOR UPDATE句を指定した場合に，mに加算した1に対応するAiは12バイトとします。

(7) 拡張SQLエラー情報出力機能使用時に必要なメモリ所要量の求め方

拡張SQLエラー情報出力機能を使用した場合，次のときにプロセス固有領域を確保します。

(a) OPEN文実行時

計算式

```
●32ビットモードの場合
（16＋16×m）＋a
●64ビットモードの場合
```

（24＋24×m）＋a
（単位：バイト）

a：？パラメタ又は埋込み変数のデータ長の合計

$$a=\sum_{i=1}^{m}(ai)$$

m：SQL 文中の？パラメタ又は埋込み変数の数

ai：i 番目の？パラメタ又は埋込み変数のデータ長

データ長については，表「埋込み変数又は？パラメタのデータ長」を参照して算出します。

(b) 定義系 SQL の PREPARE 文実行時

計算式

SQL文長＋20　（単位：バイト）

(8) 部分構造インデクスの定義，又は部分構造インデクスを定義した表の更新時に必要なメモリ所要量の求め方

(a) 部分構造インデクスの定義時

定義系 SQL の CREATE INDEX で部分構造インデクスを定義する場合に確保するプロセス固有領域は，次に示す計算式で求められます。

計算式

（インデクスキー長※×100＋64）　（単位：バイト）

注※

表に定義する部分構造インデクスの最大定義長です。

(b) 部分構造インデクスを定義した表の更新時

操作系 SQL の INSERT，UPDATE，又は DELETE で部分構造インデクスを定義した表を更新する場合に確保するプロセス固有領域は，次に示す計算式で求められます。

計算式

（インデクスキー長※1×100＋64＋128）＋Σ（インデクスキー長＋128）※2　（単位：バイト）

注※1

表に定義している部分構造インデクスの最大定義長です。

注※2

USING UNIQUE TAG 指定の部分構造インデクス数です。

(9) 圧縮列に対して操作系 SQL を実行する場合に必要なメモリ所要量の求め方

SQL 実行時，データの格納，及び抽出対象に圧縮列が含まれる場合，次に示すメモリサイズのプロセス固有領域を確保します。

計算式

```
MIN（圧縮分割サイズ，圧縮列の定義長）※×C＋L　　（単位：バイト）
```

C：次に示すどれかの条件に該当する場合は 2。該当しない場合は 1。

- SUBSTR 関数を使用している
- POSITION 関数を使用している
- 後方削除更新をしている

L：SQL の実行対象となる圧縮表が格納されている RD エリアのページ長

複数の RD エリアが対象になる場合は，最大のページ長で計算する。

注※

SQL の実行対象となる全圧縮列の中で最大になる値で計算します。

(10) バックエンドサーバで使用する SQL 実行用通信メモリ所要量の求め方

SQL 文実行時，FES－BES 間，及び BES－BES 間で使用する通信で，次に示すメモリサイズのプロセス固有領域を確保します。

計算式

```
4×1024×
（2×1SQLで指定する表の最大数
　×表の最大分割BES数÷システム内BES数
　＋フロータブルサーバ数※×システム内BES数×2
　＋2×フロータブルサーバ数※）　　（単位：バイト）
```

注※

フロータブルサーバを使用する SQL の中で指定している表の最大数を指定してください。フロータブルサーバを使用しない SQL だけの場合は 0 を指定してください。フロータブルサーバを使用する SQL の詳細は，マニュアル「HiRDB UAP 開発ガイド」の「フロータブルサーバの割り当て方法」を参照してください。

14.2.7　SQL 前処理時に必要なメモリ所要量の計算式

(1)　ストアドプロシジャを使用しない場合に必要なメモリ所要量の求め方

ストアドプロシジャを使用しない場合，SQL 前処理時に確保するメモリサイズは，次に示す計算式で求められます。

計算式

```
↑ {
    (2539＋Si×70＋Pi×20＋Ti×980＋Ci×68＋Wi×818＋Ki×416＋Li×5
   ＋Di×116＋Ari×108＋Gi×44＋Ori×10＋Sli×40＋Upi×96＋Fi×90
   ＋Ti×Cwi×48＋MAX (Pi, Wpi) ×52＋MAX (Ti, Sli-1) ×96
   ＋MAX (Ti×2, Wi) ×24＋MAX (Ti×3, Wi) ×24
   ＋MAX {MAX (Ti, Ori＋Gi＋Si＋Fi) , Sli-1} ×24
 } ×1.2 ÷1024↑×CLS
                                                    （単位：キロバイト）
```

Si：SQL 文中の検索項目数

Pi：SQL 文中の埋込み変数，？パラメタ又は SQL パラメタの数

Ti：SQL 文中の表名の数

Ci：SQL 文中の列名の数

Wi：SQL 文中の論理演算子（AND 及び OR）に出てくる述語の数

Ki：SQL 文中の定数の数

Li：SQL 文中の定数の長さの合計（単位：バイト）

Di：SQL 文中に定義された格納条件の総数

Ari：SQL 文中の四則演算及び連結演算の数

Gi：SQL 文中の GROUP BY 句に指定した列の数

Ori：SQL 文中の ORDER BY 句に指定した列指定又はソート項目指定番号の数

Fi：SQL 文中の集合関数及びスカラ関数の総数

Sli：SQL 文中の問合せ指定の数

Upi：SQL 文中の更新列数

Cwi：SQL 文中の CASE 式中の WHEN の数

Wpi：SQL 文中の WITH 句に対応する変数の数

CLS：SQL オブジェクト内の一つのアクセスパスが生成される領域の数※

注※

SQL オブジェクト内の一つのアクセスパスが生成される領域の数は，次に示す計算式で求められます。

計算式

```
SELECT_APSLが適用されている場合※
  a＋b×4＋c＋d＋e×2
SELECT_APSLが適用されていない場合※
  a＋b＋c＋d＋e
```

a：フロントエンドサーバ数

フロントエンドサーバ数は 1 を指定します。

b：表数

表数は次に示す計算式で求めます。

実表数＋相関名数

c：集合演算サーバ数

集合関数の指定がある場合は 1 を，指定がない場合は 0 を指定します。

d：GROUP BY 句，DISTINCT 又は ORDER BY 句がある問合せ指定の数

e：ジョインサーバ数

ジョインサーバ数は次に示す計算式で求めます。

b－SQL 文中の問合せ指定の数

注※

SELECT_APSL が適用され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（pdvwopt）を使用すると分かります。アクセスパス表示ユティリティ（pdvwopt）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。

(2)　ストアドプロシジャを使用する場合に必要なメモリ所要量の求め方

ストアドプロシジャを使用する場合，SQL 前処理時に確保するメモリサイズ（単位：キロバイト）は，「ストアドプロシジャを使用しない場合に必要なメモリ所要量の求め方」の計算式で求めた値に，ストアドプロシジャごとのプロシジャ制御用オブジェクト長を加算します。プロシジャ制御用オブジェクト長の計算式については，システム共通定義の pd_sql_object_cache_size オペランドの 1 ストアドプロシジャのプロシジャ制御用オブジェクト長を参照してください。1 ストアドプロシジャのプロシジャ制御用オブジェクト長については，マニュアル「HiRDB システム定義」の「1 ルーチンのルーチン制御用オブジェクト長の計算式」を参照してください。

14.2.8 BLOB 型データの検索又は更新時に必要なメモリ所要量の計算式（フロントエンドサーバの場合）

BLOB 型データの検索又は更新時に必要なメモリ所要量は次に示す計算式で求められます。

計算式

ａ＋ｂ＋7　　（単位：キロバイト）

ａ：1SQL 文中に指定する BLOB 型入力変数又は出力変数で，実行する SQL 文の中で次に示す計算式の結果が最大となる値です。

```
↑ {
 c
Σ （BLOB型入力変数iの実長※1 ×2＋58） ＋
i=1
 d
Σ （BLOB型出力変数jの定義長※2 ＋26）
j=1
} ÷1024↑
```

注※ 1

埋込み変数で UAP から HiRDB サーバに受け渡された BLOB 型データの実際の長さです。

注※ 2

HiRDB から UAP に返却する BLOB 型データを受け取る UAP の埋込み変数の宣言長です。

ｂ：同時オープン中のカーソルで結合検索を行う SQL 文の組み合わせで，次に示す計算式の結果が最大となる値です。

256×同時オープン中のカーソル数

ｃ：入力変数の数

ｄ：出力変数の数

14.2.9 BLOB 型データの検索又は更新時に必要なメモリ所要量の計算式（バックエンドサーバ又はディクショナリサーバの場合）

BLOB 型データの検索又は更新時に必要なメモリ所要量は次に示す計算式で求められます。

計算式

ａ＋ｂ

（単位：キロバイト）

ａ：1SQL 文中に指定する BLOB 型入力変数又は出力変数について，実行する SQL 文の中で，次に示す計算式の結果が最大となる値です。

```
↑ {
 c
Σ （BLOB型入力変数iの実長※＋118＋70×出力変数の数）
i=1
} ÷1024↑
```

注※

埋込み変数で UAP から HiRDB サーバに受け渡された BLOB 型データの実際の長さです。

ｂ：同時オープン中のカーソルで BLOB 型データの検索を行う SQL 文の組み合わせで，次に示す計算式の結果が最大となる値です。

```
d
Σ {280＋184×（SQLiに記述した表数＋1）}
i=1
```

ｃ：入力変数の数

ｄ：カーソル数

14.2.10　ブロック転送又は配列 FETCH で必要なメモリ所要量の計算式（フロントエンドサーバの場合）

ブロック転送又は配列 FETCH で必要なメモリ所要量は，次の計算式で求められます。

条件		PDBLKBUFFSIZE オペランドの指定値	
		省略又は 0	1 以上
FETCH 文の INTO 句に配列型の埋込み変数を指定する		計算式 1	
FETCH 文の INTO 句に配列型の埋込み変数を指定しない	PDBLKF オペランドを省略又は 1	－	計算式 2
	PDBLKF オペランドが 2 以上	計算式 1	

（凡例）－：該当しません。

計算式 1

```
↑ {864＋16×ａ＋（6×ａ＋2×d＋ｂ）×ｃ} ÷1024↑
                                （単位：キロバイト）
```

ａ：SELECT 句で指定する検索項目数

ｂ：FETCH 文で受け取る検索結果中の 1 行のデータ長（各列の最大長の合計。単位はバイト）

ｃ：PDBLKF オペランドの指定値又は配列数

d：SELECT 句で指定する検索項目で，BINARY 型を指定した選択式の数

計算式 2

```
MAX（X1，X2）
                                                （単位：キロバイト）
```

X_1：↑（864 ＋ 22×a ＋ 2×c ＋ b）÷1024↑

X_2：PDBLKBUFFSIZE オペランドの値

a：SELECT 句で指定する検索項目数

b：FETCH 文で受け取る検索結果中の 1 行のデータ長（実際に取得する各列の長さの合計。単位はバイト）

c：SELECT 句で指定する検索項目で，BINARY 型を指定した選択式の数

14.2.11　インメモリデータ処理で必要なメモリ所要量

インメモリデータ処理で必要なメモリ所要量は次に示す計算式で求められます。

HiRDB/パラレルサーバの場合は，サーバマシンごとにインメモリ化する RD エリアを見積もってください。

計算式

```
計算式 1 ＋D×2　　（単位：キロバイト）
```

計算式 1

```
n
Σ　↑｛736＋32×A＋48＋448×B＋2048＋C×B｝÷1024↑　　（単位：キロバイト）
i=1
```

n：インメモリ RD エリアの数

A：インメモリ RD エリアを構成する HiRDB ファイル数

B：インメモリ RD エリアの総ページ数

C：インメモリ RD エリアのページサイズ

D：計算式 2 の値

計算式 2（インメモリデータバッファが使用する共用メモリセグメント数）

```
↑計算式 1 の値÷（SHMMAXオペランドの値×1024）↑
```

計算式 2 で求めた値は，pd_max_resident_rdarea_shm_no オペランド又は OS のオペレーティングシステムパラメタの見積もりに使います。

15 RD エリアの容量の見積もり

この章では，各 RD エリアの容量の見積もり方法について説明します。

15.1　ユーザ用 RD エリアの容量の見積もり

ここでは，ユーザ用 RD エリアの容量の見積もり方法について説明します。

15.1.1　ユーザ用 RD エリアの容量の計算方法

(1)　ユーザ用 RD エリアの容量の求め方

ユーザ用 RD エリアの容量は，次に示す計算式で求めます。

計算式

```
ユーザ用RDエリアの容量（単位：バイト）
　　＝ユーザ用RDエリアのページ長※1×ユーザ用RDエリアの総ページ数※2
```

注※1

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）の create rdarea 文で指定するページ長です。

注※2

「ユーザ用 RD エリアの総ページ数を求める計算式」を参照してください。

(2)　ユーザ用 RD エリアの総ページ数を求める計算式

ユーザ用 RD エリアの総ページ数は，次に示す計算式で求めます。

計算式

```
ユーザ用RDエリアの総ページ数（単位：ページ）
　　＝ディレクトリページ部分の総ページ数＋データページ部分の総ページ数
```

(a)　ディレクトリページ部分の総ページ数の計算式

$$\text{ディレクトリページ部分の総ページ数（単位：ページ）} = 6 \times (n+1) + \uparrow 20480 \div P \uparrow \times 2 + \sum_{i=1}^{n} \{\uparrow d_i \div b \uparrow + \uparrow d_i \div f \uparrow\}$$

n：ユーザ用 RD エリアを構成する HiRDB ファイル数

P：ユーザ用 RD エリアのページ長（バイト）

b：$\downarrow (P-20) \div (\uparrow S \div 32 \uparrow \times 8 + 56) \downarrow$

f：$\downarrow (125 \times P) \div (16 \times b) \downarrow \times b$

d_i：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）の create rdarea 文で指定する各 HiRDB ファイルのセグメント数

S：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）の create rdarea 文で指定する 1 セグメントのページ数（セグメントサイズ）

(b)　データページ部分の総ページ数の計算式

$$
\begin{aligned}
&\text{データページ部分の総ページ数（単位：ページ）} = \\
&\sum_{i=1}^{e} \{\uparrow \alpha i \div S \uparrow \times S\} \\
&+ \sum_{i=1}^{e} \{\uparrow \beta i \div S \uparrow \times S\} \\
&+ \sum_{i=1}^{k} \{\uparrow (\gamma i + 1) \div S \uparrow \times S\}
\end{aligned}
$$

e：ユーザ用 RD エリアに格納する表の総数

k：ユーザ用 RD エリアに格納するインデクスの総数

S：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）の create rdarea 文で指定する 1 セグメントのページ数（セグメントサイズ）

α_i：各表の分岐するとした BINARY 列以外の列を格納するために必要なページ数

「表の格納ページ数の計算方法」を参照してください。

β_i：各表の分岐するとした BINARY 列を格納するために必要なページ数

「表の格納ページ数の計算方法」を参照してください。

γ_i：各インデクスを格納するために必要なページ数

「インデクスの格納ページ数の計算方法」を参照してください。

15.1.2　表の格納ページ数の計算方法

CREATE TABLE で FIX 指定をするかどうかによって，表の格納ページ数の計算方法が異なります。それぞれの計算方法を「FIX 指定がない場合」と「FIX 指定がある場合」に説明します。「FIX 指定がない場合」及び「FIX 指定がある場合」の計算式中で使用する変数については，「計算式中で使用する変数」で説明しています。表の格納ページ数の計算例については，「表の格納ページ数の計算例」で説明しています。また，リバランス機能を使用する場合の RD エリア容量見積もりを「リバランス機能を使用する場合のエリア容量見積もり」で説明しています。

なお，表を横分割する場合，格納 RD エリアごとにページ数を求めてください。

(1)　FIX 指定がない場合

FIX 指定がない場合の表の格納ページ数は，次に示す計算式で求めます。

計算式

```
表の格納ページ数＝
　分岐するとしたBINARY列以外を格納するページ数
＋分岐するとしたBINARY列を格納するページ数　　（単位：ページ）
```

・分岐するとした BINARY 列以外を格納するページ数

$$\left\lceil \frac{(P+SPN1+\sum_{i=1}^{n} PS_i) \times g}{g-\left\lfloor \frac{g \times h}{100} \right\rfloor} \right\rceil$$

・分岐するとした BINARY 列を格納するページ数

```
SPN2
```

(a)　P の求め方

P の求め方を次に示します。なお，P の分母の括弧部は 1 ページに格納される行数であり，最小 1，最大 255 とします。

$$P=\left\lceil \frac{a}{MIN\left(255,\left(\left\lfloor \frac{b \times (100-c)}{100} \right\rfloor -48-Z \right) / \left(\left\lceil \frac{\sum_{i=1}^{f} d_i}{2} \right\rceil \times 2+8+2 \times f \right) \right)^{※1}} \right\rceil$$

注※1　次の条件を満たすようにｂ，ｃの値を決定してください。
ｂの値を決定する場合，マニュアル「HiRDB Version 9 SQLリファレンス」のCREATE TABLEの共通規則に記載されている，列の長さの合計による制限も満たす必要があります。

$$\left\lfloor \frac{b \times (100-c)}{100} \right\rfloor -48-Z \geqq \left\lceil \frac{\sum_{i=1}^{f} d_i}{2} \right\rceil \times 2+8+2 \times f$$

(b)　PS_i の求め方

PS_i の求め方を次に示します。次に示す計算式で各 PS_i を計算し，その総和を求めてください。なお，n は表「可変長文字列型のデータ長一覧（抽象データ型及び繰返し列を除く）」に該当する列の数を示しています。

$$PS_i = a \times \lceil e_i \div (b-62) \rceil$$

(2)　FIX 指定がある場合

FIX 指定がある場合の表の格納ページ数は，次に示す計算式で求めます。

計算式

$$\text{表の格納ページ数} = \left\uparrow \frac{Q \times g}{g - \left\downarrow \frac{g \times h}{100} \right\downarrow} \right\uparrow \quad \text{（単位：ページ）}$$

(a)　Q の求め方

Q の求め方を次に示します。なお，Q の分母の括弧部は 1 ページに格納される行数であり，最小 1，最大 255 とします。

$$Q = \left\uparrow \frac{a}{MIN\left(255, \left(\left\downarrow \frac{b \times (100 - c)}{100} \right\downarrow - 48 - Z \right)^{※1} \right)} \right\uparrow$$

注※1　次の条件を満たすようにｂ，ｃの値を決定してください。
ｂの値を決定する場合，マニュアル「HiRDB Version 9 SQLリファレンス」のCREATE TABLEの共通規則に記載されている，列の長さの合計による制限も満たす必要があります。

$$\left\downarrow \frac{b \times (100 - c)}{100} \right\downarrow - 48 - Z \geqq \left\uparrow \frac{\sum_{i=1}^{f} d_i}{2} \right\uparrow \times 2 + 6$$

(3)　計算式中で使用する変数

a：表に格納する行の総数（件）

b：ユーザ用 RD エリアのページ長（バイト）

c：CREATE TABLE で指定する未使用領域の比率（%）

未使用領域の比率を指定しない場合は，30%を仮定して計算します。

d_i：各列のデータ長（バイト）

表「データ長一覧」を参照して，すべての列について求めてください。

抽象データ型の列のデータ長については，「抽象データ型の列のデータ長の求め方」を参照してください。

繰返し列のデータ長については，「繰返し列のデータ長の求め方」を参照してください。

e_i：列のデータ長の平均値（バイト〉

- 既定義型で定義された列の場合，表「可変長文字列型のデータ長一覧（抽象データ型及び繰返し列を除く）」を参照して，表中に示したデータ型の列についてだけ求めてください。

- 抽象データ型で定義された列の場合，表「可変長文字列型のデータ長一覧（抽象データ型の場合）」を参照して，表中に示したデータ型の列についてだけ求めてください。
- 繰返し列の場合，表「可変長文字列型のデータ長一覧（繰返し列の場合）」を参照して，表中に示したデータ型の列についてだけ求めてください。

f：表に定義する列の総数（個）

g：表を格納する RD エリアのセグメントサイズ（ページ）

h：CREATE TABLE で指定するセグメント内の空きページ比率（%）

セグメント内の空きページ比率を指定しない場合は，10%を仮定して計算します。ここでいう空きページとは，未使用ページのことです。

Z：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ALL，又は，ONLY_USER の場合：0
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合：510

SPN1：分岐するとした列（BINARY 以外）を格納するページ数

なお，分岐する条件については表「データ長一覧」の注※5 で説明しています。

```
SPN1＝
   f
  Σ ↑分岐するとしたdiの値÷（b-61）↑×a×SF
  i=1
```

SPN2：分岐するとした BINARY 列を格納するページ数

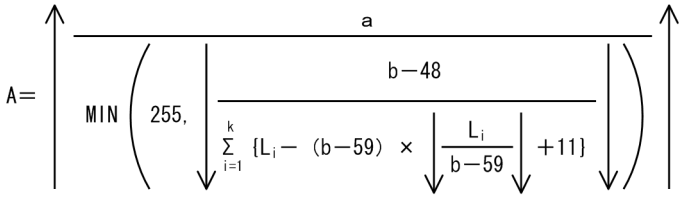
なお，分岐する条件については表「データ長一覧」の注※5 で説明しています。

```
SPN2＝SPN2A＋SPN2B＋SPN2C

・INSERT SQLによる分岐ページ数
 分岐するとしたBINARY列について計算します。
 SPN2A＝
    k
  {Σ ↓Li÷（b-59）↓×a＋A} ×SF
   i=1

・pdload又はpdrorgによる分岐ページ数
 Li＞（b-2853）÷255 のとき
 SPN2B＝
      k
 ↑ {Σ （Li＋11）×a} ÷（b-48）↑×SF
     i=1
 Li≦（b-2853）÷255 のとき
 SPN2C＝
 ↑a÷255↑×SF
```

A の計算式を次に示します。



k：分岐するとした列の数

L_i：各列の実際のデータ長（バイト）

圧縮列の場合は次の計算式の値になります。

圧縮後のデータ長＋（↑圧縮前のデータ長÷圧縮分割サイズ↑）×8

SF：1.3

ただし，次に示す場合は 1.3 より大きくしてください。

- 抽象データ型の列を大量に更新する場合
- 繰返し列に対して要素のデータ長が大きくなる更新又は要素数が増える更新を大量に実行する場合
- VARCHAR，NVARCHAR，MVARCHAR，又は BINARY 型の列に対してデータ長が大きくなる更新を大量に実行する場合
- BINARY 型の列に，データ長が大きくなる更新を大量に実行する場合
- 列ごとのデータ抑制が実行された列に対して，データ長が大きくなる更新を大量に実行する場合
- 前記以外のデータ型で，NULL 値から非 NULL 値への更新を大量に実行する場合

表 15-1　データ長一覧

分類	データ型及び条件			データ長 (単位：バイト)
数値データ	INTEGER			4
	SMALLINT			2
	LARGE DECIMAL(m,n)　※1			↓m÷2↓ ＋1※2
	FLOAT 又は DOUBLE PRECISION			8
	SMALLFLT 又は REAL			4
文字データ	CHARACTER(n)			n※3
	VARCHAR(n)	d≦255	繰返し列の要素	d＋2
			上記以外	d＋1
		d≧256		6
	VARCHAR(n) ノースプリット オプション	n≦255	抽象データ型の属性	d＋3
			繰返し列の要素	d＋2

分類	データ型及び条件				データ長 (単位：バイト)
	指定あり		上記以外		d＋1
		n≧256	分岐する場合※5		6
			分岐しない場合※5	抽象データ型の属性	d＋3
				繰返し列の要素	d＋2
				上記以外	d＋3
各国文字データ	NCHAR(n)又は NATIONAL CHARACTER(n)				2×n ※4
	NVARCHAR(n)	d≦127	繰返し列の要素		2×d＋2
			上記以外		2×d＋1
		d≧128			6
	NVARCHAR(n) ノースプリット オプション 指定あり	n≦127	抽象データ型の属性		2×d＋3
			繰返し列の要素		2×d＋2
			上記以外		2×d＋1
		n≧128	分岐する場合※5		6
			分岐しない場合※5	抽象データ型の属性	2×d＋3
				繰返し列の要素	2×d＋2
				上記以外	2×d＋3
混在文字データ	MCHAR(n)				n※3
	MVARCHAR(n)	d≦255	繰返し列の要素		d＋2
			上記以外		d＋1
		d≧256			6
	MVARCHAR(n) ノースプリット オプション 指定あり	n≦255	抽象データ型の属性		d＋3
			繰返し列の要素		d＋2
			それ以外		d＋1
		n≧256	分岐する場合※5		6
			分岐しない場合※5	抽象データ型の属性	d＋3
				繰返し列の要素	d＋2
				上記以外	d＋3
日付データ	DATE				4
時刻データ	TIME				3
日間隔データ	INTERVAL YEAR TO DAY				5

分類	データ型及び条件			データ長 (単位：バイト)
時間隔データ	INTERVAL HOUR TO SECOND			4
時刻印データ	TIMESTAMP(n)			7 ＋（n÷2）
長大データ	BLOB			9
バイナリデータ	BINARY(n)	n≦255		d ＋ 3
		n≧256	分岐する場合※5	15
			分岐しない場合※5	d ＋ 3
	BINARY(n) 圧縮指定あり	分岐する場合※5		15
		分岐しない場合※5		γ ＋ 9

d：実際のデータ長（文字数）

m，n：正の整数

γ：圧縮後のデータ長（文字数）

注※1

全体のけた数が m けたで，小数点以下のけた数が n けたの固定小数点数です。m を省略した場合は 15 を仮定します。

注※2

表定義時に表オプションに SUPPRESS DECIMAL を指定した場合，データ長は「⌊k÷2⌋ ＋ 2」になります。k は，格納時の有効けた数（先頭の 0 の部分を除いたけた数）を示します。なお，次に示す場合は SUPPRESS DECIMAL を使用しないでください。計算式中の a は，SUPPRESS DECIMAL 又は列データ抑制指定を使用しない場合の，表中の列のデータ長の合計値です。

32717 ＜（a ＋表中の列数×2 ＋ 8）

注※3

列データ抑制指定をして，データ抑制された場合，n は「n－b ＋ 4」になります。なお，データ抑制は，列データ抑制指定時，列データの最後の文字が空白の場合，その最後の文字と連続している半角の空白が 4 文字以上あるときだけ実行されます。b は，列データの最後の文字と連続している空白の数を示します。

ただし，列データ抑制指定をして，データ抑制されなかった場合は，列ごとに 1 バイトの付加情報が追加されます。

なお，次に示す場合は列データ抑制指定を使用しないでください。計算式中の a は，SUPPRESS DECIMAL 又は列データ抑制指定を使用しない場合の，表中の列のデータ長の合計値です。

32717 ＜（a ＋表中の列数×2 ＋ 8）

注※4

列データ抑制指定をして，データ抑制された場合，2×n は「2×n−2×b ＋ 5」になります。なお，データ抑制は，列データ抑制指定時，列データの最後の文字が空白の場合，その最後の文字と連続している全角の空白が 3 文字以上あるときだけ実行されます。b は，列データの最後の文字と連続している空白の数を示します。

ただし，列データ抑制指定をして，データ抑制されなかった場合は，列ごとに 1 バイトの付加情報が追加されます。

なお，次に示す場合は列データ抑制指定を使用しないでください。計算式中の a は，SUPPRESS DECIMAL 又は列データ抑制指定を使用しない場合の，表中の列のデータ長の合計値です。

32717 ＜（a ＋表中の列数×2 ＋ 8）

注※5

通常は分岐しない場合で計算します。次に示す計算式が成立する場合に分岐します。なお，圧縮列の場合，データ長は圧縮前のデータ長で計算してください。

```
BL＞ページ長-50
                  f
ＢＬ（バイト）＝Σ di＋2×f＋6
                 i=1
```

この分岐条件が成立した場合は，分岐しないとした列を列番号の小さい方から分岐条件が成立しなくなるまで分岐するとして BL を計算し直してください。

表 15–2　可変長文字列型のデータ長一覧（抽象データ型及び繰返し列を除く）

データ型		データ長（バイト）
VARCHAR(n)	d≧256	d ＋ 2
	ノースプリットオプション指定あり	0
NVARCHAR(n)	d≧128	2×d ＋ 2
	ノースプリットオプション指定あり	0
MVARCHAR(n)	d≧256	d ＋ 2
	ノースプリットオプション指定あり	0

d：実際のデータ長（文字数）

(4)　抽象データ型の列のデータ長の求め方

抽象データ型の列のデータ長 d_i は，次に示す計算式で求めます。

計算式

$$di = \sum_{k=1}^{h} ADT_k + 5$$

h：抽象データ型の継承の数（個）

継承なしの場合は 1

CREATE TYPE 文で UNDER オペランドを指定して別の抽象データ型を継承した場合，最も上位の抽象データ型を h 番目，最も下位の抽象データ型を 1 番目としてください。

ADT_k：抽象データ型のデータ長（バイト）

次に示す計算式で求めてください。

$$ADT_k = \sum_{i=1}^{m} att_j + 10 + 2 \times m$$

m：抽象データ型の全属性数（個）

att_j：抽象データ型の各属性のデータ長（バイト）

継承がない場合は，m=1 であり，ADT_1 を計算します。

各属性のデータ長については，表「データ長一覧」を参照してください。ただし，データ型が表「可変長文字列型のデータ長一覧（抽象データ型の場合）」で示す条件を満たしている場合は，表「可変長文字列型のデータ長一覧（抽象データ型の場合）」に従ってデータ長を計算してください。

また，対応する $atte_j$ の値を次に示す計算式に代入して，分岐行格納ページ数 ADTLS を P に加算してください。

$$ADTLS = \sum_{i=1}^{h} \uparrow atte_j \div (b-62) \uparrow \times a$$

属性が抽象データ型で定義されている場合は，次に示す計算式で属性のデータ長を求めてください。

$$att_j（バイト） = \sum_{k=1}^{h} ADT_k + 5$$

表 15-3　可変長文字列型のデータ長一覧（抽象データ型の場合）

データ型	条　件	データ長 att_j （バイト）	分岐部分のデータ長 $atte_j$（バイト）
VARCHAR(n)	d≧256	8	d + 2
	ノースプリットオプション指定あり	d + 3	0
NVARCHAR(n)	d≧128	8	2×d + 2
	ノースプリットオプション指定あり	2×d + 3	0
MVARCHAR(n)	d≧256	8	d + 2

データ型	条　件	データ長 att_j （バイト）	分岐部分の データ長 $atte_j$（バイト）
	ノースプリットオプション指定あり	d＋3	0

d：実際のデータ長（文字数）

(5)　繰返し列のデータ長の求め方

繰返し列のデータ長は，次に示す計算式で求めます。

計算式

$$d_i = 4 + (el_i + 1) \times en_i$$

el_i：繰返し列のデータ長

表「データ長一覧」から求めてください。

ただし，可変長文字列型の場合は，表「可変長文字列型のデータ長一覧（繰返し列の場合）」から求めてください。

en_i：繰返し列の平均要素数

表 15-4　可変長文字列型のデータ長一覧（繰返し列の場合）

データ型	条件	データ長 el_i （バイト）	分岐部分の データ長 es_j（バイト）
VARCHAR(n)	d≧256	5	d＋2
	ノースプリットオプション指定あり	d＋2	0
NVARCHAR(n)	d≧128	5	2×d＋2
	ノースプリットオプション指定あり	2×d＋2	0
MVARCHAR(n)	d≧256	5	d＋2
	ノースプリットオプション指定あり	d＋2	0

d：実際のデータ長（文字数）

可変長文字列型の繰返し列で，el_i の値が表「可変長文字列型のデータ長一覧（繰返し列の場合）」の条件を満たす列について，次に示す計算式の値を P に加算してください。

$$\uparrow \sum_{i=1}^{m} \{ es_i \times en_i + 14 \times (en_i - 1) \} \div (b - 62) \uparrow \times a$$

m：表「可変長文字列型のデータ長一覧（繰返し列の場合）」の条件を満たす可変長文字列型の繰返し列数

es_i：1 要素当たりの実際のデータ長の平均値

表「可変長文字列型のデータ長一覧（抽象データ型及び繰返し列を除く）」に示したデータ長を適用します。

(6) リバランス機能を使用する場合のエリア容量見積もり

ハッシュ関数 HASHA，HASHB，HASHC，HASHD，HASHE，HASHF を使用した分割表の場合，データは 1,024 個のハッシュ要素値に分けられ，値ごとに別々のセグメントに格納されます。

各分割 RD エリアには，平均（1024÷分割数）のハッシュ要素数のデータが格納されます。このため，各 RD エリアには，少なくともその RD エリアに格納される要素数分のセグメントを割り当てる必要があります。

リバランス機能を使用する場合の RD エリア容量は次のように見積もります。

1. データ件数N，行長 L，ページ長 P から，必要な総セグメント数 Sn を見積もります。
2. RD エリア当たりに必要なセグメント数 Ssn を見積もります。

   Ssn ＝↑Sn÷Srn↑×Srn

   Srn：↑1024÷Dvn↑

   Dvn：RD エリア分割数
3. 余裕値を考慮して，RD エリア当たりの使用中セグメント数 S を見積もります。

   S ＝↑（Ssn×K）÷Srn↑×Srn

   K：係数（例：余裕率 20%の場合，1.2）

(7) 表の格納ページ数の計算例

(a) 例題

次に示す在庫表の表格納ページ数を求めます。

品番	商品名	規格	単価	数量	原価
20180	掃除機	C20	20000	26	15000
20190	掃除機	C77	28000	105	23000
20130	冷蔵庫	P10	30000	70	25000
20220	テレビ	K18	35000	12	30000
20200	掃除機	C89	35000	30	30000
20140	冷蔵庫	P23	35000	60	30000
20280	アンプ	L10	38000	200	33000
20150	冷蔵庫	P32	48000	50	43000

品番	商品名	規格	単価	数量	原価
20290	アンプ	L50	49800	260	45000
20230	テレビ	K20	50000	15	45000
20160	冷蔵庫	P35	55800	120	50000

計算条件

1. 表に格納する行の総数：10000 件
2. ユーザ用 RD エリアのページ長：8192 バイト
3. CREATE TABLE で指定する未使用領域の比率：30%
4. 列数：6 列
5. 表を格納する RD エリアのセグメントサイズ：100 ページ
6. CREATE TABLE で指定するセグメント内の空きページ比率：40%
7. 列のデータ型：次に示します。
   品番：CHARACTER(5)
   商品名：NCHAR(4)
   規格：CHARACTER(3)
   単価：INTEGER
   数量：INTEGER
   原価：INTEGER
8. システム共通定義 pd_dbreuse_remaining_entries オペランドを指定していない

FIX 指定がない場合

1. 行長の計算

5（品番）＋ 2×4（商品名）＋ 3（規格）＋ 4（単価）＋ 4（数量）＋ 4（原価）＝ 28 バイト

2. P の計算

$$\left\lceil \frac{10000}{\left\lfloor \frac{8192 \times (100-30)}{100} \right\rfloor - 48} \right\rceil = 85$$

3. 表の格納ページ数の計算

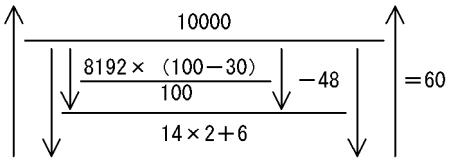
$$\left\lceil \frac{85 \times 100}{100 - \left\lceil \frac{100 \times 40}{100} \right\rceil} \right\rceil = 142\text{ページ}$$

FIX 指定がある場合

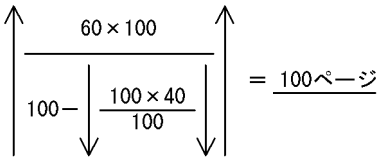
1. 行長の計算

5（品番）＋2×4（商品名）＋3（規格）＋4（単価）＋4（数量）＋4（原価）＝28バイト

2. Qの計算



3. 表の格納ページ数の計算



15.1.3 インデクスの格納ページ数の計算方法

インデクスの格納ページ数の計算方法を「計算方法」で説明します。「計算方法」の計算式中で使用する変数については「計算式中で使用する変数」で説明しています。インデクスの格納ページ数の計算例については「インデクスの格納ページ数の計算例」で説明しています。

なお，CREATE TABLEでクラスタキーを指定する場合，インデクスの格納ページ数を求める方法と同じ方法で，クラスタキーの格納ページ数を求めてください。

また，インデクスを横分割する場合，格納RDエリアごとにページ数を求めてください。

注意事項

インデクスページスプリットが発生すると，インデクスページ内のキーの格納比率を50：50にして二つのインデクスページに分割します。このため，インデクスの追加又は更新が多く発生すると，インデクスの格納ページ数は最大で見積もり式の2倍の容量が必要となります。また，最大キーが格納されたリーフページのインデクスページスプリットは，UAPからのINSERTであってもPCTFREEオペランドの値が考慮されます。

なお，インデクスページスプリットの発生回数を削減する方法の一つにアンバランスインデクススプリットがあります。インデクスページスプリット及び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。

(1) 計算方法

インデクスの格納ページ数は，次に示す計算式で求めます。

計算式

$$\text{インデクスの格納ページ数（単位：ページ）} = \sum_{i=1}^{n} Pi + P_d$$

Pi は，**計算式 1** に示す漸化式から求めます。

$P_n = 1$ となるまで P_{i+1} の計算をし，その計算結果の総計を求めてください。

P_d は，キー値の重複が 201 以上の場合に計算します。P_d の求め方を**計算式 2** に示します。

計算式 1

$$P_1 = \left\uparrow \frac{c-h}{MAX\left(1, \left\downarrow \frac{\left\downarrow \frac{a\times(100-b)}{100}\right\downarrow - g - 68}{18+g+4\times d}\right\downarrow\right)}\right\uparrow + \left\uparrow \frac{e}{MAX\left(1, \left\downarrow \frac{\left\downarrow \frac{a\times(100-b)}{100}\right\downarrow - g - 68}{18+g}\right\downarrow\right)}\right\uparrow + \left\uparrow \frac{h}{MAX\left(1, \left\downarrow \frac{\left\downarrow \frac{a\times(100-b)}{100}\right\downarrow - g - 68}{14+g}\right\downarrow\right)}\right\uparrow$$

$$P_{i+1} = \left\uparrow \frac{P_i}{MAX\left(1, \left\downarrow \frac{\left\downarrow \frac{a\times(100-b)}{100}\right\downarrow - g - 68}{14+g}\right\downarrow + 1\right)}\right\uparrow$$

計算式 2

$$Pd = \left\{ \left\uparrow \frac{f}{\left\downarrow \frac{\left\downarrow \frac{a\times 95}{100}\right\downarrow - 70}{4}\right\downarrow}\right\uparrow + 1 \right\} \times e$$

繰返し列を含むインデクスの場合の 1 行当たりの繰返し要素の重複数について

繰返し列を含むインデクスの場合，1 行当たりの繰返し要素の重複数は，次に示す計算式の値を超えないようにしてください。

```
重複数＝↓（↓a×0.95↓-82）÷4↓-1
```

(2) 計算式中で使用する変数

a：ユーザ用 RD エリアのページ長（バイト）

b：CREATE INDEX で指定する未使用領域の比率※1（%）

c：キー値の重複が 200 以下のキーの種類の個数（個）※2，3，4

d：キー値の重複が 200 以下のキーの重複数の平均値（個）※3，5

e：キー値の重複が 201 以上のキーの種類の個数（個）※3，4

f：キー値の重複が 201 以上のキーの重複数の平均値（個）※3，5

g：DB 格納キー長※6（バイト）

h：次に示すどちらかを代入します。

- ユニークインデクスの場合：ナル値以外のキーの種類の個数（個）
  なお，複数列インデクスの場合は，構成列にナル値を含まない全キー数となります。
- ユニークインデクス以外の場合：0

注※1

未使用領域の比率を指定しない場合は，30%を仮定して計算します。また，クラスタキーを指定する場合は，CREATE TABLE で指定する未使用領域の比率とします。

注※2

ユニークインデクスの重複がないキーを含める必要があります。

注※3

c×d ＋ e×f の値が，インデクスのキーの総数以上になるように計算してください。

注※4

ユニークインデクスの重複があるキー（キー値にナル値を含むことで重複するキー）を含める必要があります。

注※5

小数点未満は整数値に切り上げてください。

注※6

表「インデクスのキー長一覧」を参照してください。DB 格納キー長は，次に示す計算式で求めます。

- 単一列インデクス及び固定長複数列インデクスの場合
  ↑キー長÷4↑×4
- 可変長複数列インデクスで，かつキー長 255 バイト以下の場合
  ↑（キー長＋ 1）÷4↑×4
- 可変長複数列インデクスで，かつキー長 256 バイト以上の場合

↑（キー長＋ 2）÷4↑×4

ただし，複数列インデクスのキー長は，表「インデクスのキー長一覧」を基に全構成列のキー長を加算したものとなります。

表 15-5　インデクスのキー長一覧

分類	データ型	キー長（単位：バイト）					
		キー長が 255 バイト以下			キー長が 256 バイト以上		
		単一列インデクス	複数列インデクス		単一列インデクス	複数列インデクス	
			固定長※1	可変長※2		固定長※1	可変長※2
数値データ	INTEGER	4	5	6	−	5	7
	SMALLINT	2	3	4	−	3	5
	LARGE DECIMAL (m,n)※3	↓m÷2↓＋1	↓m÷2↓＋2	↓m÷2↓＋3	−	↓m÷2↓＋2	↓m÷2↓＋4
	FLOAT 又は DOUBLE PRECISION	8	×	×	−	×	×
	SMALLFLT 又は REAL	4	×	×	−	×	×
文字データ	CHARACTER (n)	n	n＋1	n＋2	n	n＋1	n＋3
	VARCHAR(n)	a＋1	−	a＋2	a＋2	−	a＋3
各国文字データ	NCHAR(n) 又は NATIONAL CHARACTER (n)	2×n	2×n＋1	2×n＋2	2×n	2×n＋1	2×n＋3
	NVARCHAR (n)	2×b＋1	−	2×b＋2	2×b＋2	−	2×b＋3
混在文字データ	MCHAR(n)	n	n＋1	n＋2	n	n＋1	n＋3
	MVARCHAR (n)	a＋1	−	a＋2	a＋2	−	a＋3
日付データ	DATE	4	5	6	−	5	7
時刻データ	TIME	3	4	5	−	4	6

分類	データ型	キー長（単位：バイト）					
		キー長が 255 バイト以下			キー長が 256 バイト以上		
		単一列インデクス	複数列インデクス		単一列インデクス	複数列インデクス	
			固定長※1	可変長※2		固定長※1	可変長※2
日間隔データ	INTERVAL YEAR TO DAY	5	6	7	－	6	8
時間隔データ	INTERVAL HOUR TO SECOND	4	5	6	－	5	7
時刻印データ	TIMESTAMP(p)	7 ＋（p÷2）	8 ＋（p÷2）	9 ＋（p÷2）	－	8 ＋（p÷2）	10 ＋（p÷2）

a：実際のデータ長

b：実際の文字数

m，n，p：正の整数

×：インデクス定義時にエラーになります。

－：該当しません。

注

最初は「キー長が 255 バイト以下」で計算してください。その結果，キー長が 256 バイト以上になる場合は，「キー長が 256 バイト以上」で再計算してください。

注※1

構成列が固定長の列だけのインデクスのキー長です。

注※2

構成列に可変長の列を含むインデクスのキー長です。

注※3

全体のけた数が m けたで，小数点以下のけた数が n けたの固定小数点数です。m を省略した場合は 15 を仮定します。

参考

非ユニークインデクスは，インデクスのデータの格納領域中に重複数を格納する領域を持つため，その分容量が大きくなります。一方，ユニークインデクスは重複数を格納する領域を持ちません。そのため，非ユニークインデクスよりもユニークインデクスの方が容量が小さくなります。

(3) インデクスの格納ページ数の計算例

(a) 例題 1

次に示す在庫表の「品番」をユニークインデクス（重複するキーがない）とする場合のインデクス格納ページ数を求めます。

品番	商品名	規格	単価	数量	原価
20180	掃除機	C20	20000	26	15000
20190	掃除機	C77	28000	105	23000
20130	冷蔵庫	P10	30000	70	25000
20220	テレビ	K18	35000	12	30000
20200	掃除機	C89	35000	30	30000
20140	冷蔵庫	P23	35000	60	30000
20280	アンプ	L10	38000	200	33000
20150	冷蔵庫	P32	48000	50	43000
20290	アンプ	L50	49800	260	45000
20230	テレビ	K20	50000	15	45000
20160	冷蔵庫	P35	55800	120	50000

計算条件

1. インデクスのキーの総数：10,000 件
2. ユーザ用 RD エリアのページ長：8,192 バイト
3. CREATE INDEX で指定する未使用領域の比率：30%
4. インデクスのデータ型：CHARACTER
5. インデクスのキー長：5 バイト
6. キーの重複数：1

計算式

ＤＢ格納キー長（ｇ）＝⌈５÷４⌉×４＝８

$$P_1 = \left\lceil \frac{10000-10000}{\mathrm{MAX}\left(1,\ \left\lfloor \frac{\left\lfloor \frac{8192\times(100-30)}{100} \right\rfloor - 8 - 68}{18+8+4\times1} \right\rfloor\right)} \right\rceil + \left\lceil \frac{0}{\mathrm{MAX}\left(1,\ \left\lfloor \frac{\left\lfloor \frac{8192\times(100-30)}{100} \right\rfloor - 8 - 68}{18+8} \right\rfloor\right)} \right\rceil + \left\lceil \frac{10000}{\mathrm{MAX}\left(1,\ \left\lfloor \frac{\left\lfloor \frac{8192\times(100-30)}{100} \right\rfloor - 8 - 68}{14+8} \right\rfloor\right)} \right\rceil$$

＝０＋０＋３９＝３９

$$P_2 = \left\lceil \frac{39}{\mathrm{MAX}\left(1,\ \left\lfloor \frac{\left\lfloor \frac{8192\times(100-30)}{100} \right\rfloor - 8 - 68}{14+8} \right\rfloor + 1\right)} \right\rceil = 1$$

Ｐｄ＝０（キー値の重複が２００以下となるため）

インデクスの格納ページ数＝３９＋１＋０＝　４０ページ

(b)　例題２

例題１に示す在庫表の「商品名」をインデクス（重複するキーがある）とする場合のインデクス格納ページ数を求めます。

計算条件

1. インデクスのキーの総数：10,000 件
2. ユーザ用 RD エリアのページ長：8,192 バイト
3. CREATE INDEX で指定する未使用領域の比率：30%
4. インデクスのデータ型：NCHAR
5. インデクスのキー長：4 文字（漢字）
6. キー値の重複が 201 以上のキーの種類の個数

（このときの平均重複数は 250）：1

7. キー値の重複が 200 以下のキーの種類の個数

（このときの平均重複数は 5）：（10000－250）÷5 ＝ 1950

計算式

ＤＢ格納キー長（ｇ）＝ ↑（２×４）÷４↑ ×４＝８

$$P_1 = \left\uparrow \frac{1950-0}{\mathrm{MAX}\left(1,\ \left\downarrow \frac{\left\downarrow \frac{8192\times(100-30)}{100}\right\downarrow -8-68}{18+8+4\times5}\right\downarrow\right)}\right\uparrow + \left\uparrow \frac{1}{\mathrm{MAX}\left(1,\ \left\downarrow \frac{\left\downarrow \frac{8192\times(100-30)}{100}\right\downarrow -8-68}{18+8}\right\downarrow\right)}\right\uparrow + \left\uparrow \frac{0}{\mathrm{MAX}\left(1,\ \left\downarrow \frac{\left\downarrow \frac{8192\times(100-30)}{100}\right\downarrow -8-68}{14+8}\right\downarrow\right)}\right\uparrow$$

＝１６＋１＋０＝１７

$$P_2 = \left\uparrow \frac{17}{\mathrm{MAX}\left(1,\ \left\downarrow \frac{\left\downarrow \frac{8192\times(100-30)}{100}\right\downarrow -8-68}{14+8}\right\downarrow +1\right)}\right\uparrow = 1$$

$$Pd = \left\{ \left\uparrow \frac{250}{\left\downarrow \frac{\left\downarrow \frac{8192\times95}{100}\right\downarrow -70}{4}\right\downarrow}\right\uparrow +1 \right\} \times 1 = 2$$

インデクスの格納ページ数＝１７＋１＋２＝ ２０ページ

次に示す会員表の「性別」と「入会年度」を複数列インデクスとする場合のインデクス格納ページ数を求めます。

会員番号	氏名	年齢	性別	入会年度
0001	安東洋子	18	F	1983

会員番号	氏名	年齢	性別	入会年度
0002	浅井順一	25	M	1967
0003	石井公子	24	F	1987
0004	石井一郎	25	M	1964
・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・	・ ・ ・
1000	渡辺英雄	30	M	1995

計算条件

1. インデクスのキーの総数：10,000 件
2. ユーザ用 RD エリアのページ長：8,192 バイト
3. CREATE INDEX で指定する未使用領域の比率：30%
4. 1964 年度の入会人数：1,000 人
5. ほかの年度の入会人数：200 人以下
6. 入会年度：1965～1995 の 31 年間
7. 入会する男女の数は，毎年同数とします。
8. 列のデータ型：次に示します。
   会員番号：CHARACTER(5)
   氏名：NCHAR(4)
   年齢：INTEGER
   性別：CHARACTER(4)
   入会年度：INTEGER

計算式

1. 1965 年度以降の 31 年間に入会した人（男，女含めて 200 人以下）のキーの種類の個数（c）は，c ＝ 31×2 ＝ 62 となります。
2. 重複数の平均値（d）は，d ＝（10000－1000）÷62 ＝ 146 となります。
3. 1964 年の 1 年間に入会した人（男，女含めて 1000 人）のキーの種類の個数（e）は，e ＝ 2 となります。
4. 重複数の平均値（f）は，f ＝ 1000÷2 ＝ 500 となります。
5. 性別と入会年度の DB 格納キー長（g）は，
   g ＝↑（4 ＋ 1 ＋ 5）÷4↑×4 ＝ 12 となります。

$$P_1 = \left\lceil \frac{62-0}{MAX\left(1,\ \left\lfloor \frac{\left\lfloor \frac{8192\times(100-30)}{100} \right\rfloor -12-68}{18+12+4\times146} \right\rfloor\right)} \right\rceil + \left\lceil \frac{2}{MAX\left(1,\ \left\lfloor \frac{\left\lfloor \frac{8192\times(100-30)}{100} \right\rfloor -12-68}{18+12} \right\rfloor\right)} \right\rceil + \left\lceil \frac{0}{MAX\left(1,\ \left\lfloor \frac{\left\lfloor \frac{8192\times(100-30)}{100} \right\rfloor -12-68}{14+12} \right\rfloor\right)} \right\rceil$$

＝７＋１＋０＝８

$$P_2 = \left\lceil \frac{8}{MAX\left(1,\ \left\lfloor \frac{\left\lfloor \frac{8192\times(100-30)}{100} \right\rfloor -12-68}{14+12} \right\rfloor + 1\right)} \right\rceil = 1$$

$$Pd = \left\{ \left\lceil \frac{500}{\left\lfloor \frac{\left\lfloor \frac{8192\times95}{100} \right\rfloor -70}{4} \right\rfloor} \right\rceil + 1 \right\} \times 2 = 4$$

インデクスの格納ページ数＝８＋１＋４＝　１３ページ

15.2　データディクショナリ用 RD エリアの容量の見積もり

データディクショナリ用 RD エリアは，データベース構成変更ユティリティ（pdmod）の create rdarea 文の指定によって次に示す 2 種類作成できます。

- 通常のデータディクショナリ用 RD エリア
  create rdarea 文に datadictionary，又は datadictionary of routines を指定
- 解析情報表及び運用履歴表を格納するデータディクショナリ用 RD エリア
  create rdarea 文に datadictionary of dbmanagement を指定

上記の RD エリアは，それぞれの種類ごとに容量を見積もる必要があります。

15.2.1　通常のデータディクショナリ用 RD エリアの容量の見積もり

(1)　データディクショナリ用 RD エリアの容量の求め方

create rdarea 文に datadictionary 又は datadictionary of routines を指定する場合のデータディクショナリ用 RD エリアの容量は，次に示す計算式で求めます。

計算式

```
データディクショナリ用RDエリアの容量（単位：バイト）
　＝a×b×1.3＋c×125＋1600000
●拡張システム定義スカラ関数を定義する場合に加算します。
　＋1933312
```

a：データディクショナリ用 RD エリアのページ長※1

b：データディクショナリ用 RD エリアの総ページ数※2

c：データディクショナリ用 RD エリアのセグメントサイズ※3

注※1
　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）の create rdarea 文で指定するページ長です。

注※2
　「表の格納ページ数＋インデクスの格納ページ数」です。「表の格納ページ数の計算方法」及び「インデクスの格納ページ数の計算方法」を参照してください。

注※3
　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）の create rdarea 文で指定するセグメントサイズです。

(2) 表の格納ページ数の計算方法

表の格納ページ数（単位：ページ）は，計算式 1～計算式 24 の総和になります。

(a) 計算式 1

ディクショナリ表名	計算式
SQL_TABLES	$\uparrow \frac{a}{\downarrow \frac{b-48}{\uparrow (h\times 2+i\times 2+e+d+f+w+657) \div 2 \uparrow} \times 2} \downarrow$
SQL_COLUMNS	●DEFAULT句を指定しない場合，及びDEFAULT句に指定した既定値のデータ長が255バイト以下の場合 $\uparrow \frac{a\times g}{\downarrow \frac{b-48}{\uparrow (h\times 2+i+c+j+k+m+n+p+t+y+A+310) \div 2 \uparrow} \times 2} \downarrow$ ●DEFAULT句に指定した既定値のデータ長が256バイト以上の場合 $\uparrow \frac{a\times g}{\downarrow \frac{b-48}{\uparrow (h\times 2+i+c+j+k+m+n+p+t+299+\downarrow (y+A) \div 32000 \downarrow \times 6+8) \div 2 \uparrow} \times 2} \downarrow + a\times g\times \uparrow (w+y) \div (b-61) \uparrow$
SQL_DIV_TABLE	$\uparrow \frac{a\times C}{\downarrow \frac{b-48}{\uparrow (h+i+d+v+B\times 2+84) \div 2 \uparrow} \times 2} \downarrow$

a：表の総数（個）

b：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長－510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

c：列の名称長の平均値（バイト）

d：表を格納する RD エリアの名称長の平均値（バイト）

e：表の注釈長の平均値（バイト）

f：表の横分割条件を指定する列の名称長の平均値（バイト）

g：表の列数の平均値（個）

h：認可識別子の長さの平均値（バイト）

i：表識別子の長さの平均値（バイト）

j：列の注釈長の平均値（バイト）

k：ビュー表の基になる表の認可識別子の長さの平均値（バイト）

m：ビュー表の基になる表の表識別子の長さの平均値（バイト）

n：ビュー表の基になる表の列の名称長の平均値（バイト）

p：ユーザ定義型の名称長の平均値（バイト）

q：表の横分割条件数の平均値（個）

t：PLUGIN 句指定長の平均値（バイト）

v：分割キーの長さの平均値（バイト）

w：挿入履歴保持列の名称の平均値（バイト）

y：DEFAULT 句に指定した既定値の実長の平均値（バイト）

実長の算出方法は，マニュアル「HiRDB UAP 開発ガイド」の「SQL 記述領域に設定するデータコードとデータの長さ」を参照してください。

A：DEFAULT 句に指定した既定値の長さの平均値（バイト）

指定した既定値が定数の場合は，定数表現の長さです。既定値の長さを長く変更する可能性がある場合は変更後の長さを考慮して算出してください。文字型の定数の場合，各国文字列定数を表す N，混在文字列定数を表す M，16 進文字列定数を表す X，引用符（’）を長さに含みます。それ以外は指定した既定値のバイト数です。

（例）

'HiRDB'：7 バイト

N'Ｈｉ ＲＤＢ'：13 バイト

X'4869524442'：13 バイト

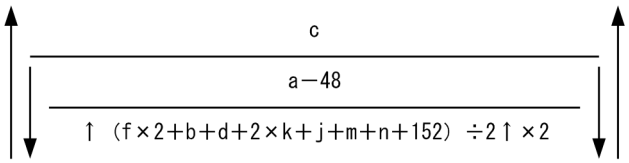
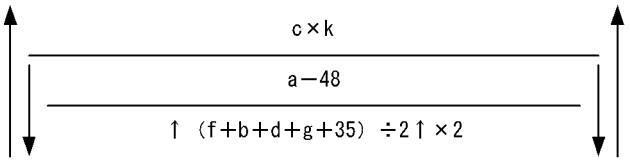
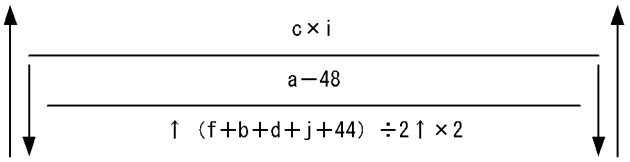
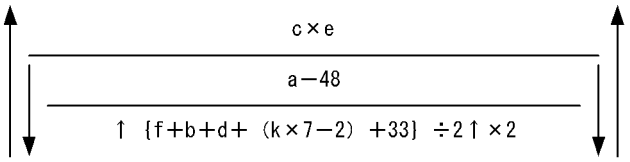
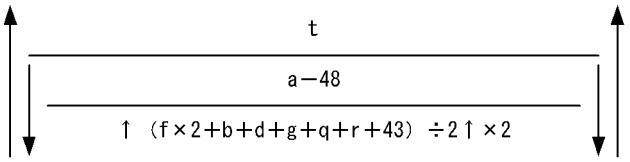
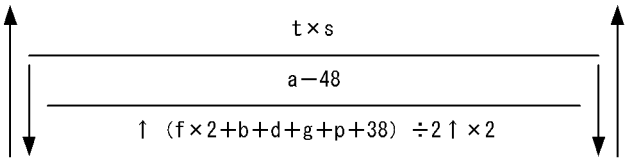
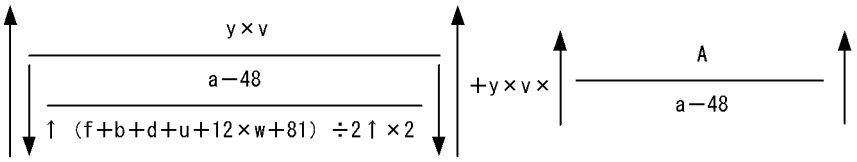
CURRENT_TIME：12 バイト

100：3 バイト

B：表の横分割条件長の平均値（バイト）

C：表格納用 RD エリア指定数（個）

(b) 計算式 2

ディクショナリ表名	計算式
SQL_INDEXES	
SQL_INDEX_ COLINF	
SQL_DIV_INDEX	
SQL_EXCEPT	
SQL_INDEX_ DATATYPE	
SQL_INDEX_ FUNCTION	
SQL_INDEX_ XMLINF	

a：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合

データディクショナリ用 RD エリアのページ長－510（バイト）

- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

b：表識別子の長さの平均値（バイト）

c：インデクスの総数（個）

d：インデクス識別子の長さの平均値（バイト）

e：1 インデクス当たりのインデクス除外キー値数の平均値（個）

f：認可識別子の長さの平均値（バイト）

g：列の名称長の平均値（バイト）

i：表の横分割条件数の平均値（個）

j：インデクスを格納する RD エリアの名称の長さの平均値（バイト）

k：インデクスを構成する列数の平均値（個）

m：インデクス型名称の長さの平均値（バイト）

n：PLUGIN 句指定長の平均値（バイト）

p：プラグインインデクス適用関数名称の長さの平均値（バイト）

q：抽象データ型名称の長さの平均値（バイト）

r：属性名称の長さの平均値（バイト）

s：1 プラグインインデクス当たりの適用関数の数（個）

t：プラグインインデクスの総数（個）

u：部分構造パス長の平均値（バイト）

v：部分構造インデクスを構成する部分構造パス数の平均値（個）

w：データ長が 256 バイト以上で，かつ分岐するバイナリデータ（部分構造パス用解析ツリー）数の平均値（個）

バイナリデータの格納ページ数の分岐条件については，表「データ長一覧」を参照してください。

y：部分構造インデクスの総数（個）

A：部分構造パス用解析ツリー長（バイト）

次の計算式で求められる値になります。

S×120 ＋ P ＋ L ＋ S×4 ＋ 32

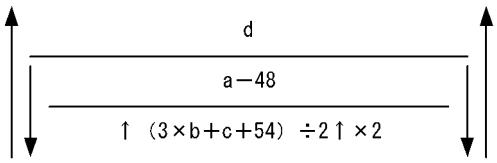
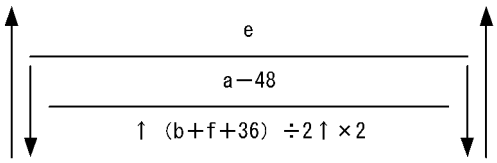
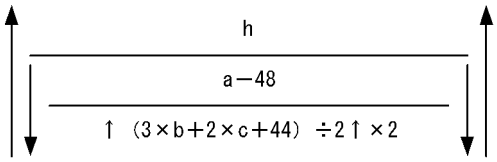
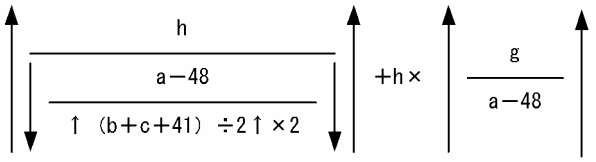
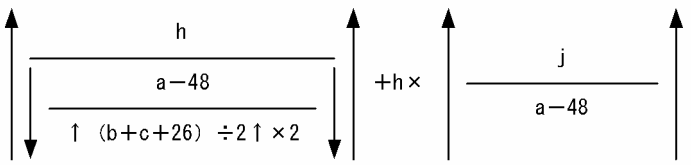
L：ステップ式の修飾名に指定した局所名の文字列表現長の合計値（バイト）※

P：接頭辞に関連づけた XML 名前空間 URI の文字列表現長の合計値（バイト）※
　接頭辞を省略した場合は，省略時に関連づけられる XML 名前空間 URI

S：ステップ式の指定数（個）

注※　４の倍数に切り上げてください。

(c)　計算式３

ディクショナリ表名	計算式
SQL_TABLE_ PRIVILEGES	
SQL_RDAREA_ PRIVILEGES	
SQL_VIEW_ TABLE_USAGE	
SQL_VIEWS	
SQL_VIEW_DEF※	

a：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長－510（バイト）

- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

b：認可識別子の長さの平均値（バイト）

c：表識別子の長さの平均値（バイト）

d：アクセス権限の定義数（個）

- 一つの表に対して n 人に権限を与えた場合は，権限を与えた表数×n 個と計算します。
- PUBLIC に対して権限を与えた場合は，1 人として計算します。
- グループに対して権限を与えた場合は，一つのグループ ID を 1 人として計算します。

e：RD エリアの総数（個）

f：表を格納する RD エリアの名称の長さの平均値（バイト）

g：ビュー表を定義するときの SQL 文の長さの平均値（バイト）

h：ビュー定義の総数（個）

j：ビュー解析情報の平均長（バイト）
1 ビュー表当たりのビュー解析情報長については，マニュアル「HiRDB システム定義」の「ビュー解析情報用バッファ長（pd_view_def_cache_size）の見積もり式」を参照してください。

注※　システムが使用する表です。

(d)　計算式 4

ディクショナリ表名	計算式
SQL_REFERENTIAL_CONSTRAINTS	$\left\uparrow \frac{f}{\left\uparrow \frac{a-48}{(b+3\times c+2\times d+E+F+G+H+186)\div 2\uparrow \times 2}\right\downarrow}\right\downarrow$

E：{e×h ＋ 2×h ＋ （h－1)} ＋ 1

F：{e×i ＋ 2×i ＋ （i－1)} ＋ 1

G：{2×h ＋ （h－1)} ＋ 1

H：{2×i ＋ （i－1)} ＋ 1

a：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合

データディクショナリ用 RD エリアのページ長－510（バイト）

- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

b：制約名称の長さの平均値（バイト）

c：認可識別子の長さの平均値（バイト）

d：表識別子の長さの平均値（バイト）

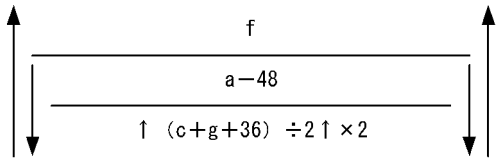
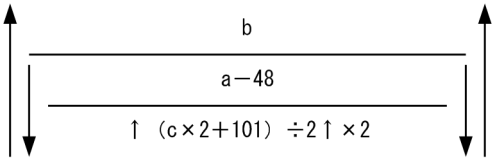
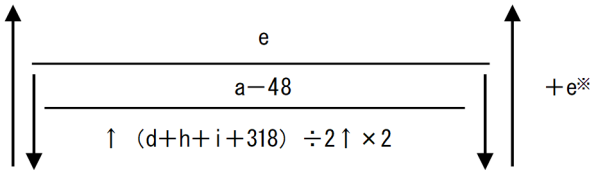
e：外部キーを定義した列名の長さの平均値（バイト）

f：主キーを定義した列名の長さの平均値（バイト）

h：外部キーを構成する列数の平均値（個）

i：主キーを構成する列数の平均値（個）

(e) 計算式 5

ディクショナリ表名	計算式
SQL_PHYSICAL_FILES	
SQL_RDAREAS	
SQL_USERS	 注※　CONNECT関連セキュリティ機能のパスワードの文字列制限でパスワード再利用禁止の個数(REUSE MAX指定値)が8以上の場合に加算してください。

a：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長－510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合

データディクショナリ用 RD エリアのページ長（バイト）

b：RD エリアの総数（個）

c：RD エリアの名称の長さの平均値（バイト）

d：スキーマの認可識別子の長さの平均値（バイト）

e：スキーマ総数（個）

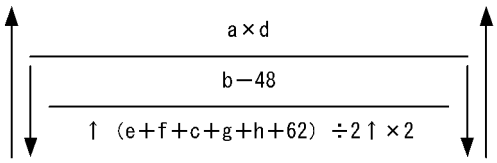
f：全 RD エリアを構成する HiRDB ファイルの総数（個）

g：全 RD エリアを構成する HiRDB ファイルの名称の長さの平均値（バイト）

h：パスワード長の平均値（バイト）

i：スキーマ操作権限対象のスキーマ名長の平均値（バイト）

(f) 計算式 6

ディクショナリ表名	計算式
SQL_DIV_TABLE_ REGULARIZE※	

a：横分割表の総数（個）

b：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長−510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

c：表を格納する RD エリアの名称長の平均値（バイト）

d：表の横分割条件数の平均値（個）

e：認可識別子の長さの平均値（バイト）

f：表識別子の長さの平均値（バイト）

g：表の横分割条件を指定する列が文字列のときの条件値長の平均値（バイト）

h：表の横分割条件を指定する列が数値列のときの条件値長の平均値（バイト）

注※　システムが使用する表です。

(g)　計算式７

ディクショナリ表名	計算式
SQL_TABLE_ STATISTICS※1	$\left\uparrow \frac{a}{\left\uparrow \frac{b-48}{\uparrow (e+f+50) \div 2 \uparrow \times 2} \right\downarrow} \right\downarrow$
SQL_COLUMN_ STATISTICS※1	$\left\uparrow \frac{g}{\left\uparrow \frac{b-48}{\uparrow (e+f+c+34) \div 2 \uparrow \times 2} \right\downarrow} \right\downarrow + g \times \left\uparrow \frac{MIN(64+24 \times i, 2872)}{b-48} \right\downarrow$
SQL_INDEX_ STATISTICS※1	$\left\uparrow \frac{g}{\left\uparrow \frac{b-48}{\uparrow (e+f+h+76) \div 2 \uparrow \times 2} \right\downarrow} \right\downarrow$

a：最適化情報を取得する表の総数（個）

b：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長－510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

c：列の名称長の平均値（バイト）

e：認可識別子の長さの平均値（バイト）

f：表識別子の長さの平均値（バイト）

g：最適化情報を取得する表に定義するインデクスの総数（個）

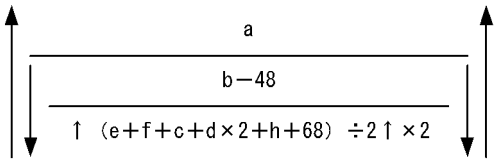
h：インデクス識別子の長さの平均値（バイト）

i：最適化情報を取得する表に定義するインデクスのキー列値数※2（個）

注※1　システムが使用する表です。

注※2　キー列値数 ＜ 100 の場合，i ＝キー列値数
　　キー列値数 ≧ 100 の場合，i ＝ 100

(h)　計算式 8

ディクショナリ表名	計算式
SQL_DIV_ COLUMN	

a：LOB 列を定義した表の総数（個）

b：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長−510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

c：列の名称長の平均値（バイト）

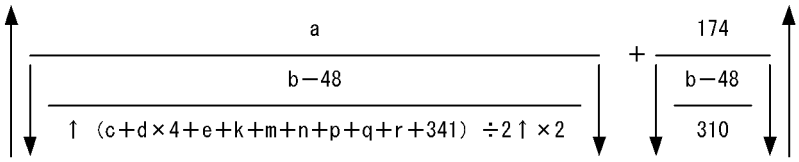
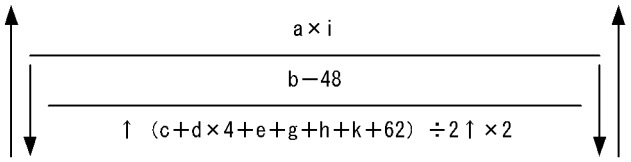
d：表を格納する RD エリアの名称の長さの平均値（バイト）

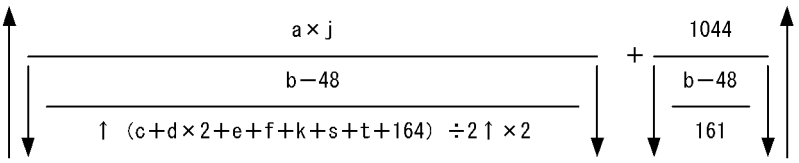
e：認可識別子の長さの平均値（バイト）

f：表識別子の長さの平均値（バイト）

h：コンポネント名称の長さの平均値（バイト）

(i)　計算式 9

ディクショナリ表名	計算式
SQL_ROUTINES	
SQL_ROUTINE_ RESOURCES	

ディクショナリ表名	計算式
SQL_ROUTINE_ PARAMS	

a：ルーチンの総数（個）

b：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長−510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

c：ルーチン名称の長さの平均値（バイト）

d：認可識別子の長さの平均値（バイト）

e：特定名※1 の長さの平均値（バイト）

f：パラメタ名称の長さの平均値（バイト）

g：リソース※2 の所有者の認可識別子の長さの平均値（バイト）

h：リソース※2 名称の長さの平均値（バイト）

i：1 ルーチン当たりのリソース※2 数の平均値（個）

j：1 ルーチン当たりのパラメタ数の平均値（個）

k：抽象データ型の名称長の平均値（バイト）

m：ユーザ定義型の名称長（戻り値）の平均値（バイト）

n：外部ルーチン名称長の平均値（バイト）

p：Java クラス名称長の平均値（バイト）

q：Java アーカイブ名称長の平均値（バイト）

r：Java 戻り値のデータ型名称長の平均値（バイト）

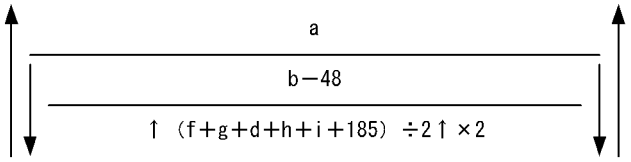
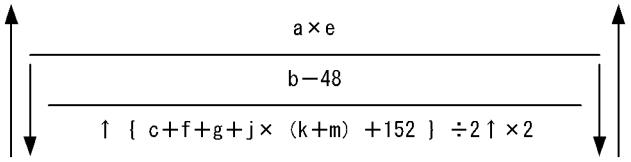
s：Java パラメタデータ型の名称長の平均値（バイト）

t：新旧値相関名で使用した列名称の長さの平均値（バイト）

注※1　「認可識別子.ルーチン識別子」を表しています。

注※2　表及びインデクスを表しています。

(j)　計算式 10

ディクショナリ表名	計算式
SQL_DATATYPES	
SQL_DATATYPE_ DESCRIPTORS	

a：ユーザ定義型の総数（個）

b：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長−510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

c：属性又はフィールド名の長さの平均値（バイト）

d：ユーザ定義型の注釈の長さの平均値（バイト）

e：1 データ型当たりの属性数の平均値（個）

f：認可識別子の長さの平均値（バイト）

g：データ型識別子の長さの平均値（バイト）

h：スーパタイプの抽象データ型の認可識別子の長さの平均値（バイト）

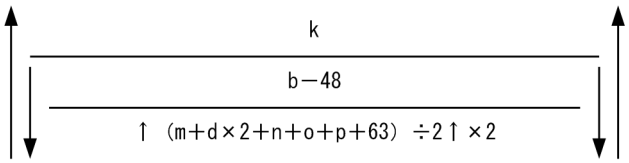
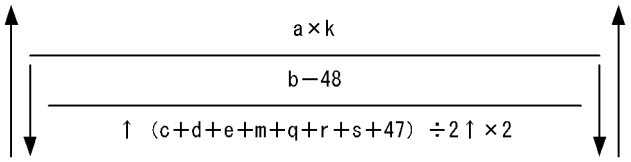
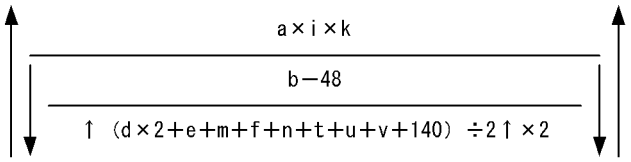
i：スーパタイプの抽象データ型のデータ識別子の長さの平均値（バイト）

j：ユーザ定義型で定義された属性数（個）

k：ユーザ定義型で定義された属性の抽象データ型の認可識別子の長さの平均値（バイト）

m：ユーザ定義型で定義された属性の抽象データ型のデータ識別子の長さの平均値（バイト）

(k) 計算式 11

ディクショナリ表名	計算式
SQL_PLUGINS	
SQL_PLUGIN_ ROUTINES	
SQL_PLUGIN_ ROUTINE_PARAMS	

a：1 プラグイン当たりのプラグインルーチン数の平均値（個）

b：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長－510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

c：ルーチン名称の長さの平均値（バイト）

d：認可識別子の長さの平均値（バイト）

e：特定名※の長さの平均値（バイト）

f：1 プラグイン当たりのパラメタ名称の長さの平均値（バイト）

i：1 ルーチン当たりのリソース数の平均値（個）

k：プラグインの総数（個）

m：プラグイン名称の長さの平均値（バイト）

n：抽象データ型／インデクス型名称の長さの平均値（バイト）

o：プラグインライブラリパス名称の長さの平均値（バイト）

p：プラグインの注釈の長さの平均値（バイト）

q：契機指示子の長さの平均値（バイト）

r：オペレーション修飾子の平均値（バイト）

s：オペレーション修飾コード長の平均値（バイト）

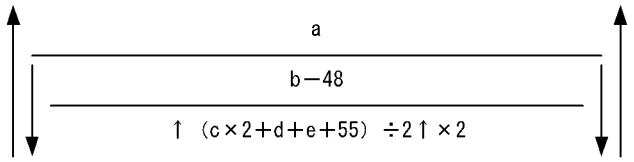
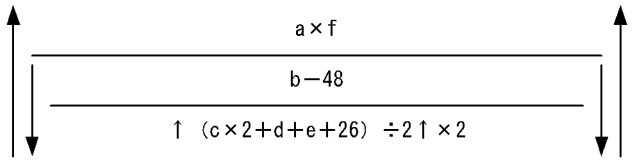
t：パラメタ修飾情報長の平均値（バイト）

u：バインドオペレーション名称長の平均値（バイト）

v：パラメタ修飾情報コード長の平均値（バイト）

注※ 「認可識別子.ルーチン識別子」を表しています。

(l) 計算式 12

ディクショナリ表名	計算式
SQL_INDEX_TYPES	
SQL_INDEX_TYPE_FUNCTION	

a：インデクス型の総数（個）

b：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長－510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

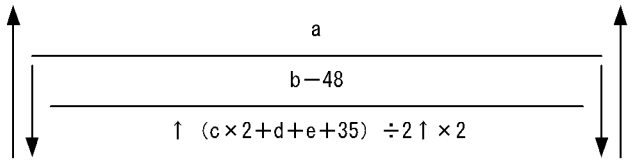
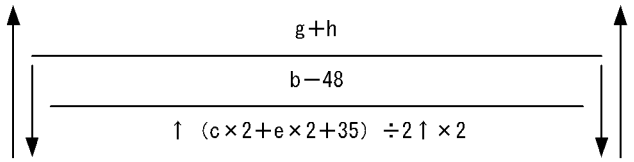
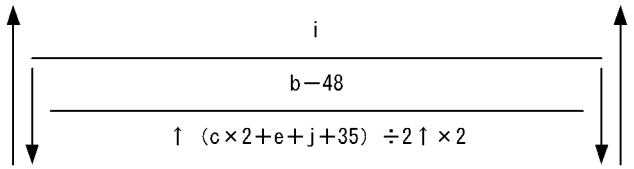
c：認可識別子の長さの平均値（バイト）

d：インデクス型識別子の長さの平均値（バイト）

e：抽象データ型名称の長さの平均値（バイト）

f：1 インデクス型当たりの適用関数の数（個）

(m) 計算式 13

ディクショナリ表名	計算式
SQL_INDEX_ RESOURCES	
SQL_TYPE_ RESOURCES	
SQL_TABLE_ RESOURCES	

a：プラグインインデクスの総数（個）

b：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長−510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

c：認可識別子の長さの平均値（バイト）

d：インデクス型識別子の長さの平均値（バイト）

e：抽象データ型名称の長さの平均値（バイト）

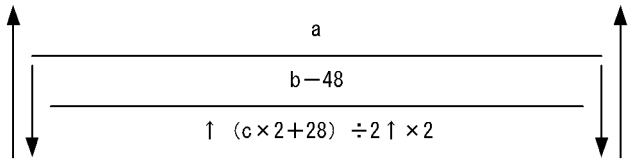
g：抽象データ型で定義された属性の総数（個）

h：サブタイプとして定義された抽象データ型の総数（個）

i：抽象データ型の総数（個）

j：表識別子の平均長（バイト）

(n) 計算式 14

ディクショナリ表名	計算式
SQL_IOS_GENERATIONS	

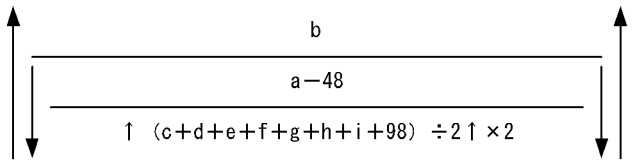
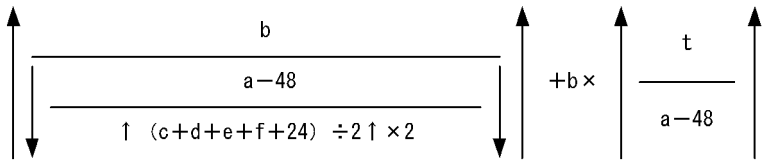
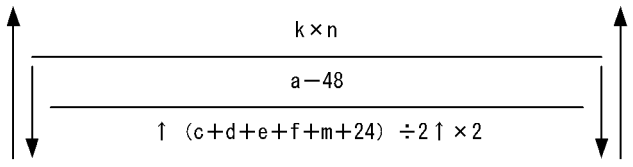
a：レプリカ RD エリアを構成する HiRDB ファイル数（個）

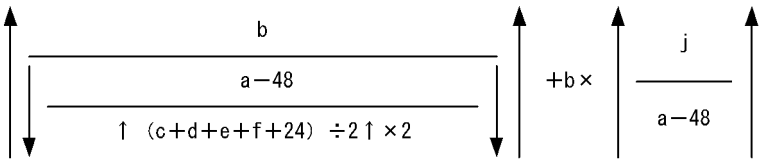
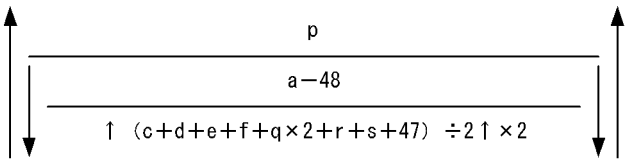
b：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長−510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

c：RD エリアを構成する HiRDB ファイルの名称長の平均値（バイト）

(o) 計算式 15

ディクショナリ表名	計算式
SQL_TRIGGERS	
SQL_TRIGGER_ACTCOND※	
SQL_TRIGGER_COLUMNS	

ディクショナリ表名	計算式
SQL_TRIGGER_DEF_SOURCE	
SQL_TRIGGER_USAGE	

a：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長−510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

b：トリガ定義の総数（個）

c：トリガ認可識別子の長さの平均値（バイト）

d：トリガ名称の長さの平均値（バイト）

e：トリガを定義した表の認可識別子の長さの平均値（バイト）

f：トリガを定義した表名称の長さの平均値（バイト）

g：旧値相関名称の長さの平均値（バイト）

h：新値相関名称の長さの平均値（バイト）

i：トリガ動作手続きの特定名称の長さの平均値（バイト）

j：トリガ定義時の SQL 文の長さの平均値（バイト）

k：UPDATE 文を契機とするトリガの定義数（個）

m：トリガの実行契機となる列に指定した列名称の長さの平均値（バイト）

n；トリガの実行契機となる列に指定した列数の平均値（個）

p：トリガ動作の探索条件中のリソース数（個）

q：トリガ動作の探索条件中のリソースの認可識別子の長さの平均値（バイト）

r：トリガ動作の探索条件中のリソースの表名称の長さの平均値（バイト）

s：トリガ動作の探索条件中のリソースの特定名称の長さの平均値（バイト）

t：トリガ動作条件の解析ツリー長（バイト）

次の計算式で求められる値になります。なお，この計算式中の変数はすべて WHEN の探索条件での指定内容です。

S×36 ＋ T ＋ U×48 ＋ V×128

＋ F1×420 ＋ F2×132 ＋ F3×124 ＋ F4×296 ＋ F5×F4×132

＋ A×140 ＋ B×200 ＋ 1000

A：コンポーネント指定の属性数（個）

B：値式に現れる抽象データ型の数（個）

F1：システム定義スカラ関数の数（個）

F2：システム定義スカラ関数の引数の数（個）

F3：ユーザ定義関数の数（個）

F4：ユーザ定義関数の候補となる関数の数（個）

F5：ユーザ定義関数の引数の数（個）

S：論理演算子，算術演算子，定数，及びシステム組込みスカラ関数の合計数（個）

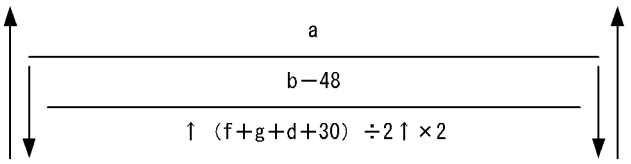
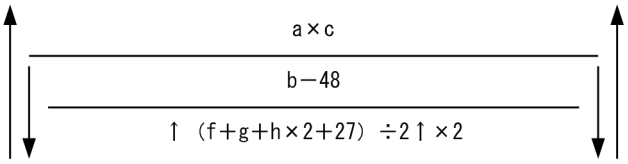
T：定数（HiRDB で解釈するデータ型）の合計長（バイト）

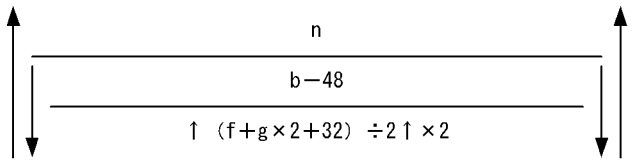
U：スカラ関数 VALUE，CASE 式，CAST 指定中の値式の数（個）

V：列指定の数（個）

注※　システムが使用する表です。

(p)　計算式 16

ディクショナリ表名	計算式
SQL_PARTKEY	
SQL_PARTKEY_DIVISION	

ディクショナリ表名	計算式
SQL_DIV_TYPE	

a：作成するマトリクス分割表の数（個）

b：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長−510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

c：表の横分割条件値数の平均値（個）

d：表の横分割条件を指定する列の名称長の平均値（バイト）

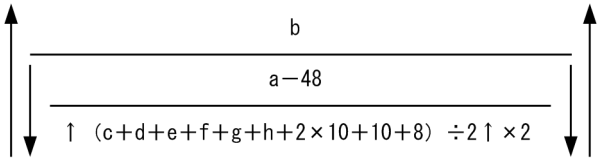
f：認可識別子の長さの平均値（バイト）

g：表識別子の長さの平均値（バイト）

h：表の横分割条件値の長さの平均値（バイト）

n：境界値指定の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数（個）

(q) 計算式 17

ディクショナリ表名	計算式
SQL_AUDITS	

a：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長−510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

b：監査対象イベントの数（個）（CREATE AUDIT の実行回数（回））

c：イベントタイプ名称の長さの平均値（バイト）

d：イベントサブタイプ名称の長さの平均値（バイト）

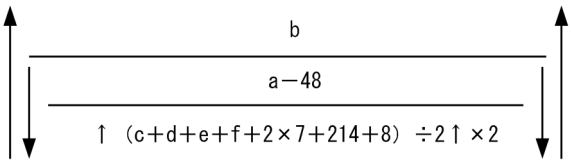
e：オブジェクトタイプ名称の長さの平均値（バイト）

f：オブジェクトの所有者名称の長さの平均値（バイト）

g：オブジェクト名称の長さの平均値（バイト）

h：イベント実行者の名称の長さの平均値（バイト）

(r)　計算式 18

ディクショナリ表名	計算式
SQL_AUDIT_REGULARIZE※	

a：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長－510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

b：監査証跡の絞り込みに指定したオブジェクト数とイベント実行者の総数 ＋ 1（個）

c：オブジェクトタイプ名称の長さの平均値（バイト）

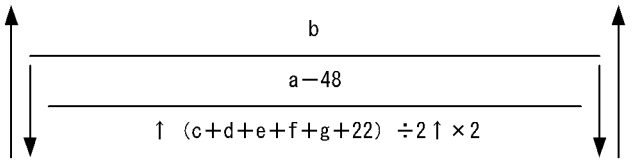
d：オブジェクトの所有者名称の長さの平均値（バイト）

e：オブジェクト名称の長さの平均値（バイト）

f：イベント実行者の名称の長さの平均値（バイト）

注※　システムが使用する表です。

(s) 計算式 19

ディクショナリ表名	計算式
SQL_KEYCOLUMN_USAGE	

a：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長−510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

b：制約定義の総数（個）

c：制約認可識別子の長さの平均値（バイト）

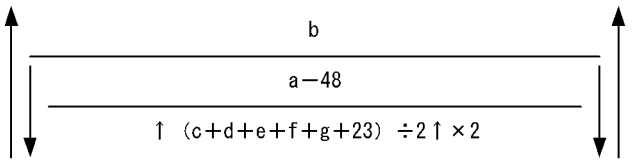
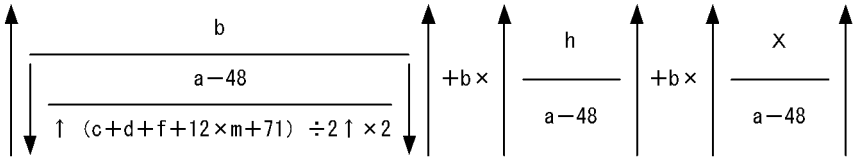
d：制約名称の長さの平均値（バイト）

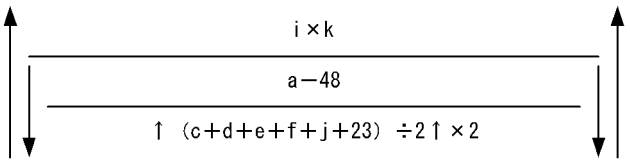
e：制約を定義した表の認可識別子の長さの平均値（バイト）

f：制約を定義した表名称の長さの平均値（バイト）

g：制約の種類名称の長さの平均値（バイト）

(t) 計算式 20

ディクショナリ表名	計算式
SQL_TABLE_CONSTRAINTS	
SQL_CHECKS	

ディクショナリ表名	計算式
SQL_CHECK_COLUMNS	

a：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長−510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

b：制約定義の総数（個）

c：制約認可識別子の長さの平均値（バイト）

d：制約名称の長さの平均値（バイト）

e：制約を定義した表の認可識別子の長さの平均値（バイト）

f：制約を定義した表名称の長さの平均値（バイト）

g：制約の種類名称の長さの平均値（バイト）

h：検査制約定義時の SQL 文の長さの平均値（バイト）

i：検査制約の定義数（個）

j：検査制約を定義した列に指定した列名称の長さの平均値（バイト）

k：検査制約を定義した列に指定した列数の平均値（個）

m：バイナリデータ（検査制約の探索条件，検査制約用解析ツリー）のデータ長が 256 以上，かつ分岐する平均バイナリデータ数

バイナリデータの格納ページ数の分岐条件については，「表の格納ページ数の計算方法」を参照してください。

X：検査制約用解析ツリー長（バイト）

次の計算式で求められる値になります。なお，この計算式中の変数はすべて検査制約の探索条件の指定です。

32 ビットモードの場合：S×36 ＋ T ＋（U1 ＋ U2 ＋ U3）×48 ＋ V×128 ＋ 1000

64 ビットモードの場合：S×72 ＋ T ＋（U1 ＋ U2 ＋ U3）×96 ＋ V×184 ＋ 1400

S：論理演算子，算術演算子（＋，－，＊，／，又は｜｜），及びシステム組み込みスカラ関数の合計数（個）

T：定数（HiRDB で解釈するデータ型）の長さの合計値（バイト）

U1：CASE 式，CAST 指定中の値式の数（個）

U2：スカラ関数（VALUE，SUBSTR，BIT_AND_TEST，POSITION，TIMESTAMP，VARCHAR_FORMAT，TIMESTAMP_FORMAT）の値式の数（個）

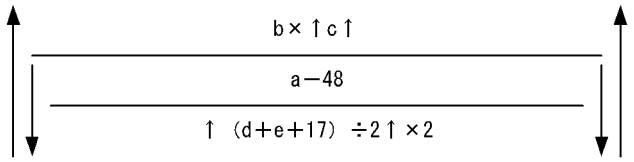
U3：日時書式の数（個）

V：列指定の数（個）

(u) 計算式 21

ディクショナリ表名	計算式
SQL_SYSPARAMS	2

(v) 計算式 22

ディクショナリ表名	計算式
SQL_PUBLICVIEW_SAME_USERS	

a：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長−510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

b：パブリックビュー表の総定義数（個）

c：パブリックビュー表ごとの重複名数の平均値※（バイト）

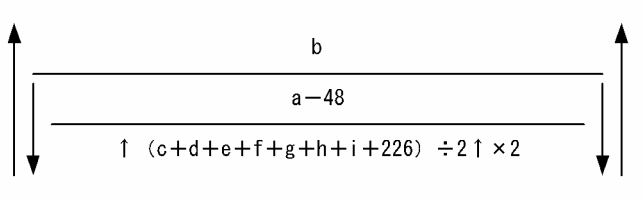
d：パブリックビュー表の表識別子の長さの平均値（バイト）

e：認可識別子の長さの平均値（バイト）

注※

パブリックビュー表の表識別子ごとに SQL_TABLES 表の TABLE_NAME 列が同じ値となる行数の平均値

(w) 計算式 23

ディクショナリ表名	計算式
SQL_SEQUENCES	

a：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長−510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合
  データディクショナリ用 RD エリアのページ長（バイト）

b：順序数生成子の総定義数（個）

c：順序数生成子識別子の長さの平均値（バイト）

d：認可識別子の長さの平均値（バイト）

e：順序数生成子開始値の長さの平均値（バイト）

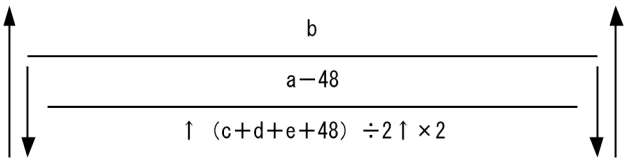
f：順序数生成子最大値の長さの平均値（バイト）

g：順序数生成子最小値の長さの平均値（バイト）

h：順序数生成子増分値の長さの平均値（バイト）

i：RD エリア名称の長さの平均値（バイト）

(x) 計算式 24

ディクショナリ表名	計算式
SQL_ACCESS_SECURITY	

a：次に示すどちらかを代入します。

- pd_dbreuse_remaining_entries オペランドの指定値が ONLY_USER 又は NOTHING の場合
  データディクショナリ用 RD エリアのページ長−510（バイト）
- pd_dbreuse_remaining_entries オペランドの指定値が上記以外の場合

データディクショナリ用 RD エリアのページ長（バイト）

b：接続制約の総数（個）

c：接続制約名称の長さの平均値（バイト）

d：IP アドレスの長さの平均値（バイト）

e：認可識別子の長さの平均値（バイト）

(3)　インデクスの格納ページ数の計算方法

インデクスの格納ページ数は，次に示す計算式で求めます。

計算式

インデクスの格納ページ数（単位：ページ）
＝ディクショナリ表のインデクスを格納するページ数※＋12

注※

「インデクスの格納ページ数の計算方法」を参照し，ディクショナリ表のインデクスを格納するページ数を計算してください。また，このときの計算条件を次に示します。

1. 計算式には，表「インデクスの格納ページ数を求める計算式に代入する変数一覧」に示す変数を代入します。
2. 変数 b（CREATE INDEX で指定する未使用領域の比率）は 30 とします。
3. 変数 e 及び f（キー値の重複が dx ＋ 1 以上のキーの種類の個数及びキーの重複数の平均値）は 12 とします。

表 15-6　インデクスの格納ページ数を求める計算式に代入する変数一覧

表名	種別	キー長※3 (変数 g ※1)	キーの種類の個数 (変数 c ※1)	キーの重複数の平均値 (変数 d ※1)
SQL_PHYSICAL_FILES	1	8	サーバ数	サーバ内の HiRDB ファイル数の平均値
	2	g ＋ 1	RD エリア数	RD エリアを構成する HiRDB ファイル数の平均値
SQL_RDAREAS	3	g ＋ 1	RD エリア数	1
	4	4		
	151	8	オリジナル RD エリア数×インナレプリカグループ内のレプリカ RD エリアの数の平均値 ＋ 2	↑RD エリアの総数÷キーの種類の数↑

表名	種別	キー長※3 (変数 g ※1)	キーの種類の個数 (変数 c ※1)	キーの重複数の平均値 (変数 d ※1)
	152	4	オリジナル RD エリア数＋ 1	↑RD エリアの総数÷キーの種類の数↑
SQL_TABLES	5	d ＋ e ＋ 2	表の総数＋ 80	1
	6	4		
SQL_COLUMNS	7	d ＋ e ＋ f ＋ 3	a×b	1
	8	d ＋ e ＋ 6		
	9	4		b
SQL_INDEXES	10	d ＋ e ＋ 2	a	↑h÷a↑
	11	d ＋ i ＋ 2	h	1
	12	4		
SQL_USERS	13	d ＋ 1	認可識別子の数	1
	153	d ＋ 1	スキーマ操作権限対象のスキーマ数	↑スキーマ操作権限を付与されたユーザ数÷スキーマ操作権限対象のスキーマ数↑
SQL_RDAREA_PRIVILEGES	14	d ＋ 1	データベース初期設定ユティリティ（pdinit）の USER USED BY オペランドで指定する認可識別子を重複排除した数	1 ユーザ当たりの平均 RD エリア利用権限使用数
	15	g ＋ 1	データベース初期設定ユティリティ（pdinit）の USER USED BY オペランドで指定する RD エリアの数	1RD エリア当たりの平均使用ユーザ数
SQL_TABLE_PRIVILEGES	16	d ＋ 1	a	↑y÷a↑
	17	2×d ＋ e ＋ 3	y	1
SQL_DIV_TABLE	18	d ＋ e ＋ 6	表の横分割数の総数	1
	19	g ＋ 1	表を横分割したときに指定する RD エリアを重複排除した数	RD エリアに格納する表数の平均値
	20	4		

表名	種別	キー長※3 (変数 g ※1)	キーの種類の個数 (変数 c ※1)	キーの重複数の平均値 (変数 d ※1)
SQL_DIV_TABLE_REGULARIZE	21	d ＋ e ＋ 6	表の横分割数の総数	1
	22	4	横分割表の数	表の横分割数の平均値
SQL_INDEX_COLINF	23	d ＋ e ＋ 6	インデクス定義のある表を重複排除した数	インデクスを構成する列数の平均値
	24	d ＋ i ＋ 6	インデクスの構成列数	1
SQL_TABLE_STATISTICS	25	d ＋ e ＋ 2	表の総数（ディクショナリ表もカウントの対象とします）	1
SQL_COLUMN_STATISTICS	26	d ＋ e ＋ f ＋ 3	h	1
SQL_INDEX_STATISTICS	27	d ＋ e ＋ 2	インデクスを定義する表の総数	↑h÷a↑
	28	d ＋ i ＋ 2	h	1
SQL_VIEW_TABLE_USAGE	29	d ＋ e ＋ 2	z	1
	30	d ＋ e ＋ 2	ビュー定義をする基表の数	1 表当たりのビュー定義数の平均値
	31	4	z	1
SQL_VIEWS	32	d ＋ e ＋ 2	z	1
	33	4		
SQL_VIEW_DEF	34	d ＋ e ＋ 2	z	1
	35	10		
SQL_REFERENTIAL_CONSTRAINTS	39	d ＋ e ＋ 2	参照制約の総数	1
	40	d ＋ e ＋ 2	参照表の総数	1 表当たりの参照表数の平均値
	41	d ＋ e ＋ 2	被参照表の総数	1 表当たりの被参照表数の平均値
SQL_EXCEPT	86	d ＋ e ＋ 2	除外値指定をしたインデクスの数	一つの表に除外値指定をしたインデクスの数

表名	種別	キー長※3 (変数 g ※1)	キーの種類の個数 (変数 c ※1)	キーの重複数の平均値 (変数 d ※1)
	87	d＋i＋2		1
	88	4	除外値指定をしたインデクスを持つ表を重複排除した数	一つの表に除外値指定をしたインデクスの数
SQL_DIV_INDEX	36	d＋e＋6	横分割インデクス数×分割数	1
	37	d＋i＋2	横分割インデクスの総数	1 表当たりの平均分割数
SQL_DIV_COLUMN	38	d＋e＋f＋3	LOB 列の定義数	1 表当たりの平均横分割数
	52	d＋e＋9	LOB 属性の定義数	1
SQL_ROUTINES	43	d＋MAX（q，7）	p＋174	1
	44	d＋MAX（t，18）	u＋65	1
	45	4	p＋174	1
	53	d＋UDT	抽象データ型の数＋1（NULL 値）	1 抽象データ型当たりのルーチン数の平均値＋NULL 値数
SQL_ROUTINE_RESOURCES	46	d＋q	p×s	1 ルーチン当たりの使用リソース数の平均値
	47	d＋t		
	48	d＋q		
	49	4		
SQL_ROUTINE_PARAMS	50	d＋MAX（q，8）	p×r＋347	1 ルーチン当たりの平均パラメタ数
	51	d＋MAX（t，19）	ルーチン数	1 特定名当たりのパラメタ数の平均値（3 未満の場合は 3 としてください）
	106	e＋4＋2	トリガ SQL オブジェクト数×r＋1（NULL 値）	1 特定名当たりのパラメタ数の平均値（3 未満の

表名	種別	キー長※3 (変数 g ※1)	キーの種類の個数 (変数 c ※1)	キーの重複数の平均値 (変数 d ※1)
				場合は 3 としてください）＋ NULL 値数
SQL_DATATYPES	54	d ＋ UDT	抽象データ型の数	1
	55	4		
	56	d ＋ UDT	サブタイプを持つ抽象データ型の数＋ 1（NULL 値）	1 抽象データ型当たりのサブタイプ数の平均値＋ NULL 値数
SQL_DATATYPE_DESCRIPTORS	57	d ＋ UDT ＋ ATT	NUDT×NATT	1
	58	4	抽象データ型の数	1 抽象データ型当たりの属性数の平均値
SQL_TABLE_RESOURCES	59	d ＋ e	ユーザ定義データ型を使用する表の総数	1 表当たりの使用ユーザ定義データ型数の平均値
	60	d ＋ UDT	UDT	1 ユーザ定義データ型当たりの使用表数の平均値
	61	4		
SQL_PLUGINS	62	d ＋ PLG	プラグイン数	1
	63	4		
	64	{(d ＋ UDT）÷ IXT}	データ型プラグイン数＋ インデクス型プラグイン数	
SQL_PLUGIN_ROUTINES	65	t	NPLG×NFPLG	1
	66	PLG ＋ TMD ＋ 2		
	67	PLG ＋ 4		
	68	POPR	オペレーション数	プラグイン数
SQL_PLUGIN_ROUTINE_PARAMS	69	t ＋ PRM	NPLG×NPPAR	1
	70	PLG	NPLG	1 プラグイン当たりの平均パラメタ数
	71	t ＋ 4	NPLG×NPPAR	1

表名	種別	キー長※3 (変数 g ※1)	キーの種類の個数 (変数 c ※1)	キーの重複数の平均値 (変数 d ※1)
SQL_REGISTRY_CONTEXT	72	CNM ＋ 1	コンテキスト数	1
SQL_REGISTRY_KEY	73	KNM ＋ 6	キー数	1
SQL_INDEX_TYPES	74	d ＋ IXT	作成するインデクス型の数	1
	75	4		
SQL_INDEX_RESOURCES	76	d ＋ IXT	プラグインインデクス数	インデクス型を使用するインデクス定義数の平均値
	77	4		
SQL_INDEX_DATATYPE	78	d ＋ e	プラグインインデクスを定義する表の定義数	同一表に対する平均プラグインインデクス数
	79	d ＋ i	プラグインインデクス数	1
SQL_INDEX_FUNCTION	80	d ＋ e	プラグインインデクスを定義する表の定義数	1 表当たりのプラグインインデクス数の平均値×プラグインインデクス適用関数数の平均値
	81	d ＋ i	プラグインインデクス数	1 プラグインインデクス当たりの適用関数数の平均値
SQL_TYPE_RESOURCES	82	d ＋ e	ユーザ定義データ型を使用するユーザ定義データ型の数	ユーザ定義データ型で属性に指定する平均ユーザ定義データ型数
	83	d ＋ UDT	ユーザ定義データ型数	ユーザ定義データ型を使用する平均ユーザ定義データ型数
	84	4		
SQL_INDEX_TYPE_FUNCTION	85	d ＋ IXT	インデクス型の数	1 インデクス当たりの適用関数数の平均値
SQL_TRIGGERS	90	d ＋ e ＋ 2+16	トリガ数	1
	91	d ＋ TRIG ＋ 2		
	92	d ＋ t ＋ 2		
	93	4		

表名	種別	キー長※3 (変数 g ※1)	キーの種類の個数 (変数 c ※1)	キーの重複数の平均値 (変数 d ※1)
SQL_TRIGGER_ACTCOND	94	d ＋ TRIG ＋ 2 ＋ 4	動作条件ありのトリガ数	1
	95	d ＋ e ＋ 2	トリガを定義した表数	1 表当たりのトリガ数の平均値
SQL_TRIGGER_COLUMNS	96	d ＋ TRIG ＋ 2	列指定がある，UPDATE 文を契機とするトリガ数	1 トリガ当たりの指定列の平均値
	97	d ＋ e ＋ f ＋ 3	UPDATE 文を契機とするトリガの指定列数	1
SQL_TRIGGER_DEF_SOURCE	98	d ＋ TRIG ＋ 2 ＋ 4	トリガ数	1
	99	d ＋ e ＋ 2	トリガを定義した表数	1 表当たりのトリガ数の平均値
SQL_TRIGGER_USAGE	100	d ＋ TRIG ＋ 2	トリガ動作の探索条件中でリソースを参照するトリガ数	1 トリガ当たりの参照リソース数の平均値
	101	d ＋ e ＋ 2	トリガ動作の探索条件中でリソースを参照するトリガを定義した表数	1 表当たりの参照リソース数の平均値
	102	d ＋ e ＋（t 又は e）＋ 2	使用リソース数	1
	103	8		
SQL_PARTKEY	104	d ＋ e ＋ f ＋ 3	マトリクス分割表の数×分割キーの数	1
SQL_PARTKEY_DIVISION	105	d ＋ e ＋ 6	マトリクス分割表の数×境界値数×分割キーの数	1
SQL_AUDITS	107	ETP ＋ EST ＋ 2	監査対象イベント数	1
	108	OTP ＋ OSC ＋ ONM ＋ 3	監査対象オブジェクト数	1 オブジェクト当たりの監査対象イベント数の平均値
	109	d	監査対象ユーザ数	1 ユーザ当たりの監査対象イベント数の平均値

表名	種別	キー長※3 (変数 g ※1)	キーの種類の個数 (変数 c ※1)	キーの重複数の平均値 (変数 d ※1)
SQL_AUDITS_REGULARIZE	110	OTP ＋ OSC ＋ ONM ＋ 3	監査対象オブジェクト数	1 オブジェクト当たりの監査対象イベント数の平均値
	111	d	監査対象ユーザ数	1 ユーザ当たりの監査対象イベント数の平均値
SQL_KEYCOLUMN_USAGE	112	d ＋ CNS ＋ 2	制約数	1
	113	d ＋ e ＋ 2	制約を定義している表数	1 表当たりの制約数の平均値
SQL_TABLE_CONSTRAINTS	114	d ＋ CNS ＋ 2	制約数	1
	115	d ＋ e ＋ 2	制約を定義した表数	1 表当たりの制約数の平均値
SQL_CHECKS	116	d ＋ CNS ＋ 2	検査制約数	1
	117	d ＋ e ＋ 2	検査制約を定義している表数	1 表当たりの検査制約数の平均値
SQL_CHECK_COLUMNS	118	d ＋ CNS ＋ 2	検査制約数	1 検査制約当たりの指定列数の平均値
	119	d ＋ e ＋ f ＋ 3	検査制約で使用している列数	1 表当たりの検査制約で使用している列の平均長複数
SQL_SYSPARAMS	121	8	2	1
SQL_PUBLICVIEW_SAME_USERS	124	d ＋ e ＋ 2	パブリックビュー表数×認可識別子数	1
SQL_INDEX_XMLINF	125	d ＋ e ＋ 2	NPSIT	1 インデクス当たりの平均インデクス構成部分構造パス数
	126	d ＋ i ＋ 7	NPSS	1
SQL_SEQUENCES	127	d ＋ SEQN ＋ 2	システム内の順序数生成子数	1
	128	4	システム内の順序数生成子数	1
SQL_ACCESS_SECURITY	149	CSTN	接続制約数	1

表名	種別	キー長※3 (変数 g ※1)	キーの種類の個数 (変数 c ※1)	キーの重複数の平均値 (変数 d ※1)
	150	1	2	接続制約数
	154	d	指定した認可識別子の数	1 認可識別子当たりの平均接続制約登録数

a：表の総数（個）

b：表の列数の平均値（個）

c：データディクショナリ用 RD エリアのページ長（バイト）

d：認可識別子の長さの平均値（バイト）

e：表識別子の長さの平均値（バイト）

f：列の名称長の平均値（バイト）

g：RD エリアの名称長の平均値（バイト）

h：インデクスの総数（個）

i：インデクス識別子の長さの平均値（バイト）

n：RD エリアを構成する HiRDB ファイル名称の長さの平均値（バイト）

p：作成するルーチン数（個）

q：ルーチン名称の長さの平均値（バイト）

r：1 ルーチン当たりのパラメタ数の平均値（個）

s：1 ルーチン当たりの使用リソース数の平均値（個）

t：特定名※2の長さの平均値（バイト）

u：特定名※2の総数（個）

y：アクセス権限の定義数（個）
　一つの表に対して n 人に権限を与えた場合，表数×n 個と数えます。

z：ビュー定義の総数（個）

NUDT：作成するユーザ定義データ型の数（個）

UDT：ユーザ定義データ型名称の長さの平均値（バイト）

NATT：1 ユーザ定義データ型当たりの属性数の平均値（個）

ATT：ユーザ定義データ型属性名称の長さの平均値（バイト）

PLG：プラグイン名称の長さの平均値（バイト）

NPLG：作成するプラグイン数（個）

IXT：インデクス型名称の長さの平均値（バイト）

NFPLG：プラグイン関数の数の平均値（個）

POPR：プラグイン関数名の長さの平均値（バイト）

NPPAR：1 プラグイン関数当たりのパラメタ数の平均値（個）

PRM：1 プラグイン関数当たりのパラメタ名の長さの平均値（バイト）

TMD：契機記述長の平均値（バイト）

CNM：コンテキスト名称長の平均値（バイト）

KNM：キー名称長の平均値（バイト）

TRIG：トリガ名称長の平均値（バイト）

ETP：イベントタイプ名称の長さの平均値（バイト）

EST：イベントサブタイプ名称の長さの平均値（バイト）

OTP：オブジェクト種別名称の長さの平均値（バイト）

OSC：オブジェクトの所有者名称の長さの平均値（バイト）

ONM：オブジェクト名称の長さの平均値（バイト）

CNS：制約名称の長さの平均値（バイト）

NPSIT：部分構造インデクス定義表数（個）

NPSS：部分構造インデクス構成部分構造パス数（個）

SEQN：順序数生成子識別子の長さの平均値（バイト）

CSTN：接続制約名称の長さの平均値（バイト）

注※1

「インデクスの格納ページ数の計算方法」の「計算式中で使用する変数」に記載されている変数のことです。

注※2

「認可識別子.ルーチン識別子」を表しています。

注※3

キー長は４バイト単位で切り上げになります。次に示す計算式で求めてください。

- ↑キー長÷4↑×4

15.2.2　解析情報表及び運用履歴表を格納するデータディクショナリ用 RD エリアの容量の見積もり

create rdarea 文に datadictionary of dbmanagement を指定する場合のデータディクショナリ用 RD エリアの容量は，次に示す計算式で求めます。

計算式

```
データディクショナリ用RDエリアの容量＝（↑（a×1.3）÷b↑）×b×4096
（単位：バイト）
```

a：データディクショナリ用 RD エリアの総ページ数※1

b：データディクショナリ用 RD エリアのセグメントサイズ※2

注※1

「表の格納ページ数＋インデクスの格納ページ数」です。「表の格納ページ数の計算方法」及び「インデクスの格納ページ数の計算方法」を参照してください。

注※2

データベース構成変更ユティリティ（pdmod）の create rdarea 文で指定するセグメントサイズです。

(1)　表の格納ページ数の計算方法

表の格納ページ数（単位：ページ）は，計算式 1 と計算式 2 の和になります。

計算式 1	$\uparrow \frac{(a+c+e+g) \times (\uparrow b \uparrow +1) +120) \times 30}{96} \uparrow$
計算式 2	$\uparrow \frac{(a+c+e+g) \times (\uparrow b \uparrow +1) +120) \times 30}{9} \uparrow$

a：作成する表の数＋ 61（個）

b：表を格納する RD エリアの分割数の平均値（個）

分割していない場合は 1 とします。また，平均値は切り上げます。

c：作成するインデクス数＋ 124（個）

e：作成する表に定義した BLOB 型の列の総数＋ 3（個）

g：作成する表に定義した BLOB 属性の総数（個）

(2) インデクスの格納ページ数の計算方法

インデクスの格納ページ数（単位：ページ）は次の計算式で求めます。

ディクショナリ表のインデクスを格納するページ数※×2

注※

「インデクスの格納ページ数の計算方法」を参照し，ディクショナリ表のインデクスを格納するページ数を計算してください。また，このときの計算条件を次に示します。

1. CREATE INDEX で指定する未使用領域の比率は 0 とします。
2. 計算式には，次に示す変数を代入します。

キー長 （変数 g※）	キーの種類の個数 （変数 c※）	キーの重複数の平均値 （変数 d※）
12	（a ＋ c ＋ e ＋ g）×（b ＋ 1）＋ 120	30

a：作成する表の数（個）

b：表を格納する RD エリアの分割数の平均値（個）

c：作成するインデクス数（個）

e：作成する表に定義した BLOB 型の列の総数（個）

g：作成する表に定義した BLOB 属性の総数（個）

なお，解析情報表及び運用履歴表にはユニークインデクスを定義してないため，変数 h を加算する必要はありません。

注※

「インデクスの格納ページ数の計算方法」の「計算式中で使用する変数」に記載されている変数のことです。

15.3　マスタディレクトリ用 RD エリアの容量の見積もり

マスタディレクトリ用 RD エリアの容量は，次に示す計算式で求めます。

計算式

```
マスタディレクトリ用RDエリアの容量（単位：バイト）
＝｛
    ↑（a＋2）÷800↑×51＋↑（b＋120）÷6000↑×51＋↑（c＋240）÷6000↑×51
    ＋↑（d＋240）÷64000↑×51＋↑e÷64000↑×51＋↑ｆ÷50↑×51＋2＋6×n
    ｝※1×4096※2
```

ａ：データディクショナリ用 RD エリアの総数＋ユーザ用 RD エリアの総数

インナレプリカ機能使用時はレプリカ RD エリアの総数も加算してください。

ｂ：定義する表の総数

ｃ：定義するインデクスの総数

ｄ：定義するビュー表の総数

ｅ：定義するデータ型及びインデクス型の合計

ｆ：オリジナル RD エリア及びレプリカ RD エリアの総数

インナレプリカ機能を使用しない場合は 0

オリジナル RD エリアの情報は，すべてのレプリカ RD エリアを削除しても消えません（オリジナル RD エリアを削除するまで消えません）。

ｎ：マスタディレクトリ用 RD エリアを構成する HiRDB ファイル数

注※1　マスタディレクトリ用 RD エリアの総ページ数です。

注※2　マスタディレクトリ用 RD エリアのページ長です。

15.4　データディレクトリ用 RD エリアの容量の見積もり

データディレクトリ用 RD エリアの容量は，次に示す計算式で求めます。

計算式

データディレクトリ用RDエリアの容量（単位：バイト）

$$= \{\uparrow (\sum_{i=1}^{e} gi + \sum_{j=1}^{f} pj + 86) \div 3000 \uparrow \times 51 + 6 \times n + 1 \} \text{※1} \times 4096\text{※2}$$

$$gi = \uparrow (5 \times a_i + 2 \times b_i + 2 \times c_i + 48) \div 32 \uparrow$$

$$pj = \uparrow (d_j + 12) \div 16 \uparrow$$

a_i：インデクスを構成する列数

b_i：インデクスを格納する RD エリア数

c_i：インデクスを定義する表を格納する RD エリア数

d_j：表を格納する RD エリア数

e：定義するインデクスの総数

f：横分割する表の総数

n：データディレクトリ用 RD エリアを構成する HiRDB ファイル数

注※1　データディレクトリ用 RD エリアの総ページ数です。

注※2　データディレクトリ用 RD エリアのページ長です。

15.5　データディクショナリ LOB 用 RD エリアの容量の見積もり

15.5.1　データディクショナリ LOB 用 RD エリアの容量の計算方法

(1)　ソース格納用のデータディクショナリ LOB 用 RD エリアの容量の見積もり

ソース格納用のデータディクショナリ LOB 用 RD エリアの容量は，次に示す計算式で求めます。

計算式

```
ソース格納用のデータディクショナリLOB用RDエリアの容量（単位：バイト）
＝｛
        a
     [ Σ  ↑Si÷64000↑×96 ＋7＋3×（a-1）] ※1
       i=1
          b
    ＋[ Σ  ↑（Cj＋1024）÷8192↑] ※2
        j=1
    ｝※3×8192※4
```

a：ソース格納用のデータディクショナリ LOB 用 RD エリアを構成する HiRDB ファイル数

b：次の総数

- 手続き（CREATE PROCEDURE）
- 抽象データ型内の関数と手続き（各 FUNCTION（ただし，プラグイン関数を除きます），PROCEDURE）
- ユーザ定義関数（CREATE FUNCTION）

S_i：セグメント数

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）の create rdarea 文で指定する各 HiRDB ファイルのセグメント数

C_j：次の長さ

- 手続き（各 CREATE PROCEDURE の長さ）
- 抽象データ型内の関数と手続き（各 FUNCTION（ただし，プラグイン関数を除きます），PROCEDURE の長さ）
- ユーザ定義関数（CREATE FUNCTION の長さ）

注※1　ディレクトリページ部分の総ページ数です。

注※2　データページ部分の総ページ数です。

注※3　ソース格納用のデータディクショナリ LOB 用 RD エリアの総ページ数です。

注※4　ソース格納用のデータディクショナリ LOB 用 RD エリアのページ長です。

(2)　オブジェクト格納用のデータディクショナリ LOB 用 RD エリアの容量の見積もり

オブジェクト格納用のデータディクショナリ LOB 用 RD エリアの容量は，次に示す計算式で求めます。

計算式

```
オブジェクト格納用のデータディクショナリLOB用RDエリアの容量（単位：バイト）
＝ {
        a
     [ Σ  ↑Si÷64000↑×96＋7＋3×（a-1）] ※1
       i=1
          b
    ＋ [ Σ  ↑（Cj＋1024）÷8192↑] ※2
         j=1
    } ※3×8192※4＋500000※5
```

a：オブジェクト格納用のデータディクショナリ LOB 用 RD エリアを構成する HiRDB ファイル数

b：手続き（CREATE PROCEDURE），抽象データ型内の関数と手続き（各 FUNCTION（ただし，プラグイン関数を除きます），PROCEDURE），ユーザ定義関数（CREATE FUNCTION）及びトリガ定義（CREATE TRIGGER）の総数

S_i：セグメント数

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）の create rdarea 文で指定する各 HiRDB ファイルのセグメント数

C_j：QO_i ＋ PR（QO_i の計算式を「QOi（SQL オブジェクト長）の計算式」に，PR の計算式を「PR（ルーチン制御用オブジェクト長）の計算式」に示します。これらの計算式で使用する変数を「PR 及び QOi の計算式で使用する変数」に示します）

注※1　ディレクトリページ部分の総ページ数です。

注※2　データページ部分の総ページ数です。

注※3　オブジェクト格納用のデータディクショナリ LOB 用 RD エリアの総ページ数です。

注※4　オブジェクト格納用のデータディクショナリ LOB 用 RD エリアのページ長です。

注※5　抽象データ型，又はプラグイン使用時に加算します。

(3)　QO_i（SQL オブジェクト長）の計算式

```
QOi（単位：バイト）＝
 a
Σ {
```

```
i=1
    1840＋46×RCN＋298×Si＋20×Pi＋1138×Ti＋76×Ti×Di＋80×Ci＋40×Ii＋534×Wi
    ＋20×Ki＋Li＋8×TCi＋656×Di＋48×nFF＋100×nFP＋148×nFC＋696×nPFF
    ＋16×（nAT＋nPAT）＋20×nCAT＋28×（nAF＋nCAF）＋20×（nAA＋nPAA＋nCAA）
    ＋1057×nSPA＋120×nSPP＋287×nSFF＋8×nSFP＋813×nJFC＋20×nJFP
     〔＋1057×nTR＋120×（nTSN＋nTSO）＋20×（nTCN＋nTCO）〕※1
     〔＋760＋376×RCC＋1880×RCT〕※2
     〔＋32×Si＋16〕※3
     〔＋↑（42×SiT）＋｛52＋152×（SiTA＋SiSA＋SiNA）×（SiT＋SiS＋SiN）｝↑〕※4
  }
```

a：ストアドプロシジャ内の SQL 文数

注※１　トリガを使用する場合に加算する計算式です。

注※２　参照制約を使用する場合に加算する計算式です。

注※３　列名記述領域長の計算式です。動的 SQL の場合に加算します。

注※４　型名記述領域長の計算式です。動的 SQL の場合に加算します。

(4)　PR（ルーチン制御用オブジェクト長）の計算式

(a)　ユーザが定義する場合

ストアドプロシジャ，ストアドファンクション，又はトリガを定義した場合の，ルーチン制御用オブジェクト長は次に示す計算式から求めます。

```
PR（単位：バイト）＝
 a
Σ {
i=1
    600＋28×sRi＋32×（sRUi＋sDi）＋56×sSXi＋sCUi＋sSi＋sPi＋sLA
    ＋sKi＋sL＋80×sWi＋24×sCM＋32×sCCR＋2×sDCR＋60×sCHD＋72×sDHD＋64×sHCN
    ＋8×sCHD×sHCN＋48×nRFF＋100×nRFP＋148×nRFC＋200×nPRFF＋8×nPRFP
    ＋196×nPA＋64×nPP＋36×nPPI＋20×nPPO＋200×nPPA＋8×nPPP＋20×nAR＋48×nARA
    ＋16×nRPAT＋20×nCAT＋28×（nRPAF＋nRCAF）＋20×（nRPAA＋nRCAA）＋287×nRSFF
    ＋8×nRSFP＋813×nPJA＋20×nPJP＋813×nRJFC＋20×nRJFP
     〔＋28×（nTSN×2＋nTSO）〕※
  }
```

a：次の SQL 文数

- 手続き（CREATE PROCEDURE）
- 抽象データ型内の関数と手続き（各 FUNCTION（ただし，プラグイン関数を除きます），PROCEDURE）
- ユーザ定義関数（CREATE FUNCTION）
- トリガ定義（CREATE TRIGGER）

注※　トリガを使用する場合に加算する計算式です。

(b)　HiRDB が自動的に作成する場合

表定義時，参照動作に CASCADE を指定した場合，HiRDB が制約制御のためにトリガを作成したときのルーチン制御用オブジェクト長は次に示す計算式から求めます。

```
PR（単位：バイト）＝
 a
Σ {240＋608×RCC＋（5120＋100×RDi＋256×RIi）×RCP×RCT}
i=1
```

a：次の SQL 文数

- 手続き（CREATE PROCEDURE）
- 抽象データ型内の関数と手続き（各 FUNCTION（ただし，プラグイン関数を除きます），PROCEDURE）
- ユーザ定義関数（CREATE FUNCTION）
- トリガ定義（CREATE TRIGGER）

(5)　PR 及び QO_i の計算式で使用する変数

変数名	説明
RCN	SQL オブジェクトで使用する表，及びインデクスの合計数
Si	SQL 文中にある検索項目数（SQL 文で指定する列がインデクス列の場合はその列数）
Pi	SQL 文中にある埋込み変数又はパラメタ数
Ti	SQL 文中にある表名数
Ci	SQL 文中にある列名数
TCi	SQL 文中にある表の構成列数
Wi	SQL 文中にある論理演算子数※
Ki	SQL 文中にある定数の数※
Li	SQL 文中にある定数の合計長※（単位：バイト）
Ii	SQL 文実行時に使用するインデクス数（SQL 文で指定する表のうち，検索条件に指定するインデクス数）
Di	SQL 文中の表に定義された格納条件の総数（マトリクス分割表は 2 倍する）
SiT	SQL 文中にある選択式の抽象データ型数
SiS	SQL 文中にある選択式の抽象データ型のスーパタイプ数
SiN	SQL 文中にある選択式のサブタイプである抽象データ型のスーパタイプの総数
SiTA	SQL 文中にある選択式の抽象データ型の属性数

変数名	説明
SiSA	SQL 文中にある選択式の抽象データ型のスーパタイプの属性数
SiNA	SQL 文中にある選択式のサブタイプである抽象データ型のコンポネント指定総数
nSPA	SQL 文中にある手続き文の呼び出し数
nSPP	SQL 文中にある手続き文の引数の総数
nFF	SQL 文中にある関数の呼び出し数※
nFP	SQL 文中にある関数の引数の総数※
nFC	SQL 文中にある関数の総関数定義候補数（関数呼出し数 nFF に，引数が抽象データ型の各関数呼出しに対して，サブタイプを引数とする関数定義数を加算する）
nPFF	SQL オブジェクトが使用するプラグイン関数呼出し数（SQL 文中にあるプラグイン関数呼出し数＋ SELECT の場合は 1，INSERT，UPDATE，DELETE の場合は 6）
nSFF	SQL 文中にあるシステム定義スカラ関数の呼び出し数※
nSFP	SQL 文中にあるシステム定義スカラ関数の引数の総数※
nJFC	SQL 文中にある外部 Java 関数の呼び出し数
nJFP	SQL 文中にある外部 Java 関数の引数の総数
nAT	SQL 文中にあるコンポネント指定で使用する抽象データ型数（スーパタイプ，抽象データ型属性によって現れる抽象データ型を除く）
nAA	SQL 文中にあるコンポネント指定で使用する抽象データ型数（スーパタイプ，抽象データ型属性によって現れる抽象データ型を含む）
nAF	SQL 文中にあるコンポネント指定で使用する属性総数
nPAT	SQL オブジェクトが使用するプラグイン関数の引数で使用する抽象データ型数（スーパタイプ，抽象データ型属性によって現れる抽象データ型を除く）
nPAA	SQL オブジェクトが使用するプラグイン関数の引数で使用する抽象データ型数（スーパタイプ，サブタイプを含む）
nCAT	SQL 文中にあるコンストラクタ関数の呼び出し数
nCAA	SQL 文中にあるコンストラクタ関数の抽象データ型数（スーパタイプを含む）
nCAF	SQL 文中にあるコンストラクタ関数の抽象データ型の属性総数
nTR	SQL 文の実行によって起動されるトリガ数
nTSN	SQL 文の実行によって起動される各トリガのトリガ SQL 文中の新値相関名によって修飾された列の総数
nTSO	SQL 文の実行によって起動される各トリガのトリガ SQL 文中の旧値相関名によって修飾された列の総数
nTCN	SQL 文の実行によって起動される各トリガのトリガ動作条件中の新値相関名によって修飾された列の総数
nTCO	SQL 文の実行によって起動される各トリガのトリガ動作条件中の旧値相関名によって修飾された列の総数
RCC	SQL 文中で，更新対象の表を参照する表の外部キーの構成列数と，主キーの構成列数の総数
RCT	SQL 文中で，更新対象の表を参照する表，及び更新対象の表が参照する表の総数

変数名	説明
RCP	参照表定義時に，参照動作に指定した CASCADE の総数
RIi	参照表定義時，参照指定をする被参照表に定義されたインデクスの総数
RDi	参照表定義時，参照指定をする被参照表に定義された分割格納条件の総数（マトリクス分割表の場合は，2 倍する）
sRi	手続き及び関数中の SQL パラメタ数（INOUT 指定の SQL パラメタ数は 2 倍する）
sRUi	手続き及び関数中の SQL パラメタの総数（又は，トリガ定義中のトリガ SQL 文中の新旧値相関名によって修飾された列の総数）
sDi	手続き，関数，及びトリガ SQL 文中の SQL 変数（declare）の総数
sSXi	手続き，関数，及びトリガ SQL 文中の SQLCODE，SQLCOUNT 変数の総数
sCUi	手続き，関数，及びトリガ SQL 文中の CURRENT_TIME 値関数，CURRENT_DATE 値関数の総数
sSi	手続き及びトリガ SQL 文中の操作系 SQL の数（カーソル宣言を除く SQL 文：OPEN，FETCH，CLOSE，UPDATE，DELETE，INSERT 文など）
sPi	手続き，関数，及びトリガ SQL 文中のルーチン制御 SQL 数（BEGIN，SET，IF，ELSEIF，WHILE など）
sLA	手続き，関数，及びトリガ SQL 文中のラベル数
sKi	手続き，関数，及びトリガ SQL 文中の定数の数（手続き及びトリガ SQL 文中に記述された操作系 SQL 文の定数を除く）
sL	手続き，関数，及びトリガ SQL 文中の定数の長さの合計（手続き及びトリガ SQL 文中に記述された操作系 SQL 文の定数を除く）
sWi	手続き，関数，及びトリガ SQL 文中の条件述語数
sCM	手続き，関数，及びトリガ SQL 文中の複合文の数
sCCR	手続き及びトリガ SQL 文のカーソル宣言を記述した複合文の数
sDCR	手続き及びトリガ SQL 文のカーソル宣言の数
sCHD	手続き，関数，及びトリガ SQL 文のハンドラ宣言を記述した複合文の数
sDHD	手続き，関数，及びトリガ SQL 文のハンドラ宣言の数
sHCN	手続き，関数，及びトリガ SQL 文のハンドラ宣言中に記述された条件値の数
nRFF	ルーチン中にある関数呼出し数
nRFP	ルーチン中にある関数の引数の総数
nRFC	ルーチン中にある関数の関数定義候補の総数（関数呼出し数 nFF に，引数が抽象データ型の各関数呼出しに対して，サブタイプを引数とする関数定義数を加算する）
nPRFF	ルーチンの SQL オブジェクトが使用するプラグイン関数呼出し数
nPRFP	ルーチンの SQL オブジェクトが使用するプラグイン関数のプラグインパラメタ総数
nPA	ルーチン中にある手続き呼び出し数
nPP	ルーチン中にある手続きのパラメタ総数

変数名	説明
nPPI	ルーチン中にある手続きの入力パラメタ総数（入出力パラメタを含む）
nPPO	ルーチン中にある手続きの出力パラメタ総数（入出力パラメタを含む）
nPPA	ルーチンの SQL オブジェクトが使用するプラグイン手続き呼び出し数
nPPP	ルーチンの SQL オブジェクトが使用するプラグイン手続きのプラグインパラメタ総数
nRSFF	ルーチン中にあるシステム定義スカラ関数の呼び出し数
nRSFP	ルーチン中にあるシステム定義スカラ関数の引数の総数
nPJA	ルーチン中にある外部 Java 手続きの呼び出し数
nPJP	ルーチン中にある外部 Java 手続きの引数の総数
nRJFC	ルーチン中にある外部 Java 関数の呼び出し数
nRJFP	ルーチン中にある外部 Java 関数の引数の総数
nAR	ルーチン中にあるコンポネント指定で使用する抽象データ型数（スーパタイプ，抽象データ型属性によって現れる抽象データ型を除く）
nARA	ルーチン中にあるコンポネント指定で使用する属性総数
nRPAT	ルーチンの SQL オブジェクトが使用するプラグインルーチンのパラメタで使用する抽象データ型数（ただし，スーパタイプ，抽象データ型属性の抽象データ型を除く）
nRPAA	ルーチンの SQL オブジェクトが使用するプラグインルーチンのパラメタで使用する抽象データ型数（スーパタイプを含む）
nRPAF	ルーチンの SQL オブジェクトが使用するプラグインルーチンのパラメタで使用する抽象データ型の属性総数
nRCAT	ルーチン中にあるコンストラクタ関数の呼び出し数
nRCAA	ルーチン中にあるコンストラクタ関数の抽象データ型数（スーパタイプを含む）
nRCAF	ルーチン中にあるコンストラクタ関数の抽象データ型の属性総数

注※

トリガを使用する場合は，SQL 文の実行によって起動される各トリガのトリガ動作条件についても数える必要があります。

15.6　ユーザ LOB 用 RD エリアの容量の見積もり

15.6.1　ユーザ LOB 用 RD エリアの容量の計算方法

ユーザ LOB 用 RD エリアの容量は，次に示す計算式で求めます。

計算式

```
ユーザLOB用RDエリアの容量（単位：バイト）
  ＝（ディレクトリページ部分の総ページ数＋データページ部分の総ページ数）×8192※
```

注※　ユーザ LOB 用 RD エリアのページ長です。

(1)　ディレクトリページ部分の総ページ数

計算式

```
ディレクトリページ部分の総ページ数
      a
＝    Σ  ↑Si÷64000↑×96＋7＋3×（a-1）
     i=1
```

a：ユーザ LOB 用 RD エリアを構成する HiRDB ファイル数

S_i：セグメント数

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）の create rdarea 文で指定する各 HiRDB ファイルのセグメント数

(2)　データページ部分の総ページ数

計算式

```
データページ部分の総ページ数
      b
＝    Σ  ↑（Cj＋1024）÷8192↑
     j=1
```

b：LOB 列の行の総数

データ長が 0 の行もカウントします。NULL 値の行はカウントしません。

C_j：各 BLOB データのデータ長（バイト）

15.7　レジストリ用 RD エリアの容量の見積もり

15.7.1　レジストリ用 RD エリアの容量の計算方法

レジストリ用 RD エリアの容量は，次に示す計算式で求めます。

計算式

```
レジストリ用RDエリアの容量（単位：バイト）
  ＝レジストリ用RDエリアのページ長※×レジストリ用RDエリアの総ページ数×1.3
```

注※

レジストリ機能初期設定ユティリティ（pdreginit）の create rdarea 文で指定するページ長です。

レジストリ用 RD エリアの総ページ数の求め方

```
レジストリ用RDエリアの総ページ数（単位：ページ）
      a
  ＝ Σ  {↑Si÷d↑＋↑Si÷e↑＋6×（a＋1）＋2×↑20480÷b↑
     i=1
  ＋レジストリ管理表の格納ページ数＋レジストリ管理表のインデクス格納ページ数
```

a：レジストリ用 RD エリアを構成する HiRDB ファイル数

b：レジストリ用 RD エリアのページ長（単位：バイト）

c：レジストリ機能初期設定ユティリティ（pdreginit）の create rdarea 文で指定するセグメントサイズ

d：↓（b−20）÷ {(↑c÷32↑×8）＋ 56} ↓

e：↓（125×b）÷（16×d）↓×d

Si：レジストリ機能初期設定ユティリティ（pdreginit）の create rdarea 文で指定する各 HiRDB ファイルのセグメント数

各表及びインデクスの確保単位は，セグメントです。表やインデクスごとに求めた値をセグメント単位に切り上げます。

(1)　レジストリ管理表の格納ページ数

計算式

$$\left\downarrow \frac{a}{\left\downarrow \frac{b-48}{\uparrow (c+d+20) \div 2 \uparrow \times 2} \right\downarrow} \right\uparrow \quad \cdots \text{（SQL_REGISTRY_CONTEXT表）}$$

$$\left\downarrow \frac{e}{\left\downarrow \frac{b-48}{\uparrow (f+g+6\times 2+13+8+2) \div 2 \uparrow \times 2} \right\downarrow} \right\uparrow \quad \cdots \text{（SQL_REGISTRY_KEY表）}$$

a：レジストリ管理表のコンテキスト数

b：レジストリ管理表のページ長

c：レジストリコンテキスト名称の長さ

d：アクセスパスワードの長さ

e：レジストリ管理表のキー値数（レジストリ管理表に登録したキー名称の数）

f：レジストリキー名称長

g：レジストリキー値長（レジストリキー値長が 32,000 バイト以下の場合に加算する）

(2)　レジストリ管理表のインデクス格納ページ数

計算式

```
レジストリ管理表のインデクス格納ページ数（単位：ページ）
  ＝SQL_REGISTRY_CONTEXT表のインデクス格納ページ数
  ＋SQL_REGISTRY_KEY表のインデクス格納ページ数
```

SQL_REGISTRY_CONTEXT 表のインデクス格納ページ数，及び SQL_REGISTRY_KEY 表のインデクス格納ページ数については，「インデクスの格納ページ数の計算方法」を参照してください。ただし，CREATE INDEX 文で指定する未使用領域の比率は 30%として計算してください。

インデクスを格納するページ数の計算式に使用する値を次に示します。

レジストリ用 RD エリアのインデクス格納ページ数の計算式に使用する値

表名	種別	キー長	キーの種類	平均重複数
SQL_REGISTRY_CONTEXT	72	a＋1	コンテキスト名の数（レジストリ管理表のコンテキスト数）	1
SQL_REGISTRY_KEY	73	f＋6	キー値の数（レジストリ管理表のキー値数）	1

a：レジストリコンテキストの長さ

f：レジストリキー名称の長さ

15.8　レジストリ LOB 用 RD エリアの容量の見積もり

15.8.1　レジストリ LOB 用 RD エリアの容量の計算方法

レジストリ LOB 用 RD エリアの容量は，次に示す計算式で求めます。

計算式

```
レジストリLOB用RDエリアの容量（単位：バイト）
  ＝（ディレクトリページ部分の総ページ数＋データページ部分の総ページ数）×8192※
```

注※　レジストリ LOB 用 RD エリアのページ長です。

(1)　ディレクトリページ部分の総ページ数

計算式

```
ディレクトリページ部分の総ページ数
    a
＝ Σ  ↑Si÷64000↑×96＋7＋3×（a-1）
  i=1
```

a：レジストリ LOB 用 RD エリアを構成する HiRDB ファイル数

S_i：レジストリ LOB 用 RD エリアのセグメント数

(2)　データページ部分の総ページ数

計算式

```
データページ部分の総ページ数
    b
＝ Σ  ↑（Cj＋1024）÷8192↑
  j=1
```

b：32000 バイトを超えるレジストリキー値の数

C_j：32000 バイトを超えるレジストリキー値の長さ

15.9　リスト用 RD エリアの容量の見積もり

リスト用 RD エリアの容量は，次に示す計算式で求めます。

計算式

```
リスト用RDエリアの容量（単位：バイト）
＝｛
      a
      Σ（↑Si÷f↑＋↑Si÷g↑）
     i=1
    ＋6×（a＋1）＋↑（1024×n）÷（25×b）↑＋↑20480÷b↑＋c×e
    ｝×b
```

a：リスト用 RD エリアを構成する HiRDB ファイル数

b：リスト用 RD エリアのページ長※（バイト）

c：リスト用 RD エリアのセグメント数

e：リスト用 RD エリアのセグメントサイズ※

f：↓｛b－20｝÷｛(↑e÷32↑×8）＋56｝↓

g：↑（125×b）÷（16×f）↑×f

n：1 リスト用 RD エリアに作成できる最大リスト数※

リスト用 RD エリアの最大リスト数は，次の計算式で求めます。

↑サーバ内に保持するリストの総数÷サーバ内に配置するリスト用 RD エリアの数↑

S_i：リスト用 RD エリアを構成する HiRDB ファイルのセグメント数※

注※

上記の計算式で求めた値以上となる最も小さい 500 の整数倍の値を，データベース初期設定ユティリティ（pdinit）若しくはデータベース構成変更ユティリティ（pdmod）の create rdarea 文，又はデータベース構成変更ユティリティの initialize rdarea 文で指定してください。

16 システムファイル及び監査証跡ファイルの容量の見積もり

この章では，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などのシステムファイルの容量の見積もり方法，及び監査証跡ファイルの容量の見積もり方法について説明します。

16.1　システムログファイルの容量の見積もり

ここでは，システムログファイルの容量の見積もり方法について説明します。説明する項目を次に示します。

- システムログファイルの総容量
- 表定義時に出力されるシステムログ量
- インデクス定義時に出力されるシステムログ量
- 表データ更新時に出力されるシステムログ量
- ユティリティによるデータベース作成時に出力されるシステムログ量
- SQL 操作に応じて出力されるシステムログ量
- 拡張システム定義スカラ関数の定義時に出力されるシステムログ量
- RD エリアの自動増分機能使用時に出力されるシステムログ量
- PURGE TABLE 文実行時に出力されるシステムログ量
- 空きページ解放ユティリティ実行時に出力されるシステムログ量
- 再編成時期予測機能使用時に出力されるシステムログ量
- 更新可能バックアップ閉塞中に出力されるシステムログ量
- pdchpathn コマンド実行時に出力されるシステムログ量
- ディクショナリ表のメンテナンス実行時に出力されるシステムログ量

なお，システムログファイルの容量を見積もる上で，更新可能なオンライン再編成をする場合の注意事項については，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。

16.1.1　システムログファイルの総容量

(1)　システムログファイルの総容量の求め方

システムログファイルの総容量は，次に示す計算式で求めます。

計算式

全システムログファイルの総容量（単位：バイト）＝（↑ａ÷ｃ↑×3）※×ｂ

ａ：出力されるシステムログ量
求め方については，「システムログ量の求め方」を参照してください。

ｂ：pd_log_rec_leng オペランドで指定するシステムログファイルのレコード長です。

ｃ：次に示す値を代入してください。
pd_log_rec_leng ＝ 1024 の場合：1000

pd_log_rec_leng ＝ 2048 の場合：2000

pd_log_rec_leng ＝ 4096 の場合：4000

注

- システムログファイル中に空き領域が発生するため，ここで求めた容量の 1.2 倍以上の容量を準備することをお勧めします。
- インナレプリカ機能使用時はログに拡張情報を付加するため，ここで求めた容量の 1.1 倍の容量を準備してください。
- ここで得られる総容量を単位時間当たりの容量に変換して，一つのシステムログファイルの容量と数を見積もってください。また，容量と数を見積もるときに，システムログファイルのアンロード間隔についても考慮してください。

注※

システムログファイルの総レコード数を求める計算式です。

(2) システムログ量の求め方

出力されるシステムログ量は，次に示す計算式で求めます。

計算式

```
出力されるシステムログ量（単位：バイト）＝Σ｛ｂ＋ｅ＋↑ｂ÷（ａ-256）↑×256＋ｄ｝＋c
●回復不要FESの場合だけ加算します。
＋f
```

Σ はトランザクションごとの合計を意味します。ただし，実行するトランザクションが次の条件をすべて満たす場合，その分のシステムログは出力されません。

- 検索をするトランザクションで，かつ COMMIT 文で決着する
- トランザクションを実行するサーバがシングルサーバ，ディクショナリサーバ，又はバックエンドサーバである

a：pd_log_max_data_size オペランドの値

b：1336 ＋データベース操作に応じて出力されるシステムログ量

ただし，回復不要 FES の場合は 0 になります。

データベース操作に応じて出力されるシステムログ量は，「表定義時に出力されるシステムログ量」～「ユティリティによるデータベース作成時に出力されるシステムログ量」で求めたシステムログ量を一つのトランザクション当たりで合計した値です。

c：SQL 操作に応じて出力されるシステムログ量

「SQL 操作に応じて出力されるシステムログ量」を参照してください。

d：サーバ種別ごとに次に示す計算式で算出した値

- HiRDB/シングルサーバの場合

  2×pd_log_rec_leng オペランドの値

- HiRDB/パラレルサーバの場合

  ・フロントエンドサーバのとき

マルチスレッド対応で，X/Open に準拠した接続方式を実現する HiRDB ライブラリを使用した UAP の実行，かつトランザクションの移行機能の使用有無	回復不要 FES の使用有無	計算式
使用	未使用（使用できない）	（↑1336÷pd_log_rec_leng オペランドの値↑＋3）×pd_log_rec_leng オペランドの値
未使用	使用	0
未使用	未使用	2×pd_log_rec_leng オペランドの値

・バックエンドサーバ又はディクショナリサーバのとき

システム内の，回復不要 FES を使用するフロントエンドサーバの有無	pd_rpl_reflect_mode オペランドの指定値	計算式
あり	uap	pd_log_rec_leng オペランドの値＋↑｛176＋128×（一つのトランザクションで更新する BES の最大数＋1）｝÷pd_log_rec_leng オペランドの値↑×pd_log_rec_leng オペランドの値
	server	
なし	uap	
	server	2×pd_log_rec_leng オペランドの値

e：

- HiRDB/パラレルサーバの場合

  フロントエンドサーバのとき：8＋72×一つのトランザクションで参照，又は更新する BES 及び DS の最大数

  フロントエンドサーバ以外のとき：0
- HiRDB/シングルサーバの場合：0

f：10×pd_log_rec_leng の値

注　システムログは次に示す操作をする場合に出力されます。

- 表の定義時
- インデクスの定義時
- 表データの更新時
- ユティリティによるデータベースの作成時

なお，上記の操作によってデータベースを更新中に，ロールバックが発生した場合，それまでデータベースを更新した分のシステムログが新たに追加で出力されます。このことを考慮してシステムログ量を見積もってください。

(3) 容量見積もり時の注意事項

次の SQL を実行すると，システムログを大量に出力します。システムログファイルの容量が不足するおそれがあるため，これらの SQL は出力するシステムログ量を見積もってから実行してください。また，見積もりの結果，システムログファイルの容量が不足する場合は，次に示す方法で容量不足を回避してください。

システムログを大量に出力する SQL	システムログファイル容量不足の回避方法
CREATE INDEX　形式 1（インデクス定義）	インデクスオプションに EMPTY を指定してインデクスを定義し，データベース再編成ユティリティ（pdrorg）を使用してインデクスを作成します。そのとき，ログ取得方式に更新前ログ取得モードを指定してインデクスを作成してください。
CREATE INDEX　形式 2（インデクス定義） CREATE INDEX　形式 3（部分構造インデクス定義）	ログレスモードで実行してください。 なお，左記の SQL 実行時は，特に大量のシステムログを出力します。そのため，システムログファイルの容量の過不足に関係なく，ログレスモードで運用することをお勧めします。ログレスモードで UAP 又はユティリティを実行するときの運用については，マニュアル「HiRDB システム運用ガイド」を参照してください。
DROP SCHEMA（スキーマ削除）	指定した認可識別子のスキーマ内の次に示す定義を個別に削除し，一度に出力するシステムログ量を減らしてください。 • 実表 • ビュー表（パブリックビューも含む） • インデクス • アクセス権限 • ルーチン（DROP SCHEMA で指定した認可識別子が定義したパブリック手続き，パブリック関数を含む） • トリガ • 抽象データ型 • インデクス型

16.1.2 表定義時に出力されるシステムログ量

表定義時に出力されるシステムログ量は，次に示す計算式で求めます。

(1) HiRDB/シングルサーバの場合

条件	システムログ量（単位：バイト）
表を横分割しない場合	((1256×b + 2500) ×1.2) ×a + (632×a×d) ※
表を横分割する場合	((1256×b + 1800×c + 2500) ×1.2) ×a + (632×a×d) ※

a：定義する表の総数（個）

b：定義する表の列数の平均値（個）

c：定義する表の平均分割数（個）

d：LOB 列を格納する RD エリア数（個）

注※　LOB 列を定義する表がある場合に加算します。

(2)　HiRDB/パラレルサーバの場合

条件	システムログ量（単位：バイト）
ディクショナリサーバで出力されるシステムログ量	((1256×b + 1800×c + 2500) ×1.2) ×a
バックエンドサーバで出力されるシステムログ量	912×a×d + (632×a×e) ※

a：定義する表の総数（個）

b：定義する表の列数の平均値（個）

c：定義する表の平均分割数（個）

d：定義する表のバックエンドサーバ内での分割数（個）

e：このバックエンドサーバ内にある RD エリアで，LOB 列を格納する RD エリア数（個）

注※　このバックエンドサーバ内に定義する表で，LOB 列を定義する表がある場合に加算します。

16.1.3　インデクス定義時に出力されるシステムログ量

インデクス定義時（主キーの追加を含みます）に出力されるシステムログ量は，次に示す計算式で求めます。

(1)　HiRDB/シングルサーバの場合

条件	システムログ量（単位：バイト）
インデクスを横分割しない場合	$\sum_{i=1}^{n} \{ 5624 + 800 \times b + 1256 \times \lceil b \times 5 \div 24 \rceil + 1256^{※} + 288 \times g + R + KP + 272 \times W_i + 1940 \times \lceil W_i \div V_i \rceil \}$
インデクスを横分割する場合	$\sum_{i=1}^{n} \{ 5124 + 800 \times b + 500 \times c + 800 \times a + 1256 \times \lceil b \times 5 \div 24 \rceil + 1256^{※} + 288 \times g + R + a$

条件	システムログ量（単位：バイト）
	$\Sigma_{j=1}^{} \{KP2 + 272 \times W_{ij} + 1940 \uparrow W_{ij} \div V_{ij} \uparrow\}$

注※　除外値指定がない場合は 0 になります。

a：インデクスを定義する表の横分割数（個）

b：インデクスの構成列数（個）

c：インデクスを格納する RD エリア数（個）

f：インデクス定義時に指定する PCTFREE オペランドの値（%）
　ページ内未使用領域比率の値です。

g：インデクスを格納する RD エリアのレプリカ世代数（個）
　レプリカ RD エリアがない場合は 0 になります。

n：定義するインデクスの総数（個）

V：インデクスを格納するユーザ用 RD エリアのセグメントサイズ（ページ）

W：インデクスの格納ページ数（ページ）
　「インデクスの格納ページ数の計算方法」を参照してください。
　なお，CREATE INDEX で EMPTY を指定した場合は 0 になります。

X：インデクスを格納するユーザ用 RD エリアのページ長（バイト）

h：$\downarrow$pd_log_max_data_size オペランドの値/1000$\downarrow \times 1000$

R：次のどちらかの値

- ログ取得モードの場合：0
- ログレスモードの場合：((a ＋ c) ×2) ×h

KP：次のどちらかの値

- ログ取得モードの場合：(132 ＋ Xi×↑(100−f) ÷100↑) ×Wi
- ログレスモードの場合：0

KP2：次のどちらかの値

- ログ取得モードの場合：(132 ＋ Xij×↑(100−f) ÷100↑) ×Wij
- ログレスモードの場合：0

(2) HiRDB/パラレルサーバの場合

条件	システムログ量（単位：バイト）
ディクショナリサーバで出力されるシステムログ量	$\sum_{i=1}^{n}\{5124+800\times b+500\times c+800\times a+(1256\times\uparrow b\times 5\div 24\uparrow)+1256^{※}+288\times g\}$
バックエンドサーバで出力されるシステムログ量	$\sum_{i=1}^{n}\{814+R+KP+272\times W_i+1940\times\uparrow W_i\div V_i\uparrow\}$

注※　除外値指定がない場合は 0 になります。

a：インデクスを定義する表の横分割数（個）

b：インデクスの構成列数（個）

c：インデクスを格納する RD エリア数（個）

f：インデクス定義時に指定する PCTFREE オペランドの値（%）
　ページ内未使用領域比率の値です。

g：インデクスを格納する RD エリアのレプリカ世代数（個）
　レプリカ RD エリアがない場合は 0 になります。

n：定義するインデクスの総数（個）

V：インデクスを格納するユーザ用 RD エリアのセグメントサイズ（ページ）

W：インデクスの格納ページ数（ページ）
　「インデクスの格納ページ数の計算方法」を参照してください。
　なお，CREATE INDEX で EMPTY を指定した場合は 0 になります。

X：インデクスを格納するユーザ用 RD エリアのページ長（バイト）

h：↓pd_log_max_data_size オペランドの値/1000↓×1000

R：次のどちらかの値

- ログ取得モードの場合：0
- ログレスモードの場合：((a ＋ c) ×2) ×h

KP：次のどちらかの値

- ログ取得モードの場合：(132 + Xi×↑(100−f)÷100↑)×Wi
- ログレスモードの場合：0

16.1.4　表データ更新時に出力されるシステムログ量

表中の行を操作すると次の表に示すシステムログが出力されます。

HiRDB/シングルサーバの場合は，ここで求めるシステムログ量をシングルサーバで出力するログ量に加算してください。HiRDB/パラレルサーバの場合は，更新対象の表及びインデクスが格納されているRDエリアを管理する，バックエンドサーバ及びディクショナリサーバで出力するログ量に，ここで求めるシステムログ量をそれぞれ加算してください。

ただし，UAPがログレスモードの場合には，ログレスモードの表データ更新時に出力されるシステムログ量を求めてください。

表16-1　表中の行を操作したときに出力されるシステムログの種類

システムログの種類	説　明
基本行ログ	基本行ログは，表中の行データを追加，削除又は更新するときに出力されます。
分岐行ログ	分岐行ログは，次に示すデータ型のデータを操作するときに出力されます。 • VARCHAR※1 • NVARCHAR※2 • MVARCHAR※1 • 繰返し列 • 抽象データ型 • BINARY型※3
インデクスログ	インデクスログは，インデクスのキーを追加，削除又は更新するときに出力されます。インデクスログ量は，データベースの操作内容（INSERT文，DELETE文及びUPDATE文）によって，表「データベースの操作内容と求めるログ量」に示すとおりに求めてください。
イベントログ	イベントログは，HiRDB Datareplicatorを使用しているとき，繰返し列を含む行データを追加，削除又は更新すると出力されます。

注※1　次に示すどちらかの条件を満たす場合に分岐行ログが出力されます。

- ノースプリットオプションの指定がなく，実際のデータ長が256バイト以上の場合
- ノースプリットオプションの指定があり，実際の1行のデータ長の合計がページ長を超える場合

注※2　次に示すどちらかの条件を満たす場合に分岐行ログが出力されます。

- ノースプリットオプションの指定がなく，実際のデータ長が128文字以上の場合
- ノースプリットオプションの指定があり，実際の1行のデータ長の合計がページ長を超える場合

注※3　実際の 1 行のデータ長の合計がページ長を超える場合に分岐行ログが出力されます。

表 16–2　データベースの操作内容と求めるログ量

操作内容（SQL 文）	求めるログ量
キーの追加（INSERT 文）	追加ログ量
キーの削除（DELETE 文）	削除ログ量
キーの更新（UPDATE 文）	更新前キーの削除ログ量＋更新後キーの追加ログ量

データベースを更新（INSERT，DELETE 及び UPDATE）する場合に出力されるシステムログ量は，操作内容（INSERT，DELETE 及び UPDATE）によって異なります。システムログ量の計算式を次に示します。

なお，ログを取得しない UAP 実行時に出力されるシステムログ量は，セグメント確保が発生する場合のログ量として 460 バイトとなります。

条件	計算式（単位：バイト）
表に n 行追加（INSERT）する場合又は表から n 行削除（DELETE）する場合に出力されるシステムログ量	（a ＋ b ＋ c）×n
表中の n 行を更新（UPDATE）する場合に出力されるシステムログ量	（a※1 ＋ d※2 ＋ e※3）×n

a：基本行ログ量（バイト）

b：全分岐行ログ量の合計値（バイト）

c：全インデクスログ量の合計値（バイト）

d：更新対象分岐行ログ量の合計値（バイト）

e：更新対象インデクスログ量の合計値（バイト）

n：操作する行数（件）

注※1　更新（UPDATE）する列値が基本行内にある場合に加算します。

注※2　更新（UPDATE）する列値が分岐行内にある場合に加算します。

注※3　更新（UPDATE）する列にインデクスを定義する場合に加算します。

(1)　基本行ログ量の見積もり

データ 1 件当たりの基本行ログ量の計算式を次の表に示します。

表 16–3　データ 1 件当たりの基本行ログ量の計算式

データの操作内容（SQL 文）		出力されるログ量（単位：バイト）
データの追加（INSERT 文）		k ＋ 152
データの削除（DELETE 文）		
データの更新 （UPDATE 文）	FIX 表で f≦12	f Σ d_i ＋ i=1 f Σ d_j ＋ 4×f ＋ 152 j=1
	非 FIX 表 又は f ＞ 12	k1 ＋ k2 ＋ 160 ●pd_nowait_scan_option オペランドに LOCK を指定する場合に加算します。 314×2
	行インタフェース使用時	2×k1 ＋ 160

k：追加又は削除する行長

k1：更新する行の更新前行長

k2：更新する行の更新後行長

f：更新する列数

d_i：更新する列の更新前データ長

d_j：更新する列の更新後データ長

注 1

k，k1，k2 の値は，操作する表に FIX 指定をするかどうかによって異なります。それぞれの計算式を次に示します。なお，表のデータ長については次を参照してください。

- 表「データ長一覧」
- 表「可変長文字列型のデータ長一覧（抽象データ型及び繰返し列を除く）」
- 表「可変長文字列型のデータ長一覧（抽象データ型の場合）」
- 表「可変長文字列型のデータ長一覧（繰返し列の場合）」
- **FIX 指定をする場合**
  表の全列のデータ長の合計値＋ 4
- **FIX 指定をしない場合**
  表の全列のデータ長の合計値＋ 6 ＋ 2×表の全列数

注 2

HiRDB Datareplicator を使用している場合，又は更新可能なオンライン再編成実行中の場合，FIX 表に対して 12 列以下の UPDATE を列単位で実行すると，13 列以上の UPDATE と同量のログが出力されます。

注 3

表に BLOB 列が定義されている場合，表「データ 1 件当たりの基本行ログ量の計算式」の行長は BLOB 列のとき 9 バイト固定とし，次の表に示すログ量を加算してください。

表 16-4　BLOB 列のデータ 1 件当たりのログ量の計算式

データの操作内容 (SQL 文)	recovery の指定	出力されるログ量（単位:バイト）
データの追加 (INSERT 文)	なし又は partial	604 + 180×p5
	all	2348 + p1 + 8340×p2 +（148 + lt）×p3
	no	300
データの削除 (DELETE 文)	なし又は partial	460 + 180×p6
	all	
	no	468
データの更新 (UPDATE 文)	なし又は partial	604 + 180×p5 + 460 + 180×p6
	all	2496 + p1 + 8340×p2 +（148 + lt）×p3
	no	312
データの連結演算 (UPDATE 文) Bb≦7168	なし又は partial	2344
	all	8340×a + 1600 + d + 8340×p4 +（148 + lt2）×p3
	no	428
データの連結演算 (UPDATE 文) Bb ＞ 7168	なし又は partial	2772
	all	2772 + Ba + 8340×p4 +（148 + lt2）×p3
	no	428
データの後方削除更新 (UPDATE 文) Bb≦7168	なし又は partial	2344
	all	9512
	no	428
データの後方削除更新 (UPDATE 文) Bb ＞ 7168	なし又は partial	2492 + 180×↑（Bb-Bd）÷8192↑
	all	2492 + 180×↑（Bb-Bd）÷8192↑
	no	428 + 180×↑（Bb-Bd）÷8192↑

Bi：BLOB データ長（単位：バイト）

Ba：次のどちらかの値

- Bb ＞ 7168 の場合：8192－｛(Bb－7168）－↓（Bb－7168）÷8192↓×8192｝
- Bb≦7168 の場合：0

Bb：BLOB 更新前データ長（単位：バイト）

Bc：BLOB 追加データ長（単位：バイト）

Bd：更新後データ長（SUBSTR 関数の値式 3 の指定値）（単位：バイト）

lt：次のどちらかの値

- Bi ＞ 7168 の場合：Bi－7168－↓（Bi－7168）÷8192↓×8192
- Bi≦7168 の場合：0

lt2：次のどちらかの値

- Bc ＋ Bb ＞ 7168 の場合：(Bc ＋ Bb－Ba－7168）－↓（Bc ＋ Bb－Ba－7168）÷8192↓×8192
- Bc ＋ Bb≦7168 の場合：0

p1：次のどちらかの値

- Bi ＞ 7168 の場合：7168
- Bi≦7168 の場合：Bi

p2：次のどちらかの値

- Bi ＞ 7168 の場合：↓（Bi－7168）÷8192↓
- Bi≦7168 の場合：0

p3：次のどちらかの値

- lt ＝ 0 又は lt2 ＝ 0 の場合：0
- lt ＞ 0 又は lt2 ＞ 0 の場合：1

p4：次のどちらかの値

- Bc ＋ Bb ＞ 7168 の場合：↓（Bc ＋ Bb－Ba－7168）÷8192↓
- Bc ＋ Bb≦7168 の場合：0

p5：次のどちらかの値

- Bi ＞ 0 の場合：↑（Bi ＋ 1024）÷8192↑
- Bi ＝ 0 の場合：0

p6：次のどちらかの値

- Bb ＞ 0 の場合：↑（Bb ＋ 1024）÷8192↑
- Bb ＝ 0 の場合：0

a：次のどちらかの値

- Bb ＞ 0 の場合：1
- Bb ＝ 0 の場合：0

d：次のどちらかの値

- Bc ＋ Bb≦7168 の場合：Bc ＋ Bb
- Bc ＋ Bb ＞ 7168 の場合：7168

注 4

pd_idx_without_rollback=Y を指定し，インデクスを定義した WITHOUT ROLLBACK 指定表に対して，INSERT 文，DELETE 文又はインデクス構成列を更新する UPDATE 文を実行する場合に 312 を加算してください。

(2) 分岐行ログ量の見積もり

(a) ノースプリットオプション指定なしの VARCHAR，NVARCHAR 及び MVARCHAR の分岐行ログ量

発生する分岐行分のログ量を算出してその合計値を求めます。1 分岐行ログ量の計算式を次の表に示します。

表 16–5 1 分岐行のログ量の計算式（その 1）

データの操作内容（SQL 文）		出力されるログ量 （単位：バイト）
データの追加（INSERT 文）		k ＋ 152
データの削除（DELETE 文）		
データの更新 （UPDATE 文）	更新によって新たな分岐行ができる場合	k2 ＋ 160
	分岐行を更新する場合	k1 ＋ k2 ＋ 160
	更新によって分岐行が削除される場合	k1 ＋ 160

k：追加又は削除する一つの分岐行長

k1：更新する一つの更新前分岐行長

k2：更新する一つの更新後分岐行長

注 k，k1 及び k2 の行長は，次に示す計算式で求めます。

8 ＋ MIN（平均データ長の実長 , RD エリアのページ長－48 ）

(b) 抽象データ型の列，繰返し列，BINARY 型の列，ノースプリットオプション指定ありの VARCHAR，NVARCHAR 及び MVARCHAR の分岐行ログ量

発生する分岐行分のログ量を算出してその合計値を求めます。1 分岐行ログ量の計算式を次の表に示します。

ただし，表格納 RD エリアが複数あり，ページ長が各 RD エリアで異なる場合は，同じページ長の RD エリアごとに計算して，その合計を分岐行ログ量として求めてください。

表 16-6　1 分岐行のログ量の計算式（その 2）

データの操作内容（SQL 文）	出力されるログ量（単位：バイト）
データの追加（INSERT 文）	SPN×（b ＋ 152）
データの削除（DELETE 文）	
データの更新（UPDATE 文）	SPN×（b ＋ 160）
データの連結演算（UPDATE 文）※1	（b ＋ 160）＋（SPN−1）×（b ＋ 152） ●pd_rpl_hdepath オペランドを指定する場合に加算します。 160
データの後方削除更新（UPDATE 文）※1，※2	（b×2 ＋ 160）＋（↓（c−d）÷（b−57）↓）×（b ＋ 152） ●pd_rpl_hdepath オペランドを指定する場合に加算します。 160

注※1

BINARY 型の列の場合だけです。

注※2

圧縮列の場合は該当しません。圧縮列のデータの後方削除更新（UPDATE 文）は，データの更新（UPDATE 文）で算出してください。

b：RD エリアのページ長

c：更新前データ長

d：更新後データ長（SUBSTR 関数の値式 3 の指定値）

SPN：求め方を次に示します。

分岐する列がある場合（分岐する条件については表「データ長一覧」の注※5 で説明），INSERT 文及び DELETE 文は表を構成するすべての列について，UPDATE 文は更新対象列について次に示す値を求めてください。ただし，BINARY 型の列の連結演算の場合，di は追加するデータ長として計算してください。

```
SPN＝SPN1＋SPN2
```

SPN1，及び SPN2 については表「表の格納ページ数の計算方法」の「計算式中で使用する変数」を参照してください。ただし，a の値には 1 を設定してください。

(c)　更新可能なオンライン再編成時に追加出力されるログ量

更新可能なオンライン再編成中（オンライン再編成閉塞状態の副系 RD エリアにアクセスしている間）に更新可能なオンライン再編成実行対象 RD エリアに対して UAP がアクセスするときに出力するログ量見

積もりは，通常出力するログ量に加えて，pdorend 反映プロセス向けのイベントログ量を追加する必要があります。追加するログ量について説明します。

- 更新可能なオンライン再編成中は，1 行分の更新で行更新開始イベントログ及び行更新終了イベントログを出力します。また，ロールバックするとき 1 行分の回復で行回復開始イベントログ及び行回復終了イベントログを出力します。そのため，システムログファイルの容量見積もり時，表「データ 1 件当たりの基本行ログ量の計算式」の計算式に，次の表に示すログ量を加算してください。

表 16-7　更新可能なオンライン再編成時にデータ 1 件当たりに追加されるシステムログ量

データの操作内容（SQL 文）	追加出力されるログ量（単位：バイト）
INSERT	320
DELETE	
UPDATE	320 + c※1 + 160※2
INSERT のロールバック	320
DELETE のロールバック	
UPDATE のロールバック	496 +行更新前データ長+ c※1 + 160※2

注※1

オンライン再編成データベース静止化コマンド（pdorbegin）の-e オプション指定時に c を加算します。c の計算式を次に示します。

（↑（↑（a−b + 1）÷8↑+ 6）÷4↑）×4×2

a：更新する列のうちで最も大きい列の ID

b：更新する列のうちで最も小さい列の ID

注※2

定義長 32001 以上の BINARY 型の列に対し連結演算，又は後方削除更新を行った場合に加算します。

- 更新可能なオンライン再編成中は，表に BLOB 列が定義されている場合は BLOB 更新のイベントログが追加されます。そのため，システムログファイルの容量見積もり時，表「BLOB 列のデータ 1 件当たりのログ量の計算式」の計算式に，次の表に示すログ量を加算してください。

表 16-8　更新可能なオンライン再編成時に BLOB 列のデータ 1 件当たりに追加されるログ量

データの操作内容（SQL 文）	追加出力されるログ量（単位：バイト）
INSERT	0
DELETE	224
UPDATE	224 + 160※2
INSERT のロールバック	0
DELETE のロールバック	148 + p1 + 8340×p2 +（148 + lt）×p3※1
UPDATE のロールバック	148 + p1 + 8340×p2 +（148 + lt）×p3※1 + 160※2

注※1

変数 p1，p2，p3，lt については，表「BLOB 列のデータ 1 件当たりのログ量の計算式」を参照してください。また，これらの変数には削除する BLOB データ長を代入してください。

注※2

連結演算，又は後方削除更新を行った場合に加算します。

- 更新可能なオンライン再編成中は，次の場合は新たに繰返し列更新のイベントログを出力します。そのため，システムログファイルの容量見積もり時，表「1 行を操作したときに出力されるイベントログ量」の計算式に，次の表に示すログ量を加算してください。
  - 繰返し列を含む行を削除した場合
  - 繰返し列全体を指定して更新する場合
  - 繰返し列の要素を指定して更新する場合
  - 繰返し列の要素を指定して削除する場合
  - 繰返し列の指定した要素だけを指定して更新する場合

表 16-9　更新可能なオンライン再編成時に繰返し列を含むデータ 1 件当たりに追加されるログ量

データの操作内容（SQL 文）	追加出力されるログ量（単位：バイト）
UPDATE-SET（列指定）	164
UPDATE-ADD（要素指定） （更新によって新たな要素が追加される場合）	$\sum_{i=1}^{n} 164$
UPDATE-DELETE（要素指定） （更新によって新たな要素が削除される場合）	$\sum_{i=1}^{m} (p4 + 160)$
UPDATE-SET（要素指定） （指定した要素だけを更新する場合）	$\sum_{i=1}^{n} ((p4 + 160) \times 2)$
DELETE	156
UPDATE-SET（列指定）のロールバック	156
UPDATE-ADD（要素指定）のロールバック （更新によって新たな要素が削除される場合）	$\sum_{i=1}^{n} (p4 + 180)$
UPDATE-DELETE（要素指定）のロールバック （更新によって新たな要素が追加される場合）	$\sum_{i=1}^{m} 184$
UPDATE-SET（要素指定）のロールバック （指定した要素だけを更新する場合）	$\sum_{i=1}^{n} (p4 + 180)$

データの操作内容（SQL 文）	追加出力されるログ量（単位：バイト）
	i=1
DELETE のロールバック	156

n：更新対象の繰返し列数

m：削除対象の要素数

p4：↑｛↑（指定する最大添字番号の平均値－指定する最小添字番号の平均値＋ 1）÷8↑｝÷4↑×4

(3) インデクスログ量の見積もり

1 本のインデクスの 1 行ごとの操作で出力するインデクスログ量は，次の表に示す計算式で求めます。

表 16-10 1 本のインデクスログ量の計算式

キーの操作内容（SQL 文）			出力されるログ量（単位：バイト）		
			pd_indexlock_mode オペランドの指定値		
			KEY	NONE	
				非ユニーク属性のインデクス	ユニーク属性のインデクス
キーの追加（INSERT 文）	新しいキーを追加する場合		k1 ＋ 156	k1 ＋ 156	k1×2 ＋ 356
	追加するキーと同じ行が既にある場合	d≦200	k1 ＋ 156	k1 ＋ 156	－
		d ＞ 200	k1 ＋ 292	k1 ＋ 292	－
キーの削除（DELETE 文）	キー値を削除する場合		k2 ＋ 156	k2 ＋ 156	k2 ＋ 160 ＋ a
	削除するキー値と同じ値を持つ行がほかにある場合	d≦200	k2 ＋ 156	k2 ＋ 156	－
		d ＞ 200	k2 ＋ 292	k2 ＋ 308	－
キーの更新（UPDATE 文）			キー削除のログ量＋キー追加のログ量		

d：キー値の重複数

k1：追加するキー長（バイト）

k2：削除するキー長（バイト）

a：次に示すどちらかを代入します。

- pd_unique_check_mode オペランドの指定値が 1 の場合：20
- pd_unique_check_mode オペランドの指定値が 0 の場合： 0

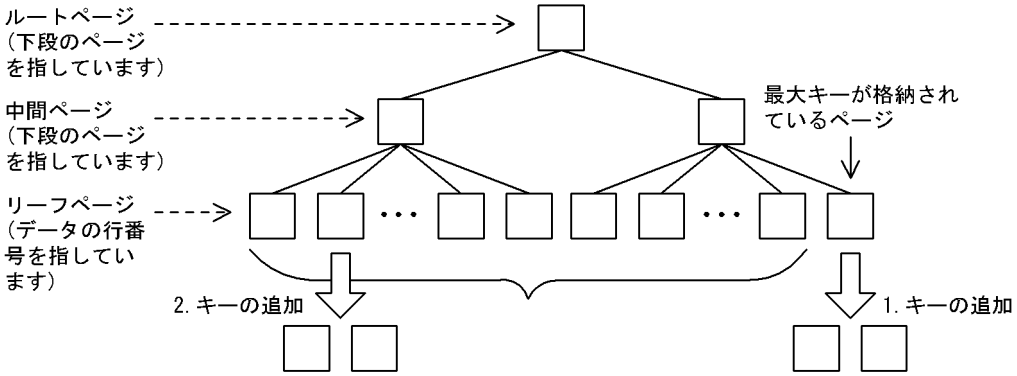
注

ここでいうキー長とは，DB 格納キー長のことです。キー長の求め方については，「インデクスの格納ページ数の計算方法」を参照してください。

(a)　インデクスページスプリット時のインデクスログ量の見積もり

インデクスページスプリットの概念を次の図に示します。

図 16–1　インデクスページスプリットの概念



［説明］

1. リーフページの一番右側（図中の最大キーが格納されているページ）にキーが追加され，ページが二つに分割されていることを最大キーが入っているページスプリットといいます。
2. そのほかのリーフページにキーが追加され，ページが二つに分割されていることを最大キーが入っていないページスプリットといいます。

インデクス格納ページをスプリットする場合，HiRDB は次に示す二つの方法でキー値を格納します。

●最大キー値が入っていないページスプリットの場合

キー値を追加又は削除すると，キーと未使用領域の比率をおよそ 50：50 にスプリットしてキー値を格納します。最大キー値が入っていないページスプリットが発生する条件を次に示します。

- インデクス格納ページ内にキーが入らないとき
- 重複するキー値が 201 件以上あり，同じキー値を持つ行を a 件以上追加するとき
  a は次に示す計算式で求めます。このとき，a 件に 1 回はスプリットが発生します。

計算式

a ＝↑インデクスを格納する RD エリアのページ長（バイト）÷4↑

●最大キー値が入っているページスプリットの場合

最大キー値が格納されているインデクス格納ページに対してキー値を追加又は更新すると，キーと未使用領域の比率を CREATE INDEX の PCTFREE オペランドの指定値に従ってスプリットし，キー値を格納します。

例えば，PCTFREE ＝ 30 と指定すると，キーと未使用領域の比率をおよそ 70：30 にスプリットしてキー値を格納します。

キーの追加によってCREATE INDEXのPCTFREEオペランドに指定する分の空き領域が確保できないときに，最大キー値が入っているページスプリットが発生します。ただし，上位レベルページは該当しません。

スプリット種別による1回当たりのインデクスログ量の計算式を次の表に示します。

表16–11　スプリット種別による1回当たりのインデクスログ量の計算式

スプリット種別	条件			出力されるログ量（単位：バイト）
最大キーが入っているページのスプリット	インデクス中のキー値と異なるキーを追加した場合	上位ページにキーを追加するための空き領域が未使用領域にある場合	2×k1 ＋ a ＋ 8×（m ＋ 1）≦31516の場合	2×k1 ＋ 472 ＋ a ＋ 8×（m ＋ 1）
			2×k1 ＋ a ＋ 8×（m ＋ 1）＞ 31516の場合	2×k1 ＋ 632 ＋ a ＋ 8×（m ＋ 1）
		上位ページにキーを追加するための空き領域が未使用領域にない場合	2×k1 ＋ a ＋ 8×（m ＋ 1）≦31516の場合	n-1 Σ（288 ＋ a） i=2 ＋ 2×k1 ＋ 472 ＋ a ＋ 8×（m ＋ 1）
			2×k1 ＋ a ＋ 8×（m ＋ 1）＞ 31516の場合	n-1 Σ（288 ＋ a） i=2 ＋ 2×k1 ＋ 628 ＋ a ＋ 8×（m ＋ 1）
	インデクス中に追加するキー値と同じキーがある場合	d1≦200	上位ページにキーを追加するための空き領域が未使用領域にある場合	2×k1 ＋ 472 ＋ a ＋ 8×（m ＋ 1）
			上位ページにキーを追加するための空き領域が未使用領域にない場合	n-1 Σ（288 ＋ a） i=2 ＋ 2×k1 ＋ 472 ＋ a ＋ 8×（m ＋ 1）
		d1 ＞ 200	下位ページにキーを追加するための空き領域が未使用領域にある場合	k1 ＋ 472 ＋ a
			下位ページにキーを追加するための空き領域が未使用領域にない場合	k1 ＋ 462 ＋ 2×a
最大キーが入っていないページのスプリット	キーを追加するための空き領域が未使用領域にない場合	上位ページにキーを追加するための空き領域が未使用領域にある場合	2×k1 ＋ a ＋ 8×（m ＋ 1）≦31516の場合	2×k1 ＋ 332 ＋ a ＋ 8×（m ＋ 1）
			2×k1 ＋ a ＋ 8×（m ＋ 1）＞ 31516の場合	2×k1 ＋ 492 ＋ a ＋ 8×（m ＋ 1）

スプリット種別	条件			出力されるログ量（単位：バイト）
		上位ページにキーを追加するための空き領域が未使用領域にない場合	2×k1 ＋ a ＋ 8×（m ＋ 1）≦31516 の場合	n-1 Σ（288 ＋ a） i=2 ＋ 2×k1 ＋ 332 ＋ a ＋ 8 ×（m ＋ 1）
			2×k1 ＋ a ＋ 8×（m ＋ 1）＞ 31516 の場合	n-1 Σ（288 ＋ a） i=2 ＋ 2×k1 ＋ 492 ＋ a ＋ 8 ×（m ＋ 1）

a：インデクスを格納している RD エリアのページ長（バイト）

d1：キー値の重複数

k1：追加するキー長（バイト）

ここでいうキー長とは，DB 格納キー長のことです。キー長の求め方については，「インデクスの格納ページ数の計算方法」を参照してください。

m：スプリットが発生した時点のインデクス段数

n：スプリットが波及した上位ページの段数

リーフページのスプリットによって波及した上位ページが，さらにスプリットした場合は n ＝ 3（n≧3）となります。

注

この計算式は，1 件当たりの行及びインデクス部分の更新ログ量の見積もり式です。このため，この計算式には，追加及び更新によって，新しいページ又はセグメントを確保したときのシステム管理情報に関するログ量を含んでいません。したがって，大量のデータを扱う場合は，次の表に示すログ量を加算して見積もる必要があります。

表 16-12　ページ／セグメントの確保ログ量の計算式

条件	出力されるログ量（単位：バイト）
データの追加（INSERT）又は更新（UPDATE）によって，行の格納ページを新しく確保する場合	440
データの追加（INSERT）又は更新（UPDATE）によって，インデクスのページスプリットが発生する場合	544×n ＋ 272
上記のページ確保に対して，セグメントの確保が発生する場合（セグメントサイズ数分のページ確保をするごとに発生します）	1940

n：ページスプリットが発生した時点のインデクスの段数＋ 1

(4) イベントログ量の見積もり

イベントログは HiRDB Datareplicator を使用しているときに，繰返し列を含む行データを追加，削除又は更新すると出力されます。1 行を操作したときに出力されるイベントログ量を次の表に示します。

表 16–13　1 行を操作したときに出力されるイベントログ量

データ操作内容		イベントログ量 （単位：バイト）
データの追加（INSERT 文）		156×n
データの更新（UPDATE 文）	更新によって新しい要素が追加される場合（UPDATE ADD）	164×n
	更新によって要素が削除される場合 （UPDATE DELETE）	n Σ （p5 +160） i=1
	指定した要素だけを更新する場合 （UPDATE SET）	n Σ （p5 +160） i=1
	指定した繰返し列を更新する場合 （UPDATE SET）	164×n

n：更新対象となる繰返し列の数

p5：↑｛↑（指定する最大添字番号の平均値－指定する最小添字番号の平均値＋ 1）÷8↑｝÷4↑×4

(5) ログレスモードの表データ更新時に出力されるシステムログ量の見積もり

ログレスモードの表データ更新時に出力されるシステムログ量は，次に示す計算式で求めます。

計算式

```
（a＋b＋c）×h×2
＋（（d＋c）×156）
＋（（e＋f）×148）
＋m
```

a：更新対象表を格納する RD エリア数（個）

b：更新対象表のインデクスを格納する RD エリア数（個）

c：更新対象表の LOB 列を格納する RD エリア数（個）

d：更新対象表数（個）

e：表データ更新による表の使用中セグメントの増加数：
↑（データ更新後の表の格納ページ数－データ更新前の表の格納ページ数）

÷更新対象表を格納する RD エリアのセグメントサイズ↑

f：表データ更新によるインデクスの使用中セグメントの増加数：

↑（データ更新後のインデクスの格納ページ数

－データ更新前のインデクスの格納ページ数）

÷更新対象表のインデクスを格納する RD エリアのセグメントサイズ↑

h：↓pd_log_max_data_size オペランドの値/1000↓×1000

m：インナレプリカ機能を使用している環境の場合だけ，次の値を加算してください。

（a ＋ b ＋ c）×h

16.1.5 ユティリティによるデータベース作成時に出力されるシステムログ量

次に示すユティリティを実行する場合，表「ユティリティによるデータベース作成時に出力されるシステムログ量の計算式」に示すシステムログが出力されます。

- データベース作成ユティリティ（pdload コマンド）
- データベース再編成ユティリティ（pdrorg コマンド）
- リバランスユティリティ（pdrbal コマンド）

システムログ量は表「ユティリティによるデータベース作成時に出力されるシステムログ量の計算式」で求めた値に，表「システムログ量の算出時に加算する値と加算する場合の条件」に示す値を加算して算出します。なお，表及びインデクスを横分割している場合は，表及びインデクスを格納する RD エリアごとにシステムログ量を計算します。

- HiRDB/シングルサーバの場合

  RD エリアごとに計算したシステムログ量の合計を，シングルサーバで出力するログ量に加算してください。
- HiRDB/パラレルサーバの場合

  RD エリアごとに計算したシステムログ量の合計を，処理対象の表又はインデクスを管理するバックエンドサーバのログ量に加算してください。ただし，データベース再編成ユティリティ（pdrorg）でディクショナリ表の再編成を行う場合は，ディクショナリサーバで出力するログ量に加算してください。

表 16-14 ユティリティによるデータベース作成時に出力されるシステムログ量の計算式

項番	条件	出力されるシステムログ量（単位：バイト）	
		-l オプションに a を指定	-l オプションに p を指定
1	インデクスを一括作成する場合（-i オプションに c を指定） 対象ユティリティ	$\sum_{i=1}^{n}\{[132 + \uparrow(100-f)\div 100\uparrow \times X_i] \times W_i$	$\sum_{i=1}^{n}\{280 \times W_i + 1940 \times \uparrow W_i \div V_i\uparrow$

項番	条件	出力されるシステムログ量（単位：バイト）	
		-l オプションに a を指定	-l オプションに p を指定
	• pdload • pdrorg • pdrbal	＋280×W_i＋1940×↑W_i÷V_i↑ } ＋280×m＋1940×↑m÷s↑＋a×r	} ＋280×m＋1940×↑m÷s↑＋c×r

2	インデクス更新モードでインデクスを作成する場合（-i オプションに s を指定）
	対象ユティリティ
	• pdload
	• pdrorg
	• pdrbal
		n
		Σ {
		i=1
		280×W_i＋1940×↑W_i÷V_i↑
		＋b×r＋e×d
		}
		＋280×m＋1940×↑m÷s↑＋a×r
			n
			Σ {
			i=1
			280×W_i＋1940×↑W_i÷V_i↑
			＋b×r＋e×d
			}
			＋280×m＋1940×↑m÷s↑＋c×r

3	インデクスを作成しない場合（-i オプションに n 又は x を指定） 対象ユティリティ • pdload • pdrorg • pdrbal	280×m＋1940×↑m÷s↑＋a×r	280×m＋1940×↑m÷s↑＋c×r
4	インデクスを再作成，又は再編成する場合（pdload 0 件ロード，又は pdrorg -k オプションに ixrc 若しくは ixor を指定）	n Σ {280wi+1940×↑Wi÷Vi↑ i=1 ＋(「インデクス定義時に出力されるシステムログ量」) ＋(表「ページ／セグメントの確保ログ量の計算式」) ＋(表「PURGE TABLE 文実行時に出力されるシステムログ量」のインデクスのシステムログ量) }	n Σ {280wi+1940×↑Wi÷Vi↑ i=1 ＋(表「ページ／セグメントの確保ログ量の計算式」) ＋(表「PURGE TABLE 文実行時に出力されるシステムログ量」のインデクスのシステムログ量) }

注

インデクスの一括作成又はインデクスの作成時のシステムログ量は，作成するインデクス数分の計算が必要です。

a：表「データ 1 件当たりの基本行ログ量の計算式」，表「BLOB 列のデータ 1 件当たりのログ量の計算式」で求めた，データを 1 件追加するときに出力されるログ量

b：表「1 本のインデクスログ量の計算式」で求めたデータを 1 件追加するときに出力されるログ量

c：表「BLOB 列のデータ 1 件当たりのログ量の計算式」で求めた，データを 1 件追加するときに出力されるログ量

d：インデクススプリット時に出力されるシステムログ量

表「スプリット種別による 1 回当たりのインデクスログ量の計算式」及び表「ページ／セグメントの確保ログ量の計算式」を参照してください。

e：インデクスのスプリット回数

f：インデクス定義時に指定する PCTFREE オペランド（ページ内未使用領域比率）の値（%）

m：表の格納ページ数（ページ）

「表の格納ページ数の計算方法」を参照してください。

n：表に定義されているインデクス数（個）

r：表に格納する行数（行）

s：表を格納するユーザ用 RD エリアのセグメントサイズ（ページ）

V_i：インデクスを格納するユーザ用 RD エリアのセグメントサイズ（ページ）

W_i：インデクスの格納ページ数（ページ）

「インデクスの格納ページ数の計算方法」を参照してください。

X_i：インデクスを格納するユーザ用 RD エリアのページ長（バイト）

表「ユティリティによるデータベース作成時に出力されるシステムログ量の計算式」の項番 1，及び項番 2 の場合に，システムログ量の算出時に加算する値と，加算する場合の条件を次の表に示します。

表 16-15　システムログ量の算出時に加算する値と加算する場合の条件

加算する場合の条件	システムログ量に加算する値
LOB 列を定義している表の場合	次に示すログ量を（LOB 列数×行数）の数分加算してください。 • LOB 列の回復属性が recovery no 以外の場合 3544×（LOB データ長÷31744）（単位：バイト）
データベース作成ユティリティ（pdload）を-d オプション指定で，又はデータベース再編成ユティリティ（pdrorg）を実行する場合	表「PURGE TABLE 文実行時に出力されるシステムログ量」の「表」又は「リバランス表」と，対象表に定義されている全インデクス，及び全 LOB 列（又は LOB 属性）のログ量を加算してください。
データベース再編成ユティリティ（pdrorg）で LOB 列，又は LOB 属性がある表を再編成する場合で次の二つの条件を満たすとき • -j オプションを指定しないで再編成する	次に示すログ量を LOB 用 RD エリアの構成ファイル数分加算してください。 Σ｛17000×↑（HiRDB ファイルのセグメント数÷64000）↑×95｝
	次に示すログ量を（LOB 列又は LOB 属性の数×行数）の数分加算してください。 **●LOB 列が recovery all 指定かつ-l a 指定の場合** ↑17600÷sr↑×sr **●上記以外の場合** ↑3200÷sr↑×sr

加算する場合の条件	システムログ量に加算する値
• LOB 列が recovery no 指定でなく，かつ-l オプションが n 指定でない	sr：pd_log_rec_leng オペランドで指定するシステムログファイルのレコード長
HiRDB Datareplicator 連携機能を使用（pd_rpl_hdepath オペランドを指定）していて，表に繰返し列を含む場合	表「1 行を操作したときに出力されるイベントログ量」に示すログ量を追加行数分加算してください。

16.1.6 SQL 操作に応じて出力されるシステムログ量

システムログは，システム共通定義の pdhibegin オペランドに-k cnc を指定している状態で，CONNECT，DISCONNECT 又は set session authorization を実行した場合に出力されます。

SQL 操作に応じて出力されるシステムログ量を求める計算式を次に示します。HiRDB/シングルサーバの場合はシングルサーバで出力するログ量，HiRDB/パラレルサーバの場合はフロントエンドサーバで出力するログ量に，このログ量を加算してください。

計算式

```
システムログ量（単位：バイト）＝568×（CONNECT回数＋set session authorization実行回数）
```

16.1.7 拡張システム定義スカラ関数の定義時に出力されるシステムログ量

pdextfunc コマンドで拡張システム定義スカラ関数を定義した場合に出力されるシステムログ量を次に示します。HiRDB/シングルサーバの場合はシングルサーバで出力するログ量，HiRDB/パラレルサーバの場合はディクショナリサーバで出力するログ量に，このログ量を加算してください。

```
システムログ量（単位：バイト）＝233529
```

16.1.8 RD エリアの自動増分機能使用時に出力されるシステムログ量

RD エリアの自動増分機能を使用すると，自動増分の実行時にシステムログが出力されます。

出力されるシステムログ量を求める計算式を次の表に示します。HiRDB/シングルサーバの場合はシングルサーバで出力するログ量，HiRDB/パラレルサーバの場合は対象となる RD エリアを管理するバックエンドサーバ及びディクショナリサーバで出力するログ量に，このログ量を加算してください。

表 16–16　RD エリアの自動増分機能使用時に出力されるシステムログ量

RD エリアの種類	システムログ量（単位：バイト）
LOB 用 RD エリアの場合	1956
LOB 用 RD エリア以外の場合	1372 ＋（144 ＋ p）×2

p：自動増分する RD エリアのページサイズ

16.1.9　PURGE TABLE 文実行時に出力されるシステムログ量

PURGE TABLE 文実行時に出力されるシステムログ量は，表に定義されているすべてのインデクス，LOB 列，又は LOB 属性分のログ量を合計して求めます。分割表，及び分割インデクスの場合は RD エリアごとに求めたログ量を合計して求めます。

求めた値を，HiRDB/シングルサーバの場合はシングルサーバで出力するログ量に加算してください。HiRDB/パラレルサーバの場合は，対象となる表が格納されている RD エリア（表に定義されているインデクス，LOB 列，及び LOB 属性が格納されている RD エリアも含む）を管理するバックエンドサーバのログ量に加算してください。

PURGE TABLE 文実行時のシステムログ量の計算式を次の表に示します。なお，計算式中の変数 A，B，C は RD エリア構成ファイルごとに求めてください。

表 16–17　PURGE TABLE 文実行時に出力されるシステムログ量

種別	システムログ量（単位：バイト）
表	1000 ＋ 1100×確保セグメント数
リバランス表	1000 ＋ 1100×確保セグメント数＋ 400×↑1024÷分割 RD エリア数↑
インデクス※	1000 ＋ 1100×確保セグメント数
LOB 列又は LOB 属性※	1000 ＋ 17000×A ＋ 160×B ＋ 2×C

A：↑（HiRDB ファイルの使用中セグメント数÷64000）↑×95

B：↑（HiRDB ファイルの使用中セグメント数÷64000）↑

C：↑（HiRDB ファイルの使用中セグメント数÷64000）↑×8150

注※

プラグインの場合は，各プラグインの初期化のログが出力されます。各プラグインのマニュアルを参照してください。

16.1.10　空きページ解放ユティリティ（pdreclaim）実行時に出力されるシステムログ量

空きページ解放ユティリティ（pdreclaim）による表，インデクスの空きページ，又はセグメントの解放をする場合，システムログを出力します。

指定したオプションと対象となる資源の組み合わせによるログ量を次に示します。なお，ログ量の説明の項番は，表「空きページ解放ユティリティ実行時に出力されるシステムログ量」の項番に対応しています。

オプション	対象資源	ログ量
なし	表	項番(1)の値
	インデクス	項番(2)の値
-j	表	項番(3)の値
	インデクス	
-a	表	項番(1)と(4)を合計した値
	インデクス	項番(2)と(4)を合計した値

ログ量を求める計算式を次の表に示します。なお，表及びインデクスを横分割している場合は分割した RD エリアごとに計算する必要があります。HiRDB/シングルサーバの場合は，横分割した RD エリアごとに計算したログ量の合計を求めます。これを，シングルサーバが出力するログ量に加算してください。HiRDB/パラレルサーバの場合は，横分割した RD エリアごとに計算したログ量をサーバごとに求め，その合計を求めます。これを，対象資源が格納されている RD エリアを管理するバックエンドサーバ及びディクショナリサーバのログ量に，それぞれ加算してください。

表 16-18　空きページ解放ユティリティ実行時に出力されるシステムログ量

項番	種別		システムログ量（単位：バイト）
(1)	表		140×n ●-o オプションを指定していない場合に加算します。 ＋ 504×m
(2)	インデクス	キー値の重複が 201 以上のキーが存在しない	{((k ＋ 14) ÷ (j−68)) ×1408 ＋ k ＋ 12×h ＋ 908} ×n ＋ 304×q ●-x オプションを指定している場合に加算します。 ＋ {m− n − (((k ＋ 14) ×2×m) ÷ (j−68))} × (j ＋ 76) ×0.7
		キー値の重複が 201 以上のキーが存在する	(724 ＋ k) ×n ＋ 285×MAX (n, 16×n÷ (j−78)) −159 ＋ 304×q ●-x オプションを指定している場合に加算します。 ＋ {m− n − (((k ＋ 14) ×2×m) ÷ (j−68))} × (j ＋ 76) ×0.7
(3)	セグメ	-j オプション指定時	2380×p ●-k オプションに index を指定している場合に加算します。 ＋ 304×p

項番	種別		システムログ量（単位：バイト）
(4)	ント	-a オプション指定時	3100×p ●-k オプションに index を指定している場合に加算します。 ＋ 304×p

h：インデクスの段数（段）

j：ページサイズ（バイト）

k：インデクスキーの長さ（バイト）

m：使用中ページ数（満杯ページを除く）（個）

n：使用中空きページ数（個）

p：使用中空きセグメント数（個）

解放途中セグメントの数を含みます。

q：空きありセグメント数（個）

使用中セグメント数から満杯セグメント数を引いた数となります。

使用中セグメント数と満杯セグメント数は，データベース状態解析ユティリティ（pddbst）で確認できます。

16.1.11　再編成時期予測機能使用時に出力されるシステムログ量

再編成時期予測機能を使用する場合，HiRDB/パラレルサーバの場合はディクショナリサーバで出力するログ量，HiRDB/シングルサーバの場合はシングルサーバで出力するログ量に，次の計算式で求められるログ量を加算する必要があります。

計算式

```
システムログ量（単位：バイト）＝
  n× {1604× (A＋B＋C＋D) × (↑E↑＋1) }
  ＋m× {872× (a＋b＋c＋1) }
  ＋11680×↑ { (A＋B＋C＋D) × (↑E↑＋1) } ×30÷540↑
  ＋332× { (A＋B＋C＋D) × (↑E↑＋1) } ×30
  ＋7760
```

A：作成した表数＋ 61

B：作成したインデクス数＋ 124

C：作成した表に定義した BLOB 列の総数＋ 3

D：作成した表に定義した BLOB 属性の総数

E：表を格納する RD エリアの分割数の平均値
分割していない場合は 1 とします。また，平均値は切り上げます。

a：SQL 又はコマンドが処理した表を格納している RD エリア数

b：SQL 又はコマンドが処理したインデクスを格納している RD エリア数

c：SQL 又はコマンドが処理した表を格納する LOB 用 RD エリア数

n：状態解析結果蓄積機能（pddbst -k logi -e）を実行した回数

m：運用履歴表を更新する SQL 又はコマンドの実行回数
運用履歴表を更新する SQL 及びコマンドについては，マニュアル「HiRDB システム運用ガイド」を参照してください。

16.1.12 更新可能バックアップ閉塞中に出力されるシステムログ量

更新可能バックアップ閉塞中にデータベースを更新すると，次に示す計算式の分だけシステムログが追加で出力されます。HiRDB/シングルサーバの場合は，ここで求めるシステムログ量をシングルサーバで出力するログ量に加算してください。HiRDB/パラレルサーバの場合は，更新対象の RD エリアを管理するバックエンドサーバ及びディクショナリサーバに対して，該当するログ量をそれぞれ加算してください。

ただし，バックアップ閉塞中に HiRDB が異常終了又は強制終了した場合，HiRDB の再開始時にバックアップ閉塞を引き継ぎません（参照可能バックアップ閉塞を除く）。このため，この計算式で示すシステムログは出力されません。

計算式

$$\sum_{i=1}^{a} (Si+200) \times Ti$$

a：更新可能バックアップ閉塞中に更新した RD エリア数

Si：RD エリアのページサイズ（バイト）

Ti：更新可能バックアップ閉塞中に更新した RD エリアの更新ページ数

RD エリアの更新ページ数は次に示す手順で求めます。

〈手順〉

1. 次に示すタイミングで統計解析ユティリティ（データベース操作に関する HiRDB ファイルの統計情報）を実行します。
   ・更新可能バックアップ閉塞の開始時
   ・更新可能バックアップ閉塞の解除時

2.「同期 WRITE 回数（SYNC-W）」を参照し，その差分で更新ページ数を求めます。

16.1.13　pdchpathn コマンド実行時に出力されるシステムログ量

HiRDB/シングルサーバの場合はシングルサーバで出力するログ量，HiRDB/パラレルサーバの場合はディクショナリサーバで出力するログ量に，このログ量を加算してください。

計算式

```
システムログ量（単位：バイト）
　　＝590×a＋844×b
```

a：SQL_PHYSICAL_FILES 表の行数

b：SQL_IOS_GENERATIONS 表の行数（インナレプリカ機能未使用時は 0）

16.1.14　ディクショナリ表のメンテナンス実行時に出力されるシステムログ量

HiRDB/シングルサーバの場合はシングルサーバで出力するログ量，HiRDB/パラレルサーバの場合はディクショナリサーバで出力するログ量に，このログ量を加算してください。

計算式

```
システムログ量（単位：バイト）
＝ 2×（8480＋(132＋page_len)×idx_page＋272×idx_page＋1940×↑idx_page÷seg_size↑）
```

変数の説明

idx_page：インデクスの格納ページ数（ページ）

「データディクショナリ用 RD エリアの容量の見積もり」の「インデクスの格納ページ数の計算方法」を参照してください。

page_len：データディクショナリ用 RD エリアのページ長（バイト）

seg_size：データディクショナリ用 RD エリアのセグメントサイズ（ページ数）

16.2　シンクポイントダンプファイルの容量の見積もり

16.2.1　シンクポイントダンプファイルの容量の計算方法

(1)　シンクポイントダンプファイルの容量の求め方

シンクポイントダンプファイルの容量は，次に示す計算式で求めます。

計算式

```
シンクポイントダンプファイルの容量（単位：バイト）
　　＝MAX（12，シンクポイントダンプファイルのレコード数）※1 ×4096※2
```

注※1

「シンクポイントダンプファイルのレコード数の求め方」を参照してください。ここで求めた値をpdloginit コマンドの-n オプションに指定します。求めた値が 12 未満の場合，pdloginit コマンドの-n オプションには 12 を指定してください。

注※2

シンクポイントダンプファイルのレコード長です。

(2)　シンクポイントダンプファイルのレコード数の求め方

条件		レコード数の計算式
HiRDB/シングルサーバの場合		↑（96 ＋ 112×a）÷4096↑＋ 3
HiRDB/パラレルサーバの場合	フロントエンドサーバの場合	4
	バックエンドサーバの場合	↑（96 ＋ 112×a）÷4096↑＋ 3
	ディクショナリサーバの場合	↑（96 ＋ 112×a）÷4096↑＋ 3

a ：pd_lck_until_disconnect_cnt オペランドの値

16.3　ステータスファイルの容量の見積もり

16.3.1　ステータスファイルの容量の計算方法

(1)　ステータスファイルの容量の求め方

ステータスファイルの容量は，次に示す計算式で求めます。

計算式

```
ステータスファイルの容量（単位：バイト）＝a×b
```

a：ステータスファイルのレコード数

「ステータスファイルのレコード数の求め方」を参照してください。ここで求めた値を pdstsinit コマンドの-c オプションに指定します。

b：ステータスファイルのレコード長

pdstsinit コマンドの-l オプションに指定する値です。

(2)　ステータスファイルのレコード数の求め方

ステータスファイルのレコード数は，次に示す計算式で求めます。

計算式

```
レコード数＝↑｛↑S÷（レコード長-40）↑＋↑S÷100↑＋S＋2｝×1.2↑
```

注　S については，「S の求め方」を参照してください。

(3)　S の求め方

(a)　HiRDB/シングルサーバの場合

種別	S の計算式
ユニット用	↑（2056 ＋ 128×d）÷g↑＋↑2512÷g↑＋↑40÷g↑ ＋↑308÷g↑＋↑4000÷g↑＋↑43536÷g↑＋↑32÷g↑
サーバ用	↑128÷g↑＋↑3280※1÷g↑＋↑（592※2＋j×b）÷g↑ ＋↑8192÷g↑＋↑（8192－128）÷g↑×｛↑（a＋m）÷k↑－1｝ ＋↑［↑｛(↑c÷127↑＋1）×2048｝÷8192↑×8192］÷g↑ ＋↑2048÷g↑＋↑｛244＋64×（2×h＋7)｝÷g↑ ＋↑96÷g↑＋↑｛48＋68×（2×h＋7)｝÷g↑ ＋↑［20480×｛1＋↑((152×e）－（16×f）－（80×A）＋（8×B)) ÷20476↑＋↑（4f＋4C）÷20480↑}］÷g↑

種別	S の計算式
	＋↑（12×c＋4）÷g↑ ＋w ＋↑（4＋16×c）÷g↑ ●ステータスファイルのレコード長＜4096 の場合に加算します。 ＋↓MAX｛3400÷r＋0.7，1｝↓×MAX（↓4096÷s↓，2）×n ＋↓5662310÷r＋0.7↓×MAX（↓4096÷s↓，2）×p ●4096≦ステータスファイルのレコード長＜12288 の場合に加算します。 ＋↓MAX｛(3400÷↓（s−348）÷20↓）＋0.7，1｝↓×n ＋↓（5662310÷↓（s−348）÷20↓）＋0.7↓×p ●12288≦ステータスファイルのレコード長の場合に加算します。 ＋↓MAX｛(3400÷↓（s−876）÷20↓）＋0.7，1｝↓×n ＋↓（5662310÷↓（s−876）÷20↓）＋0.7↓×p ●pd_log_auto_unload_path オペランドを指定する場合に加算します。 ＋↑20848÷g↑ ●高速系切り替え機能を使用する場合に加算します。 ＋↑（↑（256×i＋4096）÷4096↑）×4096÷g↑ ●pd_dbbuff_modify オペランドに Y を指定する場合に加算します。 ＋v ●HiRDB Staticizer Option がセットアップ済みで，かつ pd_max_reflect_process_count オペランドを指定する場合に加算します。 ＋↑（128＋704）÷g↑ ＋↑256÷g↑

a：pdlogadfg オペランドの指定数

b：pdlogadfg -d spd オペランドの指定数

c：pd_max_rdarea_no オペランドの値

d：108（シングルサーバの場合）又は 74（ユティリティ専用ユニットの場合）

e：pdbuffer オペランドの指定数

f：pdbuffer オペランドの-i オプション指定数

g：ステータスファイルのレコード長−40

h：pd_max_users の値＋ pd_max_reflect_process_count の値

i：インナレプリカ機能を使用する場合は pd_max_file_no オペランドの値
　インナレプリカ機能を使用しない場合は pd_max_rdarea_no オペランドの値

j：736

k：11

m：1

n：サーバ内の RD エリア数≦3400 の場合は 1
3401≦サーバ内の RD エリア数≦6800 の場合は 2
6801≦サーバ内の RD エリア数の場合は 3

p：サーバ内の RD エリア数≦10200 の場合は 0
10201≦サーバ内の RD エリア数≦5672510 の場合は 1
5672511≦サーバ内の RD エリア数≦11334820 の場合は 2
11334821≦サーバ内の RD エリア数の場合は 3

r：↓（s－348）÷20↓＋↓g÷20↓×（MAX（↓4096÷s↓，2）－1）

s：ステータスファイルのレコード長

v：32 ビットモードの場合：↑（24 ＋ 28×（x ＋ 512）＋ 32 ＋ 112×D）÷g↑
64 ビットモードの場合：↑（32 ＋ 32×（x ＋ 512）＋ 32 ＋ 144×D）÷g↑

w：32 ビットモードの場合：↑（(12 ＋（↑↑（c÷8）↑÷4↑）×4）×z）÷g↑
64 ビットモードの場合：↑（(12 ＋（↑↑（c÷8）↑÷8↑）×8）×z）÷g↑

x：pd_max_add_dbbuff_shm_no オペランドの値

y：pd_max_add_dbbuff_no オペランドの値

z：pdbuffer オペランドの-i オプション指定総数に，pd_max_add_dbbuff_no オペランドの値を加算した値（pd_dbbuff_modify オペランドに Y 以外を指定した場合は 0 として計算します）

A：システム共通定義の pdbuffer オペランドの-o オプション指定あり：1
システム共通定義の pdbuffer オペランドの-o オプション指定なし：0

B：システム共通定義の pdbuffer オペランドの-r オプション及び-b オプションに指定した RD エリアの総数

C：システム共通定義の pdbuffer オペランドの-i オプションに指定したインデクスを格納している RD エリアの総数

D：システム共通定義の pd_dbbuf_modify オペランドに Y を指定した場合：

- システム共通定義の pd_max_add_dbbuff_no オペランドを指定したとき：e ＋ y
- システム共通定義の pd_max_add_dbbuff_no オペランドを省略したとき：1000

システム共通定義の pd_dbbuf_modify オペランドに N を指定した場合：e

注※1　64 ビットモードの場合は 3456 です。

注※2　64 ビットモードの場合は 688 です。

(b) HiRDB/パラレルサーバの場合

種別	S の計算式
ユニット用	↑［2056＋128×｛14×（p−P＋q＋N）＋（p−P＋q＋r） ＋（25＋15×（p−P）＋7×q＋3×r）＋（s×2＋1）＋（N×16） ＋（38＋4×（p−P＋q＋r＋N）＋d＋z＋3×（p−P＋q＋r＋N））｝］÷j↑ ＋↑944÷j↑＋↑4816÷j↑＋↑40÷j↑ ＋↑308÷j↑＋↑4000÷j↑＋↑43536÷j↑ ＋↑（16＋8×M）÷j↑＋↑16÷j↑ ●ユニット内にシステムマネジャがある場合に加算します。 ＋↑（L＋16×M）÷j↑ ●影響分散スタンバイレス型系切り替え機能を使用しない場合に加算します。 ＋↑32÷j↑ ●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に加算します。 ＋↑［20480×｛1＋↑（(152×f)−（16×g）−（80×Q）＋（8×R））÷20476↑＋↑（4g＋4S）÷20480↑｝］÷j↑
フロントエンドサーバ用	↑128÷j↑＋↑3280※1÷j↑＋↑（592※2＋A×b）÷j↑ ＋↑8192÷j↑＋↑（8192−128）÷j↑×｛↑（a＋C）÷B↑−1｝ ＋↑8192÷j↑＋↑2048÷j↑ ＋↑｛244＋64×（2×e＋7)｝÷j↑＋↑96÷j↑＋↑32÷j↑ ●pd_log_auto_unload_path オペランドを指定する場合に加算します。 ＋↑20848÷j↑
ディクショナリサーバ用	↑128÷j↑＋↑3280※1÷j↑＋↑（592※2＋A×b）÷j↑ ＋↑8192÷j↑＋↑8192÷j↑×｛↑（a＋C）÷B↑−1｝ ＋↑［↑｛(↑c÷127↑＋1）×2048｝÷8192↑×8192］÷j↑ ＋↑2048÷j↑＋↑｛244＋64×（2×h＋7)｝÷j↑＋↑96÷j↑ ＋↑｛48＋68×（2×h＋7)｝÷j↑ ＋↑［20480×｛1＋↑（(152×f)−（16×g）−（80×Q）＋（8×R）） ÷20476↑＋↑（4g＋4S）÷20480↑｝］÷j↑ ＋↑（12×c＋4）÷j↑＋↑32÷j↑ ＋H ＋↑（4＋16×c）÷j↑ ●ステータスファイルのレコード長＜4096 の場合に加算します。 ＋↓MAX｛3400÷D＋0.7，1｝↓×MAX（↓4096÷n↓，2） ●4096≦ステータスファイルのレコード長＜12288 の場合に加算します。 ＋↓MAX｛(3400÷↓（n−348）÷20↓）＋0.7，1｝↓ ●12288≦ステータスファイルのレコード長の場合に加算します。 ＋↓MAX｛(3400÷↓（n−876）÷20↓）＋0.7，1｝↓ ●系切り替え機能を使用する場合に加算します。 ＋↑（↑（256×c＋4096）÷4096↑×4096）÷j↑ ●pd_log_auto_unload_path オペランドを指定する場合に加算します。 ＋↑20848÷j↑ ●pd_dbbuff_modify オペランドに Y を指定する場合に加算します。

種別	S の計算式
	＋G
バックエンド サーバ用	↑128÷j↑＋↑3280※1÷j↑＋↑（592※2＋A×b）÷j↑ ＋↑8192÷j↑＋↑8192÷j↑×｛↑（a＋C）÷B↑－1｝ ＋↑［↑｛(↑c÷127↑＋1）×2048｝÷8192↑×8192］÷j↑ ＋↑2048÷j↑＋↑｛244＋64×（2×i＋7)｝÷j↑＋↑96÷j↑ ＋↑｛48＋68×（2×i＋7)｝÷j↑＋↑（12×c＋4）÷j↑ ＋↑［20480×｛1＋↑((152×f)－(16×g)－(80×Q)＋(8×R)) ÷20476↑＋↑（4g＋4S）÷20480↑｝］÷j↑※3 ＋↑32÷j↑＋H＋↑（4＋16×c）÷j↑ ●ステータスファイルのレコード長＜4096 の場合に加算します。 ＋↓MAX｛3400÷D＋0.7，1｝↓×MAX（↓4096÷n↓，2）×k ＋↓5662310÷D＋0.7↓×MAX（↓4096÷n↓，2）×m ●4096≦ステータスファイルのレコード長＜12288 の場合に加算します。 ＋↓MAX｛(3400÷↓（n－348）÷20↓）＋0.7，1｝↓×k ＋↓（5662310÷↓（n－348）÷20↓）＋0.7↓×m ●12288≦ステータスファイルのレコード長の場合に加算します。 ＋↓MAX｛(3400÷↓（n－876）÷20↓）＋0.7，1｝↓×k ＋↓（5662310÷↓（n－876）÷20↓）＋0.7↓×m ●系切り替え機能を使用する場合に加算します。 ＋↑（↑（256×y＋4096）÷4096↑×4096）÷j↑ ●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に加算します。 ＋↑2816÷j↑＋↑80÷j↑ ＋↑32÷j↑ ●pd_log_auto_unload_path オペランドを指定する場合に加算します。 ＋↑20848÷j↑ ●pd_dbbuff_modify オペランドに Y を指定する場合に加算します。 ＋G ●HiRDB Staticizer Option がセットアップ済みで，かつ pd_max_reflect_process_count オペランドを指定する場合に加算します。 ＋↑256÷j↑ ＋↑（128＋704）÷j↑

a：pdlogadfg オペランドの指定数

b：pdlogadfg -d spd オペランドの指定数

c：pd_max_rdarea_no オペランドの値

d：3

ただし，1 サーバマシンでの HiRDB/パラレルサーバの場合は 2

e：（pd_max_users オペランドの値＋ pd_max_reflect_process_count オペランドの値）＋ 1
ただし，1 サーバマシンでの HiRDB/パラレルサーバの場合は（pd_max_users オペランドの値＋ pd_max_reflect_process_count オペランドの値）

f：pdbuffer オペランドの指定数

g：pdbuffer オペランドの-i オプション指定数

h：pd_max_dic_process オペランドの値＋ pd_max_reflect_process_count オペランドの値

i：pd_max_bes_process オペランドの値＋ pd_max_reflect_process_count オペランドの値

j：ステータスファイルのレコード長－40

k：サーバ内の RD エリア数≦3400 の場合は 1
3401≦サーバ内の RD エリア数≦6800 の場合は 2
6801≦サーバ内の RD エリア数の場合は 3

m：サーバ内の RD エリア数≦10200 の場合は 0
10201≦サーバ内の RD エリア数≦5672510 の場合は 1
5672511≦サーバ内の RD エリア数≦11334820 の場合は 2
11334821≦サーバ内の RD エリア数の場合は 3

n：ステータスファイルのレコード長

p：ユニット内のバックエンドサーバ数

q：ユニット内にディクショナリサーバがある場合は 1，ない場合は 0

r：ユニット内にフロントエンドサーバがある場合は 1，ない場合は 0

s：ユニット内にシステムマネジャがある場合は 1，ない場合は 0

y：インナレプリカ機能を使用する場合は pd_max_file_no オペランドの値
インナレプリカ機能を使用しない場合は pd_max_rdarea_no オペランドの値

z：次の場合は 1

- ユニット内にシステムマネジャがある場合
- ユニット内にシステムマネジャがない場合で，pd_mlg_msg_log_unit オペランドに local を指定しているとき

ユニット内にシステムマネジャがない場合で，pd_mlg_msg_log_unit オペランドに manager を指定しているか，又は指定を省略しているときは 0

A：736

B：11

C：1

D：↓（n－348）÷20↓＋↓j÷20↓×（MAX（↓4096÷n↓，2）－1）

G：32 ビットモードの場合：↑（24 ＋ 28×（I ＋ 512）＋ 32 ＋ 112×T）÷j↑
　64 ビットモードの場合：↑（32 ＋ 32×（I ＋ 512）＋ 32 ＋ 144×T）÷j↑

H：32 ビットモードの場合：↑（(12 ＋（↑↑（c÷8）↑÷4↑）×4）×K）÷j↑
　64 ビットモードの場合：↑（(12 ＋（↑↑（c÷8）↑÷8↑）×8）×K）÷j↑

I：pd_max_add_dbbuff_shm_no オペランドの値

J：pd_max_add_dbbuff_no オペランドの値

K：pdbuffer オペランドの-i オプション指定総数に，pd_max_add_dbbuff_no オペランドの値を加算した値（pd_dbbuff_modify オペランドに Y 以外を指定した場合は 0 として計算します）

L：pd_system_expand_unit オペランドを指定した場合：16
　pd_system_expand_unit オペランドを省略した場合：0

M：pd_system_expand_unit オペランドを指定した場合：pd_system_expand_unit オペランドに指定したユニット数
　pd_system_expand_unit オペランドを省略した場合：0

N：pd_ha_max_act_guest_servers オペランドを指定した場合：pd_ha_max_act_guest_servers オペランドの指定値
　pd_ha_max_act_guest_servers オペランドを省略した場合：0

P：ユニット内の代替 BES の数（1：1 スタンバイレス型系切り替え機能使用時の代替 BES）

Q：システム共通定義の pdbuffer オペランドの-o オプション指定あり：1
　システム共通定義の pdbuffer オペランドの-o オプション指定なし：0

R：システム共通定義の pdbuffer オペランドの-r オプション及び-b オプションに指定した RD エリアの総数

S：システム共通定義の pdbuffer オペランドの-i オプションに指定したインデクスを格納している RD エリアの総数

T：システム共通定義の pd_buf_modify オペランドに Y を指定した場合
- システム共通定義の pd_max_add_dbbuff_no オペランドを指定したとき：f ＋ J
- システム共通定義の pd_max_add_dbbuff_no オペランドを省略したとき：1000

　システム共通定義の pd_buf_modify オペランドに N を指定した場合：f

注※1　64 ビットモードの場合は 3456 です。

注※2　64 ビットモードの場合は 688 です。

注※3　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，加算しないでください。

16.4　監査証跡ファイルの容量の見積もり

監査証跡ファイル用の HiRDB ファイルシステムの容量は，次に示す計算式で求めます。

計算式

監査証跡ファイル用のHiRDBファイルシステムの容量（単位：メガバイト）＝ａ＋19

ａ：最も多い場合の監査証跡のデータ量（単位：メガバイト）

↑（Σ｛ｂ×ｃ｝）÷（1024×1024）↑

Σ は監査証跡を出力する監査対象イベントごとの合計を意味します。なお，監査証跡ファイルの入出力エラーが発生した場合や，pdaudswap コマンドで監査証跡ファイルのスワップを実行した場合は，監査証跡ファイルの容量が一杯になる前にスワップします。この場合，監査証跡表へのデータ登録が終了するまで，その空き容量は使用できません。そのため，あらかじめａの値を 2 倍にしておくことを推奨します。

ｂ：監査証跡のレコードサイズ（単位：バイト）

監査証跡のレコードサイズは，次に示す計算式で求めます。

464＋ｄ＋ｅ＋ｆ＋ｇ＋ｈ＋ｉ

ｃ：監査証跡のレコード数

監査証跡イベントの種類ごとの記録レコード数です。

ｄ：監査証跡のレコードに出力する SQL 文の長さ（単位：バイト）

pd_aud_sql_source_size の値が 0，又は指定していない場合：0

レコード項目に SQL 文がない場合：0

レコード項目に SQL 文があるが，SQL 文が NULL 値の場合：0

上記以外の場合：

↑Min（pd_aud_sql_source_size の値，SQL 文の平均長）÷4↑×4＋8

ｅ：監査証跡のレコードに出力する SQL データの長さ（単位：バイト）

pd_aud_sql_data_size の値が 0，又は指定していない場合：0

レコード項目に SQL データがない場合：0

レコード項目に SQL データがあるが，SQL データが NULL 値の場合：0

上記以外の場合：

↑Min（pd_aud_sql_data_size の値，SQL データの平均長）÷4↑×4＋8

ｆ，ｇ，ｈ：監査証跡のレコードに出力するユーザ付加情報 1，2，3 の長さ（単位：バイト）

レコード項目にユーザ付加情報がない場合，又はユーザ付加情報が NULL 値の場合：0

上記以外の場合：

↑ユーザ付加情報の平均長÷4↑×4＋8

ｉ：監査証跡のレコードに出力する関連製品付加情報の長さ（単位：バイト）

レコード項目に関連製品付加情報がない場合，又は関連製品付加情報が NULL 値の場合※1：0

上記以外の場合：

↑関連製品付加情報の平均長※2÷4↑×4＋8

注※1

関連製品付加情報が NULL 値になる場合の条件については，関連製品のマニュアルを参照してください。

注※2

関連製品付加情報の平均長については，関連製品のマニュアルを参照してください。マニュアルに該当する記載がない場合は，関連製品付加情報の最大長で計算してください。

監査証跡のレコード項目，及び監査証跡ファイルに出力される情報については，マニュアル「HiRDB システム運用ガイド」を参照してください。

17 作業表用ファイルの容量の見積もり

この章では，作業表用ファイルの容量の見積もり方法について説明します。

17.1　作業表用ファイルの概要

ここでは，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**作業表用ファイル**について説明します。

17.1.1　作業表用ファイルの作成契機

作業表用ファイルとは，次に示す操作をするときに発生する一時的な情報を格納するファイルです。

- 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
- ディクショナリ表のメンテナンス

注※

作業表用ファイルは，SELECT 文で複数の表を結合して検索する場合や CREATE INDEX 実行時など，特定の SQL 実行時に使用されます。作業表用ファイルを必要とする SQL を次に示します。

ただし，次に該当しない SQL でも，アクセスパスによっては作業表用ファイルを必要とする場合があります。作業表用ファイルの要否については，アクセスパスの Work Table の項目を確認してください。また，SQL に LIMIT 句を指定しアクセスパス中に MEM(SUBSORT)の表示がある場合も，作業表用ファイルを必要とすることがあります。アクセスパスの出力形式の詳細については，マニュアル「HiRDB コマンドリファレンス」の「アクセスパス表示ユティリティ（pdvwopt)」の「出力形式」を参照してください。

1. UNION [ALL]句又は EXCEPT [ALL]句を指定して検索する場合
2. DROP SCHEMA
3. DROP TABLE
4. DROP INDEX
5. REVOKE でアクセス権限を取り消す場合
6. CREATE INDEX
7. ASSIGN LIST 文で実表からリストを作成する場合
8. SELECT 文で次に示す指定をする場合
    - 複数の表を結合して検索するとき
    - インデクスを定義していない列に対して ORDER BY 句を指定するとき
    - 横分割表に対して ORDER BY 句を指定するとき

- 選択式中に集合関数を含む値式を指定しているとき（HiRDB/パラレルサーバの場合にだけ該当します）
- 選択式中にウィンドウ関数 COUNT(*) OVER()を含む値式を指定しているとき
- GROUP BY 句を指定するとき
- DISTINCT 句を指定するとき
- インデクスを定義した複数の列に検索条件を指定するとき
- 繰返し列インデクスを定義した列に検索条件を指定するとき
- SQL 最適化オプションに「プラグイン提供関数からの一括取得機能」を指定し，探索条件にプラグインインデクスを使用するプラグイン提供関数を指定して検索するとき
- FOR UPDATE 句を指定するか又はこのカーソルを使用した更新があり，インデクスを定義した列に検索条件を指定するとき
- FOR READ ONLY 句を指定するとき
- 限定述語の副問合せを指定するとき
- IN 述語の副問合せを指定するとき
- ビュー表の検索，又は WITH 句を指定した検索で内部導出表を作成するとき

9. INSERT 文の挿入元の問合せ本体に，次に示す指定をする場合
   - 外への参照がある副問合せに更新表を指定するとき
   - 挿入元の問合せ式本体の主問合せに更新表を指定するとき
10. UPDATE 文に次に示す指定をする場合
    - 探索条件中又は更新値中に，外への参照がある副問合せを指定し，その副問合せ中に更新表を指定するとき
    - 探索条件中に限定述語の副問合せを指定するとき
    - 探索条件中に IN 述語の副問合せを指定するとき
    - インデクスを定義した列を更新対象及び探索条件に指定し，かつそのインデクスを使用するとき
11. DELETE 文に次に示す指定をする場合
    - 探索条件中の外への参照がある副問合せに，更新表を指定するとき
    - 探索条件中に限定述語の副問合せを指定するとき
    - 探索条件中に IN 述語の副問合せを指定するとき
    - インデクスを定義した列を探索条件に指定し，かつそのインデクスを使用するとき
12. ALTER TABLE ADD PRIMARY KEY
13. ALTER TABLE DROP PRIMARY KEY

17.1.2　作業表用ファイルの格納先

作業表用ファイルは，HiRDB が HiRDB ファイルシステム領域に作成します。HiRDB 管理者は次に示すことをしてください。

- pdfmkfs コマンドで，作業表用ファイルを作成する HiRDB ファイルシステム領域を初期設定します。
- HiRDB システム定義の pdwork オペランドで，上記の HiRDB ファイルシステム領域の名称を指定します。

以降，pdfmkfs コマンドのオプションに指定する値の見積もり方法について説明します。pdfmkfs コマンドのオプションと作業表用ファイルの内容の関係を次の表に示します。

表 17–1　pdfmkfs コマンドのオプションと作業表用ファイルの内容の関係

オプション	内　容
-n	作業表用ファイルを作成する HiRDB ファイルシステム領域のサイズ
-l	HiRDB ファイルシステム領域に作成する HiRDB ファイル（作業表用ファイル）の最大数
-e	HiRDB ファイルシステム領域の最大増分回数
-a	HiRDB ファイルシステム領域を自動的に拡張するかどうか

17.2 HiRDB ファイルシステム領域サイズの見積もり（pdfmkfs -n コマンド）

作業表用ファイルを作成する HiRDB ファイルシステム領域のサイズは，pdfmkfs コマンドの-n オプションで設定します。

HiRDB ファイルシステム領域のサイズは，次に示す計算式で求めます。

計算式

```
HiRDBファイルシステム領域のサイズ（単位：バイト）＝A＋B
```

A：

SQL 文が使用する作業表用ファイルの容量です。求め方については，「SQL 文が使用する作業表用ファイルの容量」を参照してください。

B：

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）で C 以外の場合，及びリバランスユティリティ（pdrbal）が使用する作業表用ファイルの容量です。求め方については，「ユティリティが使用する作業表用ファイルの容量」を参照してください。

C：

データベース再編成ユティリティ（pdrorg）でディクショナリ表のメンテナンスを実行する場合に，データベース再編成ユティリティが使用する作業表用ファイルの容量です。求め方については，「ディクショナリ表のメンテナンスが使用する作業表用ファイルの容量」を参照してください。

なお，作業表用ファイルを使用する SQL 文と作業表用ファイルを使用するユティリティの操作を同時に実行しない場合は，A 又は B のどちらか大きい値を HiRDB ファイルシステム領域のサイズとしてください。

注意事項

ここで求めた HiRDB ファイルシステム領域のサイズが大き過ぎて，一つの HiRDB ファイルシステム領域に収まらない場合は，複数の HiRDB ファイルシステム領域を pdfmkfs コマンドで初期設定し，HiRDB システム定義の pdwork オペランドに指定してください。このときに注意することを次に示します。

- 各 HiRDB ファイルシステム領域のサイズを同じにしてください。
- 一つの HiRDB ファイルシステム領域のサイズは，一つの作業表（列情報格納用）の容量より大きくしてください。
- HiRDB ファイルシステム領域を分割し過ぎると，未使用領域が複数の HiRDB ファイルシステム領域に分割されます。したがって，領域すべてが有効活用されないため，容量が不足することがあります。
- 1 作業表用ファイルの容量が 2 ギガバイトを超える場合は，ラージファイルを使用してください。1 作業表用ファイルは複数の HiRDB ファイルシステム領域に分割できません。

17.2.1　SQL 文が使用する作業表用ファイルの容量

SQL 文が使用する作業表用ファイルの容量は次に示す計算式で求めます。

計算式

```
SQL文が使用する作業表用ファイルの容量（単位：バイト）＝MAX（ａ，ｂ）×ｃ
```

a：1SQL 文が使用する作業表用ファイルの容量の最大値

SQL 文ごとに作業表用ファイルの容量を計算します。その中で最も大きい値を a に代入します。「1SQL 文が使用する作業表用ファイルの容量の求め方」を参照して求めてください。

b：ASSIGN LIST 文が使用する作業表用ファイルの容量の最大値

ASSIGN LIST 文ごとに作業表用ファイルの容量を計算します。その中で最も大きい値を b に代入します。「ASSIGN LIST 文が使用する作業表用ファイルの容量の求め方」を参照して求めてください。

c：pd_max_users オペランドの値＋ pd_max_reflect_process_count オペランドの値

ただし，マルチフロントエンドサーバを使用している場合のバックエンドサーバでは，（pd_max_bes_process オペランドの値＋ pd_max_reflect_process_count オペランドの値）になります。

(1)　1SQL 文が使用する作業表用ファイルの容量の求め方

1SQL 文が使用する作業表用ファイルの容量は，次に示す計算式で求めます。

計算式

```
1SQL文が使用する作業表用ファイルの容量（単位：バイト）＝ａ×ｂ＋ｃ×ｄ
```

a：列情報作業表の容量

b：列情報作業表の最大数

c：位置情報作業表の容量

d：位置情報作業表の最大数

(a)　列情報作業表の容量の求め方

列情報作業表の容量は次に示す計算式で求めます。

計算式

```
列情報作業表の容量（単位：バイト）
                    ＝↑a÷MIN｛↓（b-48）÷c↓，255｝↑×b×2
```

a：列情報作業表の行数（表「列情報作業表の行数の求め方」から求めてください）

b：作業表ページ長（**計算式 1** から求めてください）

c：作業表の行長（**計算式 2** から求めてください）

計算式 1

- pdwork に指定した HiRDB ファイルシステム領域のセクタ長が 4096 の場合※1

```
作業表ページ長※2＝↑（作業表の行長＋48）÷4096↑×4096
```

- pdwork に指定した HiRDB ファイルシステム領域のセクタ長が 4096 ではない場合

```
作業表ページ長※2＝MAX｛↑（作業表の行長＋48）÷2048↑×2048 , 4096 ｝
```

注※1　複数指定した場合，1 つでもセクタ長が 4096 の領域がある場合は，「セクタ長が 4096 の場合」の計算式を使用してください。

注※2　作業表ページ長は，32,768 バイト以下でなければなりません。

計算式 2

$$作業表の行長※ = \sum_{i=1}^{n} Ai + 2 \times n + 6$$

Ai：作業表の各列のデータ長（表「作業表の各列のデータ長及び列数の求め方」から求めてください）

n：作業表の列数（表「作業表の各列のデータ長及び列数の求め方」から求めてください）

注※　作業表の行長は，32,720 バイト以下でなければなりません。

なお，LIMIT 句指定時，（オフセット行数＋ LIMIT 句に指定した行数）の値が 32,768 以上になる場合は，計算式 2 で算出した作業表の行長に 12 を加算してください。ただし，次の場合には 12 を加算する必要はありません。

- 検索対象の表に EX モードで排他が掛かっている場合
- WITHOUT LOCK を指定した場合
- グループ分け高速化機能を指定した場合
- 複数の表を結合する場合

表 17-2　列情報作業表の行数の求め方

SQL 文	列情報作業表の行数
SELECT 文	検索する個々の表から探索する行数の合計です。ただし，複数の表を結合していて結合結果行数の方が多い場合は，その行数となります。
• CREATE INDEX • ALTER TABLE ADD PRIMARY KEY	表に格納している行数です。ただし，繰返し列に対するインデクスの場合は，インデクス構成列のうち一つの繰返し列の要素の総数となります。

表 17-3　作業表の各列のデータ長及び列数の求め方

SQL 文	n	A_i
SELECT 文	選択式に指定した列数＋ GROUP BY 句に指定した列数※＋ ORDER BY 句に指定した列数※＋	左記の列ごとの実際の長さ ただし，長大データ（BLOB），定義長が 256 バイト以上の文字データ（各国・混在文字デー

SQL 文	n	A_i
	HAVING 句に指定した列数※＋ FOR UPDATE 句を指定した場合※は 1 ただし，選択式に ROW を指定している場合は表の全列数	タも含む），バイナリデータのうち，下記に属さない列又は位置情報列の場合は 12 • 結合条件中に指定する列（結合列） • DISTINCT 句指定の選択式 • 限定述語の副問合せ選択式中に指定する列 • IN 述語の副問合せ選択式中に指定する列 • UNION [ALL] 又は EXCEPT [ALL] によって，集合演算対象となっている問合せ指定中の選択式 • ORDER BY 句に指定した列
• CREATE INDEX • ALTER TABLE ADD PRIMARY KEY	1（インデクス情報列）＋ 1（位置情報列）	• インデクス情報列はインデクス構成列のデータ長の合計値 • 位置情報列は 12

注　各列のデータ長については，次に示す表を参照してください。

- 表「データ長一覧」
- 表「可変長文字列型のデータ長一覧（抽象データ型及び繰返し列を除く）」
- 表「可変長文字列型のデータ長一覧（抽象データ型の場合）」
- 表「可変長文字列型のデータ長一覧（繰返し列の場合）」

注※

選択式に指定した列と同一の列の場合は，新たに加算する必要はありません。

(b)　列情報作業表の最大数の求め方

列情報作業表の最大数の求め方を次の表に示します。

表 17–4　列情報作業表の最大数の求め方

SQL 文	列情報作業表の最大数※1
SELECT 文	1. ～10. の指定がない場合は，0 になります。 1. ～10. の指定がある場合は，対応する作業表増加数をすべて加算した値になります。 1. 複数の表を結合して検索する場合 作業表増加数（HiRDB/シングルサーバの場合）＝結合表数＋ 1 作業表増加数（HiRDB/パラレルサーバの場合）＝結合表数×2 ただし，結合キーとなる列にインデクスがあり，制限条件がある場合は 0 2. ORDER BY 句を指定する場合 作業表増加数＝ 2 ORDER BY 句に指定した列をすべて含むインデクスを検索に使用する場合＝ 0 3. GROUP BY 句を指定しないで，選択式中に集合関数を含む値式を指定する場合※2 作業表増加数＝ 1 4. GROUP BY 句を指定する場合

SQL 文	列情報作業表の最大数※1
	作業表増加数＝ GROUP BY 句指定数×2 5. DISTINCT 句を指定する場合 作業表増加数＝ DISTINCT 句指定数×2 6. UNION［ALL］句又は EXCEPT［ALL］句を指定する場合 作業表増加数（HiRDB/シングルサーバの場合）＝ UNION［ALL］句又は EXCEPT［ALL］句指定数＋ 2 作業表増加数（HiRDB/パラレルサーバの場合）＝（UNION［ALL］句又は EXCEPT［ALL］句指定数＋ 1）×2 7. FOR UPDATE 句を指定するか又はこのカーソルを使用した更新があり，インデクスを定義した列に検索条件を指定する場合※2 作業表増加数＝ 2 8. FOR READ ONLY 句を指定する場合 作業表増加数＝ 1 9. 副問合せ（限定述語）を指定する場合 増加作業表数＝副問合せ指定数＋（インデクスを定義した列に対する＝ ANY の限定述語の数）＋（インデクスを定義した列に対する IN 述語の副問合せ指定数）＋（インデクスを定義した列に対する＝ SOME の限定述語の数） 10. 選択式にウィンドウ関数 COUNT(*) OVER()を指定する場合 増加作業表数＝選択式にウィンドウ関数を指定した問合せ指定数
• CREATE INDEX • ALTER TABLE ADD PRIMARY KEY	2

注※1　HiRDB が見積もるアクセスコストによっては作業表を作成しない場合があります。

注※2　HiRDB/パラレルサーバの場合にだけ該当します。

(c)　位置情報作業表の容量の求め方

位置情報作業表の容量は次に示す計算式で求めます。

計算式

位置情報作業表の容量（単位：バイト）＝↑a÷184※↑×4096×2

注※　探索条件にインデクス型プラグイン専用関数を指定する場合は 155

a：位置情報作業表の行数

位置情報作業表の行数の求め方を次に示します。

SQL 文	位置情報作業表の行数の求め方
• SELECT 文 • UPDATE 文 • DELETE 文	探索時の条件中にインデクスを定義した列を含む述語が一つの場合，述語が真となる行数になります。二つ以上の場合，次の行数の合計値になります。 • 述語を OR 論理演算している場合，少なくとも一つの述語が真となる行数の合計

SQL 文	位置情報作業表の行数の求め方
	• 述語を AND 論理演算している場合，述語のうち真となる行数が多い方の行数として合算した行数

(d)　位置情報作業表の最大数の求め方

位置情報作業表の最大数の求め方を次の表に示します。

表 17-5　位置情報作業表の最大数の求め方

SQL 文	位置情報作業表の最大数
SELECT 文	1. インデクスを定義した複数の列に検索条件を指定する場合 2. FOR UPDATE 句を指定するか又はこのカーソルを使用した更新があり，インデクスを定義した列に検索条件を指定する場合※ 3. 繰返し列インデクスを定義した列に検索条件を指定する場合 4. SQL 最適化オプションに「プラグイン提供関数からの一括取得機能」を指定し，探索条件にプラグインインデクスを使用するプラグイン提供関数を指定して検索する場合 1～4 で，探索時に使用するインデクス数＋ 1 になります。
• UPDATE 文 • DELETE 文	探索条件中にインデクスを定義した列がある場合，探索時に使用するインデクス数＋ 1 になります。

注※　HiRDB/シングルサーバの場合にだけ該当します。

(2)　ASSIGN LIST 文が使用する作業表用ファイルの容量の求め方

ASSIGN LIST 文が使用する作業表用ファイルの容量は，次に示す計算式で求めます。

計算式

```
ASSIGN LIST文が使用する作業表用ファイルの容量（単位：バイト）=
 n
Σ (Bi×2)
i=1
```

n：ASSIGN LIST 文の探索条件中の述語の数

Bi：探索条件中の i 番目の述語を処理する作業表の容量。次に示す計算式で求めます。

B_i ＝↑リストの基になる表で i 番目の述語が真となる行数※÷504↑×4096

×1.5（単位：バイト）

注※　述語が繰返し列に対する条件の場合，真となる要素の総数となります。

17.2.2　ユティリティが使用する作業表用ファイルの容量

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，又はリバランスユティリティでデータの再配置をする場合は，次に示す計算式の分の作業表用ファイルが必要になります。

計算式

```
ユティリティが使用する作業表用ファイルの容量（単位：バイト）＝｛（A＋B）×2×D ｝÷C
```

A：インデクス作成時に必要な作業表の行数 1

B：インデクス作成時に必要な作業表の行数 2

C：作業表ページ内の行数

D：作業表ページ長

注

- 1 回のユティリティで複数のインデクスの一括作成又は再作成をする場合は，インデクスのキー長が最も長いインデクスについて求めてください。
- インデクスの一括作成又は再作成を同時に実行する場合は，それぞれで必要な作業表用ファイルの容量を求めて加算してください。
- 複数のユティリティを同時実行する場合は，各ユティリティが使用する作業表用ファイルの容量を計算して合計してください。

(1)　インデクス作成時に必要な作業表の行数 1 の求め方

インデクス作成時に必要な作業表の行数 1 は，次に示す計算式で求めます。

計算式

```
作業表の行数1＝↑c÷｛↓↑a×（100-b）×0.01↑÷（d＋22）↓｝↑
```

a：インデクスを格納するユーザ用 RD エリアのページサイズ

b：CREATE INDEX の PCTFREE オペランドで指定する未使用領域の比率

c：データ件数

繰返し列に対するインデクスの場合は，インデクス構成列のうち一つの繰返し列の各行の要素数の合計

d：インデクスのキー長

インデクスのキー長については表「インデクスのキー長一覧」を参照してください。なお，データベースに格納されるキー長は 4 バイトバウンダリされるため，**↑キー長÷4↑×4** となります。

複数インデクスのキー長は，表「インデクスのキー長一覧」を基に全構成列のキー長を加算してください。

(2)　インデクス作成時に必要な作業表の行数 2 の求め方

インデクス作成時に必要な作業表の行数 2 は，次に示す計算式で求めます。

計算式

```
作業表の行数2＝↑c÷｛↓↑a×（100-b）×0.01↑÷（d＋14）↓｝↑
```

a：インデクスを格納するユーザ用 RD エリアのページサイズ

b：CREATE INDEX の PCTFREE オペランドで指定する未使用領域の比率

c：インデクス作成時に必要な作業表の行数 1

（1）で求めた値を代入してください。

d：インデクスのキー長

インデクスのキー長については表「インデクスのキー長一覧」を参照してください。なお，データベースに格納されるキー長は 4 バイトバウンダリされるため，**↑キー長÷4↑×4** となります。

複数インデクスのキー長は，表「インデクスのキー長一覧」を基に全構成列のキー長を加算してください。

(3)　作業表ページ内の行数の求め方

作業表ページ内の行数は，次に示す計算式で求めます。

計算式

```
作業表ページ内の行数＝MIN｛↓（b-48）÷a↓，255｝
```

a：作業表の行長（インデクスのキー長＋ 18）

インデクスのキー長については表「インデクスのキー長一覧」を参照してください。キー長は**↑キー長÷4↑×4** となります。

複数インデクスのキー長は，表「インデクスのキー長一覧」を基に全構成列のキー長を加算してください。

b：作業表ページ長

（4）で求めてください。

(4)　作業表ページ長の求め方

作業表ページ長は，次に示す計算式で求めます。

計算式

```
作業表ページ長※＝MAX｛↑（作業表の行長＋48）÷2048↑×2048，4096｝
```

注※　作業表ページ長は，32,768 バイト以下でなければなりません。

a：作業表の行長（インデクスのキー長＋ 18）

インデクスのキー長については表「インデクスのキー長一覧」を参照してください。キー長は↑キー長÷4↑×4 となります。

複数インデクスのキー長は，表「インデクスのキー長一覧」を基に全構成列のキー長を加算してください。

17.2.3　ディクショナリ表のメンテナンスが使用する作業表用ファイルの容量

データベース再編成ユティリティ（pdrorg）でディクショナリ表のメンテナンスを行う場合は，次に示す計算式の分の作業表用ファイルが必要になります。

計算式

ディクショナリ表のメンテナンスが使用する作業表用ファイルの容量（単位：バイト）
＝↑row_num÷134↑×16384

変数の説明

row_num：SQL_RDAREAS 表の行数

17.3　最大ファイル数の見積もり（pdfmkfs -l コマンド）

17.3.1　最大ファイル数の計算方法

HiRDB ファイルシステム領域に作成する作業表用ファイルの最大ファイル数は，pdfmkfs コマンドの-l オプションで設定します。

HiRDB ファイルシステム領域に作成する作業表用ファイルの最大ファイル数は，次に示す計算式で求めます。

計算式

```
最大ファイル数 ＝ MAX（ａ，ｂ）×ｃ＋20＋2※
```

ａ：1SQL 文が使用する作業表用ファイルの数

SQL 文ごとに作業表用ファイルの数を計算します。その中で最も大きい値をａに代入します。「1SQL 文が使用する作業表用ファイルの数の求め方」を参照して求めてください。

ｂ：ASSIGN LIST 文が使用する作業表用ファイルの数

ASSIGN LIST 文ごとに作業表用ファイルの数を計算します。その中で最も大きい値をｂに代入します。「ASSIGN LIST 文が使用する作業表用ファイルの数の求め方」を参照して求めてください。

ｃ：pd_max_users オペランドの値＋ pd_max_reflect_process_count オペランドの値

ただし，マルチフロントエンドサーバを使用している場合のバックエンドサーバでは，（pd_max_bes_process オペランドの値＋ pd_max_reflect_process_count オペランドの値）になります。

注※

作業表用ファイルを使用する SQL 文と作業表用ファイルを使用するユティリティ（データベース作成ユティリティ又はデータベース再編成ユティリティ）を同時に実行する場合に加算します。

(1)　1SQL 文が使用する作業表用ファイルの数の求め方

1SQL 文が使用する作業表用ファイルの数は，次に示す計算式で求めます。

計算式

```
1SQL文が使用する作業表用ファイルの数＝列情報作業表の最大数＋位置情報作業表の最大数
```

列情報作業表の最大数，及び位置情報作業表の最大数については，「SQL 文が使用する作業表用ファイルの容量」を参照してください。

(2)　ASSIGN LIST 文が使用する作業表用ファイルの数の求め方

ASSIGN LIST 文が使用する作業表用ファイルの数は，次に示す計算式で求めます。

計算式

ASSIGN LIST文が使用する作業表用ファイルの数＝ASSIGN LIST文の探索条件中の述語の数×2

(3) 注意

作業表用ファイルを作成する HiRDB ファイルシステム領域を複数設定する場合は，次に示すことに注意してください。

- ここで求めた値が 4096 より大きい場合は，-l オプションには 4096 を指定してください。

17.4　最大増分回数の見積もり（pdfmkfs -e コマンド）

HiRDB ファイルシステム領域に作成する作業表用ファイルの最大増分回数は，pdfmkfs コマンドの-e オプションで設定します。

HiRDB ファイルシステム領域の最大増分回数は，次に示す計算式で求めます。

計算式

```
最大増分回数＝MIN（最大ファイル数×23，60000）
```

注 1

最大増分回数が見積もり値よりも少ない場合，HiRDB ファイルシステム領域に空きがあっても領域確保ができなくなることがあります。

注 2

最大ファイル数については，「最大ファイル数の見積もり（pdfmkfs -l コマンド）」の説明を基に算出した値を代入してください。

18 ユティリティ実行時の容量の見積もり

この章では，ユティリティ実行時のファイル容量及びメモリ所要量の見積もり方法について説明します。

18.1　ユティリティ実行時のファイルの容量の見積もり

18.1.1　データベース作成ユティリティ（pdload）実行時のファイルの容量

データベース作成ユティリティ（pdload）で使用するファイルの容量の計算式を次に示します。

ファイルの種類	容量の計算式（単位：バイト）
入力データファイル	h×b
インデクス情報ファイル	**B-tree インデクスの場合** （d ＋ y）×（b ＋ e）＋ 512 **プラグインインデクスの場合** （12 ＋ q）×p ＋ 1024 この計算式は，1 インデクス当たりの容量計算式です。インデクスが複数ある場合は，それぞれのインデクスに対して計算してください。
エラー情報ファイル	k×f ＋ s×200 ＋ m
エラー情報ファイル作成用一時ファイル	次に示す条件の場合は，表格納 RD エリアがあるサーバごとに，**キー重複エラー数×8 ＋プラグイン関数が検知したエラー件数×200** の容量がワークファイル出力先ディレクトリに必要です。ワークファイル出力先ディレクトリについては，「ワークファイル出力先ディレクトリの作成」を参照してください。 • HiRDB/シングルサーバでは，ユティリティ専用ユニットに入力ファイルがある場合 • HiRDB/パラレルサーバでは，入力ファイルがあるサーバと表格納 RD エリアがあるサーバが異なる場合
LOB 入力ファイル	**LOB 入力ファイルを EasyMT 上の複数ファイルとして作成する場合** a Σ（LOB データ長 i ＋ 400） i=1 **列単位 LOB 入力ファイルの場合** b Σ（LOB データ長＋ 4）i i=1
LOB 中間ファイル	b Σ ｛ i=1 c Σ（LOB ファイル名称長 ij ＋ 36）＋ 24 ｝ j=1 ＋ 1024 ＋ c×84
エラーデータファイル	MIN（f，g）×h
処理結果ファイル	1500 ＋表格納サーバ数×500

ファイルの種類	容量の計算式（単位：バイト）
ワークファイル※	[4 ＋ 2×R ＋ 2×r ＋ 4×I×R ＋ {b÷（-m オプションに指定した経過メッセージ出力間隔の値)}］×200
ソート用ワークファイル	**条件 1 の場合** インデクス情報ファイルの容量＋ 4×（b ＋ e） **条件 2 の場合** {インデクス情報ファイルの容量＋ 4×（b ＋ e)}×2 • 条件 1 1024 キロバイト≧E のとき • 条件 2 1024 キロバイト＜ E のとき E：バッファサイズ 「ソート用ワークファイル容量の計算で使用するバッファサイズ」で求めたバッファサイズ

a：入力行数×LOB 列数

b：入力行数（繰返し列の場合は入力行数×要素数）

c：LOB 列数

d：インデクスのキー長

表「インデクスのキー長一覧」を参照してください。ただし，可変長データの場合は単一列でも複数列として扱い，定義長の最大値で計算してください。

e：既存の行数（繰返し列の場合は既存行数×要素数）

f：エラーデータ件数

g：source 文の errdata オペランドで指定する出力行数

h：平均ソースレコード長

k：抽象データ型の列がある場合は 300

ない場合は 120

m：DAT 形式，又は pdrorg で出力したバイナリ形式のファイルの場合は 0

そのほかの場合は（入力ファイルの 1 行のレコード長×4）

p：インデクス格納 RD エリアを初期化した場合は（b ＋ e）

そのほかの場合は b

q：次に示す値

- LOB 用 RD エリアに格納された抽象データ型の場合は 27
- 定義長 255 バイト以下の抽象データ型の場合は（キー長＋ 2）
- 定義長 256 バイト以上の抽象データ型の場合は 2

代表的な抽象データ型の値を次に示します。

- SGMLTEXT 型の場合は 27
- FREEWORD，GEOMETRY，及び XML 型の場合は 2

r：LOB 格納 RD エリア数

s：サーバ数

y：キー構成列がすべて固定長の場合は 10
キー構成列に可変長を含む場合は 12

I：インデクス数

R：表，又はインデクスの分割 RD エリア数

注

インデクス情報ファイル及びソート用ワークファイルの容量を算出するとき，インデクス構成列が繰返し列の場合はｂ及びｅは行数ではなく，行数×要素数となります。

注※

-m オプションでインフォメーションメッセージ出力抑止レベルに lvl2 を指定した場合に出力されます。

18.1.2 データベース再編成ユティリティ（pdrorg）実行時のファイルの容量

データベース再編成ユティリティ（pdrorg）で使用するファイルの容量の計算式を次に示します。

ファイルの種類	容量の計算式（単位：バイト）
アンロードデータファイル※1 （オプション指定なし）	n Σ（Pi ＋ M）＋ Li※7 ＋ 1200 ＋ A ＋ B ＋ c×96 ＋ D ＋ I ＋ F i=1
アンロードデータファイル※1 （-W オプション指定時）	DAT 形式，又は拡張 DAT 形式の場合 { c Σ（列の最大文字変換長 i ※2 ＋ 1）×C ＋ M i=1 } ×n バイナリ形式で FIX 表の場合 { c Σ（列データ長 i ※3）＋ M i=1 } ×n

ファイルの種類	容量の計算式（単位：バイト）
	バイナリ形式で非 FIX 表の場合 { c Σ（列データ長 i ※3 ＋ G）＋ M ＋ 4×（c ＋ 1） i=1 } ×n **固定長文字形式の場合** { c Σ（列の最大文字変換長 i ※4 ＋ crlf）×C ＋ M i=1 } ×n
アンロードデータファイル※1 (-j オプション指定時又はスキーマ単位の再編成時）※5	n Σ（Pi ＋ M）＋ Li※7 ＋ i=1 n Σ { i=1 m Σ（Oij ＋ 44） j=1 } ＋ 1200 ＋ A ＋ B ＋ c×96 ＋ D ＋ I ＋ F
LOB データのアンロードデータファイル※1	n Σ { i=1 m Σ（Oij ＋ 44） j=1 } ＋ 1200 ＋ A ＋ B ＋ c×96 ＋ D ＋ I ＋ F
インデクス情報ファイル	**B-tree インデクスの場合** （K ＋ p）×n ＋ 512 **プラグインインデクスの場合** （12 ＋ X）×n ＋ 1024 この計算式は，1 インデクス当たりの容量計算式です。インデクスが複数ある場合は，それぞれのインデクスに対して計算してください。
処理結果ファイル	1700 ＋表格納サーバ数×500 ＋スキーマ内表数×1000 ＋スキーマ内総格納 RD エリア数×100
ワークファイル※6	[8 ＋ 2×S ＋ 2×｛n÷（-m オプションに指定した経過メッセージ出力間隔の値)｝＋ 3×R ＋ 4×J×R］×200
ソート用ワークファイル	**条件 1 の場合**

ファイルの種類	容量の計算式（単位：バイト）
	インデクス情報ファイルの容量＋ 4×ｎ **条件 2 の場合** {インデクス情報ファイルの容量＋ 4×ｎ｝×2 • 条件 1 1024 キロバイト≧E のとき • 条件 2 1024 キロバイト＜ E のとき E：バッファサイズ 「ソート用ワークファイル容量の計算で使用するバッファサイズ」で求めたバッファサイズ

A：キーレンジ分割の場合：48 ＋分割条件数×284

ハッシュ分割の場合：40 ＋ａ×60

マトリクス分割（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：

48 ＋（分割条件数×284）＋（40 ＋ａ×60）

B：ｎ×36（FIX 表の場合）又は（44 ＋ｃ×4）×ｎ（非 FIX 表の場合）

C：アンロードデータファイルの出力文字コードが HiRDB の既定文字コード以外の場合：2

上記以外の場合：1

D：16 ＋（LOB 列数×ａ×80）

D は LOB 列がある場合に加算します。

F：次の値を代入してください。

```
d
Σ ｛（列iのプラグインが提供する抽象データ型の属性数×84）＋
i=1      （列iのプラグインが提供する抽象データ型のLOB属性数×ａ×72）｝

＋64＋
d
Σ （84＋逆生成関数数ｉ×60）
i=1
```

G：列 I に対する逆生成関数の返却値が BLOB の属性数× 4

Ｉ：136 ＋インデクス分割数×60

インデクス引き上げ時に加算します。

Ｊ：インデクス数

K：インデクスのキー長

表「インデクスのキー長一覧」を参照してください。ただし，可変長データの場合は単一列でも複数列として扱い，定義長の最大値で計算してください。

L_i：行の実長

行の実長は，概算値又は詳細値で見積もってください。なお，BINARY 型で圧縮指定ありの場合は，詳細値で見積もると計算が複雑になるため，概算値で見積もることをお勧めします。

概算値の見積もり：

データベースに格納されているデータから，次の計算式で行の実長の全行の合計概算値（単位：バイト）を求めます。

```
表格納RDエリアの使用中ページ数×表格納RDエリアのページ長
```

表格納 RD エリアの使用中ページ数，及びページ長については，pddbst の RD エリア単位の状態解析（論理的解析），又は表単位の状態解析の実行結果から取得できます。

詳細値の見積もり：

各列のデータから，次の値を使用して行の実長を求めます。

列のデータ型	実長（単位：バイト）
BLOB	16
プラグインが提供する抽象データ型	2
BINARY※	• pdrorg -k rorg で圧縮指定あり，かつ UOC を使用しない場合 定義長＋ 8×MAX（ SQL の UPDATE 文で連結演算を使用した回数, ↑定義長÷圧縮分割サイズ↑） • 上記以外の場合 BINARY 型の列の定義長
上記以外	各列の実長

注※

行の実長の最大値を算出するため，BINARY 型のデータが定義長のデータで，圧縮率が 0%と仮定しています。

M：文字集合が指定された文字列型の列の合計データ長

次の値になります。

$$\left(\sum_{i=1}^{k} \text{列データ長}\,i\right)$$

O_{ij}：LOB データ長

P_i：プラグインが提供する抽象データ型のデータ長

R：表又はインデクスの分割 RD エリア数

S：表格納サーバ数

X：次に示す値

- LOB 用 RD エリアに格納された抽象データ型の場合は 27
- 定義長 255 バイト以下の抽象データ型の場合は（キー長＋ 2）
- 定義長 256 バイト以上の抽象データ型の場合は 2

代表的な抽象データ型の値を次に示します。

- SGMLTEXT 型の場合は 27
- FREEWORD，GEOMETRY，及び XML 型の場合は 2

a：分割 RD エリア数

c：列定義数

d：プラグインが提供する抽象データ型が定義された列数

k：文字集合が指定された文字列型の列の数

m：LOB 列数

n：行数（繰返し列の場合は行数×要素数）

p：キー構成列がすべて固定長の場合は 10，キー構成列に可変長を含む場合は 12

crlf：-W オプションに cr 又は crlf を指定した場合に加算する改行文字の長さ

次の表から改行文字の長さを求めます。

-W オプション指定値		加算値
-W dat 又は-W extdat	,cr	1
	,crlf	2
	上記の指定なし	1
-W fixtext	,cr	1
	,crlf	2
	上記の指定なし	0

注

インデクス情報ファイル及びソート用ワークファイルの容量を算出するとき，インデクス構成列が繰返し列の場合は「リロードする行数」及び n は行数ではなく，行数×要素数となります。

注※1

ファイルの容量が 2 ギガバイトを超える場合は，次に示すどちらかの対処をしてください。

- 2 ギガバイト以下のファイルを複数個作成してください。
- ラージファイルを使用してください。ラージファイルの作成方法については，「HiRDB ファイルシステム領域の作成」を参照してください。

注※2

DAT 形式（-W dat）又は拡張 DAT 形式（-W extdat）の場合の列の最大文字変換長を次の表に示します。

表 18-1　列の最大文字変換長（DAT 形式又は拡張 DAT 形式の場合）

データ型		最大文字変換長（バイト）
数データ	INTEGER	11
	SMALLINT	11
	DECIMAL	40
	FLOAT	23
	SMALLFLT	23
文字データ※1	CHARACTER	定義長＋ 2※2
	VARCHAR	実長＋ 2※2
混在文字データ※1	MCHAR	定義長＋ 2※2
	MVARCHAR	実長＋ 2※2
各国文字データ※1	NCHAR	定義長×2 ＋ 2※2
	NVARCHAR	実長＋ 2※2
日付データ	DATE	10
時刻	TIME	8
日間隔データ	INTERVAL YEAR TO DAY	9
時間隔データ	INTERVAL HOUR TO SECOND	7
時刻印データ	TIMESTAMP	19 小数秒けた数が 0 でなければ，小数秒けた数＋ 1 を加算してください。
バイナリデータ※1	BINARY	実長＋ 2※2

注※ 1

拡張 DAT 形式の場合，データ中に「”」があると，「”」の数分，文字変換長が長くなります。

注※ 2

データの長さに囲み文字数を 2 バイト加算しています。

ただし，-W dat 又は-W extdat 指定時，オペランドに sup を指定した場合の列の最大文字変換長は次に示すとおりです。なお，表中の実長とは，後方の連続スペースを圧縮した残りの長さです。スペースを圧縮した場合の出力形式については，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）の-W オプションの説明を参照してください。

データ型		最大文字変換長（バイト）
文字データ	CHARACTER	実長＋ 2
混在文字データ	MCHAR	実長＋ 2
各国文字データ	NCHAR	実長＋ 2

注※3

データ長については次を参照してください。

- 表「データ長一覧」
- 表「可変長文字列型のデータ長一覧（抽象データ型及び繰返し列を除く）」
- 表「可変長文字列型のデータ長一覧（抽象データ型の場合）」
- 表「可変長文字列型のデータ長一覧（繰返し列の場合）」

注※4

固定長文字形式（-W fixtext）の場合の列の最大文字変換長を次の表に示します。

表 18-2　列の最大文字変換長（固定長文字形式の場合）

データ型		最大文字変換長（バイト）	
数データ	INTEGER	11	
	SMALLINT	6	
	DECIMAL	けた数＋ 2	
	FLOAT	23	
	SMALLFLT	23	
文字データ	CHARACTER VARCHAR	定義長	fixtext_option に enclose オペランドを指定した場合は出力長に 2 を加算してください。
混在文字データ	MCHAR MVARCHAR	定義長	
各国文字データ	NCHAR NVARCHAR	定義長×2	
日付データ	DATE	10	
時刻	TIME	8	
日間隔データ	INTERVAL YEAR TO DAY	10	
時間隔データ	INTERVAL HOUR TO SECOND	8	
時刻印データ	TIMESTAMP	小数部 0:19　2:22　4:24　6:26	
長大データ	BLOB	0	

データ型		最大文字変換長（バイト）
バイナリデータ	BINARY	0
抽象データ型	ADT	0

注※5

スキーマ単位の再編成（アンロードも含む）をする場合は，表ごとに求めた値を合計してください。

注※6

-m オプションでインフォメーションメッセージ出力抑止レベルに lvl2 を指定した場合に出力されます。

注※7

Li を詳細値で見積もる場合は，「(Pi ＋ M) ＋ Li」を「(Li ＋ Pi ＋ M)」に置き換えてください。

18.1.3 統計解析ユティリティ（pdstedit）実行時のファイルの容量

統計解析ユティリティ（pdstedit）で使用するファイルの容量の計算式を次に示します。

ファイルの種類		容量の計算式（単位：バイト）
ワーク用一時ファイル	システムの稼働に関する統計情報	4096×取得回数※1×2
	サーバごとのシステムの稼働に関する統計情報	4096×取得回数※1×サーバ数×2
	UAP に関する統計情報	872×実行 UAP 数又は実行トランザクション数×2
	SQL に関する統計情報	512×実行 SQL 数×2
	SQL 静的最適化に関する統計情報	512×SQL オブジェクトキャッシュでミスヒットした回数
	SQL 動的最適化に関する統計情報	49624×OPEN 又は EXECUTE で発行した SELECT 文の数（INSERT SELECT も含む）
	SQL オブジェクト実行に関する統計情報	512×実行 SQL 数×サーバ数
	SQL オブジェクト転送に関する統計情報	256×実行 SQL 数×サーバ数
	SQL 文の履歴に関する統計情報	（1024 ＋平均的な SQL 長）×実行 SQL 数
	CONNECT/DISCONNECT に関する統計情報	256×CONNECT 及び DISCONNECT 回数
	グローバルバッファに関する統計情報	512×シンクポイント発生回数×2
	データベース操作に関する HiRDB ファイルの統計情報	512×シンクポイント発生回数×2
	デファードライト処理に関する統計情報	512×デファードライト発生回数×2
	インデクスに関する統計情報 （入力：STJ）	128×シンクポイント発生回数×2

ファイルの種類		容量の計算式（単位：バイト）
	インデクスに関する統計情報 （入力：FJ）	128×ページスプリット発生回数×2
	データベースへの入出力に関する統計情報	256×取得時間間隔ごとにアクセスのあった RD エリアを構成する HiRDB ファイル数の合計×2
ソート用ワークファイル	解析対象とした上記のワーク用一時ファイルのソート用作業領域	解析対象とした上記のワーク用一時ファイルのファイル容量の最大値
DAT 形式ファイル※2	システムの稼働に関する統計情報	3404×取得回数※1
	サーバごとのシステムの稼働に関する統計情報	3262×取得回数※1×サーバ数
	UAP に関する統計情報	2167×実行 UAP 数又は実行トランザクション数
	SQL に関する統計情報	447×実行 SQL 数
	SQL 静的最適化に関する統計情報	646×SQL オブジェクトキャッシュでミスヒットした回数
	SQL 動的最適化に関する統計情報	380×OPEN 又は EXECUTE で発行した SELECT 文の数（INSERT SELECT も含む）
	SQL オブジェクト実行に関する統計情報	520×実行 SQL 数×サーバ数
	SQL オブジェクト転送に関する統計情報	389×実行 SQL 数×サーバ数
	SQL 文の履歴に関する統計情報	（240 ＋平均的な SQL 長）×実行 SQL 数
	CONNECT/DISCONNECT に関する統計情報	278×CONNECT 及び DISCONNECT 回数
	グローバルバッファに関する統計情報	600×シンクポイント発生回数
	データベース操作に関する HiRDB ファイルの統計情報	356×シンクポイント発生回数
	デファードライト処理に関する統計情報	330×デファードライト発生回数
	データベースへの入出力に関する統計情報（DAT 出力）	196×取得時間間隔ごとにアクセスのあった RD エリアを構成する HiRDB ファイル数の合計

注※1

取得回数＝↓（pdstend コマンドの入力時刻－pdstbegin コマンドの入力時刻）÷-m オプションで指定した時間間隔↓

注※2

-b オプションを指定した場合は，容量計算式に 3810 を加算

(1)　ワーク用一時ファイルの容量の概算値

統計解析ユティリティ実行時に作成されるワーク用一時ファイルの容量の概算値は，次の計算式で求められます。なお，この概算値は，ユティリティの入力情報となる統計ログファイル内のすべての情報を解析した場合の値です。

ワーク用一時ファイルの容量＝統計ログファイルの容量×a（単位：バイト）

a：

統計ログファイルに含まれる統計情報の種別に応じて次の値を代入してください。統計ログファイルに複数の種別の統計情報が含まれる場合は，最も大きい値を a の値とします。

統計情報の種別	a の値
システムの稼働に関する統計情報	3.4
SQL 動的最適化に関する統計情報	135.6
上記以外	2

誤差について

統計ログファイルに複数の種別の統計情報が含まれている場合で，SQL 動的最適化に関する統計情報が含まれるとき，その含まれる比率によっては，概算値と実際の値との誤差が大きくなります。この誤差を小さくしたい場合は，SQL 動的最適化に関する統計情報の部分だけを別に算出してください。

SQL 動的最適化に関する統計情報のワーク用一時ファイルの容量は，次の計算式から求められます。

SQL動的最適化に関する統計情報のワーク用一時ファイルの容量＝SQL動的最適化に関する統計情報の情報数×49624（単位：バイト）

SQL 動的最適化に関する統計情報の情報数は，統計解析ユティリティ実行時に出力される入力ログファイルのサマリ情報から取得します。入力ログファイルのサマリ情報については，マニュアル「HiRDB コマンドリファレンス」の「統計解析ユティリティ（pdstedit)」の「統計情報の出力形式」を参照してください。

(2)　ソート用ワークファイルの容量の概算値

統計解析ユティリティ実行時に作成されるソート用ワークファイルの容量の概算値は，次の計算式で求められます。なお，この概算値は，ユティリティの入力情報となる統計ログファイル内のすべての情報を解析した場合の値です。

ソート用ワークファイルの容量＝統計ログファイルの容量×b（単位：バイト）

b：

統計ログファイルに含まれる統計情報の種別に応じて次の値を代入してください。統計ログファイルに複数の種別の統計情報が含まれる場合は，最も大きい値を b の値とします。

統計情報の種別	b の値
システムの稼働に関する統計情報	1.7
上記以外	1

(3) DAT 形式ファイルの容量の概算値

統計解析ユティリティ実行時に作成される DAT 形式ファイルの容量の概算値は，次の計算式で求められます。なお，この概算値は，ユティリティの入力情報となる統計ログファイル内のすべての情報を解析した場合の値です。

DAT形式ファイルの容量＝統計ログファイルの容量×2（単位：バイト）

18.1.4 データベース状態解析ユティリティ（pddbst）実行時のファイルの容量

データベース状態解析ユティリティ（pddbst）で使用するファイルの容量の計算式を次に示します。

ファイルの種類		容量の計算式（単位：キロバイト）
ワーク用ファイル	RD エリア単位の物理的解析	1 ＋ 4.1×a
	RD エリア単位の論理的解析	$1 + 4.1 \times \sum_{i=1}^{a} (\text{RD エリア内の表数 i} + \text{インデクス数 i} + \text{LOB 用 RD エリア数 i})$
	状態解析結果蓄積又は再編成時期予測※	$1 + 4.1 \times \sum_{i=1}^{a} (\text{RD エリア内の表数 i} + \text{インデクス数 i} + \text{LOB 用 RD エリア数 i})$
	表単位の状態解析	1 ＋ 4.1×（格納 RD エリア数）
	インデクス単位の状態解析	1 ＋ 4.1×（格納 RD エリア数）
	クラスタキーの状態解析	1 ＋ 4.1×（格納 RD エリア数）
ソート用ワークファイル	上記のワーク用ファイルのソート用作業領域	上記の計算式で求めた値×2

注※

pddbst -r ALL 指定時は，ユーザ用 RD エリア内だけでなく，ディクショナリ用 RD エリア内の資源数を加算する必要があります。また，分割した表やインデクスについては RD エリアごとに数を加算します。

a：解析対象の RD エリア数

18.1.5 データベース複写ユティリティ（pdcopy）実行時のファイルの容量

データベース複写ユティリティ（pdcopy）で使用するファイルの容量の計算式を次に示します。

ファイルの種類	容量の計算式（単位：バイト）
バックアップファイル※ フルバックアップファイル※	$\sum_{i=1}^{a}\{28\times c_i + (d_i + 28) \times (e_i + q_i)\}$ $+ 88\times a + 220\times b$
差分バックアップファイル※	w×フルバックアップファイルの容量
差分バックアップ管理ファイル	(1 + j + m) ×32768
ログポイント情報ファイル	1024

注※

バックアップファイルの容量が 2 ギガバイトを超える場合は，次に示すどちらかの対処をしてください。

- バックアップファイルにラージファイルを使用してください。ラージファイルの作成方法については，「HiRDB ファイルシステム領域の作成」を参照してください。
- 2 ギガバイト以下のパーティションを複数作成して，バックアップファイルを複数指定してください。

a：バックアップ対象 RD エリア数

b：バックアップ対象 RD エリアの HiRDB ファイルの総数

c_i：バックアップ対象 RD エリアの未使用ページ数

システム構築前に見積もる場合は 0 と仮定してください。

- ユーザ用 RD エリアの場合

  データベース状態解析ユティリティ（pddbst コマンド）の RD エリア単位の状態解析（物理的解析）を実行して求めてください。ユティリティの実行結果の「RD エリア内のページの情報」の「総ページ数－使用ページ数」の値です。

- ユーザ LOB 用 RD エリアの場合

データベース状態解析ユティリティ（pddbst コマンド）の RD エリア単位の状態解析（物理的解析）を実行して求めてください。ユティリティの実行結果の「RD エリア内のセグメントの情報」の「総セグメント数－使用セグメント数」の値です。

d_i：バックアップ対象 RD エリアのページ長

e_i：バックアップ対象 RD エリアの使用ページ数

システム構築前に見積もる場合は，（バックアップ対象 RD エリアのセグメント数×セグメントサイズ）を仮定して計算してください。

- ユーザ用 RD エリアの場合

  データベース状態解析ユティリティ（pddbst コマンド）の RD エリア単位の状態解析（物理的解析）を実行して求めてください。ユティリティの実行結果の「RD エリア内のページの情報」の「使用ページ数」の値です。

- ユーザ LOB 用 RD エリアの場合

  データベース状態解析ユティリティ（pddbst コマンド）の RD エリア単位の状態解析（物理的解析）を実行して求めてください。ユティリティの実行結果の「RD エリア内のセグメントの情報」の「使用セグメント数」の値です。

g：-b オプションに指定するバックアップファイルの名称長（バイト）

複数のバックアップファイルを指定する場合は，合計の名称長になります。

h：-b オプションに指定するバックアップファイルの数

j：↑（512 + 128×a）÷32700↑

k：差分バックアップの継続回数

m：↑｛↑（256 + 128×a + g + 8×h）÷256↑×k｝÷100↑

q_i：バックアップ対象 RD エリアのディレクトリページ数

- ユーザ用 RD エリアの場合

  6×（ti + 1）+ 2×↑（20480÷di）↑+｛↑（si÷ui）↑+↑（si÷vi）↑+ 2×ti｝

- ユーザ LOB 用 RD エリアの場合

  7 + 3×（ti－1）+｛↑（si÷64000）↑+ ti｝×96

ri：バックアップ対象 RD エリアのセグメントサイズ

si：バックアップ対象 RD エリアの全セグメント数

データベース初期設定ユティリティ（pdinit コマンド）又はデータベース構成変更ユティリティ（pdmod コマンド）の create rdarea 文で指定する HiRDB ファイルのセグメント数の合計です。RD エリアの自動増分を指定している場合は，増分したセグメント数を加算してください。

ti：バックアップ対象 RD エリアの HiRDB ファイル数

ui：↓｛di－20｝÷｛(↑ri÷32↑×8）＋56｝↓

vi：↑（125×di）÷（16×ui）↑×ui

w：バックアップ対象 RD エリアの全ページに対する更新ページの比率

wの算出式（ユーザ用 RD エリア）は次に示す計算式から求めます。

$$w = \{ \sum_{i=1}^{a} Xi \div \sum_{i=1}^{a} e_i \} \times 1.2$$

Xi：バックアップ対象 RD エリアの更新ページ数

更新ページ数とは，差分の基準となる前回の差分バックアップ取得時から現在までの間に更新されたページの数のことです。更新ページ数は更新 SQL の種類又は更新件数などから，バックアップ対象 RD エリアに格納されている表及びインデクスについて，次に示す条件に従って計算してください。

- INSERT の場合

  「ユーザ用 RD エリアの容量の見積もり」を参照して，インサート件数から格納ページ数を計算して Xi に加算してください。このとき，PCTFREE は 0 で計算してください。

- DELETE の場合

  次に示す条件に応じて計算した値を Xi に加算してください。

条件			Xi に加算する値
表	行長＜ d_i	削除行が表全体に分散する場合	MIN（削除行数，表の使用中ページ数）
		削除行が一部のページに集中する場合	MIN（削除行数÷ 1 ページで格納できる行数，表の使用中ページ数）
	行長＞ d_i		MIN（削除行数× 1 行格納するのに必要なページ数，表の使用中ページ数）
インデクス	更新されるキーがインデクス全体に分散する場合		MIN（更新キー数＋ α，インデクスの使用中ページ数）
	更新されるキーが一部のページに集中する場合		MIN（更新キー数÷ 1 ページで格納できるキー数＋ α，インデクスの使用中ページ数）

α：201 以上重複するキーの数

- UPDATE の場合

  次に示す条件に応じて計算した値を Xi に加算してください。

条件			Xi に加算する値
表	更新列長＜ d_i	更新行が表全体に分散する場合	MIN（更新行数，表の使用中ページ数）
		更新行が一部のページに集中する場合	MIN（更新行数÷ 1 ページで格納できる行数，表の使用中ページ数）

条件		Xi に加算する値
	更新列長＞ d_i	MIN（更新行数×更新列を格納するのに必要なページ数，表の使用中ページ数）
インデクス	更新されるキーがインデクス全体に分散する場合	MIN（更新キー数×2 ＋ α×2，インデクスの使用中ページ数）
	更新されるキーが一部のページに集中する場合	MIN（更新キー数×2÷ 1 ページで格納できるキー数＋ α×2，インデクスの使用中ページ数）

α：201 以上重複するキーの数

- 再編成を行った場合

  再編成を行った表又はインデクスのうち，バックアップ対象 RD エリアに格納されているものについて β を計算してください。

  β ＝再編成を行った表又はインデクスの使用中ページ数＋再編成を行った表又はインデクスの使用中セグメント数÷ui ＋再編成を行った表又はインデクスの使用中セグメント数÷vi

  β を再編成した表又はインデクスの数だけ計算して Xi に加算してください。

- PURGE の場合

  PURGE を行った表又はインデクスのうち，バックアップ対象 RD エリアに格納されているものについて γ を計算してください。

  γ ＝ PURGE を行った表又はインデクスの使用中セグメント数÷ui ＋ PURGE を行った表又はインデクスの使用中セグメント数÷vi

  γ を再編成した表又はインデクスの数だけ計算して Xi に加算してください。

18.1.6 ディクショナリ搬出入ユティリティ（pdexp）実行時のファイルの容量

ディクショナリ搬出入ユティリティ（pdexp）で使用するファイルの容量の計算式を次に示します。

ファイルの種類		容量の計算式（単位：キロバイト）
搬出ファイル※	実表の場合	0.7 ＋ 0.5×CMN ＋（0.1×↑CMN÷10↑） DEF ＋ Σ（0.1 ＋ DSn）＋ 0.1×↑LRD÷10↑ n=1 ＋ 0.6×↑DIV÷10↑＋ 2.8×REF CHK ＋ Σ（0.1 ＋ 1.0×↑CSn÷10↑） n=1 IDX ＋ Σ（2.7 ＋ 0.2×IRn） n=1

ファイルの種類		容量の計算式（単位：キロバイト）
	ビュー表の場合	0.6 ＋ 0.4×CMN ＋（0.1×↑CMN÷10↑）＋ e
	プロシジャの場合	0.6 ＋ 0.1×g ＋ h

CHK：検査制約数（0≦CHK≦254）

CMN：表の列数（1≦CMN≦30,000）

CSn：n 番目の検査制約の検索条件サイズ（0≦CSn≦2,000,000）

DEF：デフォルト値定義列数（0≦DEF≦30,000）

DIV：分割条件数（0≦DIV≦4,096）

DSn：n 番目のデフォルト列のデフォルト値サイズ（1≦DSn≦64,003）

IDX：インデクス数（0≦IDX≦254）

IRn：n 番目のインデクスの格納用 RD エリア数（0≦IRn≦4,096）

LRD：LOB 用 RD エリア数（0≦LRD≦4,096）

REF：参照制約数（0≦REF≦255）

e：ビュー表定義時のソース長（単位：キロバイト）

g：搬出するストアドプロシジャが使用するリソース数
SQL_ROUTINES 表の N_RESOURCE 列の値です。

h：ストアドプロシジャのソース長（単位：キロバイト）
SQL_ROUTINES 表の SOURCE_SIZE 列の値です。

注※
複数の表を搬出する場合，各表に対して上記の計算をしてください。その合計値が搬出ファイルの容量となります。

18.1.7 最適化情報収集ユティリティ（pdgetcst）実行時のファイルの容量

最適化情報収集ユティリティ（pdgetcst）で使用するファイルの容量の計算式を次に示します。

ファイルの種類	容量の計算式（単位：バイト）
最適化情報パラメタファイル	162 ＋ 405×a ＋ 567×b ＋ 162 ＋ 324×b×g
出力結果ファイル	**検索による最適化情報収集時（-c lvl1 指定時）** 202 ＋ 131×e **検索による最適化情報収集時（-c lvl2 指定時）**

ファイルの種類	容量の計算式（単位：バイト）
	370 ＋ 561×c ＋ 196×d 最適化情報パラメタファイルによる最適化情報登録時 370 ＋ 235×c ＋ 387×f ＋ 196×g

a：指定インデクス数

b：指定列数

c：表に定義してあるインデクスの数

d：区間数（全インデクスの区間数の合計）

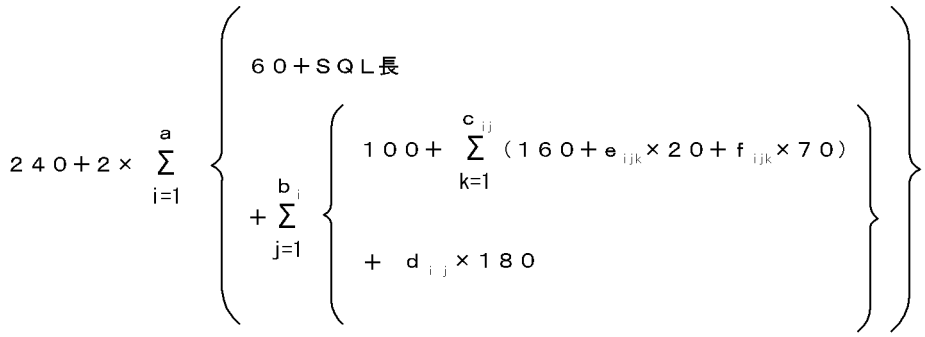
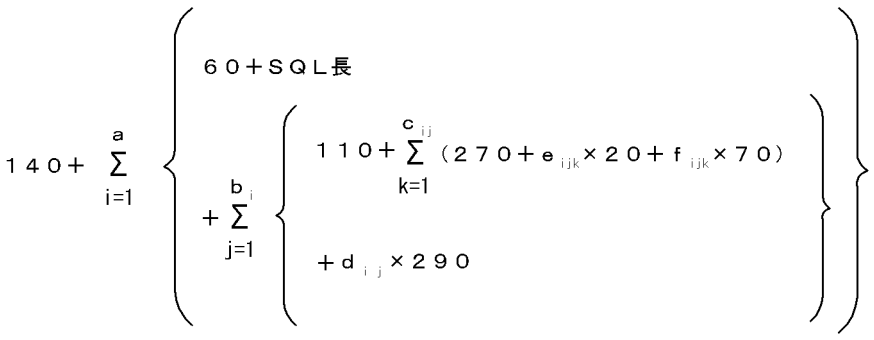
e：表数

f：表の列数

g：区間数（全列定義で指定した区間数の合計）

18.1.8 アクセスパス表示ユティリティ（pdvwopt）実行時のファイルの容量

アクセスパス表示ユティリティ（pdvwopt）で使用するファイルの容量の計算式を次に示します。

ファイルの種類	容量の計算式（単位：バイト）
アクセスパス 情報ファイル	・HiRDB/シングルサーバの場合  ・HiRDB/パラレルサーバの場合 

a：検索系 SQL 数

b_i：SQL 中の問合せ数

c_{ij}：問合せ中の表数

d_{ij}：問合せ中の結合処理数

e_{ijk}：表の格納 RD エリア数

f_{ijk}：表のインデクス定義数

18.1.9 リバランスユティリティ（pdrbal）実行時のファイルの容量

リバランスユティリティ（pdrbal）で使用するファイルの容量の計算式を次に示します。

ファイルの種類	容量の計算式（単位：バイト）
インデクス情報ファイル	**B-tree インデクスの場合** （K ＋ d）×N ＋ 512 **プラグインインデクスの場合** （12×Y）×N ＋ 1024 この計算式は，1 インデクス当たりの容量計算式です。インデクスが複数ある場合は，それぞれのインデクスに対して計算してください。
ワークファイル※1	（3 ＋ 4×I×R）×200
ソート用ワークファイル※2	**条件 1 の場合** インデクス情報ファイルの容量＋ 4×N **条件 2 の場合** {インデクス情報ファイルの容量＋ 4×N} ×2 • 条件 1 1024 キロバイト≧E のとき • 条件 2 1024 キロバイト＜ E のとき E：バッファサイズ 「ソート用ワークファイル容量の計算で使用するバッファサイズ」で求めたバッファサイズ
実行結果出力ファイル	1000 ＋表格納 RD エリア数×200

d：キー構成列がすべて固定長の場合は 10，キー構成列に可変長を含む場合は 12

I：インデクス数

K：インデクスのキー長

表「インデクスのキー長一覧」を参照してください。ただし，可変長データの場合は単一列でも複数列として扱い，定義長の最大値で計算してください。

N：リバランス処理によって移動する行数（繰返し列の場合は行数×要素数）

R：表又はインデクスの分割 RD エリア数

Y：次に示す値

- LOB 用 RD エリアに格納された抽象データ型の場合は 27
- 定義長 255 バイト以下の抽象データ型の場合は（キー長＋ 2）
- 定義長 256 バイト以上の抽象データ型の場合は 2

代表的な抽象データ型の値を次に示します。

- SGMLTEXT 型の場合は 27
- FREEWORD，GEOMETRY，及び XML 型の場合は 2

注※1

-m オプションに lvl2 を指定した場合に出力されます。

注※2

プラグインインデクスの場合，このファイルは不要です。

18.1.10 整合性チェックユティリティ（pdconstck）実行時のファイルの容量

整合性チェックユティリティ（pdconstck）で使用するファイルの容量の計算式を次に示します。

ファイルの種類	容量の計算式（単位：バイト）
処理結果ファイル	-k set/release の場合 560 ＋（REF ＋ 1）×70 ＋（CHK ＋ 1）×70 -k check の場合 700 ＋（REF ＋ 1）×70 ＋（CHK ＋ 1）×70 REF ＋ Σ（490 ＋（RCn×70） n=1 GEN ROW ＋ Σ（Σ（RCnml×70）） m=1 l=1 CHK ＋ Σ（490 ＋（CCn×70） n=1

ファイルの種類	容量の計算式（単位：バイト）
	GEN ROW ＋ Σ（Σ（CCnml×70）） m=1 l=1

REF：表に定義されている参照制約の制約数

CHK：表に定義されている検査制約の制約数

RC：参照制約の外部キー構成列数

CC：検査制約の探索条件中の列数

GEN：インナレプリカ機能を使用しない場合は 1
　インナレプリカ機能を使用する場合は，その表のレプリカ RD エリアが存在する世代数（1～10）＋ 1

ROW：制約違反となったキー値の出力数の上限値（-w オプションで指定した値）

18.1.11　パラレルローディング（pdparaload）実行時のファイルの容量

パラレルローディング（pdparaload）で使用するファイルの容量の計算式を次に示します。

ファイルの種類	容量の計算式（単位：バイト）
pdload 制御文ファイル	（pdparaload 制御文ファイルの容量＋ 200）×R
pdload が作成するファイル※	表を構成する RD エリア Σ（RD エリア単位のデータロード実行に必要なファイル容量 i） i=1

R：表を構成する RD エリア数

注※
　pdparaload コマンドは，表を構成する RD エリアの数だけ，内部で RD エリア単位のデータロード（pdload）を実行します。そのため，pdparaload は，RD エリア単位のデータロード実行に必要なファイルを，表を構成する RD エリア数分使用します。RD エリア単位のデータロード実行に必要なファイルの容量については，「データベース作成ユティリティ（pdload）実行時のファイルの容量」を参照してください。

18.1.12　ソート用ワークファイル容量の計算で使用するバッファサイズ

ソート用バッファサイズの計算式を次に示します。

●32ビットモードのHiRDBの場合

$$\text{バッファサイズ（単位：バイト）} : \frac{R+7}{2} + \sqrt{(B+8)\times n\times A+\frac{(R+7)^2}{4}} + C$$

●64ビットモードのHiRDBの場合

$$\text{バッファサイズ（単位：バイト）} : \frac{R+15}{2} + \sqrt{(B+8)\times n\times A+\frac{(R+15)^2}{4}} + C$$

n：処理するデータ件数

- pdload の場合：表の既存データと追加ロードするデータの合計
- pdrorg の場合：アンロードしたデータ件数
- pdrbal の場合：リバランスするデータ件数

なお，繰返し列の場合，データ件数は行数ではなく要素数となります。

k：キー長（最大値で計算します）。キー長の計算式については，「インデクスの格納ページ数の計算例」を参照してください。

x：キー構成列がすべて固定長の場合は 10，キー構成列に可変長を含む場合は 12

c：インデクスの構成列数

y：Linux 版の場合は 2，そのほかの場合は 1

z：可変長の複数列インデクスの場合は c×4，そのほかの場合は 0

K：可変長の複数列インデクスの場合は k ＋ c ＋ 8，そのほかの場合は k ＋ 12

N：可変長の複数列インデクスの場合は（c×2）＋ y，そのほかの場合は 3 ＋ y

R：k ＋ x ＋ z

A：32 ビットモードの HiRDB の場合は R ＋（K ＋ 8）＋ 28，64 ビットモードの HiRDB の場合は R ＋（K ＋ 8）＋ 56

B：32 ビットモードの HiRDB の場合は R ＋（K ＋ 8）＋ 56，64 ビットモードの HiRDB の場合は R ＋（K ＋ 8）＋ 104

C：32 ビットモードの HiRDB の場合は 2092 ＋（N×32）＋（K ＋ 8），64 ビットモードの HiRDB の場合は 2112 ＋（N×32）＋（K ＋ 8）

18.2　ユティリティ実行時のメモリ所要量の見積もり

18.2.1　データベース初期設定ユティリティ（pdinit）実行時のメモリ所要量

データベース初期設定ユティリティ（pdinit）実行時のメモリ所要量は，次に示す計算式で求めます。

(1)　HiRDB/シングルサーバの場合

条件	メモリ所要量の計算式（単位：キロバイト）
32 ビットモードの場合	↑{ {61440×（140×a ＋ 20×b ＋ c)} ÷61432 ＋（6004×d）÷500 ＋（36008×a）÷1000 ＋ 468×e ＋ 403888 }↑÷1024 ＋ 267
64 ビットモードの場合	↑{ {61448×（144×a ＋ 24×b ＋ c)} ÷61436 ＋（8004×d）÷500 ＋（36016×a）÷1000 ＋ 468×e ＋ 405888 }↑÷1024 ＋ 267

a：RD エリアの総数

b：全 RD エリアの HiRDB ファイル数

c：全 HiRDB ファイルの名称長の合計

d：認可識別子の総数

e：データディクショナリ用 RD エリア数

(2)　HiRDB/パラレルサーバの場合

条件		メモリ所要量の計算式（単位：キロバイト）
32 ビットモードの場合	DS	↑{ { 61440×（140×a ＋ 20×b ＋ c)}÷61432 ＋（6004×d）÷500 ＋（36008×a）÷1000 ＋ 468×e ＋ 403888 ＋ 348×f ＋ 344×g }↑÷1024 ＋ 268
	BES	↑（4×b ＋ 237220）÷1024↑＋ 268
	MGR	10
64 ビットモードの場合	DS	↑{ {61448×（144×a ＋ 24×b ＋ c)}÷61436 ＋（8004×d）÷500 ＋（36016×a）÷1000 ＋ 468×e ＋ 405888 ＋ 348×f ＋ 344×g }↑÷1024 ＋ 268

条件		メモリ所要量の計算式（単位：キロバイト）
	BES	↑（4×b + 245744）÷1024↑ + 268
	MGR	10

a：RD エリアの総数

b：全 RD エリアの HiRDB ファイル数

c：全 HiRDB ファイルの名称長の合計

d：認可識別子の総数

e：データディクショナリ用 RD エリア数

f：バックエンドサーバの総数

g：(144×a + 24×b + c) ÷7780 を各バックエンドサーバごとに計算した総計

18.2.2 データベース定義ユティリティ（pddef）実行時のメモリ所要量

データベース定義ユティリティ（pddef）実行時のメモリ所要量は，次に示す計算式で求めます。

条件	メモリ所要量の計算式（単位：キロバイト）
HiRDB/シングルサーバの場合	32 ビットモードの場合：1956 64 ビットモードの場合：1957
HiRDB/パラレルサーバの場合	

18.2.3 データベース作成ユティリティ（pdload）実行時のメモリ所要量

データベース作成ユティリティ（pdload）実行時のメモリ所要量は，次に示す計算式で求めます。計算式の変数については「計算式で使用する変数」を参照してください。

(1) HiRDB/シングルサーバの場合

条件		メモリ所要量の計算式（単位：キロバイト）
32 ビットモードの場合	シングルサーバ	6352 + ↑{(α + γ) ÷1024}↑
	ユティリティ専用ユニット※	1968 + ↑{(β + δ) ÷1024}↑
64 ビットモードの場合	シングルサーバ	12172 + ↑{(α + γ) ÷1024}↑
	ユティリティ専用ユニット※	2423 + ↑{(β + δ) ÷1024}↑

注※

ユティリティ専用ユニットを使用しない場合は，計算値をシングルサーバに加算してください。

(2) HiRDB/パラレルサーバの場合

条件		メモリ所要量の計算式（単位：キロバイト）
32 ビットモードの場合	MGR	2255 ＋↑{α÷1024}↑
	入力ファイルがあるサーバマシン	2045 ＋↑{(β ＋ δ）÷1024}↑
	BES※	3762 ＋↑{γ÷1024}↑
64 ビットモードの場合	MGR	2575 ＋↑{α÷1024}↑
	入力ファイルがあるサーバマシン	2406 ＋↑{(β ＋ δ）÷1024}↑
	BES※	8247 ＋↑{γ÷1024}↑

注※

1 サーバマシン内に複数のバックエンドサーバがある場合は，バックエンドサーバの数だけ繰り返し計算（加算）してください。

(3) 計算式で使用する変数

α（単位：バイト）：

{3056 ＋A＋B＋（516×a）＋（572×b）＋（312×c）＋（144×d）＋（8×e）＋（1032×f）＋（44×g）＋（272×h）＋（224×i）＋（44×j）＋（60×k）＋（260×m）＋（56×n）＋（196×p）＋（236×q）＋（744×r）＋（620×s)} ×2

β（単位：バイト）：

{6908 ＋ α ＋（C×t）＋K＋（48×a）＋（22×b）＋（8×e）＋（240×i）＋（48×j）＋（4×k）＋（224×m）＋（47416×t）＋（1032×u）＋（4×v)}

γ（単位：バイト）：

{37700 ＋（α÷2）＋C＋D＋F＋H＋P＋Q＋T＋（80×a）＋（1871×b）＋（120×c）＋（26×g）＋（1532×i）＋（36×j）＋（44×k）＋（1212×m）＋（40×n）＋（344×p）＋（30×q）＋（16×u）＋（88×v）＋（20×w)}

δ（単位：バイト）：

{69436 ＋ α ＋D＋K＋E＋L＋M＋N＋S＋（U×2）＋8＋（48×a）＋（32×a）＋（88×c）＋（4×g）＋（2156×k）＋（24×t）＋（1024×u）＋（4×v）＋（50×y）＋（50×z)}

a：列数

b：抽象データ型の列数

c：コンストラクタ又は逆コンストラクタ関数のパラメタ数

d：コマンドライン及び制御情報ファイルに指定したファイルパス名の数

e：LOB 中間ファイル指定数

f：列単位 LOB ファイル指定数

g：表格納 RD エリア数

h：表の横分割条件数

i：インデクス数

j：インデクス格納 RD エリア数

k：BLOB 型の列数

m：プラグインインデクス数

n：LOB 属性の抽象データ型を格納するユーザ LOB 用 RD エリア数

p：プラグインが提供する関数の数

q：プラグイン提供関数のパラメタ数

r：データ型プラグイン数

s：インデクス型プラグイン数

t：表格納サーバ数

u：使用するコンストラクタ関数のパラメタのうちで BLOB 型のパラメタ数

v：LOB 列を格納するユーザ LOB 用 RD エリア数

w：プラグインインデクスを格納するユーザ LOB 用 RD エリア数

y：BINARY 型の列数
skipdata 制御文で入力データ中の処理対象外とした BINARY データの列数と，実際に表に定義されている列数の合計です。

z：プラグイン提供関数の BINARY 属性のパラメタ数
skipdata 制御文で入力データ中の処理対象外としたプラグイン提供関数の BINARY データの列数と，実際に表に定義されている列数の合計です。

A：コマンドライン上で指定したファイルサイズの合計

B：コマンドライン及び制御情報ファイルに指定したファイルパス名の長さの合計

C：次に示す条件を満たす場合は（pd_utl_buff_size×1024 + 4096）×2，そのほかの場合は 0

- HiRDB/シングルサーバの場合

  ユティリティ専用ユニットを使用している

- HiRDB/パラレルサーバの場合

  source 文に指定したサーバ名と表格納 RD エリアがあるバックエンドサーバ名が異なる pdload コマンド実行時，又は-g オプションを指定した pdrorg コマンド実行時

D：行長

表を構成する全列の定義長の合計です。ただし，BLOB 型や抽象データ型は 8 バイトとします。BINARY 型は pdload の場合は定義長，pdrorg の場合は MIN（定義長，32500）バイトとします。また，FIX 表以外は（a + 1）×4 を加算します。

E：EasyMT のメモリ所要量

-f オプションに easymt を指定した場合に加算します。

F：550×1024 + 1024×1024 + G

-i オプションに c を指定した場合に加算します。

G：256（32 ビットモードの場合）又は 512（64 ビットモードの場合）

H：一括入出力用ローカルバッファの指定値×RD エリアのページ長× J +ランダムアクセス用ローカルバッファの指定値×RD エリアのページ長

-n オプションを指定した場合に加算します。RD エリアのページ長が横分割した RD エリアごとに異なる場合，最大のページ長で計算してください。

J：次に示す表から求めてください。

-n オプションの指定値	表の分割種別				
	非分割表	キーレンジ分割表	ハッシュ分割表		
			リバランスハッシュ（HASHA～HASHF）		非リバランスハッシュ（HASH0～HASH6，HASHZ）
			FIX ハッシュ	フレキシブルハッシュ	
div 指定あり	1	表のサーバ内横分割数	HiRDB/シングルサーバの場合：1024 HiRDB/パラレルサーバの場合：（↑1024÷g↑）×（該当サーバ内の表格納 RD エリア数）	表のサーバ内横分割数	表のサーバ内横分割数
div 指定なし		1			1

K：パラメタ長

抽象データ型に格納する値を生成するコンストラクタ関数の引数の合計長です。ただし，BLOB 型のパラメタは 8 バイトで計算します。

L：次に示す条件をすべて満たす場合は 1 又は 2 の値を加算，そのほかの場合は 0

- source 文に errdata オペランドを指定
- ユティリティ専用ユニットを使用している（HiRDB/シングルサーバの場合），又は source 文に指定したサーバ名と表格納 RD エリアがあるバックエンドサーバ名が異なる（HiRDB/パラレルサーバの場合）
- 表に抽象データ型の列がある又はユニークインデクスを定義している

1. errwork オペランド指定時：errwork オペランドの指定値×1024
2. errwork オペランド省略時：pd_utl_buff_size×1024×3× t

M：UOC のメモリ所要量

UOC を使用した場合に加算します。

N：maxreclen オペランドを指定した場合に次に示す値を加算します。

入力データファイルが拡張 DAT 形式のとき：

maxreclen オペランド指定値×1024

入力データファイルが DAT 形式のとき：

maxreclen オペランド指定値×1024×3

対象となる表が BINARY 型の列を持ち，入力データファイルがバイナリ形式のとき：

次のどちらか小さい方を加算してください。

- maxreclen オペランド指定値×1024
- 変数 D（行長）

上記以外のとき：

0

P：プラグインのメモリ所要量

プラグインが提供する抽象データ型の列がある場合に加算します。プラグインのメモリ所要量については，プラグインのマニュアルを参照してください。

コンストラクタ関数の引数が BLOB 型又は BINARY 型の場合は，1 行に定義されている（全抽象データ型に格納する実際のパラメタ長×2）を加算します。

Q：出力バッファ用メモリ所要量

インデクス作成方法にインデクス一括作成モード，又はインデクス情報出力モードを指定していて，次の条件を満たす場合，2 メガバイトを加算します。

- 表分割数×インデクス定義数＞プロセスのオープン数の上限-576

S：(4×バッファ長＋ 1.1）×1024

バッファ長は次の値になります（値は 32 キロバイト単位に切り上げてください)。

- データベース作成ユティリティ（pdload）の option 文の file_buff_size の指定値
- 上記の指定がない場合は 1024

T：圧縮列がある場合は，圧縮分割サイズ×2 ＋ RD エリアのページ長，圧縮列がない場合は 0

圧縮分割サイズは，全圧縮列中の最大値で計算してください。また，表を横分割した RD エリアごとにページ長が異なるときは，ページ長の最大値で計算してください。

U：入力ファイルの行長（単位：バイト）

DAT 形式の場合：maxreclen オペランドの指定値×1024（maxreclen オペランドの指定がないときは 32768）

固定長形式の場合：入力ファイルの行長

バイナリ形式の場合：0

18.2.4 データベース再編成ユティリティ（pdrorg）実行時のメモリ所要量

データベース再編成ユティリティ（pdrorg）実行時のメモリ所要量は，次に示す計算式で求めます。計算式の変数については，「計算式で使用する変数」を参照してください。

(1) HiRDB/シングルサーバの場合

条件		メモリ所要量の計算式（単位：キロバイト）
32 ビットモードの場合	シングルサーバ	7536 ＋ ↑{(α ＋ γ) ÷1024}↑
	ユティリティ専用ユニット※	2119 ＋ ↑{(β ＋ δ) ÷1024}↑
64 ビットモードの場合	シングルサーバ	13196 ＋ ↑{(α ＋ γ) ÷1024}↑
	ユティリティ専用ユニット※	2593 ＋ ↑{(β ＋ δ) ÷1024}↑

注※

ユティリティ専用ユニットを使用しない場合は，計算値をシングルサーバに加算してください。

(2) HiRDB/パラレルサーバの場合

条件			メモリ所要量の計算式（単位：キロバイト）
32 ビットモードの場合	MGR		2200 ＋ ↑{α ÷1024}↑
	-g オプションなし	DS	1940 ＋ ↑{β ÷1024}↑
		BES※1※2	4976 ＋ ↑{(γ ＋ δ) ÷1024}↑
	-g オプションあり	アンロードデータファイルがあるサーバマシン	1940 ＋ ↑{(β ＋ δ) ÷1024}↑
		BES※2	4976 ＋ ↑{γ ÷1024}↑
64 ビットモードの場合	MGR		2569 ＋ ↑{α ÷1024}↑

条件			メモリ所要量の計算式（単位：キロバイト）
	-g オプションなし	DS	2294 ＋ ↑{β ÷1024}↑
		BES※1※2	9441 ＋ ↑{(γ ＋ δ) ÷1024}↑
	-g オプションあり	アンロードデータファイルがあるサーバマシン	2294 ＋ ↑{(β ＋ δ) ÷1024}↑
		BES※2	9441 ＋ ↑{γ ÷1024}↑

注※1

ディクショナリ表を再編成する場合は，計算値をディクショナリサーバがあるサーバマシンに加算してください。

注※2

1 サーバマシン内に複数のバックエンドサーバがある場合は，バックエンドサーバの数だけ繰り返し計算（加算）してください。

(3) 計算式で使用する変数

α（単位：バイト）：

{2592 ＋A＋B＋（116×a）＋（260×b）＋（6×c）＋（272×d）＋（44×g）＋（272×h）＋（224×i）＋（44×j）＋（60×k）＋（260×m）＋（56×n）＋（196×p）＋（236×q）＋（744×r）＋（620×s）＋（24×t)} ×2

β（単位：バイト）：

{40940 ＋ α ＋（C×t）＋（D×t）＋（136×a）＋（56×g）＋（2200×j）＋（4×k）＋（548×t)}

γ（単位：バイト）：

{101140 ＋（α ÷2）＋C＋（D×2）＋F＋H＋P＋Q＋T＋64010 ＋（48×a）＋（128×a）＋（1949×b）＋（120×c）＋（154×g）＋（336×i）＋（216×j）＋（32056×k）＋（1212×m）＋（131224×n）＋（344×p）＋（30×q）＋（20×w)}

δ（単位：バイト）：

{33104 ＋ α ＋（K×2）＋E＋S＋72 ＋（48×a）＋（204×a）＋（688×b）＋（306×c）＋（44×g）＋（272×h）＋（224×i）＋（44×j）＋（56×n）＋（716×r）＋（152×v)}

注

- 変数については，「データベース作成ユティリティ（pdload）実行時のメモリ所要量」の「計算式で使用する変数」で説明しています。
- ディクショナリ再編成又はスキーマ単位の再編成で 1 回のコマンドで複数の表を処理する場合，変数 a ～ z は処理対象すべての表の合計で計算してください。

18.2.5　データベース構成変更ユティリティ（pdmod）実行時のメモリ所要量

データベース構成変更ユティリティ（pdmod）実行時のメモリ所要量は，次に示す計算式で求めます。

(1)　HiRDB/シングルサーバの場合

条件	メモリ所要量の計算式（単位：キロバイト）
32 ビットモードの場合	↑{ 4×a＋56016×b＋53016×c＋2440×d＋1724×e ＋（94008×f）÷500＋（4008×g）÷1000＋440720＋h＋i＋j＋k }÷1024↑＋9.8
64 ビットモードの場合	↑{ 4×a＋56024×b＋53024×c＋3040×d＋1736×e ＋（100016×f）÷500＋（4012×g）÷1000＋450720＋h＋i＋j＋k }÷1024↑＋9.8

a：pd_max_rdarea_no の値

b：initialize rdarea 文実行時の自 RD エリア内のインデクス数＋他 RD エリアのインデクス数

c：initialize rdarea 文実行時の LOB 列の総数

d：initialize rdarea 文実行時の LOB 属性の抽象データ型の総数

e：initialize rdarea 文実行時のプラグイン列とプラグインインデクスの総数

f：initialize rdarea 文実行時の抽象データ型の総数

g：initialize rdarea 文実行時の自 RD エリア格納表の ASSIGN LIST の総数

h：8×a＋30720

alter HiRDB mode to parallel 文で HiRDB/シングルサーバから HiRDB/パラレルサーバへ移行する場合に加算します。

i：46744

create rdarea 文でデータディクショナリ LOB 用 RD エリアを追加する場合に加算します。

j：88064

alter system 文でディクショナリ表の参照権限を変更する場合に加算します。

k：54732

alter system 文でディクショナリ表の列属性 MCHAR にする場合に加算します。

(2) HiRDB/パラレルサーバの場合

条件		メモリ所要量の計算式（単位：キロバイト）
32 ビットモードの場合	DS	↑ { 4×a + 56016×b + 53016×c + 2440×d + 1724×e +（94008×f）÷500 +（4008×g）÷1000 + 440720 + h + i + j + 108428×m } ÷1024↑
	BES	↑（4×a + 252755 + k）÷1024↑
	FES	0.52
	MGR	9.8
64 ビットモードの場合	DS	↑{ 4×a + 56024×b + 53024×c + 3040×d + 1736×e +（100016×f）÷500 +（4012×g）÷1000 + 450720 + h + i + j + 108432×m }÷1024↑
	BES	↑（4×a + 261112 + k）÷1024↑
	FES	0.53
	MGR	9.8

a：pd_max_rdarea_no の値

b：initialize rdarea 文実行時の自 RD エリア内のインデクス数＋他 RD エリアのインデクス数

c：initialize rdarea 文実行時の LOB 列の総数

d：initialize rdarea 文実行時の LOB 属性の抽象データ型の総数

e：initialize rdarea 文実行時のプラグイン列とプラグインインデクスの総数

f：initialize rdarea 文実行時の抽象データ型の総数

g：initialize rdarea 文実行時の自 RD エリア格納表の ASSIGN LIST の総数

h：46744

create rdarea 文でデータディクショナリ LOB 用 RD エリアを追加する場合に加算します。

i：88064

alter system 文でディクショナリ表の参照権限を変更する場合に加算します。

j：54732

alter system 文でディクショナリ表の列属性 MCHAR にする場合に加算します。

k：2200

initialize rdarea 文を実行する場合に加算します。

m：move rdarea 文を実行する場合に次に示す計算式を加算します。

move rdarea 文を実行しない場合 0 になります。

↑（192×移動対象 RD エリア数＋ 160×移動対象 RD エリアの総 HiRDB ファイル数＋ 136×移動対象 RD エリアのうちレプリカ RD エリアの総 HiRDB ファイル数＋ 8×移動対象 RD エリアに格納されている表の総数＋ 8×移動対象 RD エリアに格納されているインデクスの総数＋ 8×移動対象 RD エリアに格納されている LOB 列の総数）÷102400↑

18.2.6 統計解析ユティリティ（pdstedit）実行時のメモリ所要量

統計解析ユティリティ（pdstedit）実行時のメモリ所要量は，次に示す計算式で求めます。

条件	メモリ所要量の計算式（単位：キロバイト）
HiRDB/シングルサーバの場合	16384※＋ 1.5×ホスト数×HiRDB サーバ数×UAP 数
HiRDB/パラレルサーバの場合	

注※

64 ビットモードの場合は 18432 です。

18.2.7 データベース状態解析ユティリティ（pddbst）実行時のメモリ所要量

データベース状態解析ユティリティ（pddbst）実行時のメモリ所要量は，次に示す計算式で求めます。計算式の変数については「計算式で使用する変数」を参照してください。

(1) HiRDB/シングルサーバの場合

条件	メモリ所要量の計算式（単位：キロバイト）
32 ビットモードの場合	9425 ＋↑｛(α ＋ β）÷1024｝↑
64 ビットモードの場合	15311 ＋↑｛(α ＋ β）÷1024｝↑

(2) HiRDB/パラレルサーバの場合

条件		メモリ所要量の計算式（単位：キロバイト）
32 ビットモードの場合	MGR	5117 ＋ ↑{ α ÷ 1024 }↑
	DS	4177 ＋ ↑{ β ÷ 1024 }↑
	BES※	

条件		メモリ所要量の計算式（単位：キロバイト）
64 ビットモードの場合	MGR	5645 ＋↑｛α÷1024｝↑
	DS	8329 ＋↑｛β÷1024｝↑
	BES※	

注※

解析対象の表やインデクスが格納されているバックエンドサーバごとに計算する必要があります。

(3) 計算式で使用する変数

α（単位：バイト）：

{28000 ＋ A ＋（10592×a）＋（10592×b）＋（10592×c）＋（2264×d）＋（848×e）＋（272×f）＋（432×g）＋（304×h)}

β（単位：バイト）：

{100000 ＋（1024×d）＋（2784×i）＋（2784×j）＋（2784×k)}

注

- そのほかの変数については，「データベース作成ユティリティ（pdload）実行時のメモリ所要量」の「計算式で使用する変数」で説明しています。

18.2.8 最適化情報収集ユティリティ（pdgetcst）実行時のメモリ所要量

最適化情報収集ユティリティ（pdgetcst）実行時のメモリ所要量は，次に示す計算式で求めます。

条件		メモリ所要量の計算式（単位：キロバイト）
HiRDB/シングルサーバの場合		6060 ＋ 0.1×表格納 RD エリア数＋ 0.07×インデクス格納 RD エリア数＋ 1.0×インデクス数＋ 0.04×スキーマ内表数＋ 1.0×サーバ数＋ 11×列数
HiRDB/パラレルサーバの場合	BES	1813 ＋ 0.06×表格納 RD エリア数＋ 0.05×インデクス格納 RD エリア数＋ 0.03×インデクス数
	MGR	3336 ＋ 0.1×表格納 RD エリア数＋ 0.07×インデクス格納 RD エリア数＋ 1.0×インデクス数＋ 0.04×スキーマ内表数＋ 1.0×サーバ数＋ 11×列数
	DS	16402

注

上記のほかに次の SQL が使用するメモリ所要量を加算します。

```
SELECT 内部情報※,インデクス第一構成列名 FROM 認可識別子.表識別子
    ORDER BY インデクス第一構成列名 WITHOUT LOCK NOWAIT;
SELECT FLOAT (COUNT(*)) FROM 認可識別子.表識別子 WITHOUT LOCK NOWAIT;
SELECT FLOAT (COUNT(インデクス第一構成列名))
```

```
    FROM 認可識別子.表識別子 WITHOUT LOCK NOWAIT;
ALTER TABLE 認可識別子.表識別子 CHANGE LOCK ROW;
```

注※

内部情報は 12 バイトです。したがって，インデクス第一構成列のほかに 12 バイトの文字列（CHAR(12)）を検索する SQL を発行したものとして見積もってください。

SQL が使用するメモリ所要量については，次の箇所を参照してください。

- HiRDB/シングルサーバの場合
  「SQL 実行時に必要なメモリ所要量の計算式」
  「SQL 前処理時に必要なメモリ所要量の計算式」
- HiRDB/パラレルサーバの場合
  「SQL 実行時に必要なメモリ所要量の計算式」
  「SQL 前処理時に必要なメモリ所要量の計算式」

18.2.9 データベース複写ユティリティ（pdcopy）実行時のメモリ所要量

データベース複写ユティリティ（pdcopy）実行時のメモリ所要量は，次に示す計算式で求めます。

(1) HiRDB/シングルサーバの場合

条件	メモリ所要量の計算式（単位：キロバイト）
シングルサーバ	88 ＋バックアップファイル数×2×MAX（32，pd_utl_buff_size の値） ＋バックアップファイル数×｛(バックアップ対象 RD エリア数＋ 9）÷10｝×6 ＋｛(全バックアップ対象 RD エリア構成ファイル数＋ 25）÷16｝×8 ＋ 100 ＋ 49 ＋バックアップファイル数×2×MAX（32，pd_utl_buff_size の値）＋ 64 ＋バックアップファイル数×｛(バックアップ対象 RD エリア数＋ 9）÷10｝×6 ＋｛(全バックアップ対象 RD エリア構成ファイル数＋ 25）÷16｝×8 ＋ 100 **●バックアップファイルがこのサーバマシンにある場合に加算** ＋ 63 ＋バックアップファイル数×（2×MAX（32，pd_utl_buff_size の値）×2 ＋バックアップファイル数×｛(バックアップ対象 RD エリア数＋ 9）÷10｝×6 ＋｛(全バックアップ対象 RD エリア構成ファイル数＋ 25）÷16｝×8 ＋ 100 ＋ 1024 **●差分バックアップを取得する場合に加算** ＋ 32×2 ＋↑（512 ＋ 128×バックアップ対象 RD エリア数）÷32768↑×32 ＋↑（256 ＋ 128×バックアップ対象 RD エリア数＋ a ＋ 8×b）÷32768↑×32×2
ユティリティ専用ユニット	63 ＋バックアップファイル数×（2×MAX（32，pd_utl_buff_size の値）×2 ＋バックアップファイル数×｛(バックアップ対象 RD エリア数＋ 9）÷10｝×6 ＋｛(全バックアップ対象 RD エリア構成ファイル数＋ 25）÷16｝×8 ＋ 100 ＋ 1024

a：-b オプションに指定するバックアップファイルの名称長（バイト）

複数のバックアップファイルを指定する場合は，合計の名称長になります。

b：-b オプションに指定するバックアップファイルの数

(2) HiRDB/パラレルサーバの場合

条件	メモリ所要量の計算式（単位：キロバイト）
MGR	88 ＋バックアップファイル数×2×MAX（32，pd_utl_buff_size の値） ＋バックアップファイル数×｛(バックアップ対象 RD エリア数＋ 9）÷10｝×6 ＋｛(全バックアップ対象 RD エリア構成ファイル数＋ 25）÷16｝×8 ＋ 100 ●差分バックアップを取得する場合に加算 ＋ 32×2 ＋↑（512 ＋ 128×バックアップ対象 RD エリア数）÷32768↑×32 ＋↑（256 ＋ 128×バックアップ対象 RD エリア数＋ a ＋ 8×b）÷32768↑×32×2
DS BES	49 ＋バックアップファイル数×2×MAX（32，pd_utl_buff_size の値）＋ 64 ＋バックアップファイル数×｛(バックアップ対象 RD エリア数＋ 9）÷10｝×6 ＋｛(全バックアップ対象 RD エリア構成ファイル数＋ 25）÷16｝×8 ＋ 100
バックアップファイルがあるサーバマシン	63 ＋バックアップファイル数×2×MAX（32，pd_utl_buff_size の値） ×(バックアップ対象サーバ数＋ 1） ＋バックアップファイル数×｛(バックアップ対象 RD エリア数＋ 9）÷10｝×6 ＋｛(全バックアップ対象 RD エリア構成ファイル数＋ 25）÷16｝×8 ＋ 100 ＋ 1024

a：-b オプションに指定するバックアップファイルの名称長（バイト）
　複数のバックアップファイルを指定する場合は，合計の名称長になります。

b：-b オプションに指定するバックアップファイルの数

18.2.10 データベース回復ユティリティ（pdrstr）実行時のメモリ所要量

データベース回復ユティリティ（pdrstr）実行時のメモリ所要量は，次に示す計算式で求めます。

(1) HiRDB/シングルサーバの場合

条件	メモリ所要量の計算式（単位：キロバイト）
シングルサーバ	65 ＋｛(回復対象 RD エリア数＋ 9）÷10｝×6 ＋｛(回復対象 RD エリアの構成ファイル数＋ 25）÷16｝×8 ＋ 50 ＋ 98 ＋ 2×MAX（32，pd_utl_buff_size の値） ＋｛(回復対象 RD エリア数＋ 9）÷10｝×6 ＋ c ＋｛(回復対象 RD エリアの構成ファイル数＋ 25）÷16｝×8 ＋ 100 ＋｛(回復対象 RD エリア数＋ 99）÷100｝×5 ●バックアップファイルがこのサーバマシンにある場合に加算 ＋ 100 ＋ 2×MAX（32，pd_utl_buff_size の値） ＋｛(回復対象 RD エリア数＋ 9）÷10｝×6 ＋｛(回復対象 RD エリアの構成ファイル数＋ 25）÷16｝×8 ＋ 100 ＋ 1024

条件	メモリ所要量の計算式（単位：キロバイト）
	●アンロードログファイル，又はシステムログファイルを入力する場合に加算 ＋ 57 ＋ 2×MAX（32，pd_utl_buff_size の値） ＋ {(回復対象 RD エリア数＋ 9）÷10｝×6 ＋ 64 ＋ {(回復対象 RD エリアの構成ファイル数＋ 25）÷16｝×8 ＋ 100 ＋ d ＋ 0.6×回復対象 RD エリア数＋ソート用ワークバッファサイズ（-y オプションの値） **●差分バックアップを使用した回復をする場合に加算** ＋ 32×2 ＋↑（512 ＋ 128×バックアップ対象 RD エリア数）÷32768↑×32 ＋↑（256 ＋ 128×バックアップ対象 RD エリア数＋ a ＋ 8×b）÷32768↑×32 ＋↑（32×差分バックアップの継続回数）÷1024↑
ユティリティ専用ユニット	100 ＋ 2×MAX（32，pd_utl_buff_size の値） ＋ {(回復対象 RD エリア数＋ 9）÷10｝×6 ＋ {(回復対象 RD エリアの構成ファイル数＋ 25）÷16｝×8 ＋ 100 ＋ 1024

a：-b オプションに指定するバックアップファイルの名称長（バイト）

複数のバックアップファイルを指定する場合は，合計の名称長になります。

b：-b オプションに指定するバックアップファイルの数

c：書き込みバッファサイズを指定している場合は MAX（64，書き込みバッファサイズ），指定していない場合は 60

書き込みバッファサイズは-Y オプションの指定値になります。

d：

- 32 ビットモードの場合：
  640 ＋ 8×↑最大同時実行トランザクション数÷100↑
  ＋ 5×↑回復対象 RD エリア数÷100↑
  ＋回復対象 RD エリアの最大ページサイズ×54
  ＋ 9×ロールバック対象トランザクション数
  ＋ 0.02×回復対象 RD エリアの構成ファイル数
  ＋↑（304 ＋ 36 ＋ 4×（回復対象 RD エリア数－1）
  ＋ 352 ＋ 304×（回復対象 RD エリア数－1）
  ＋ 96 ＋ 4×（回復対象 RD エリア数－1）
  ＋ 384 ＋ 320×（回復対象 RD エリア数－1）＋ 16）÷1024↑
- 64 ビットモードの場合：
  640 ＋ 11×↑最大同時実行トランザクション数÷100↑
  ＋ 6×↑回復対象 RD エリア数÷100↑
  ＋回復対象 RD エリアの最大ページサイズ×54
  ＋ 9×ロールバック対象トランザクション数
  ＋ 0.03×回復対象 RD エリアの構成ファイル数

＋↑（304 ＋ 40 ＋ 8×（回復対象 RD エリア数－1）
＋ 400 ＋ 336×（回復対象 RD エリア数－1）
＋ 168 ＋ 8×（回復対象 RD エリア数－1）
＋ 408 ＋ 336×（回復対象 RD エリア数－1）＋ 16）÷1024↑

(2) HiRDB/パラレルサーバの場合

条件	メモリ所要量の計算式（単位：キロバイト）
MGR	65 ＋ {(回復対象 RD エリア数＋ 9）÷10} ×6 ＋ {(回復対象 RD エリアの構成ファイル数＋ 25）÷16} ×8 ＋ 50 ●差分バックアップを使用した回復をする場合に加算 ＋ 32×2 ＋↑（512 ＋ 128×バックアップ対象 RD エリア数）÷32768↑×32 ＋↑（256 ＋ 128×バックアップ対象 RD エリア数＋ a ＋ 8×b）÷32768↑×32 ＋↑（32×差分バックアップの継続回数）÷1024↑
DS	35 ＋ 2×MAX（32，pd_utl_buff_size の値）＋ 100 ＋ 98 ＋ 2×MAX（32，pd_utl_buff_size の値） ＋ {(回復対象 RD エリア数＋ 9）÷10} ×6 ＋ c ＋ {(回復対象 RD エリアの構成ファイル数＋ 25）÷16} ×8 ＋ 100 ＋ {(回復対象 RD エリア数＋ 99）÷100} ×5 ●アンロードログファイル，又はシステムログファイルを入力する場合に加算 ＋ 57 ＋ 2×MAX（32，pd_utl_buff_size の値） ＋ {(回復対象 RD エリア数＋ 9）÷10} ×6 ＋ 64 ＋ {(回復対象 RD エリアの構成ファイル数＋ 25）÷16} ×8 ＋ 100 ＋ d ＋ 0.6×回復対象 RD エリア数＋ソート用ワークバッファサイズ（-y オプションの値）
BES	98 ＋ 2×MAX（32，pd_utl_buff_size の値） ＋ {(回復対象 RD エリア数＋ 9）÷10} ×6 ＋ c ＋ {(回復対象 RD エリアの構成ファイル数＋ 25）÷16} ×8 ＋ 100 ＋ {(回復対象 RD エリア数＋ 99）÷100} ×5 ●アンロードログファイル，又はシステムログファイルを入力する場合に加算 ＋ 57 ＋ 2×MAX（32，pd_utl_buff_size の値） ＋ {(回復対象 RD エリア数＋ 9）÷10} ×6 ＋ 64 ＋ {(回復対象 RD エリアの構成ファイル数＋ 25）÷16} ×8 ＋ 100 ＋ d ＋ 0.6×回復対象 RD エリア数＋ソート用ワークバッファサイズ(-y オプションの値）
バックアップファイルがあるサーバマシン	100 ＋ 2×MAX（32，pd_utl_buff_size）×回復対象サーバ数 ＋ {(回復対象 RD エリア数＋ 9）÷10} ×6 ＋ {(回復対象 RD エリア構成ファイル数＋ 25）÷16} ×8 ＋ 100 ＋ 1024

a：-b オプションに指定するバックアップファイルの名称長（バイト）
　複数のバックアップファイルを指定する場合は，合計の名称長になります。

b：-b オプションに指定するバックアップファイルの数

c：書き込みバッファサイズを指定している場合はMAX（64，書き込みバッファサイズ），指定していない場合は60

書き込みバッファサイズは-Yオプションの指定値になります。

d：

- 32ビットモードの場合：

  640 ＋ 8×↑最大同時実行トランザクション数÷100↑
  ＋ 5×↑回復対象RDエリア数÷100↑
  ＋回復対象RDエリアの最大ページサイズ×54
  ＋ 9×ロールバック対象トランザクション数
  ＋ 0.02×回復対象RDエリアの構成ファイル数
  ＋↑（304 ＋ 36 ＋ 4×（回復対象RDエリア数－1）
  　＋ 352 ＋ 304×（回復対象RDエリア数－1）
  　＋ 96 ＋ 4×（回復対象RDエリア数－1）
  　＋ 384 ＋ 320×（回復対象RDエリア数－1）
  　＋ 16 ＋ 32×回復対象RDエリア数）÷1024↑

- 64ビットモードの場合：

  640 ＋ 11×↑最大同時実行トランザクション数÷100↑
  ＋ 6×↑回復対象RDエリア数÷100↑
  ＋回復対象RDエリアの最大ページサイズ×54
  ＋ 9×ロールバック対象トランザクション数
  ＋ 0.03×回復対象RDエリアの構成ファイル数
  ＋↑（304 ＋ 40 ＋ 8×（回復対象RDエリア数－1）
  　＋ 400 ＋ 336×（回復対象RDエリア数－1）
  　＋ 168 ＋ 8×（回復対象RDエリア数－1）
  　＋ 408 ＋ 336×（回復対象RDエリア数－1）
  　＋ 16 ＋ 48×回復対象RDエリア数）÷1024↑

18.2.11 ディクショナリ搬出入ユティリティ（pdexp）実行時のメモリ所要量

ディクショナリ搬出入ユティリティ（pdexp）実行時のメモリ所要量は，次に示す計算式で求めます。

(1) HiRDB/シングルサーバの場合

条件		メモリ所要量の計算式（単位：キロバイト）
32ビットモード	SDS	1307 ＋ 6.7×↑CTL÷100↑＋ 0.07×CTL

条件		メモリ所要量の計算式（単位：キロバイト）
	搬出ファイルのあるホスト	4499 ＋ 0.07×CTL ＋ 0.11×CHK ＋ 1.5×FKY ＋ CSZ ＋ 0.5×CMN ＋ 0.01×（CHK ＋ FKY ＋ CMN ＋ DIV ＋ 10）＋ 0.6×DIV ＋ DEF ＋ 0.06×LOB ＋ 0.2×TBL ＋ SQL ＋ 8791
64 ビットモード	SDS	1494 ＋ 6.7×↑CTL÷100↑＋ 0.07×CTL
	搬出ファイルのあるホスト	4582 ＋ 0.07×CTL ＋ 0.11×CHK ＋ 1.5×FKY ＋ CSZ ＋ 0.5×CMN ＋ 0.01×（CHK ＋ FKY ＋ CMN ＋ DIV ＋ 10）＋ 0.6×DIV ＋ DEF ＋ 0.06×LOB ＋ 0.2×TBL ＋ SQL ＋ 8791

(2) HiRDB/パラレルサーバの場合

条件		メモリ所要量の計算式（単位：キロバイト）
32 ビットモード	MGR	1714 ＋ 6.7×↑CTL÷100↑＋ 0.07×CTL
	搬出ファイルのあるホスト	4907 ＋ 0.07×CTL ＋ 0.11×CHK ＋ 1.5×FKY ＋ CSZ ＋ 0.5×CMN ＋ 0.01×（CHK ＋ FKY ＋ CMN ＋ DIV ＋ 10）＋ 0.6×DIV ＋ DEF ＋ 0.06×LOB ＋ 0.2×TBL ＋ SQL ＋ 8791
64 ビットモード	MGR	2016 ＋ 6.7×↑CTL÷100↑＋ 0.07×CTL
	搬出ファイルのあるホスト	5293 ＋ 0.07×CTL ＋ 0.11×CHK ＋ 1.5×FKY ＋ CSZ ＋ 0.5×CMN ＋ 0.01×（CHK ＋ FKY ＋ CMN ＋ DIV ＋ 10）＋ 0.6×DIV ＋ DEF ＋ 0.06×LOB ＋ 0.2×TBL ＋ SQL ＋ 8791

(3) 計算式で使用する変数

CMN：列数

CHK：検査制約数

CSZ：検査制約の検索条件の合計サイズ（SQL_TABLES 表の CHK_SOURCE_LEN の値）（単位：バイト）

CTL：制御文ファイルに指定された制御数

DEF：DEFAULT 句の最大定義長（単位：バイト）

DIV：分割条件数

FKY：外部キーの数

LOB：LOB 格納用ＲＤエリア数

SQL：次の SQL 文が使用するメモリ所要量

- 表の搬出入の場合：CREATE TABLE 文
- プロシジャの搬出入の場合：CREATE PROCEDURE 文
- トリガの搬出入の場合：CREATE TRIGGER 文

これらのメモリ所要量については，「SQL 実行時に必要なメモリ所要量の計算式」，及び「SQL 前処理時に必要なメモリ所要量の計算式」を参照してください。

TBL：実際に搬出入する表，プロシジャ，トリガ数（プロシジャ，トリガの場合は CTL と同じ）

18.2.12　アクセスパス表示ユティリティ（pdvwopt）実行時のメモリ所要量

アクセスパス表示ユティリティ（pdvwopt）実行時のメモリ所要量は，次に示す計算式で求めます。

条件		メモリ所要量の計算式（単位：キロバイト）
HiRDB/シングルサーバの場合		a Σ b_i×0.7 ＋ 200 i=1
HiRDB/パラレルサーバの場合	FES	

a：SQL 中の問合せ数

b_i：問合せ中の表数

18.2.13　リバランスユティリティ（pdrbal）実行時のメモリ所要量

リバランスユティリティ（pdrbal）実行時のメモリ所要量は，次に示す計算式で求めます。

(1)　HiRDB/シングルサーバの場合

メモリ所要量の計算式（単位：キロバイト）
8756※3 ＋ 536 ＋ 0.02×列数＋ 0.2×移動先 RD エリア数 ＋ 1.7×移動元 RD エリア数＋ 0.26×移動先 RD エリア数×インデクス数 ＋（0.09 ＋↑平均 index 文ファイル長÷1024↑）×index 文数 ＋（0.02 ＋↑平均ディレクトリ長÷1024↑）×（idxwork 文数＋ sort 文数） ＋↑（制御情報ファイル長＋実行結果ファイル長）÷1024↑ ＋ 0.05×列数＋ 0.05×RD エリア数＋ 0.15×インデクス数 ＋ 0.05×インデクス格納 RD エリア数 ＋ 550×1024 ＋ソートワークサイズ※1 **●-n オプションを指定した場合** ＋ RD エリアのページ長※2 ×一括入出力バッファ面数× y **●対象表に LOB 列がある場合** ＋ 64 ＋ 0.01×LOB 列数＋ 0.18×移動先 RD エリア数 ＋ 0.09×移動元 RD エリア数＋ 0.08×LOB 格納 RD エリア数 **●対象表に BINARY 列がある場合** ＋ 33×（BINARY 列数×移動先 RD エリア数×移動元 RD エリア数） ＋ BINARY 列の場合に使用するバッファ長※5

メモリ所要量の計算式（単位：キロバイト）

●対象表にプラグインが提供する抽象データ型がある場合
＋ 40 ＋（0.27 ＋ 2×抽象データ型長）×抽象データ型の列数
＋ 0.3×unld_func 指定数
＋（128 ＋ 0.11×LOB 属性数＋ 0.1×unld_func 指定関数数
＋ 0.07×抽象データ型属性数）×抽象データ型の列数
＋（33×BINARY 属性数×移動先 RD エリア数×移動元 RD エリア数）×2
＋ 0.01×プラグインインデクス数＋ 0.19×unld_func 文数
＋（↑平均 unld_func 文長÷1024↑×unld_func 文数）
＋（↑平均 reld_func 文長÷1024↑×reld_func 文数）
＋抽象データ型列数×1 ＋（LOB 属性数×0.05）×RD エリア数
＋データ型プラグイン数×10 ＋プラグインインデクス数×10
＋プラグインのメモリ所要量
●インデクス作成方法に，インデクス一括作成モード又はインデクス情報出力モードを指定していて，次の条件を満たす場合
・表分割数×インデクス定義数＞プロセスのオープン数の上限-576
＋ 2048
●圧縮列がある場合
＋圧縮分割サイズ※4×z ＋ RD エリアのページ長※2

y：次に示すどちらかの値を代入してください。

- FIX ハッシュ分割表にリバランス機能を使用した場合
  （↑1024÷表全体の格納 RD エリア数↑）×該当サーバ内の表格納 RD エリア数
- そのほかの場合
  1

z：次に示すどちらかの値を代入してください。

- 占有モード（-k exclusive）の場合：2
- 共有モード（-k share）の場合：1

注※1

インデクスの一括作成時（-i c 又は省略）に加算します。

注※2

表を横分割した RD エリアごとにページ長が異なる場合は，ページ長の最大値で計算してください。

注※3

64 ビットモードの場合は 9764 です。

注※4

圧縮分割サイズは，全圧縮列中の最大値で計算してください。

注※5

BINARY 列の場合に使用するバッファ長は，次の値を使用してください。

- 占有モードの場合

0

- 共用モードの場合

  n
  Σ（定義長 i）
  i=1
  n：BINARY 列数

(2) HiRDB/パラレルサーバの場合

条件	メモリ所要量の計算式（単位：キロバイト）
MGR	1498※3＋2＋0.05×列数＋0.05×RD エリア数＋0.15×インデクス数 ＋0.05×インデクス格納 RD エリア数 ＋（0.09＋↑平均 index 文ファイル長÷1024↑）×index 文数 ＋（0.02＋↑平均ディレクトリ長÷1024↑）×（idxwork 文数＋sort 文数） ＋↑(制御情報ファイル長＋実行結果ファイル長）÷1024↑ **●対象表に LOB 列がある場合** ＋0.08×LOB 格納 RD エリア数 **●対象表にプラグインが提供する抽象データ型がある場合** ＋0.19×unld_func 文数 ＋（↑平均 unld_func 文長÷1024↑×unld_func 文数） ＋（↑平均 reld_func 文長÷1024↑×reld_func 文数）
DS	1455※4＋32＋0.33×移動先 BES 数＋0.3×移動元 BES 数 ＋0.2×移動先 RD エリア数＋0.22×移動元 RD エリア数＋0.34×FES 数 ＋（0.09＋↑平均 index 文ファイル長÷1024↑）×index 文数 ＋（0.02＋↑平均ディレクトリ長÷1024↑）×（idxwork 文数＋sort 文数）＋0.05×列数＋0.05×RD エリア数＋0.15×インデクス数 ＋0.05×インデクス格納 RD エリア数 **●対象表に LOB 列がある場合** ＋0.08×LOB 格納 RD エリア数 **●対象表に BINARY 列がある場合** ＋33×（BINARY 列数×移動先 RD エリア数×移動元 RD エリア数） **●対象表にプラグインが提供する抽象データ型がある場合** ＋0.01×移動先 BES 数＋0.19×unld_func 文数 ＋（↑平均 unld_func 文長÷1024↑×unld_func 文数） ＋（↑平均 reld_func 文長÷1024↑×reld_func 文数） ＋抽象データ型列数×1＋（LOB 属性数×0.05）×RD エリア数 ＋データ型プラグイン数×10＋プラグインインデクス数×10
BES	6601※5＋50＋（517＋0.01×列数）×移動先 BES 数 ＋（33＋0.01×列数）×移動元 BES 数＋0.2×移動先 RD エリア数 ＋1.7×移動元 RD エリア数＋0.01×列数 ＋0.26×移動先 RD エリア数×インデクス数 ＋（0.09＋↑平均 index 文ファイル長÷1024↑）×index 文数

条件	メモリ所要量の計算式（単位：キロバイト）
	＋（0.02 ＋↑平均ディレクトリ長÷1024↑）×（idxwork 文数＋ sort 文数） ＋ 0.05×列数＋ 0.05×RD エリア数＋ 0.15×インデクス数 ＋ 0.05×インデクス格納 RD エリア数 ＋ 550×1024 ＋ソートワークサイズ※1 **●-n オプションを指定した場合** ＋ RD エリアのページ長※2 ×一括入出力バッファ面数× y **●対象表に LOB 列がある場合** ＋ 32 ＋ 0.01×LOB 列数＋（32 ＋ 0.01×LOB 列数）×移動先 BES 数 ＋ 0.18×移動先 RD エリア数＋ 0.1×移動元 RD エリア数 ＋ 0.08×LOB 格納 RD エリア数 **●対象表に BINARY 列がある場合** ＋ 33×（BINARY 列数×移動先 RD エリア数×移動元 RD エリア数） ＋ BINARY 列の場合に使用するバッファ長※7 **●対象表にプラグインが提供する抽象データ型がある場合** ＋ 40 ＋（0.27 ＋ 2×抽象データ型長）×抽象データ型の列数 ＋ 0.3×unld_func 指定数 ＋ {(64 ＋ 0.05×LOB 属性数）×抽象データ型列数} ×移動先 BES 数 ＋ {(64 ＋ 0.01×unld_func 指定関数数＋ 0.07×抽象データ型属性数 ＋ 0.05×LOB 属性数）×抽象データ型列数} ×移動元 BES 数 ＋（33×BINARY 属性数×移動先 RD エリア数×移動元 RD エリア数）×2 ＋ 0.01×プラグインインデクス数＋ 0.19×unld_func 文数 ＋（↑平均 unld_func 文長÷1024↑×unld_func 文数） ＋（↑平均 reld_func 文長÷1024↑×reld_func 文数） ＋抽象データ型列数×1 ＋（LOB 属性数×0.05）×RD エリア数 ＋データ型プラグイン数×10 ＋プラグインインデクス数×10 ＋プラグインのメモリ所要量 **●インデクス作成方法に，インデクス一括作成モード又はインデクス情報出力モードを指定していて，次の条件を満たす場合** **・表分割数×インデクス定義数＞プロセスのオープン数の上限-576** ＋ 2048 **●圧縮列がある場合** ＋圧縮分割サイズ※6 ×z ＋ RD エリアのページ長※2

y：次に示すどちらかの値を代入してください。

- FIX ハッシュ分割表にリバランス機能を使用した場合
  （↑1024÷表全体の格納 RD エリア数↑）×該当サーバ内の表格納 RD エリア数
- そのほかの場合
  1

z：次に示すどちらかの値を代入してください。

- 占有モード（-k exclusive）の場合：2

- 共有モード（-k share）の場合：1

注※1

インデクスの一括作成時（-i c 又は省略）に加算します。

注※2

表を横分割した RD エリアごとにページ長が異なる場合は，ページ長の最大値で計算してください。

注※3

64 ビットモードの場合は 1790 です。

注※4

64 ビットモードの場合は 1671 です。

注※5

64 ビットモードの場合は 6908 です。

注※6

圧縮分割サイズは，全圧縮列中の最大値で計算してください。

注※7

BINARY 列の場合に使用するバッファ長は，次の値を使用してください。

- 占有モードの場合

  $$\left(\sum_{i=1}^{m}(\text{定義長 } i) + \sum_{i=1}^{n}(\text{定義長 } i \times 9)\right) \times \text{移動先 BES 数}$$

  m：圧縮指定を指定していない BINARY 列の数

  n：圧縮指定を指定している BINARY 列の数

- 共用モードの場合

  $$\sum_{i=1}^{n}(\text{定義長 } i) \times \text{移動先 BES 数}$$

  n：BINARY 列数

18.2.14 空きページ解放ユティリティ（pdreclaim）及びグローバルバッファ常駐化ユティリティ（pdpgbfon）実行時のメモリ所要量

空きページ解放ユティリティ（pdreclaim）及びグローバルバッファ常駐化ユティリティ（pdpgbfon）実行時のメモリ所要量は，次に示す計算式で求めます。

条件		メモリ所要量の計算式 （単位：キロバイト）
HiRDB/シングルサーバの場合（32 ビットモードの場合）		800 ＋ W
HiRDB/シングルサーバの場合（64 ビットモードの場合）		850 ＋ W
HiRDB/パラレルサーバの場合 （32 ビットモードの場合）	システムマネジャ	800 ＋ X
	-s オプションで指定したサーバ※1	Y
	バックエンドサーバ※2	Z
HiRDB/パラレルサーバの場合 （64 ビットモードの場合）	システムマネジャ	850 ＋ X
	-s オプションで指定したサーバ※1	Y
	バックエンドサーバ※2	Z

注※1

-s オプション省略時は，処理対象表格納 RD エリアを 1 個目に定義したサーバです。

注※2

バックエンドサーバが複数ある場合は，バックエンドサーバ数だけメモリ所要量を加算します。

W：データベース再編成ユティリティ（pdrorg）実行時のシングルサーバのメモリ所要量

X：データベース再編成ユティリティ（pdrorg）実行時の MGR のメモリ所要量

Y：データベース再編成ユティリティ（pdrorg）実行時の DS のメモリ所要量

Z：データベース再編成ユティリティ（pdrorg）実行時の BES のメモリ所要量

データベース再編成ユティリティ（pdrorg）実行時のメモリ所要量については，「データベース再編成ユティリティ（pdrorg）実行時のメモリ所要量」を参照してください。

18.2.15　整合性チェックユティリティ（pdconstck）実行時のメモリ所要量

整合性チェックユティリティ（pdconstck）実行時のメモリ所要量は，次に示す計算式で求めます。

条件			メモリ所要量の計算式 （単位：キロバイト）
HiRDB/シングルサーバの場合	32 ビットモードの場合		13995 ＋ α
	64 ビットモードの場合		14766 ＋ α
HiRDB/パラレルサーバの場合	32 ビットモードの場合	MGR	6675 ＋ α
		DS	5910 ＋ β
	64 ビットモードの場合	MGR	7252 ＋ α

条件			メモリ所要量の計算式 (単位：キロバイト)
		DS	8740 + β

α：次に示す計算式から求められる値

2175

＋ 0.14×列数

＋ 0.09×表格納 RD エリア数

＋ 0.23×インデクス数

＋ 0.09×インデクス格納 RD エリア数

＋ 0.09×LOB 列格納 RD エリア数

外部キー数

＋ Σ（42 ＋ 0.47×外部キー構成列数 r）

r=1

検査制約数

＋ Σ（5 ＋ 0.29×探索条件中の列数 c ＋ 0.85

c=1

×探索条件中の AND 及び OR の数 c ＋探索条件長 c）

β：次に示す計算式から求められる値

0.2

＋ 0.02×表格納 RD エリア数

＋ 0.02×インデクス格納 RD エリア数

＋ 0.02×LOB 列格納 RD エリア数

18.2.16 パラレルローディング（pdparaload）実行時のメモリ所要量

パラレルローディング（pdparaload）実行時のメモリ所要量は，次に示す計算式で求めます。

```
パラレルローディング実行時のメモリ所要量（単位：キロバイト）
    ＝1000＋pdparaload制御文ファイルの容量＋10＋10＋R
        R
      ＋Σ（RDエリア単位のデータロード実行に必要なメモリ容量i）
       i=1
```

R：表を構成する RD エリア数

注

pdparaload コマンドは，表を構成する RD エリアの数だけ，内部で RD エリア単位のデータロード（pdload）を実行します。そのため，pdparaload は，RD エリア単位のデータロード実行に必要なメ

モリ所要量を，表を構成する RD エリア数分使用します。RD エリア単位のデータロード実行に必要なメモリ所要量については，「データベース作成ユティリティ（pdload）実行時のメモリ所要量」を参照してください。

19 OS のオペレーティングシステムパラメタの見積もり

この章では，OS のオペレーティングシステムパラメタ（カーネルパラメタ）の見積もり方法について説明します。

19.1　AIX のオペレーティングシステムパラメタの見積もり

19.1.1　AIX のオペレーティングシステムパラメタの指定方法

ここでは，AIX のオペレーティングシステムパラメタ（カーネルパラメタ）の見積もりについて説明します。オペレーティングシステムパラメタの値が小さいと，HiRDB が正しく動作しないことがあります。AIX のオペレーティングシステムパラメタの指定値の目安を次の表に示します。

表 19–1　AIX のオペレーティングシステムパラメタの指定値の目安

オペレーティングシステムパラメタ	指定値の目安
data_hard	パラメタの省略値が-1（unlimit）のため，特に必要がなければ変更しないでください。
stack_hard	パラメタの省略値で十分大きいため，特に必要がなければ変更しないでください。
nofiles	HiRDB が計算して設定するため，指定する必要はありません。
nofiles_hard	パラメタの省略値が-1（unlimit）のため，特に必要がなければ変更しないでください。
maxuproc	MAX（pd_max_server_process の値＋ｅ，512）以上を指定してください。 ただし，サーバマシン上で稼働するほかのプログラムが必要とする値の方が大きい場合は，その値を指定してください。
EXTSHM 環境変数	32 ビットモードの HiRDB の場合，ON を指定してください（ただし，1 プロセスでアタッチする共用メモリセグメント数が 11 未満の場合は指定しないでください）。64 ビットモードの HiRDB の場合，使用する機能によって設定値が変わります。

ｅ：同時実行するコマンド（ユティリティを含む）の最大数

注

- システムで同時にオープンできる最大ファイル数は，「maxuproc×nofiles×固定ライセンス数」で調整できます。
- システムにログインするユーザの最大数は固定ライセンス数で調整できます。
- システム全体で同時に実行するプロセスの総数の最大数は，「maxuproc×固定ライセンス数」で調整できます。

注意事項

TCP ポートが TIME_WAIT 状態となり，システム全体の TCP ポートが不足し，トランザクションがエラーとなったり，HiRDB が異常終了したりすることがあります。このような現象が発生した場合，オペレーティングシステムパラメタの設定によって，ポート数不足の発生を回避してください。詳細については，「ポート数不足を回避する方法」を参照してください。

(1) AIX 固有のパラメタ指定

(a) 環境変数の指定

AIX の場合，システム共通定義に，次に示す環境変数を指定する必要があります。なお，PSALLOC，NODISCLAIM，及び CORE_NOSHM 環境変数は，システム共通定義への指定は不要です。

また，次に示す環境変数は HiRDB コマンド実行環境にも設定する必要があります。システム共通定義に設定した環境変数は，HiRDB コマンド実行環境の設定値を同じ値にしてください。環境変数の設定方法については，各シェルのマニュアルを参照してください。

- EXTSHM

  32 ビットモードの HiRDB の場合，プロセス空間の共用メモリ領域数の制限をなくすことを示す「ON」を設定します。

  64 ビットモードの HiRDB の場合，共用メモリのページ固定機能を使用するときは，OS のページ固定機能を有効にするため putenv EXTSHM ON を指定しません（省略します）。また，64 ビットモードの HiRDB サーバの場合で，32 ビットモードのクライアントプロセスとの間でプロセス間メモリ通信機能を使用するときには，プロセス空間の共用メモリ領域数の制限をなくすことを示す ON を指定します。

  アドレスモードと使用する機能の組み合わせに対する EXTSHM 環境変数の指定形式を次に示します。

HiRDB サーバのアドレスモード	共用メモリのページ固定機能※1	32 ビットモードのクライアントプロセスとの間のプロセス間メモリ通信機能	EXTSHM 環境変数の指定形式
32 ビットモード	—※2	○	putenv EXTSHM ON※3
		×	
64 ビットモード	○	○	—※4
		×	指定しません
	×	○	putenv EXTSHM ON
		×	—※5

（凡例）

○：使用時

×：未使用時

－：該当しません。

注※1

共用メモリのページ固定は，pd_shmpool_attribute オペランド又は pd_dbbuff_attribute オペランドに fixed を設定した場合に使用します。詳細は，マニュアル「HiRDB システム定義」の「pd_shmpool_attribute オペランド」又は「pd_dbbuff_attribute オペランド」を参照してください。

注※2

サポートしていません。

注※3

共用メモリの 1 セグメントのサイズが 256 メガバイト未満の場合は，プロセス停止時の共用メモリデタッチ処理で，OS が 4 キロバイト単位の共用メモリページ解放を行います。そのため，CPU 使用率が上がり，系切り替えが発生することがあります。1 プロセスでアタッチする共用メモリセグメント数（pdls -d mem コマンドで確認できる共用メモリセグメント数）が 11 未満の場合は，EXTSHM=ON の指定をしないでください。

注※4

共用メモリのページ固定機能と 32 ビットモードのクライアントプロセスとの間のプロセス間メモリ通信機能を同時に使用できません。

注※5

指定不要です。

- PSALLOC

  HiRDB コマンド実行環境に，メモリ確保時に必要なページスペースをすぐに確保することを示す「early」を設定します。同時に NODISCLAIM の指定も必要です。ただし，むだなページングスペースを生む可能性があります。

- NODISCLAIM

  HiRDB コマンド実行環境に，free()に対するコールの処理方法として，nodisclaim()の発行抑止をする「true」を設定します。

- LDR_CNTRL

  32 ビットモードの場合，カーネルの従来の区分化より大きいデータエリアを扱うことができるようにするために指定します。「MAXDATA=0x40000000」を設定してください。64 ビットモードの場合，指定不要です。

- CORE_NOSHM

  HiRDB コマンド実行環境に，プロセス障害時に出力される core ファイルに，共用メモリ領域を含まないようにするために「""」を指定します。詳細は「core ファイルの出力情報の制限」を参照してください。

(b) /etc/security/limits ファイルの指定値の注意事項

root ユーザと HiRDB 管理者は，次の指定値に注意してください。

- data

  プロセスのヒープ領域がこの制限値を超えるとエラーになります。この制限値を指定する必要がない場合は，-1（unlimited）を指定してください。

- fsize，fsize_hard

  ファイルサイズがこの制限値を超えるとエラーになります。この制限値を指定する必要がない場合は，-1（unlimited）を指定してください。

(c) 仮想メモリマネージャ(VMM)のチューニングパラメタの指定

特定の機能を使用する場合，次のパラメタを指定します。VMM パラメタは AIX の vmo コマンドで設定できます。vmo コマンドについては AIX のマニュアルを参照してください。

- v_pinshm

  共用メモリ・セグメントのページ固定を可能にするための指定です。HiRDB が使用する共用メモリをページ固定する場合，このパラメタに 1 を指定してください。HiRDB が使用する共用メモリをページ固定する方法は，マニュアル「HiRDB システム定義」の「pd_shmpool_attribute オペランド」又は「pd_dbbuff_attribute オペランド」を参照してください。

- maxpin

  ページ固定する実メモリの最大パーセンテージを指定します。HiRDB が使用する共用メモリをページ固定する場合，ページ固定する HiRDB の共用メモリサイズを含む，マシン上でページ固定するすべてのメモリのサイズ合計値（OS がページ固定するメモリのサイズも含みます）を，このパラメタに指定したパーセンテージ内の実メモリサイズが上回るように，このパラメタを指定してください。HiRDB が使用する共用メモリをページ固定する方法は，マニュアル「HiRDB システム定義」の「pd_shmpool_attribute オペランド」又は「pd_dbbuff_attribute オペランド」を参照してください。

(d) core ファイルの出力情報の制限

プロセス障害時に出力される core ファイルに，共用メモリ領域を含まないように設定します。

- 設定内容

  HiRDB 管理者の環境変数（k シェルの場合）

  ```
  $ export CORE_NOSHM=
  ```

- 前提条件
  - システム属性(sys0)の fullcore パラメタが true の場合です。
- 注意事項
  - この環境変数 CORE_NOSHM が有効かどうかは，使用している AIX のバージョンに依存します。この環境変数が利用できるかどうかは，OS のマニュアルを参照してください。
  - HiRDB は OS への登録時（pdsetup コマンド実行時）に自動的に fullcore を true に設定します。その後，OS コマンドなどによって false に再設定される場合があるため，現在の fullcore パラメタが true であることを確認してください。
  - この環境変数 CORE_NOSHM の設定を/etc/environment へ記述しないでください。

(e) JFS/JFS2 ファイルシステムへの I/O 高負荷によるプログラムの沈み込みについての注意事項

JFS/JFS2 ファイルシステムへ大量の出力要求を出すプログラム（pdcopy や大きなサイズのファイルに対する compress，cp，dd コマンドなど）の実行によって，システムのディスク入出力の性能が飽和状態となって，同一システム上で動作するプログラムが数秒から数十秒沈み込む場合があります。

特に，システムの応答時間を監視する HA モニタや PowerHA によってクラスタシステム運用を行っている場合，系切り替え動作となる場合があります。

このような現象が発生した場合，次のパラメタの設定によって，プログラムの沈み込みの発生を回避してください。

- maxpout
- minpout

maxpout/minpout パラメタの設定値の詳細については，OS のマニュアルを参照してください。

(f) 時刻補正の注意事項

NTP プログラムを用いて時刻補正を行う場合は，次の点に注意してください。

- NTP の調整方式には，遅延調整方式（Slew 方式）を選択して，一度に数十秒以上の大幅な時刻補正が発生しないようにしてください。
- AIX の環境設定ファイル（/etc/environment）に「GETTOD_ADJ_MONOTONIC=1」を記述して，時刻が逆転しないようにしてください。

この環境変数 GETTOD_ADJ_MONOTONIC が有効かどうかは，使用している AIX のバージョンに依存します。

この環境変数が利用できるかどうかは，OS のマニュアルを参照してください。

19.2 Linux のオペレーティングシステムパラメタの見積もり

19.2.1 Linux のオペレーティングシステムパラメタの指定方法

ここでは，Linux のオペレーティングシステムパラメタ（カーネルパラメタ）の見積もりについて説明します。オペレーティングシステムパラメタの値が小さいと，HiRDB が正しく動作しないことがあります。Linux のオペレーティングシステムパラメタの指定値の目安を次の表に示します。

表 19–2 Linux のオペレーティングシステムパラメタの指定値の目安

オペレーティングシステムパラメタ	指定値の目安	オプション設定ファイル例※1
hard nofile	8192 を指定してください。 ただし，サーバマシン上で稼働するほかのプログラムが必要とする値の方が大きい場合は，その値を指定してください。	/etc/security/limits.conf
DefaultLimitNOFILE※2	8192 を指定してください。 ただし，サーバマシン上で稼働するほかのプログラムが必要とする値の方が大きい場合は，その値を指定してください。	/etc/systemd/system.conf
soft nofile	HiRDB が計算して設定するため，指定する必要はありません。	－
fs.aio-max-nr	次のどちらかの機能を使用する場合は（pd_max_server_process × 2）以上を指定してください。 • システムログ I/O の並列化機能 • 複製ディスク機能 このマシンに複数の HiRDB 環境がある場合，HiRDB ごとに計算して合計してください。 なお，この計算式で求めた値がパラメタのデフォルト値よりも小さいときはパラメタ値を変更する必要はありません。	/proc/sys/fs/aio-max-nr
fs.file-max	MAX｛1600，320×（h－g－i）＋［a＋（b×c）＋320］×g＋848×i＋h×2＋227＋k×m＋C｝ 上記の計算式に，次の計算式を加えて求めた値以上を指定してください。 ●HiRDB/シングルサーバの場合 pd_max_users ＋ pd_max_reflect_process_count の値 ●HiRDB/パラレルサーバの場合 （D ＋ 3×ユニット内サーバ数）×E D：次の値を求めてください。 バックエンドサーバの場合 影響分散スタンバイレス系切り替え環境かつ pd_ha_max_server_process を設定している場合 pd_ha_max_server_process の値 その他の場合 ユニット内のサーバごとに次の計算式を実行して求めた値を合計してください。	/proc/sys/fs/file-max

オペレーティングシステムパラメタ	指定値の目安	オプション設定ファイル例※1
	pd_max_bes_process ＋ pd_max_reflect_process_count の値 ユニット内に複数のバックエンドサーバがある場合は，バックエンドサーバごとに計算してください。 ディクショナリサーバの場合 ユニット内のサーバごとに次の計算式を実行して求めた値を合計してください。 pd_max_dic_process ＋ pd_max_reflect_process_count の値 pd_max_bes_process 又は pd_max_dic_process オペランドを省略している場合は pd_max_users の値で計算してください。 E：16（排他待ちスレッド数） この計算式で求めた値が，システムの上限を超える場合は，システムの上限値を指定してください。	
nproc	MAX（pd_max_server_process の値＋ e，512）以上を指定してください。 ただし，サーバマシン上で稼働するほかのプログラムが必要とする値の方が大きい場合は，その値を指定してください。	/etc/security/limits.conf
DefaultLimitNPROC※2	MAX（pd_max_server_process の値＋ e，512）以上を指定してください。 ただし，サーバマシン上で稼働するほかのプログラムが必要とする値の方が大きい場合は，その値を指定してください。	/etc/systemd/system.conf
threads-max	MAX（(pd_max_server_process の値＋ 20)，576）以上を指定してください。	/proc/sys/kernel/threads-max
msgmnb	「●msgmnb の計算式」を参照してください。ここで求めた値以上を指定してください。ただし，マルチ HiRDB の場合は，それらの HiRDB ごとに求めた値の中で最大の値を指定してください。	/proc/sys/kernel/msgmnb
msgmni	サーバマシン上で稼働する全プログラムが必要とするメッセージキュー識別子数を指定します。HiRDB が必要とするメッセージキュー識別子数については，「メッセージキュー及びセマフォ所要量の見積もり」を参照してください。そこで求めた値を加算してください。	/proc/sys/kernel/msgmni
SEMMNI	サーバマシン上で稼働する全プログラムが必要とするセマフォ識別子数を指定します。HiRDB が必要とするセマフォ識別子数については，「メッセージキュー及びセマフォ所要量の見積もり」を参照してください。そこで求めた値を加算してください。 推奨値は 1024 以上です。	/proc/sys/kernel/sem の第 4 パラメタ
SEMMNS	サーバマシン上で稼働する全プログラムが必要とするセマフォ数を指定します。HiRDB が必要とするセマフォ数については，「メッセージキュー及びセマフォ所要量の見積もり」を参照してください。そこで求めた値を加算してください。 推奨値は 7200 以上です。	/proc/sys/kernel/sem の第 2 パラメタ
SEMMSL	セマフォ識別子当たりの最大セマフォ数を指定します。HiRDB が必要とするセマフォ識別子当たりの最大セマフォ数については，250 以上を指定してください。ただし，pd_dfw_awt_process オペランドに 251 以上を指定しているサーバがある場合は，255 以上を指定してください。	/proc/sys/kernel/sem の第 1 パラメタ
shmmax	200000000 以上で，MAX（p ＋ q，r，s，t）よりも大きい値を指定してください。グローバルバッファの動的変更機能使用時は，追加するグローバルバッファのサイズ	/proc/sys/kernel/shmmax

オペレーティングシステムパラメタ	指定値の目安	オプション設定ファイル例※1
	を考慮し，設定値より追加分のサイズが大きくなる可能性があれば，予想される追加分のサイズを指定してください。 なお，HiRDB システム定義の SHMMAX オペランドには，ここで求めた shmmax の値以下を指定してください。	
shmmni	2000 以上を指定してください。 • セキュリティ監査機能使用時は 1 を加算してください。 • グローバルバッファの動的変更機能使用時は次の値を加算します。セキュリティ監査機能使用時は 1 を加算してください。 **HiRDB/シングルサーバの場合** pd_max_add_dbbuff_shm_no オペランドの値 **HiRDB/パラレルサーバの場合** n Σ 各サーバ定義に指定した pd_max_add_dbbuff_shm_no オペランドの値 i=1 n：サーバマシン内のバックエンドサーバ数＋ディクショナリサーバ数 • インメモリデータ処理の使用時は次の値を加算します。 **HiRDB/シングルサーバの場合** pd_max_resident_rdarea_shm_no オペランドの値 **HiRDB/パラレルサーバの場合** pd_max_resident_rdarea_shm_no オペランドの値×バックエンドサーバ数	/proc/sys/kernel/shmmni
SHMALL	該当するサーバマシンで HiRDB が確保する共用メモリ si に，同一サーバマシンで稼働するほかのプログラムの共用メモリの所要量を加算して指定してください。	/proc/sys/kernel/shmall

（凡例） －：該当しません。

注※1

使用している OS のバージョン，及びカーネルのバージョンごとに異なります。使用している OS のマニュアルを参照し，表中の指定値の目安で示した値を設定してください。なお，使用している OS のバージョンによっては，設定が不要になります。使用している OS で該当するカーネルパラメタが設定できない場合には，設定は不要です。

注※2

Red Hat Enterprise Linux(R) 7 以降で指定してください。

a：データベース作成ユティリティで使用する入力データファイル数と分割入力データファイル数，又はデータベース再編成ユティリティで使用するアンロードデータファイル数

b：インデクスの横分割数の最大値（データベース作成ユティリティ，データベース再編成ユティリティ，又はリバランスユティリティの処理対象のインデクス）

c：インデクス数（データベース作成ユティリティ，データベース再編成ユティリティ，又はリバランスユティリティの処理対象のインデクス）

e：同時実行するコマンド（ユティリティを含む）の最大数

g：次のどちらかの値

- HiRDB/シングルサーバの場合：pd_max_users オペランドの値＋ pd_max_reflect_process_count オペランドの値
- HiRDB/パラレルサーバの場合：次の値

  影響分散スタンバイレス系切り替え環境かつ pd_ha_max_server_process オペランドを設定している場合

  pd_ha_max_server_process オペランドの値

  その他の場合

  ユニット内の全バックエンドサーバ，ディクショナリサーバの次に示す値の合計値

  pd_max_bes_process オペランドの値＋ pd_max_reflect_process_count オペランドの値

  pd_max_dic_process オペランドの値＋ pd_max_reflect_process_count オペランドの値

h：pd_max_server_process オペランドの値

i：ユニット内のサーバ数

k：次のどちらかの値

- HiRDB/パラレルサーバでユニット内にフロントエンドサーバがある場合：pd_max_users オペランドの値＋ pd_max_reflect_process_count オペランドの値
- そのほかの場合：0

m：次のどちらかの値

- HiRDB/パラレルサーバでユニット内にフロントエンドサーバがある場合：2
- そのほかの場合：0

n：データベース作成ユティリティ，データベース再編成ユティリティ，又はリバランスユティリティで，インデクス作成方法にインデクス一括作成モード若しくはインデクス情報出力モードを指定する場合，次の計算式の値

MIN（MAX（576，システム内の HiRDB サーバ数＋ 448）＋ b×c，pd_max_open_fds オペランドの最大値）

pd_max_open_fds オペランドの最大値については，マニュアル「HiRDB システム定義」を参照してください。

p：ユニットコントローラが使用する共用メモリの大きさ

q：シングルサーバ又は各サーバが使用する共用メモリの大きさ

r：HiRDB システム定義の SHMMAX オペランドの指定値

s：グローバルバッファが使用する共用メモリの見積もり値

グローバルバッファが使用する共用メモリの見積もりについては，HiRDB/シングルサーバの場合は「グローバルバッファが使用する共用メモリの計算式」を，HiRDB/パラレルサーバの場合は「グローバルバッファが使用する共用メモリの計算式」を参照してください。

si：該当するサーバマシンで HiRDB が確保する共用メモリ

- HiRDB/シングルサーバの場合

  「ユニットコントローラが使用する共用メモリの計算式」で求めた値
  ＋「シングルサーバが使用する共用メモリの計算式」で求めた値
  ＋「グローバルバッファが使用する共用メモリの計算式」で求めた値

- HiRDB/パラレルサーバの場合

  「ユニットコントローラが使用する共用メモリの計算式」で求めた値
  ＋「各サーバが使用する共用メモリの計算式」で求めた値
  ＋「グローバルバッファが使用する共用メモリの計算式」で求めた値

t：セキュリティ監査情報用バッファ用共用メモリの見積もり値

セキュリティ監査情報用バッファ用共用メモリの見積もりについては，HiRDB/シングルサーバの場合は「メモリ所要量の計算式」を，HiRDB/パラレルサーバの場合は「メモリ所要量の計算式」を参照してください。

C：

MAX（256,（システム内 HiRDB サーバ数＋ 128））×（g＋k）＋（h－g－k）×MAX（システム内 HiRDB サーバ数，ユニット数）

●msgmnb の計算式

msgmnb の計算式を次に示します。

```
msgmnb＝MAX（（A＋B×2＋C＋1）※2×40，D※2×16，E×F×4，G×4）※1
```

注※1

ユニット内の全サーバについて求めた値で，最大値を指定します。

1：1 スタンバイレス型系切り替えの対象となるユニットの場合は，ユニット内の全サーバとユニット内で動作する代替 BES について求めた値で，最大値を指定します。

影響分散スタンバイレス型系切り替えの対象となるユニットの場合は，ユニット内の全サーバと HA グループに含まれる全バックエンドサーバについて求めた値で，最大値を指定します。

注※2

フロントエンドサーバの場合は 0 となります。

A：サーバに割り当てられているグローバルバッファの数

pdbufls コマンドで表示されるグローバルバッファ名をサーバ単位に集計すると確認できます。

B：サーバで発生するシンクポイントダンプ有効化のスキップ回数

- pd_spd_syncpoint_skip_limit オペランドに 0 以外を指定している場合：

pd_spd_syncpoint_skip_limit オペランドの指定値で計算式してください。

- pd_spd_syncpoint_skip_limit オペランドを省略しているか，又は 0 を指定している場合：
  マニュアル「HiRDB システム運用ガイド」の「UAP の状態監視（シンクポイントダンプ有効化のスキップ回数監視機能)」の「全システムログファイルの容量から計算する方法」を参照して計算してください。

C：サーバで実行される pdload，pdrorg，pdrbal，及びログレスモードの UAP の最大同時実行数

D：pd_max_ard_process オペランドが 1 以上の場合は次に示す値となります。

- サーバに割り当てられているグローバルバッファのうち，pdbuffer オペランドの-m オプションの指定値の合計

pd_max_ard_process オペランドが 0 の場合は 0 となります。

E：pd_max_users オペランドの値

F：HiRDB/シングルサーバの場合は 1，HiRDB/パラレルサーバの場合は次に示す値となります。

- ユニット内のフロントエンドサーバ数＋ユニット内のバックエンドサーバ数＋ユニット内のディクショナリサーバ数＋ pd_ha_max_act_guest_servers オペランドの値

pd_ha_max_act_guest_servers オペランドの値は，影響分散スタンバイレス型系切り替え機能を適用しているユニットの場合に加算します。

G：↓pd_trn_rcvmsg_store_buflen オペランドの値÷72↓

注意事項

TCP ポートが TIME_WAIT 状態となり，システム全体の TCP ポートが不足し，トランザクションがエラーとなったり，HiRDB が異常終了したりすることがあります。このような現象が発生した場合，オペレーティングシステムパラメタの設定によって，ポート数不足の発生を回避してください。詳細については，「ポート数不足を回避する方法」を参照してください。

(1) Linux 固有の指定

Linux 版の場合，拡張されたインターネットサービスデーモン（以降，xinetd と表記）が起動していると，その設定ファイルである xinetd.conf などで設定している値の調整が必要となります。

また，Red Hat Enterprise Linux(R) 7 で HiRDB がリモートシェル実行やファイル転送に使用するコマンドにリモートシェルを選択（システム定義 pd_cmd_rmode=rsh を指定，又は指定を省略）し，systemd の socket ユニットを構成してリモートシェルを使用する場合は，リモートシェルの socket ユニット構成ファイルに設定しているパラメタの調整が必要です。

それぞれのパラメタの指定値の目安を次の表に示します。

表 19-3　リモートシェルに関する Linux 固有の指定値の目安

パラメタ	指定値の目安	オプション設定ファイル例※1
cps （1 秒当たりに処理する接続数）	1 番目の引数に，次の値を指定してください。 HiRDB/シングルサーバの場合： ユティリティ専用ユニットがないとき：A ユティリティ専用ユニットがあるとき：MAX（A，B） HiRDB/パラレルサーバの場合： MAX（C，（D ＋ E1 ＋ E2）） なお，計算した値がパラメタのデフォルト値よりも小さい場合は，パラメタ値を変更する必要はありません。	/etc/xinetd.conf
instances （サービスの同時実行最大値）	cps パラメタで計算した値の 2 倍を指定してください。 なお，計算した値がパラメタのデフォルト値よりも小さい場合は，パラメタ値を変更する必要はありません。	/etc/xinetd.conf
MaxConnections※2 （サービスの最大値）	cps パラメタで計算した値の 2 倍を指定してください。 なお，計算した値がパラメタのデフォルト値よりも小さい場合は，パラメタ値を変更する必要はありません。	/etc/systemd/system/sockets.target.wants/rsh.socket

注※1

使用している OS のバージョン，及びカーネルのバージョンごとに異なります。xinetd.conf 及び.socket ファイルについては，OS のマニュアルを参照してください。

注※2

Red Hat Enterprise Linux(R) 7 で HiRDB がリモートシェル実行やファイル転送に使用するコマンドにリモートシェルを選択（システム定義 pd_cmd_rmode=rsh を指定，又は指定を省略）し，systemd の socket ユニットを構成してリモートシェルを使用する場合，[Socket]セクションに指定してください。

A：該当するマシンで起動するシングルサーバ数×4

B：pdcopy 及び pdrstr の最大同時実行数×2

C：該当するマシンで起動するディクショナリサーバ数
×HiRDB システムに定義した全バックエンドサーバ数×6
＋該当するマシンで起動するディクショナリサーバ数
×HiRDB システムに定義した全バックエンドサーバ数×2
＋該当するマシンで起動するバックエンドサーバ数×7
＋該当するマシンで起動するディクショナリサーバ数×4

D：pdcopy 及び pdrstr の最大同時実行数
　×（HiRDB システムに定義したバックエンドサーバ数＋ 4）

E $_1$：同時に実行する運用コマンド及びユティリティの数×3

E $_2$：pdtrndec コマンドを実行する運用の場合は次の合計値，pdtrndec コマンドを実行しない運用の場合は 0
　該当するマシンに FES が存在する場合は pd_max_users の値
　該当するマシンに BES が存在する場合は pd_max_bes_process の値
　該当するマシンに DS が存在する場合は pd_max_dic_process の値

(2)　Linux の Hugepage 機能の指定

Linux 版の HiRDB では，Linux の Hugepage 機能を用いて，共用メモリを実メモリ上に固定できます。詳細については，HiRDB/シングルサーバの場合は「Hugepage 機能を用いた共用メモリの固定」を，HiRDB/パラレルサーバの場合は「Hugepage 機能を用いた共用メモリの固定」を参照してください。ここでは，Linux の Hugepage 機能を有効にするときに指定する Linux のオペレーティングシステムパラメタと，指定値の見積もり方法について説明します。

(a)　Hugepage 機能を有効にするときに指定するオペレーティングシステムパラメタ

Linux のオペレーティングシステムパラメタの指定値の目安を次の表に示します。

表 19-4　Hugepage 機能適用時のオペレーティングシステムパラメタ指定値の目安

オペレーティングシステムパラメタ	指定値の目安	オプション設定ファイル例※
vm.nr_hugepages	下記「HiRDB が使用する hugepages の最大ページ数の求め方」を参照して求めた値を指定してください。既に値が指定されていた場合は，既定値に求めた値を加算してください。	/etc/sysctl.conf
vm.hugetlb_shm_group	HiRDB 管理者のグループ ID を指定してください。	/etc/sysctl.conf
soft memlock	HiRDB 管理者のユーザ ID を対象とし，unlimited を指定してください。	/etc/security/limits.conf
hard memlock	HiRDB 管理者のユーザ ID を対象とし，unlimited を指定してください。	/etc/security/limits.conf

注※　使用している OS のバージョン，及びカーネルのバージョンごとに異なります。使用している OS のマニュアルを参照し，表中の指定値で示した値を指定してください。

(b)　HiRDB が使用する hugepages の最大ページ数の求め方

次に示す方法で算出してください。

(i) この機能を適用しないで HiRDB を開始します。

(ii)HiRDB の開始完了後，pdls -d mem コマンドを実行して，ユニットコントローラ用共用メモリと各グローバルバッファ用共用メモリの個々のサイズを確認します。

- ユニットコントローラ用共用メモリのサイズ：

  pdls -d mem コマンドで表示される SHM-OWNER の値が，MANAGER である共用メモリの ACT-SIZE の値

- 各グローバルバッファ用共用メモリの個々のサイズ：

  pdls -d mem コマンドで表示される SHM-OWNER の値が，次に示す文字列である共用メモリの ACT-SIZE の値

- 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の場合：

  ユニット名を（）で囲んだ文字列

- 影響分散スタンバイレス型系切り替え機能を適用しないユニットの場合：

  サーバ名

(iii)次に示す計算式を用いて，HiRDB が使用する hugepages の最大ページ数を求めます。サーバマシン内にこの機能を適用する HiRDB システムを複数構成している場合，各システムで値を算出し，その合計値を用いて計算してください。

HiRDBが使用するhugepagesの最大ページ数　＝　Ａ＋Ｂ＋Ｃ＋Ｄ

A…ユニットコントローラ用共用メモリが使用する hugepages の最大ページ数

- ユニットコントローラ用共用メモリにこの機能を適用する場合：

  ↑(ii)で求めたユニットコントローラ用共用メモリのサイズ÷ 2000000↑

- ユニットコントローラ用共用メモリにこの機能を適用しない場合：0

B…グローバルバッファ用共用メモリが使用する hugepages の最大ページ数

- グローバルバッファ用共用メモリにこの機能を適用する場合：

  n

  Σ（↑(ii)で求めたグローバルバッファ用共用メモリのサイズ÷ 2000000↑）

  i=1

  n：(ii)で求めたグローバルバッファ用共用メモリの数

- グローバルバッファ用共用メモリにこの機能を適用しない場合：0

C…グローバルバッファの動的変更で確保するグローバルバッファ用共用メモリが使用する hugepages の最大ページ数

- グローバルバッファの動的変更で確保するグローバルバッファ用共用メモリにこの機能を適用する場合：

  システム共通定義の pd_max_add_dbbuff_shm_no オペランドの値

  × ↑システム共通定義の SHMMAX オペランドの値 ÷ 2 ↑

- グローバルバッファの動的変更で確保するグローバルバッファ用共用メモリにこの機能を適用しない場合：0

D…インメモリデータバッファが使用する共用メモリが使用する hugepages の最大ページ数

- インメモリデータバッファが使用する共用メモリにこの機能を適用する場合：
  システム共通定義の pd_max_resident_rdarea_shm_no オペランドの値
  × ↑ システム共通定義の SHMMAX オペランドの値 ÷ 2 ↑
- インメモリデータバッファが使用する共用メモリにこの機能を適用しない場合：0

19.3　メッセージキュー及びセマフォ所要量の見積もり

一つのサーバマシン内で使用するメッセージキュー及びセマフォ所要量は，次に示す計算式で求めます。

19.3.1　HiRDB/シングルサーバの場合の計算式

種　別	計算式
メッセージキュー識別子数	（16 ＋ f）×a ＋ 29
セマフォ識別子数	{↑ {2×（b ＋ 3）＋ 12} ÷64↑＋↑c÷64↑＋ g ＋ 5} ×a ＋ 2 ＋ d
セマフォ数（識別子ごとのセマフォ数の合計です）	{2×（b ＋ 3）＋ c ＋ h ＋ 37} ×a ＋ 3 ＋ e

a：1（シングルサーバの場合）又は 0（ユティリティ専用ユニットの場合）

b：pd_max_users オペランドの値＋ pd_max_reflect_process_count オペランドの値

c：pdbuffer オペランドの指定数（グローバルバッファの個数）

d：系切り替え機能使用時に加算します。次に示す表から値を求めてください。

e：系切り替え機能使用時に加算します。次に示す表から値を求めてください。

条件			d の値	e の値
pd_ha_acttype=monitor（又は省略）			0	0
pd_ha_acttype=server	pd_ha_agent=standbyunit		1	7
	pd_ha_agent を省略	pd_ha_server_process_standby=Y（又は省略）	1	2
		pd_ha_server_process_standby=N	0	1

f：1（pd_max_ard_process オペランドが 1 以上の場合）又は 0

g：2（pd_dfw_awt_process オペランドに値を指定する場合）又は 0

h：pd_dfw_awt_process オペランドの値＋ 1（pd_dfw_awt_process オペランドに値を指定する場合）又は 0

19.3.2　HiRDB/パラレルサーバの場合の計算式

計算式で使用している変数については，「計算式で使用する変数」を参照してください。

(1)　影響分散スタンバイレス型系切り替えを使用していない場合

種別	計算式
メッセージキュー識別子数	$\sum_{i=1}^{b} V_i + 2\times a + 3\times b + c + d + e + 26 + m$
セマフォ識別子数	$\sum_{i=1}^{b} \{\uparrow (S_i + T_i + U_i) \div 64 \uparrow + W_i\} + 6\times b + 2 + f$
セマフォ数（識別子ごとのセマフォ数の合計です）	$\sum_{i=1}^{b} (S_i + T_i + U_i + X_i) + 26\times b + 3 + g$

(2)　影響分散スタンバイレス型系切り替えを使用している場合

種別	計算式
メッセージキュー識別子数	$\sum_{i=1}^{b} V_i + 2\times a + 3\times b + c + d + e + 26 + m$
セマフォ識別子数	$\sum_{i=1}^{b} \{\uparrow \{Y_i\times (j + k)\} \div 64 \uparrow + W_i\} + 6\times b + 2 + f$
セマフォ数（識別子ごとのセマフォ数の合計です）	$\sum_{i=1}^{b} \{Y_i\times (j + k) + X_i\} + 26\times b + 3 + g$

(3)　計算式で使用する変数

a：サーバマシン内のフロントエンドサーバ数

b：サーバマシン内のディクショナリサーバ数＋n

c：4（フロントエンドサーバの場合）又は0（フロントエンドサーバ以外の場合）

d：8（ディクショナリサーバの場合）又は0（ディクショナリサーバ以外の場合）

e：16×n（バックエンドサーバの場合）又は0（バックエンドサーバ以外の場合）

f及びg：系切り替え機能使用時に加算します。次に示す表から値を求めてください。

条件	fの値	gの値
pd_ha_acttype=monitor（又は省略）	0	0

条件			f の値	g の値
pd_ha_acttype=server	pd_ha_agent=standbyunit		1	h
	pd_ha_agent=server		1	６＋２×（1：1 スタンバイレス型系切り替えを適用したユニット内のバックエンドサーバ数）
	pd_ha_agent=activeunits		0	0
	pd_ha_agent を省略	pd_ha_server_process_standby=Y（又は省略）	1	i
		pd_ha_server_process_standby=N	0	1

h：6 ＋ 2×（サーバマシン内のフロントエンドサーバ数，ディクショナリサーバ数，及びバックエンドサーバサーバ数の合計）

ｉ：1 ＋（サーバマシン内のフロントエンドサーバ数，ディクショナリサーバ数，及びバックエンドサーバサーバ数の合計）

ｊ：ホスト BES 数

ｋ：ゲスト BES 数

m：システムマネジャユニットがある場合は 3，ない場合は 0

ｎ：次のどちらかの値

- 影響分散スタンバイレス系切り替えを使用していない場合
  サーバマシン内のバックエンドサーバ数
- 影響分散スタンバイレス系切り替えを使用している場合
  サーバマシン内のホスト BES 数＋ pd_ha_max_act_guest_servers オペランドの値

Si：各サーバに配置する RD エリアに対する pdbuffer -r の定義数

Ti：各サーバに配置する RD エリアに対する pdbuffer -i の定義数

Ui：pdbuffer オペランドの-o オプションの指定数

Vi：1（pd_max_ard_process オペランドが 1 以上の場合）又は 0

Wi：2（pd_dfw_awt_process オペランドに値を指定する場合）又は 0

Xi：pd_dfw_awt_process オペランドの値＋ 1（pd_dfw_awt_process オペランドに値を指定する場合）又は 0

Y_i：pdbuffer オペランドの-c オプションの指定数

19.4 Listen キュー指定値

HiRDB への接続要求を多数同時に受け付けると，HiRDB サーバが使用する Listen キューが不足して，クライアント側のアプリケーションに KFPA11723-E メッセージで示されるエラーが通知される場合があります。この場合は，Listen キューが不足しているかどうかを OS のコマンドで確認してください。Listen キュー不足が発生した回数がエラー発生前に比べて増加していたり，エラーが継続して発生するのに伴って増加していたりする場合は，Listen キューが不足しているものと判断します。なお，確認方法については，各 OS のマニュアルを参照してください。

Listen キューが不足していることが確認された場合は，HiRDB サーバマシンの Listen キューを拡大してください。Listen キューを拡大するときは，200 程度ずつ徐々に増やすようにしてください。ただし，Listen キューを拡大できない OS の場合や，最大同時接続数の指定値まで増やしても障害が解消しない場合には，HiRDB の接続処理能力を上げるためにサーバマシンの処理能力を向上させる（CPU 追加など），又は HiRDB への接続要求数を削減するために最大同時接続数を減らすなどの対策をしてください。

Listen キューの指定値を変更する場合に，変更するパラメタ（Linux の場合はオプション設定ファイル）を次の表に示します。パラメタを変更するコマンド及びその使用方法については，OS のマニュアルを参照してください。

表 19-5 Listen キューの指定値を変更するパラメタ

OS	変更するパラメタ※
AIX	somaxconn
Linux	/etc/sysctl.conf net.core.somaxconn

注※

使用している OS のバージョン，及びカーネルのバージョンごとに異なります。使用している OS のマニュアルを参照してください。なお，使用している OS のバージョンによっては，設定が不要になります。使用している OS で該当するカーネルパラメタが設定できない場合には，設定は不要です。

19.5　セキュアシェルを使用する場合のパラメタの見積もり

HiRDB がリモートシェル実行やファイル転送に使用するコマンドにセキュアシェルを選択（システム定義 pd_cmd_rmode=ssh を指定）する場合は，ssh サーバの設定ファイルに設定するパラメタの調整が必要です。なお，パラメタの値によっては，HiRDB が正しく動作しないことがあります。

パラメタの指定値の目安を次の表に示します。

表 19-6　セキュアシェルに関する指定値の目安

パラメタ	指定値の目安	オプション設定ファイル例※1
MaxStartups (ログイン認証前に受け付ける最大接続数)	必ず計算して指定要否を検討してください。 1 番目の引数 次の式で求めた以上の値を指定してください。 • HiRDB/シングルサーバの場合： ユティリティ専用ユニットがないとき：A×2 ユティリティ専用ユニットがあるとき：MAX（A，B）×2 • HiRDB/パラレルサーバの場合： MAX（C1，（D1 ＋ E1 ＋ E2））×2 さらに，HiRDB 以外で必要な値を加算して，1 番目の引数を超過する接続が発生しないようにしてください。 なお，計算した値がパラメタのデフォルト値よりも小さい場合は，パラメタ値を変更する必要はありません。 2 番目の引数 システムごとに適切なしきい値を検討の上，値を指定してください。 3 番目の引数 1 番目の引数で求めた以上の値を指定してください。 なお，パラメタのデフォルト値よりも小さい場合は，パラメタ値を変更する必要はありません。	/etc/ssh/sshd_config
PerSourceMaxStartups※2(IP アドレスグループごとのログイン認証前に受け付ける最大接続数)	PerSourceMaxStartups による制限を行う場合だけ指定してください。 1 番目の引数 次の式で求めた以上の値を指定してください。 • HiRDB/シングルサーバの場合：	/etc/ssh/sshd_config

パラメタ	指定値の目安	オプション設定ファイル例※1
	ユティリティ専用ユニットがないとき：A×2 ユティリティ専用ユニットがあるとき：MAX（A，B）×2 • HiRDB/パラレルサーバの場合： MAX（C2，（D2 ＋ E1 ＋ E2））×2 さらに，HiRDB 以外で必要な値を加算して，1 番目の引数を超過する接続が発生しないようにしてください。 2 番目の引数 システムごとに適切なしきい値を検討の上，値を指定してください。 3 番目の引数 1 番目の引数で求めた以上の値を指定してください。	
ChrootDirectory (ルートディレクトリの変更)	none を指定するか，パラメタの指定を省略してください。	/etc/ssh/sshd_config

注※1

使用している OS のバージョン，及びカーネルのバージョンごとに異なります。sshd_config ファイルについては，OS のマニュアルを参照してください。

注※2

PerSourceMaxStartups を使用する場合は，PerSourceNetBlockSize の指定値に従ってグループごとに計算し,全グループ中の最大値を指定してください。

A：該当するマシンで起動するシングルサーバ数

B：pdcopy 及び pdrstr の最大同時実行数×2

C 1：該当するマシンで起動するディクショナリサーバ数
×HiRDB システムに定義した全バックエンドサーバ数×2
＋該当するマシンで起動するバックエンドサーバ数
＋該当するマシンで起動するディクショナリサーバ数

C 2：該当するマシンで起動するディクショナリサーバ数
×HiRDB システムに定義したバックエンドサーバのうち PerSourceNetBlockSize で指定したグループに含まれるバックエンドサーバ数×2
＋該当するマシンで起動するバックエンドサーバ数
＋該当するマシンで起動するディクショナリサーバ数

Ｄ１：pdcopy 及び pdrstr の最大同時実行数
　×（HiRDB システムに定義したバックエンドサーバ数＋ 4）

Ｄ２：pdcopy 及び pdrstr の最大同時実行数
　×（HiRDB システムに定義したバックエンドサーバのうち PerSourceNetBlockSize で指定したグループに含まれるバックエンドサーバ数＋ 4）

Ｅ１：同時に実行する運用コマンド及びユティリティの数×3

Ｅ２：pdtrndec コマンドを実行する運用の場合は次の値，pdtrndec コマンドを実行しない運用の場合は 0
　該当するマシンで起動するフロントエンドサーバ数
　＋該当するマシンで起動するバックエンドサーバ数
　＋該当するマシンで起動するディクショナリサーバ数

20 サンプルファイル

この章では，HiRDB が提供するサンプルファイル（サンプルデータベース，サンプルコンフィグレーション及びサンプル UOC）について説明します。

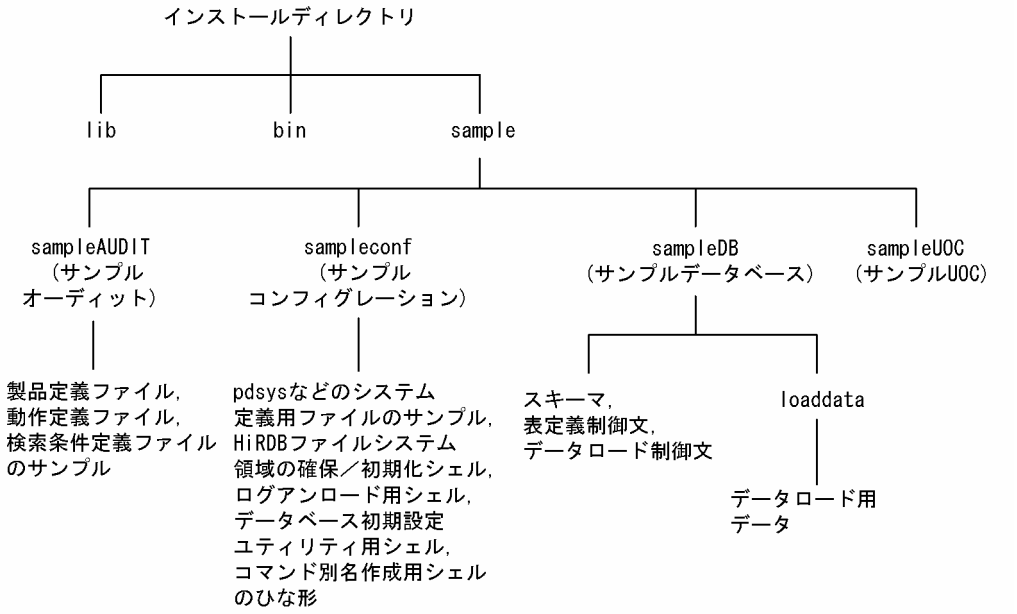
20.1　サンプルファイルの概要

HiRDB が提供するサンプルファイルを次に示します。

- サンプルオーディット
- サンプルデータベース
- サンプルコンフィグレーション
- サンプル UOC

サンプルファイルのディレクトリ構成を次の図に示します。なお，sample 以下のディレクトリは，インストールディレクトリの下にあります。pdsetup コマンドを実行しても HiRDB 運用ディレクトリにはコピーされません。

図 20–1　サンプルファイルのディレクトリ構成



注　インストールディレクトリは次のとおりです。

HiRDB/シングルサーバの場合	HiRDB/パラレルサーバの場合
/HiRDB_S/sample/sampleconf	/HiRDB_P/sample/sampleconf
/opt/HiRDB_S/sample/sampleconf	/opt/HiRDB_P/sample/sampleconf

20.1.1　サンプルファイルのファイル名

ここでは，下記のサンプルのファイル名について説明します。

- サンプルオーディット

- サンプルデータベース
- サンプルコンフィグレーション
- サンプル UOC

(1) サンプルオーディットのファイル名

サンプルオーディットは，JP1/NETM/Audit との連携で使用するサンプルファイルです。ファイル名とその内容を次の表に示します。

表 20–1 サンプルオーディットのファイル名と内容

ファイル名	内容
HiRDB.conf	製品定義ファイル
admjevlog_HiRDB.conf	動作定義ファイル
sampleaud1	検索条件定義ファイル
sampleaud2	

これらのサンプルファイルを使用した環境設定については，マニュアル「HiRDB システム運用ガイド」の「JP1/NETM/Audit との連携」を参照してください。

(2) サンプルデータベースのファイル名

サンプルで使用するディレクトリ及びファイル名とその内容を次の表に示します。

なお，サンプルで使用するファイルには，**日本語版（半角カナ使用）**と，**英語版**があります。文字コードセットがシフト JIS 以外の場合，英語版を使用してください。

表 20–2 サンプルで使用するディレクトリ及びファイル名とその内容

ディレクトリ名 又はファイル名	内容	備考
tblecreate※1	表定義文（スキーマ定義を含む）	pddef 入力形式
loadinf※2	pdload 用制御文	シェル
loaddata	データロード入力データ	ディレクトリ
SEIGYO_FILE	データロード用制御文	－

（凡例）－：該当しません。

注※1　英語版の場合，ファイル名は tblecreate_e になります。

注※2　英語版の場合，ファイル名は loadinf_e になります。

(3) サンプルコンフィグレーションのファイル名

サンプルコンフィグレーションの内容を次の表に示します。

ここで示す内容は，パラメタ間の関連性などを分かりやすくするために最小構成での指定値を例として示したもので，最適な値を示したものではありません。

表 20-3 サンプルコンフィグレーションの内容

分類	内容		ファイル名	
			HiRDB/シングルサーバ	HiRDB/パラレルサーバ
システム定義	システム共通定義		pdsys	pdsys
	ユニット制御情報定義		pdutsys	pdutsys1，pdutsys2
	サーバ共通定義		－	pdsvrc
	シングルサーバ定義		sds01	－
	フロントエンドサーバ定義		－	fes1
	バックエンドサーバ定義		－	bes1，bes2
	ディクショナリサーバ定義		－	dic1
HiRDB ファイルシステム領域の確保／初期化	RD エリアの確保シェル		fmkfile	fmkfile
	システムログ，シンクポイントダンプ及びステータスファイル確保シェル		sysfmkfs	sysfmkfs1，sysfmkfs2
	システムログ，シンクポイントダンプ及びステータスファイル初期化シェル		sysfinit	sysfint
ログアンロード	システムログファイルアンロード用シェル		logunld	feslogunld，bes1logunld，bes2logunld，diclogunld
データベース初期設定ユティリティ（pdinit）	制御文作成シェル		mkinit	mkinit
	データベース初期設定ユティリティ制御文		rdinit01※	rdinit01※
コマンドの別名での実行	コマンドを別名で実行するためのシェルスクリプトのひな形	Bourne シェル	aliascmdbsh	aliascmdbsh
		C シェル	aliascmdcsh	aliascmdcsh

（凡例）－：該当しません。

注※

データベース初期設定ユティリティ（pdinit）の制御文作成シェル（mkinit）を実行することで生成されます。

(4) サンプル UOC のファイル名

次に示す UOC を格納しています。サンプル UOC の内容を次の表に示します。

- データベース作成ユティリティ（pdload）のファイル入力の例
- データベース再編成ユティリティ（pdrorg）のファイル出力の例

UOC 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。

表 20–4 サンプル UOC の内容

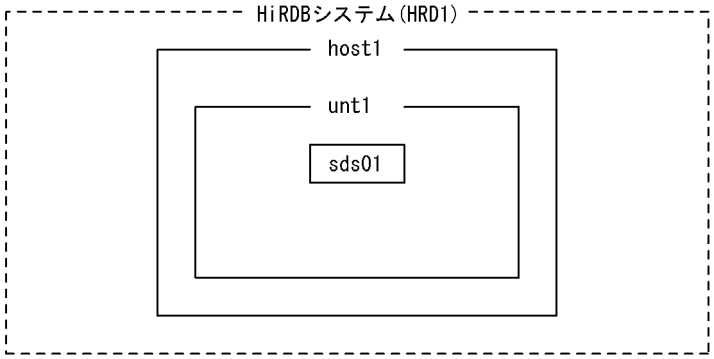
ファイル名	内容
sample1.c	DAT 形式入力ファイルをデータベース作成ユティリティ（pdload）が入力する UOC の例です。
sample2.c	バイナリ形式入力ファイルを UOC が入力する UOC の例です。
sample4.c	sample2.c が利用する関数です。
sampleA.c	不要なデータをアンロードファイルに出力しない UOC の例です。
sampleB.c	UOC データファイルを UOC で出力する例です。

20.2　システム構成と表の定義情報

20.2.1　システム構成

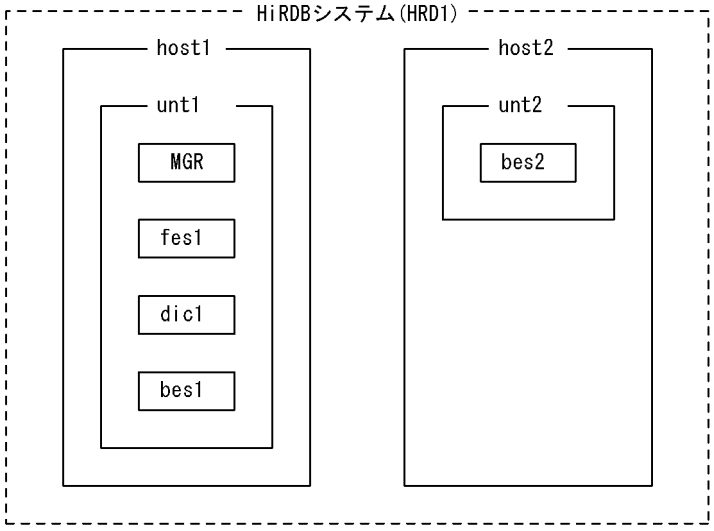
ホストとサーバの関係を示す HiRDB/シングルサーバの構成と HiRDB/パラレルサーバの場合の構成を次の図に示します。

図 20-2　HiRDB/シングルサーバの場合の構成



（凡例）HDR1　：HiRDB識別子
　　host1：ホスト名
　　unt1　：ユニット識別子
　　sds01：シングルサーバ名

図 20-3　HiRDB/パラレルサーバの場合の構成



（凡例）HDR1　　：HiRDB識別子
　　host1,host2：ホスト名
　　unt1,unt2　：ユニット識別子
　　MGR　　：システムマネージャ
　　fes1　　：フロントエンドサーバ名
　　dic1　　：ディクショナリサーバ名
　　bes1,bes2　：バックエンドサーバ名

20.2.2　表の定義情報

サンプルデータベースを使用して表定義及びデータロードをする場合，**日本語版（半角カナ使用）**と**英語版**があります。文字コードセットがシフト JIS 以外の場合，英語版を使用してください。

定義する表の内容，日本語版の列属性とインデクス，及び英語版の列属性とインデクスを次の表に示します。

ここで示す表は，すべて FIX 属性の表です。

表 20–5　定義する表の内容

表名	内容	行数
CUSTOM	得意先マスタ	100※1
GOODS	商品マスタ	100※1
VENDOR	仕入先マスタ	50※1
TAKEODR	受注	―※2
STOCK	在庫	100
WAREHUS	入庫	―※2
SHIPMNT	出庫	―※2
SENDODR	発注	―※2
LAYIN	仕入れ	―※2

（凡例）―：該当しません。

注※1　英語版では，データロード対象外です。

注※2　日本語版，英語版ともにデータロード対象外です。

表 20–6　列属性とインデクス（日本語版の場合）

表名	列名	列属性	インデクス名
CUSTOM	トクイサキ CD	CHAR(5)	UNIQUE CLUSTER KEY CUSTOMX
	トクイサキメイ	CHAR(30)	
	TELNO	CHAR(12)	
	ZIPCD	CHAR(3)	
	ジユウシヨ	CHAR(30)	
GOODS	シヨウヒン CD	CHAR(6)	UNIQUE CLUSTER KEY GOODSX
	シヨウヒンメイ	CHAR(30)	
	タンカ	DECIMAL(7,0)	

表名	列名	列属性	インデクス名
	シイレサキ CD	CHAR(5)	
VENDOR	シイレサキ CD	CHAR(5)	UNIQUE CLUSTER KEY VENDORX
	シイレサキメイ	CHAR(30)	
	TELNO	CHAR(12)	
	ZIPCD	CHAR(3)	
	ジユウショ	CHAR(30)	
TAKEODR	ジユチユウ CD	CHAR(7)	CLUSTER KEY
	トクイサキ CD	CHAR(5)	
	シヨウヒン CD	CHAR(6)	
	スウリヨウ	DECIMAL(7,0)	
	ヒキアテリヨウ	DECIMAL(7,0)	
	ジユチユウザン	DECIMAL(7,0)	
	ジユチユウヒヅケ	CHAR(6)	
	ノウキ	CHAR(6)	
STOCK	シヨウヒン CD	CHAR(6)	
	ザイコリヨウ	DECIMAL(7,0)	
	ヒキアテリヨウ	DECIMAL(7,0)	
	ハツチユウテン	DECIMAL(7,0)	
	シイレサキ CD	CHAR(5)	
WAREHUS	シヨウヒン CD	CHAR(6)	
	ニユウコスウリヨウ	DECIMAL(7,0)	
	シイレ NO	INTEGER	
	ニユウコビ	CHAR(6)	
SHIPMNT	シヨウヒン CD	CHAR(6)	
	シユツコスウリヨウ	DECIMAL(7,0)	
	ジユチユウ CD	CHAR(7)	
	ジユチユウビ	CHAR(6)	
SENDODR	ハツチユウ NO	INTEGER	CLUSTER KEY SENDODRX
	シイレサキ CD	CHAR(5)	
	シヨウヒン CD	CHAR(6)	
	ハツチユウリヨウ	DECIMAL(7,0)	

表名	列名	列属性	インデクス名
	ハツチユウビ	CHAR(6)	
	ノウキ	CHAR(6)	
LAYIN	シイレ NO	INTEGER	CLUSTER KEY LAYINX
	シイレサキ CD	CHAR(5)	
	シヨウヒン CD	CHAR(6)	
	シイレリヨウ	DECIMAL(7,0)	
	シイレビ	CHAR(6)	

注

表格納 RD エリア名は RDDATA10 になります。

インデクス格納 RD エリア名は RDINDX10 になります。

表 20–7　列属性とインデクス（英語版の場合）

表名	列名	列属性	インデクス名
CUSTOM	CUSTOM_CD	CHAR(5)	UNIQUE CLUSTER KEY CUSTOMX
	CUSTOM_NAME	CHAR(30)	
	TELNO	CHAR(12)	
	ZIPCD	CHAR(3)	
	ADDRESS	CHAR(30)	
GOODS	PRODUCT_CD	CHAR(6)	UNIQUE CLUSTER KEY GOODSX
	PRODUCT_NAME	CHAR(30)	
	PRICE	DECIMAL(7,0)	
	VENDOR_CD	CHAR(5)	
VENDOR	VENDOR_CD	CHAR(5)	UNIQUE CLUSTER KEY VENDORX
	VENDOR_NAME	CHAR(30)	
	TELNO	CHAR(12)	
	ZIPCD	CHAR(3)	
	ADDRESS	CHAR(30)	
TAKEODR	ORDER_ACCEPTED_CD	CHAR(7)	CLUSTER KEY
	CUSTOM_CD	CHAR(5)	
	PRODUCT_CD	CHAR(6)	
	QUANTITY	DECIMAL(7,0)	
	RESERVED_QUANTITY	DECIMAL(7,0)	

表名	列名	列属性	インデクス名
	SURPLUS	DECIMAL(7,0)	
	ORDER_ACCEPTED_DATE	CHAR(6)	
	DELIVERY_DATE	CHAR(6)	
STOCK	PRODUCT_CD	CHAR(6)	
	STOCK	DECIMAL(7,0)	
	RESERVED_QUANTITY	DECIMAL(7,0)	
	ORDER	DECIMAL(7,0)	
	VENDOR_CD	CHAR(5)	
WAREHUS	PRODUCT_CD	CHAR(6)	
	WAREHOUSE	DECIMAL(7,0)	
	LAY_IN_NO	INTEGER	
	WAREHOUSE_DATE	CHAR(6)	
SHIPMNT	PRODUCT_CD	CHAR(6)	
	SHIPMENT	DECIMAL(7,0)	
	ORDER_ACCEPTED_CD	CHAR(7)	
	ORDER_ACCEPTED_DATE	CHAR(6)	
SENDODR	ORDER_NO	INTEGER	CLUSTER KEY SENDODRX
	VENDOR_CD	CHAR(5)	
	PRODUCT_CD	CHAR(6)	
	ORDER_QUANTITY	DECIMAL(7,0)	
	ORDER_DATE	CHAR(6)	
	DELIVERY_DATE	CHAR(6)	
LAYIN	LAY_IN_NO	INTEGER	CLUSTER KEY LAYINX
	VENDOR_CD	CHAR(5)	
	PRODUCT_CD	CHAR(6)	
	LAY_IN_QUANTITY	DECIMAL(7,0)	
	LAY_IN_DATE	CHAR(6)	

注

表格納 RD エリア名は RDDATA10 になります。

インデクス格納 RD エリア名は RDINDX10 になります。

20.3　サンプルファイルの使用方法

20.3.1　コンフィグレーションファイルの作成

HiRDB/シングルサーバの場合に作成するコンフィグレーションのファイル名と内容，HiRDB/パラレルサーバの場合に作成するコンフィグレーションのファイル名と内容を次の表に示します。

なお，各ファイルを使用する場合，$PDDIR/conf 下に複写した後，注意に従って変更してください。

また，環境変数 PDDIR には HiRDB 運用ディレクトリのパスを設定してください。

- HiRDB/シングルサーバのとき：/opt/HiRDB_S
- HiRDB/パラレルサーバのとき：/opt/HiRDB_P

表 20–8　HiRDB/シングルサーバの場合に作成するファイル名と内容

ファイル名	内容	注意
pdsys	システム共通定義	• ポート番号は 22200 になっています。実行環境に合わせて変更してください。 • ホスト名は host1 になっています。実行環境に合わせて変更してください。
pdutsys	ユニット制御情報定義	特にありません。
sds01	シングルサーバ定義	
sysfmkfs	システムファイル用の HiRDB ファイルシステム領域の作成コマンド	
sysfint	システムファイルの初期コマンド	
logunld	システムログファイルのアンロードコマンド	• あらかじめ，アンロードログファイル格納ディレクトリ（/opt/HiRDB_S/unloadlog ）を作成しておいてください。 • アンロードログファイル格納ディレクトリを変更する場合は，pdlogunld の-o の指定を変更してください。

表 20–9　HiRDB/パラレルサーバの場合に作成するファイル名と内容

ファイル名	内容	注意
pdsys	システム共通定義	• ポート番号は 22200 になっています。実行環境に合わせて変更してください。 • ホスト名はそれぞれ host1，host2 になっています。実行環境に合わせて変更してください。
pdutsys1	host1 の unt1 用のユニット制御情報定義	host1 の環境に複写後，pdutsys というファイル名に変更してください。

ファイル名	内容	注意
pdutsys2	host2 の unt2 用のユニット制御情報定義	host2 の環境に複写後，pdutsys というファイル名に変更してください。
pdsvrc	サーバ共通定義	特にありません。
fes1	フロントエンドサーバ fes1 用の定義	
bes1	バックエンドサーバ bes1 用の定義	
bes2	バックエンドサーバ bes2 用の定義	
dic1	ディクショナリサーバ dic1 用の定義	
sysfmkfs1	host1 の unt1 で使用するシステムファイル用の HiRDB ファイルシステム領域の作成コマンド	host1 から入力してください。
sysfmkfs2	host2 の unt2 で使用するシステムファイル用の HiRDB ファイルシステム領域の作成コマンド	host2 から入力してください。
sysfint	host1，host2 で使用するシステムファイルの初期化コマンド	• host1 から入力してください。 • ホスト名はそれぞれ host1，host2 になっています。実行環境に合わせて変更してください。
feslogunld	pdfes のシステムログファイルのアンロードコマンド	• あらかじめ，アンロードログファイル格納ディレクトリ（/opt/HiRDB_P/unloadlog ）を作成しておいてください。 • アンロードログファイル格納ディレクトリを変更する場合は，pdlogunld の-o の指定を変更してください。
bes1logunld	pdbes1 のシステムログファイルのアンロードコマンド	
bes2logunld	pdbes2 のシステムログファイルのアンロードコマンド	
dic1logunld	pddic のシステムログファイルのアンロードコマンドです。	

(1)　HiRDB ファイルシステム領域の確保及び初期化

(a)　データベース用の HiRDB ファイルシステム領域の作成

次に示すように実行し，HiRDB ファイルシステム領域を作成してください。

HiRDB/シングルサーバの場合

次に示すシェルスクリプトを実行してください。

```
$PDDIR/sample/sampleconf/fmkfile
```

注　環境変数 PDDIR は必ず設定してください。

HiRDB/パラレルサーバの場合

各ホスト（host1，host2）で，次に示すシェルスクリプトを実行してください。

```
$PDDIR/sample/sampleconf/fmkfile
```

注　環境変数 PDDIR は必ず設定してください。

(b)　システムログなどのファイルの確保

次に示すように実行し，システムログなどのファイルを確保してください。

HiRDB/シングルサーバの場合

次に示すシェルスクリプトを実行してください。

```
$PDDIR/sample/sampleconf/sysfmkfs
```

注　環境変数 PDDIR は必ず設定してください。

HiRDB/パラレルサーバの場合

host1 側では，次に示すシェルスクリプトを実行してください。

```
$PDDIR/sample/sampleconf/sysfmkfs1
```

注　環境変数 PDDIR は必ず設定してください。

host2 側では，次に示すシェルスクリプトを実行してください。

```
$PDDIR/sample/sampleconf/sysfmkfs2
```

注　環境変数 PDDIR は必ず設定してください。

(c)　システムログなどのファイルの初期化

次に示すように実行し，システムログなどのファイルを初期化してください。

HiRDB/シングルサーバの場合

次に示すシェルスクリプトを実行してください。

```
$PDDIR/sample/sampleconf/sysfint
```

注　環境変数 PDDIR は必ず設定してください。

HiRDB/パラレルサーバの場合

システムマネジャを定義したホスト側（host1）で，次に示すシェルスクリプトを実行してください。

```
$PDDIR/sample/sampleconf/sysfint
```

注　環境変数 PDDIR は必ず設定してください。

(2) 初期設定制御文ファイルの作成

次に示すように実行し，初期設定制御文ファイルを作成してください。

なお，初期設定制御文ファイル名は，$PDDIR/sample/sampleconf/rdinit01 です。

HiRDB/シングルサーバの場合

次に示すシェルスクリプトを実行してください。

```
$PDDIR/sample/sampleconf/mkinit
```

注　環境変数 PDDIR は必ず設定してください。

HiRDB/パラレルサーバの場合

ディクショナリサーバ側で，次に示すシェルスクリプトを実行してください。

```
$PDDIR/sample/sampleconf/mkinit
```

注　環境変数 PDDIR は必ず設定してください。

(3) HiRDB の開始

pdstart コマンドで HiRDB を開始します。

(4) 表定義

データベース定義ユティリティ（pddef）を次に示すように実行してください。

日本語版の場合

```
$PDDIR/bin/pddef < $PDDIR/sample/sampleDB/tblecreate
```

注

環境変数 PDUSER，PDDIR，PDNAMEPORT 及び PDHOST に値を設定してください。

英語版の場合

```
$PDDIR/bin/pddef < $PDDIR/sample/sampleDB/tblecreate_e
```

注

環境変数 PDUSER，PDDIR，PDNAMEPORT 及び PDHOST に値を設定してください。

(5) データロード

次に示すシェルスクリプトを実行し，データベース作成ユティリティ（pdload）でデータロードをしてください。データロードについては，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。

日本語版の場合

```
$PDDIR/sample/sampleDB/loadinf
```

英語版の場合

```
$PDDIR/sample/sampleDB/loadinf_e
```

(6) コマンドを別名で実行するためのシェルスクリプトの作成

HiRDB のコマンド名と OS やほかのプログラムが提供しているコマンド名が同一になり，HiRDB のコマンドが実行できない場合があります。このような場合，次に示す回避策があります。

- 環境変数の設定で HiRDB のコマンドを優先します。
- 絶対パスを指定してコマンドを実行します。

上記二つの回避策が実行できない場合に，HiRDB のコマンドを任意の名称で実行する方法があります。HiRDB ではこの方法を実現するために必要なシェルスクリプトのひな形を提供しています。

(a) HiRDB が提供しているシェルスクリプトのひな形ファイル名

HiRDB では，プラットフォームに合わせたシェルスクリプトのひな形ファイルを提供しています。HiRDB が提供しているシェルスクリプトのひな形ファイルと内容を次の表に示します。ファイルの格納場所を次に示します。

- HiRDB/シングルサーバのとき：/opt/HiRDB_S/sample/sampleconf
- HiRDB/パラレルサーバのとき：/opt/HiRDB_P/sample/sampleconf

表 20-10 コマンド名の別名実行用シェルスクリプトのひな形ファイルと内容

ファイル名	内容	注意
aliascmdbsh	Bourne シェル用のひな形ファイル	HiRDB 運用ディレクトリ下の bin 及び lib の二つのディレクトリ以下にはコピーしないでください。
aliascmdcsh	C シェル用のひな形ファイル	

(b) コマンド名の別名を作成する手順

次に示す手順で，コマンドの別名を作成します。

1. シェルスクリプトのひな形ファイルを任意のディレクトリにコピーします。複数のコマンドの別名を作成したい場合は，コマンドの数だけコピーしてください。ただし，HiRDB 運用ディレクトリ下の bin 及び lib のディレクトリ以下にコピーしないでください。
2. ひな形ファイルのコピー先のディレクトリをサーチパスとして環境変数 PATH 又は path に設定します。
3. 手順 1 でコピーしたファイル名を HiRDB のコマンドの別名として使用したいコマンド名にします。例えば，データベース構成変更ユティリティ（pdmod）のコマンド名を変更したい場合，「hirmod」のように変更します。

4. コピーしたひな形ファイルを開き，次の図に示す「cc....cc」の部分を別名で実行させたい HiRDB のコマンド名に変更します。

図 20-4　Bourne シェルのひな形ファイル

```
#!/bin/sh -f

#set HiRDB command here
HIRDB_COMMAND= cc....cc    ← HiRDBのコマンドを設定する箇所(例：pdmodをhirmodに変更する場合，
                                                              pdmodを設定します)
${PDDIR}/bin/${HIRDB_COMMAND} $*
exit $?
```

図 20-5　C シェルのひな形ファイル

```
#!/bin/csh -f

#set HiRDB command here
set HIRDB_COMMAND= cc....cc    ← HiRDBのコマンドを設定する箇所(例：pdmodをhirmodに変更する
                                                          場合，pdmodを設定します)
${PDDIR}/bin/${HIRDB_COMMAND} $argv
exit $status
```

以上の設定によって，HiRDB のコマンドを任意の名称で実行できます。また，オプションも HiRDB のコマンドと同様に指定できます。

(c)　注意

1. ひな形ファイルのコピー後の名称には，HiRDB のコマンドとは異なる名称を指定してください。
2. HiRDB 運用ディレクトリ下の二つのディレクトリ（$PDDIR/bin 及び$PDDIR/lib）は，pdsetup -d コマンド実行時にディレクトリごと削除される可能性があります。このため，この二つのディレクトリ以下にひな形ファイルをコピーしないでください。
3. ひな形ファイルの内容は，HiRDB のコマンド名の設定箇所以外は変更しないでください。
4. 作成した別名コマンドを実行中に，そのコマンド処理を中断させる場合には別名プロセスの延長で起動している HiRDB コマンドプロセスを停止させてください。別名プロセスを停止させるだけでは，HiRDB コマンドプロセスは停止しません。
5. 作成した別名コマンドを実行して，HiRDB コマンドが応答入力待ち状態のときに別名プロセスを停止させると，HiRDB コマンドの実行がエラーになったり応答入力待ち状態を継続している場合があります。応答入力待ちが継続されていた場合には，HiRDB コマンドプロセスを停止させてください。

20.3.2　サンプルで使用する HiRDB ファイルシステム領域名とユーザ作成ファイル名

サンプルで使用する HiRDB ファイルシステム領域名とサイズ及びユーザ作成ファイル名について説明します。

なお，ここで示す内容は，サンプルデータベースで使用している名称を示したもので，この名称と同じにしなければならないというものではありません。

(1) HiRDB ファイルシステム領域名とサイズ

サンプルで使用する HiRDB/シングルサーバの場合の HiRDB ファイルシステム領域名とサイズ及び HiRDB/パラレルサーバの場合の HiRDB ファイルシステム領域名とサイズを次の表に示します。

表 20-11　HiRDB ファイルシステム領域名とサイズ（HiRDB/シングルサーバの例）

項番	区分	ファイル	サイズ (MB)	HiRDB ファイルシステム領域名	備考
1	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	74	/HiRDB_S/rdsys011 /HiRDB_S/rdsys012 /HiRDB_S/rdsys013 /HiRDB_S/rdsys014 /HiRDB_S/rdsys015 /HiRDB_S/rdsys016	通常ファイル
2	システム用 RD エリア	• マスタディレクトリ • データディレクトリ • データディクショナリ	20	/HiRDB_S/rdsys02	
3	作業表用ファイル	−	20	/HiRDB_S/rdsys03	
4	ユーザ用 RD エリア	−	40	/HiRDB_S/rdsys04	

（凡例）−：該当しません。

表 20-12　HiRDB ファイルシステム領域名とサイズ（HiRDB/パラレルサーバの例）

項番	区分	ファイル		サイズ（MB）	HiRDB ファイルシステム領域名	備考
1	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	host1	156	/HiRDB_P/rdsys011 /HiRDB_P/rdsys012 /HiRDB_P/rdsys013 /HiRDB_P/rdsys014	通常ファイル
			host2	74	/HiRDB_P/rdsys015 /HiRDB_P/rdsys016	
2	システム用 RD エリア	• マスタディレクトリ • データディレクトリ • データディクショナリ		20	/HiRDB_P/rdsys02	
3	作業表用ファイル	−		20	/HiRDB_P/rdsys03	
4	ユーザ用 RD エリア	−		40	/HiRDB_P/rdsys04	

（凡例）－：該当しません。

(2) ユーザ作成ファイル名

サンプルで使用する HiRDB/シングルサーバの場合のユーザ作成ファイル名と HiRDB/パラレルサーバの場合のユーザ作成ファイル名を次の表に示します。

表 20-13 ユーザ作成ファイル名（HiRDB/シングルサーバの例）

項番	名称		ファイル名	備考
1	定義ファイル	システム共通定義	$PDCONFPATH/pdsys	ディレクトリはインストールで作成済み
2		ユニット制御情報定義	$PDDIR/conf/pdutsys	
3		シングルサーバ定義	$PDDIR/conf/sds01	
4	システムログファイル		/HiRDB_S/rdsys011/log1 /HiRDB_S/rdsys012/log2 /HiRDB_S/rdsys013/log3 /HiRDB_S/rdsys014/log4	4 グループ
5	シンクポイントダンプファイル		/HiRDB_S/rdsys014/spd1 /HiRDB_S/rdsys015/spd2 /HiRDB_S/rdsys016/spd3	3 グループ
6	ステータスファイル	ユニット用	/HiRDB_S/rdsys011/utssts1a /HiRDB_S/rdsys012/utssts1b /HiRDB_S/rdsys013/utssts2a /HiRDB_S/rdsys014/utssts2b /HiRDB_S/rdsys015/utssts3a /HiRDB_S/rdsys016/utssts3b	ユニットごとに二重化×3 個
7		シングルサーバ用	/HiRDB_S/rdsys011/stsla /HiRDB_S/rdsys012/stslb /HiRDB_S/rdsys013/sts2a /HiRDB_S/rdsys014/sts2b /HiRDB_S/rdsys015/sts3a /HiRDB_S/rdsys016/sts3b	サーバごとに二重化×3 個
8	システム用 RD エリア	マスタディレクトリ	/HiRDB_S/rdsys02/rdmast	RD エリア名 ：RDMAST
9		データディレクトリ	/HiRDB_S/rdsys02/rddirt	RD エリア名 ：RDDIRT
10		データディクショナリ	/HiRDB_S/rdsys02/rddict	RD エリア名 ：RDDICT
11	作業表用ファイル		/HiRDB_S/rdsys03	－

項番	名称		ファイル名	備考
12	RPC トレースファイル		/HiRDB_S/spool/pdrpctr	－
13	ユーザ用 RD エリア	データ部	/HiRDB_S/rdsys04/rddata10	RD エリア名 ：RDDATA10
14		インデクス部	/HiRDB_S/rdsys04/rdindx10	RD エリア名 ：RDINDX10

（凡例）－：該当しません。

表 20-14　ユーザ作成ファイル名（HiRDB/パラレルサーバの例）

項番	名称			ファイル名	備考
1	定義ファイル	システム共通定義		$PDCONFPATH/pdsys	ディレクトリはインストールで作成済み
2		ユニット制御情報定義		$PDDIR/conf/pdutsys	
3		サーバ共通定義		$PDCONFPATH/pdsvrc	
4		フロントエンドサーバ定義		$PDDIR/conf/fes1	
5		ディクショナリサーバ定義		$PDDIR/conf/dic1	
6		バックエンドサーバ定義	host1	$PDDIR/conf/bes1	
			host2	$PDDIR/conf/bes2	
7	システムログファイル		host1	/HiRDB_P/rdsys011/feslog1 /HiRDB_P/rdsys012/feslog2 /HiRDB_P/rdsys013/feslog3 /HiRDB_P/rdsys014/feslog4 /HiRDB_P/rdsys011/diclog1 /HiRDB_P/rdsys012/diclog2 /HiRDB_P/rdsys013/diclog3 /HiRDB_P/rdsys014/diclog4 /HiRDB_P/rdsys011/bes1log1 /HiRDB_P/rdsys012/bes1log2 /HiRDB_P/rdsys013/bes1log3 /HiRDB_P/rdsys014/bes1log4	サーバごとに 4 グループ
			host2	/HiRDB_P/rdsys011/bes2log1 /HiRDB_P/rdsys012/bes2log2 /HiRDB_P/rdsys013/bes2log3 /HiRDB_P/rdsys014/bes2log4	
8	シンクポイントダンプファイル		host1	/HiRDB_P/rdsys014/fesspd1 /HiRDB_P/rdsys015/fesspd2 /HiRDB_P/rdsys016/fesspd3	サーバごとに 3 グループ

項番	名称			ファイル名	備考
				/HiRDB_P/rdsys014/dicspd1 /HiRDB_P/rdsys015/dicspd2 /HiRDB_P/rdsys016/dicspd3 /HiRDB_P/rdsys014/bes1spd1 /HiRDB_P/rdsys015/bes1spd2 /HiRDB_P/rdsys016/bes1spd3	
			host2	/HiRDB_P/rdsys014/bes2spd1 /HiRDB_P/rdsys015/bes2spd2 /HiRDB_P/rdsys016/bes2spd3	
9	ステータスファイル	ユニット用	host1	/HiRDB_P/rdsys011/ut1sts1a /HiRDB_P/rdsys012/ut1sts1b /HiRDB_P/rdsys013/ut1sts2a /HiRDB_P/rdsys014/ut1sts2b /HiRDB_P/rdsys015/ut1sts3a /HiRDB_P/rdsys016/ut1sts3b	ユニットごとに二重化×3個
			host2	/HiRDB_P/rdsys011/ut2sts1a /HiRDB_P/rdsys012/ut2sts1b /HiRDB_P/rdsys013/ut2sts2a /HiRDB_P/rdsys014/ut2sts2b /HiRDB_P/rdsys015/ut2sts3a /HiRDB_P/rdsys016/ut2sts3b	
10		サーバ用	host1	/HiRDB_P/rdsys011/fessts1a /HiRDB_P/rdsys012/fessts1b /HiRDB_P/rdsys013/fessts2a /HiRDB_P/rdsys014/fessts2b /HiRDB_P/rdsys015/fessts3a /HiRDB_P/rdsys016/fessts3b /HiRDB_P/rdsys011/dicsts1a /HiRDB_P/rdsys012/dicsts1b /HiRDB_P/rdsys013/dicsts2a /HiRDB_P/rdsys014/dicsts2b /HiRDB_P/rdsys015/dicsts3a /HiRDB_P/rdsys016/dicsts3b /HiRDB_P/rdsys011/bes1sts1a /HiRDB_P/rdsys012/bes1sts1b /HiRDB_P/rdsys013/bes1sts2a /HiRDB_P/rdsys014/bes1sts2b /HiRDB_P/rdsys015/bes1sts3a /HiRDB_P/rdsys016/bes1sts3b	サーバごとに二重化×3個
			host2	/HiRDB_P/rdsys011/bes2sts1a	

項番	名称			ファイル名	備考
				/HiRDB_P/rdsys012/bes2sts1b /HiRDB_P/rdsys013/bes2sts2a /HiRDB_P/rdsys014/bes2sts2b /HiRDB_P/rdsys015/bes2sts3a /HiRDB_P/rdsys016/bes2sts3b	
11	システム用RDエリア	マスタディレクトリ		/HiRDB_P/rdsys02/rdmast	RDエリア名 ：RDMAST
12		データディレクトリ		/HiRDB_P/rdsys02/rddirt	RDエリア名 ：RDDIRT
13		データディクショナリ		/HiRDB_P/rdsys02/rddict	RDエリア名 ：RDDICT
14	作業表用ファイル			/HiRDB_P/rdsys03	－
15	RPCトレースファイル			/HiRDB_P/spool/pdrpctr	－
16	ユーザ用RDエリア	bes1	データ部	/HiRDB_P/rdsys04/rddata10	RDエリア名 ：RDDATA10
17			インデクス部	/HiRDB_P/rdsys04/rdindx10	RDエリア名 ：RDINDX10
18		bes2	データ部	/HiRDB_P/rdsys04/rddata20	RDエリア名 ：RDDATA20
19			インデクス部	/HiRDB_P/rdsys04/rdindx20	RDエリア名 ：RDINDX20

（凡例）－：該当しません。

21 HiRDB サーバと HiRDB クライアント間の通信

この章では，HiRDB サーバと HiRDB クライアントの接続方法，DNS サーバやファイアウォールがある場合の設定などについて説明します。

21.1　HiRDB サーバと HiRDB クライアントの接続方法

HiRDB クライアントが HiRDB サーバに接続するには，クライアント環境定義の次に示すオペランドで**ホスト名**（又は **IP アドレス**）を指定する必要があります。

- PDHOST
- PDFESHOST

これらのオペランドに指定するホスト名は，システム共通定義の pdunit オペランドで指定したホスト名を指定します。

ただし，ネットワークの構成によっては pdunit オペランドで指定したホスト名を指定しても接続できない場合があります。DNS を使用している環境では「FQDN を指定した HiRDB サーバへの接続方法」を，HiRDB のサーバ間で使用しているネットワークと HiRDB クライアントと HiRDB サーバ間で使用しているネットワークが異なる場合は「マルチコネクションアドレス機能を使用した HiRDB サーバへの接続方法」を参照してください。

21.1.1　FQDN を指定した HiRDB サーバへの接続方法

pdunit オペランドで指定したホスト名は，HiRDB サーバにアクセスするすべてのクライアントマシンの hosts ファイルに IP アドレスとともに登録する必要があります。しかし，DNS を使用すると，hosts ファイルに登録する必要がなくなるため，登録処理や IP アドレス変更に伴う hosts ファイルの変更処理が不要になります。

ドメイン内に含まれるホスト上で稼働する HiRDB サーバと接続する場合は，PDHOST，PDFESHOST にサーバマシンの **FQDN** を指定することで，HiRDB サーバに接続できます。

クライアント環境定義に指定できる名称を次の表に示します。

表 21–1　クライアント環境定義に指定できる名称

クライアント環境定義に指定する名称	バージョン 05-02 以前	バージョン 05-03 以降
ホスト名	○	○
FQDN	×	○ ※

（凡例）

○：指定できます。

×：指定できません。

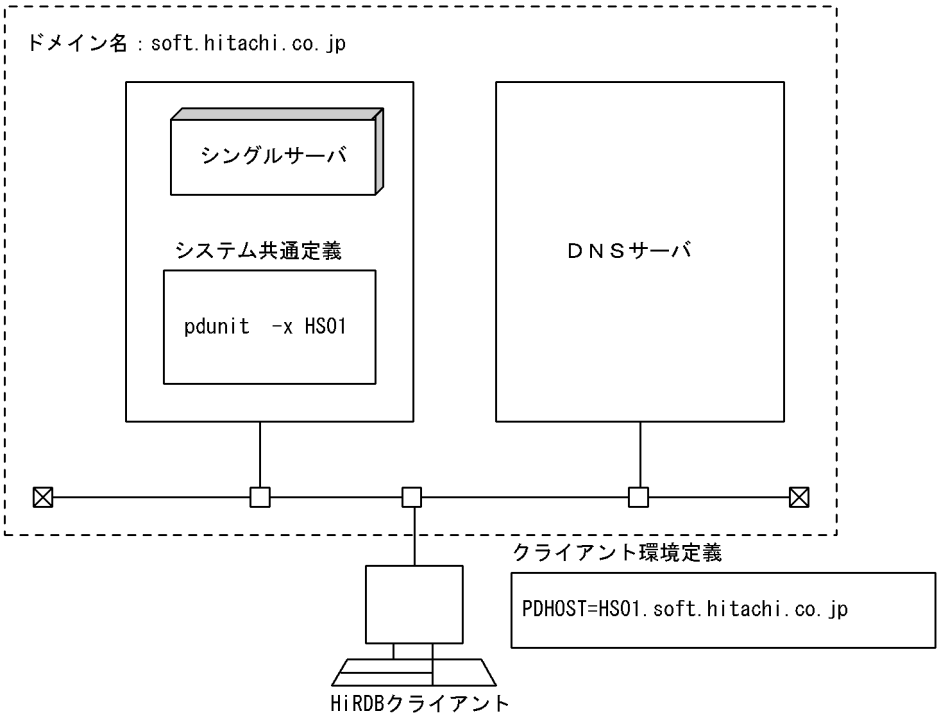
注※

大規模なネットワーク環境で，ホスト名や IP アドレスの登録，IP アドレスの変更に伴って hosts ファイルを変更したくない場合に使用します。

(1) FQDN を指定して HiRDB サーバへ接続する場合のネットワーク構成例と定義例

FQDN を指定して HiRDB サーバへ接続する場合のネットワーク構成例と定義例を次の図に示します。

図 21-1 FQDN を指定して HiRDB サーバへ接続する場合のネットワーク構成例と定義例



〔説明〕

- システム共通定義の pdunit オペランドの-x オプションには，HiRDB サーバで使用するネットワークのホスト名（HS01）を指定します。
- クライアント環境定義の PDHOST には，HiRDB サーバの FQDN（HS01.soft.hitachi.co.jp）を指定します。

(2) 注意事項

1. バージョン 05-03 より前の HiRDB サーバに接続する場合は，クライアント環境定義の PDHOST，PDFESHOST に FQDN を指定できません。指定するとクライアント最大待ち時間（PDCWAITTIME 指定値）経過後，キャンセル処理できないでサーバプロセスが残ることがあります。
2. HiRDB サーバで定義するホスト名に FQDN を指定できません。
3. HiRDB サーバが使用するネットワークと HiRDB クライアントが接続するネットワークが異なる場合は，マルチコネクションアドレス機能を使用して，HiRDB サーバに接続してください。マルチコネクションアドレス機能については，「マルチコネクションアドレス機能を使用した HiRDB サーバへの接続方法」を参照してください。

21.1.2　マルチコネクションアドレス機能を使用したHiRDBサーバへの接続方法

ネットワークの構成によってはpdunitオペランドで指定したホスト名を指定しても，HiRDBサーバと接続できないことがあります。例えば，HiRDBクライアントとHiRDBサーバ間で使用しているネットワークと，HiRDBサーバのサーバマシン間で使用しているネットワークが異なる場合がこれに該当します。

このような場合，**マルチコネクションアドレス機能**を使用します。この機能を使用すると，PDHOST又はPDFESHOSTオペランドに，pdunitオペランドと同じホスト名を指定しなくても，HiRDBサーバと接続できるようになります。

(1)　マルチコネクションアドレス機能の使用方法

マルチコネクションアドレス機能を使用するには，システム共通定義のpdstartオペランドで-mオプションを指定します。

系切り替え構成を適用する際は，HiRDBサーバ間で使用するネットワークのIPアドレスを引き継がない場合でも，HiRDBクライアントとHiRDBサーバ間で使用するネットワークのIPアドレス（クライアントの接続先となるIPアドレス）は引き継ぐ構成としてください。その場合の定義例については，「HiRDB/パラレルサーバの場合（IPアドレスを引き継ぐ系切り替えをする場合）」を参照してください。

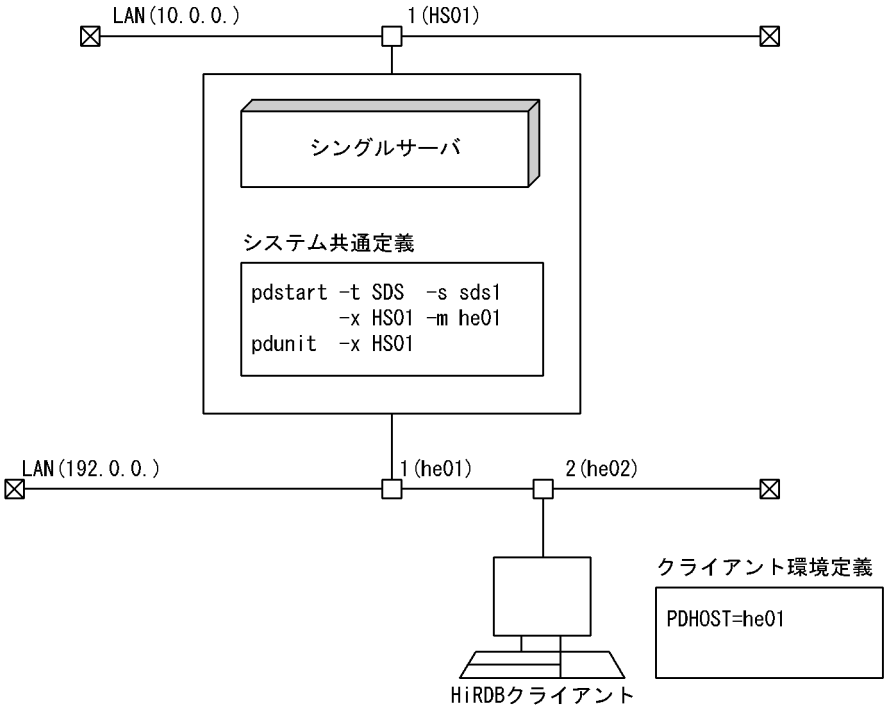
-m及び-nオプションには，HiRDBクライアントがネットワークを経由して接続できるHiRDBサーバのホスト名を指定します。pdunitオペランドに指定したホスト名と同じ必要はありません。

(2)　マルチコネクションアドレス機能を使用したネットワーク構成例と定義例

(a)　HiRDB/シングルサーバの場合

マルチコネクションアドレス機能を使用したネットワーク構成例と定義例（HiRDB/シングルサーバの場合）を次の図に示します。

図 21-2　ネットワーク構成例と定義例（HiRDB/シングルサーバの場合）



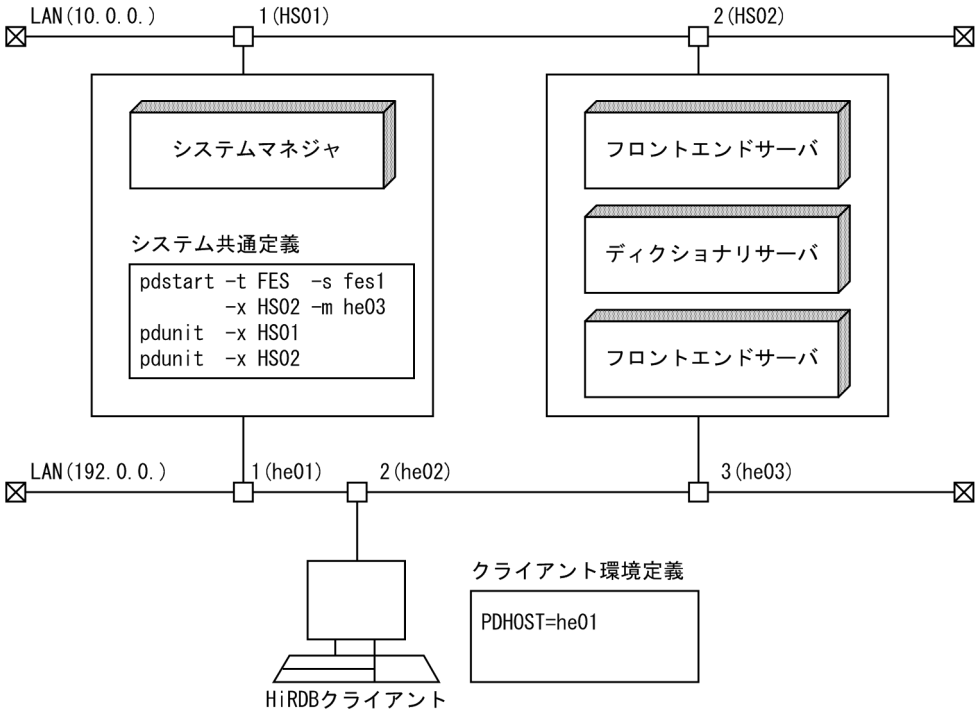
［説明］

- pdunit オペランドの-x オプションには，HiRDB サーバ間で使用するネットワークのホスト名（HS01）を指定します。
- pdstart オペランドの-m オプションには，HiRDB クライアントと HiRDB/シングルサーバ間で使用するネットワークのホスト名（he01）を指定します。
- クライアント環境定義の PDHOST オペランドには，HiRDB クライアントと HiRDB/シングルサーバ間で使用するネットワークのホスト名（he01）を指定します。

(b)　HiRDB/パラレルサーバの場合

マルチコネクションアドレス機能を使用したネットワーク構成例と定義例（HiRDB/パラレルサーバの場合）を次の図に示します。

図 21-3　ネットワーク構成例と定義例（HiRDB/パラレルサーバの場合）



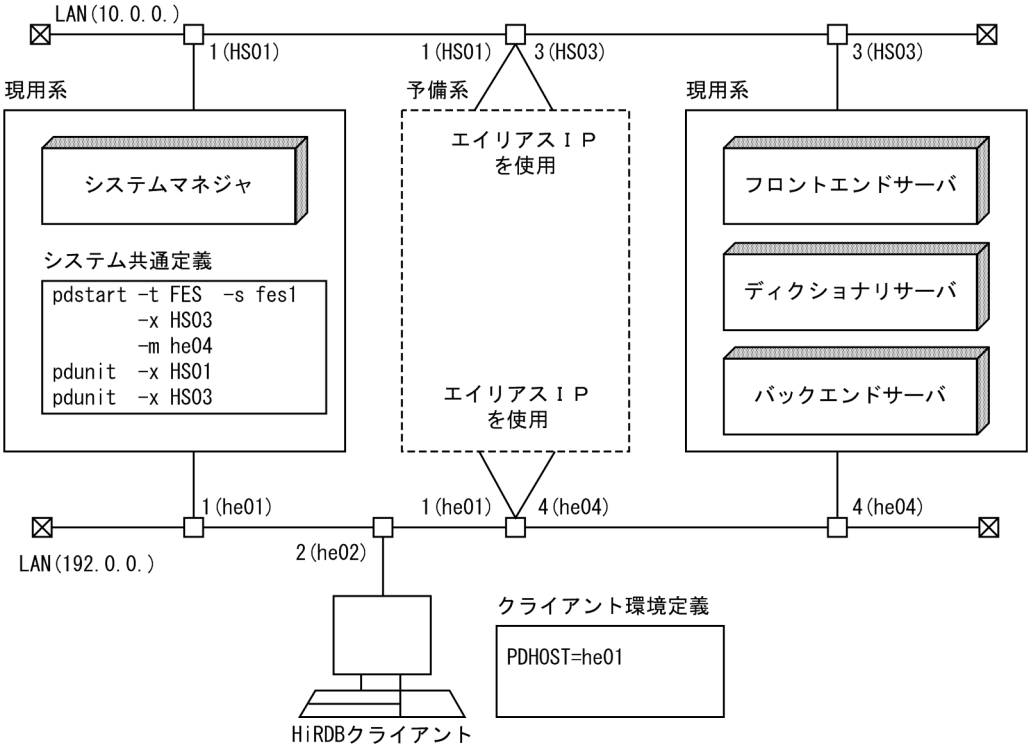
〔説明〕

- pdunit オペランドの-x オプションには，HiRDB サーバ間で使用するネットワークのホスト名（HS01，HS02）を指定します。
- pdstart オペランド（フロントエンドサーバを定義する pdstart オペランド）の-m オプションには，HiRDB クライアントと HiRDB サーバ間で使用するネットワークのホスト名（he03）を指定します。
- クライアント環境定義の PDHOST オペランドには，HiRDB クライアントと HiRDB サーバ間で使用するネットワークのホスト名（システムマネジャがあるホスト名：he01）を指定します。

(c)　HiRDB/パラレルサーバの場合（IP アドレスを引き継ぐ系切り替えをする場合）

マルチコネクションアドレス機能を使用したネットワーク構成と定義例（IP アドレスを引き継ぐ系切り替えをする場合）を次の図に示します。

図 21-4　ネットワーク構成例と定義例（IP アドレスを引き継ぐ系切り替えをする場合）



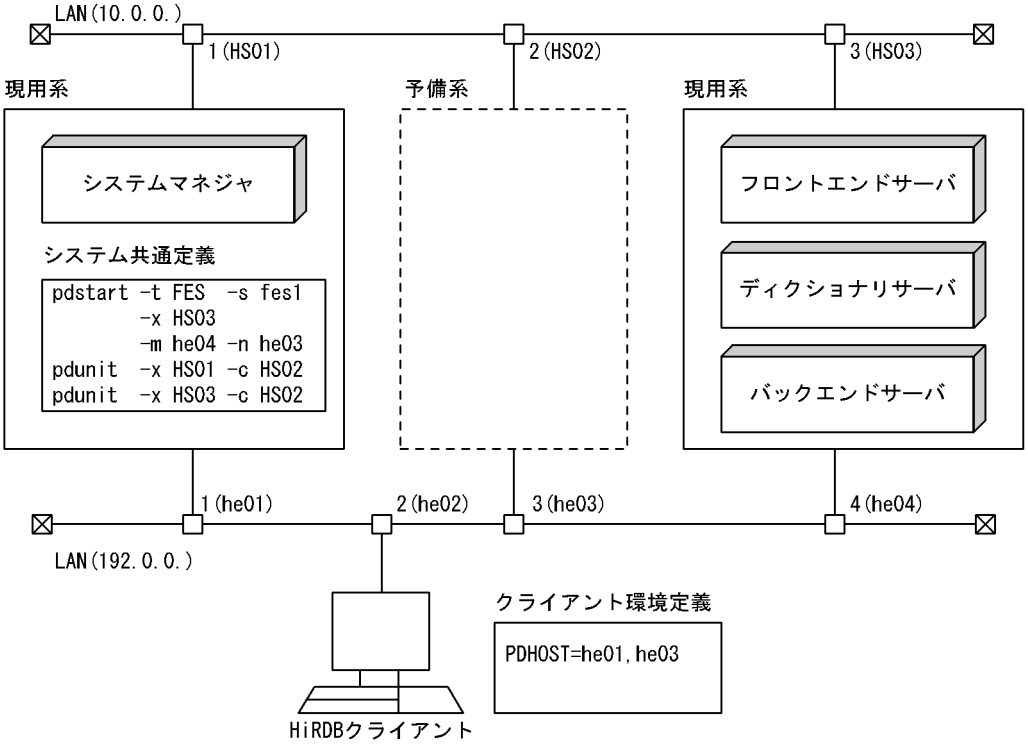
［説明］

- pdunit オペランドの-x オプションには，HiRDB サーバ間で使用するネットワークのホスト名（HS01，HS03）を指定します。
- pdstart オペランド（フロントエンドサーバを定義する pdstart オペランド）の-m オプションには，HiRDB クライアントと HiRDB サーバ間で使用するネットワークの IP アドレスに対応するホスト名（he04）を指定します。-n オプションは省略します。
- クライアント環境定義の PDHOST オペランドには，HiRDB クライアントと HiRDB サーバ間で使用するネットワークのホスト名（システムマネジャがあるホスト名：he01）を指定します。

(d)　HiRDB/パラレルサーバの場合（IP アドレスを引き継がない系切り替えをする場合）

マルチコネクションアドレス機能を使用したネットワーク構成例と定義例（IP アドレスを引き継がない系切り替えをする場合）を次の図に示します。

図 21-5　ネットワーク構成例と定義例（IP アドレスを引き継がない系切り替えをする場合）



〔説明〕

- pdunit オペランドの-x オプションには，HiRDB サーバ間で使用するネットワークのホスト名（HS01，HS03）を指定します。-c オプションには，予備系のホスト名（HS02）を指定します。
- pdstart オペランド（フロントエンドサーバを定義する pdstart オペランド）の-m オプションには，HiRDB クライアントと HiRDB サーバ間で使用するネットワークのホスト名（he04）を指定します。-n オプションには，予備系のホスト名（he03）を指定します。
- クライアント環境定義の PDHOST オペランドには，HiRDB クライアントと HiRDB サーバ間で使用するネットワークのホスト名（システムマネジャがあるホスト名：he01）を指定します。また，予備系のホスト名（he03）も指定します。

21.2　DNS サーバで IP アドレスを管理する場合の設定

DNS サーバで IP アドレスを管理する場合の HiRDB システムには次の 2 とおりがあります。

- サーバマシンが同一ドメイン内に存在する場合
- サーバマシンが複数のドメインにわたって存在する場合

それぞれの HiRDB の設定方法を説明します。

21.2.1　同一ドメイン内での HiRDB の設定方法

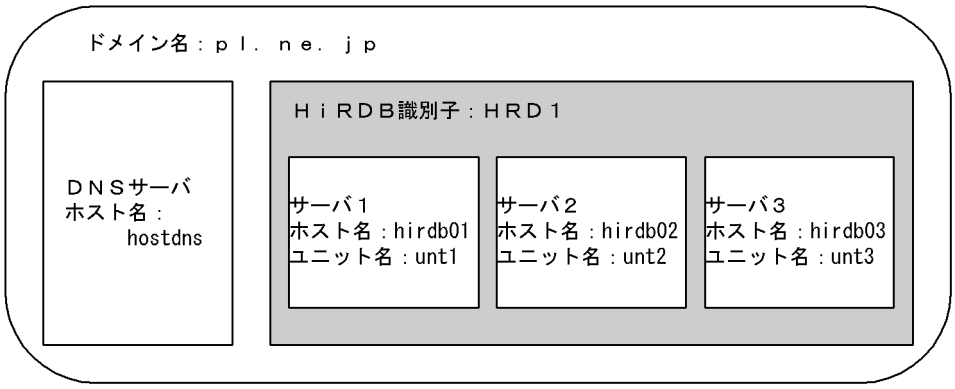
サーバマシンが同一ドメイン内に存在する場合，pdunit オペランド及び pdstart オペランドのホスト名称に「ホスト名」又は「FQDN（完全修飾子付きドメイン名称。ただし 32 文字以内)」のどちらかを指定します。これによって，DNS サーバで IP アドレスが管理でき，hosts ファイルの設定が不要になります。

具体的には，次のオプションでホスト名か FQDN を指定します。

- pdunit オペランド：-x 及び-c オプション
- pdstart オペランド：-x，-m，及び-n オプション

同一ドメインのシステム構成例を次の図に示します。

図 21-6　同一ドメインのシステム構成例



この場合の pdunit -x の指定例を次に示します。

- ホスト名指定の場合
  pdunit -x hirdb01 -u unt1 -d "運用ディレクトリ名" -p ポート番号 …
  pdunit -x hirdb02 -u unt2 -d "運用ディレクトリ名" -p ポート番号 …
  pdunit -x hirdb03 -u unt3 -d "運用ディレクトリ名" -p ポート番号 …
- FQDN 指定の場合
  pdunit -x hirdb01.p1.ne.jp -u unt1 -d "運用ディレクトリ名" -p ポート番号 …
  pdunit -x hirdb02.p1.ne.jp -u unt2 -d "運用ディレクトリ名" -p ポート番号 …

pdunit -x hirdb03.p1.ne.jp -u unt3 -d "運用ディレクトリ名" -p ポート番号 …

21.2.2　複数ドメインでの HiRDB の設定方法

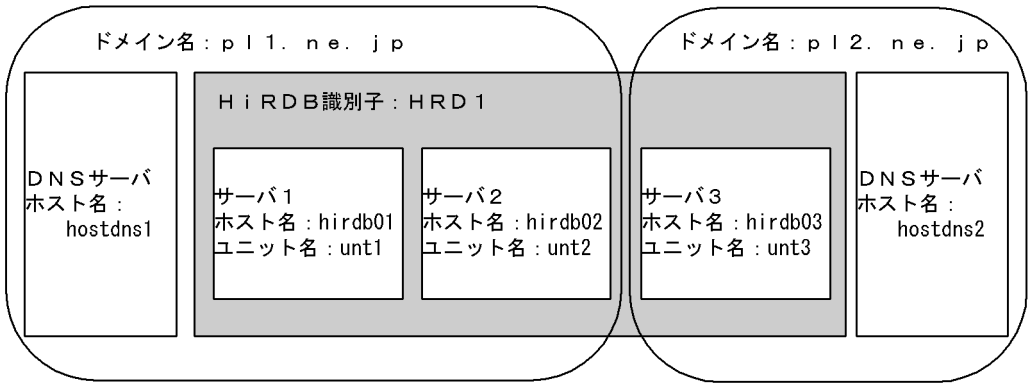
サーバマシンが複数のドメインにわたって存在する場合，pdunit オペランド及び pdstart オペランドのホスト名称に「FQDN（完全修飾子付きドメイン名称：ただし 32 文字以内)」を指定します。これによって，DNS サーバで IP アドレスが管理でき，hosts ファイルの設定が不要になります。

具体的には，次のオプションで FQDN を指定します。

- pdunit オペランド：-x，及び-c オプション
- pdstart オペランド：-x，-m，及び-n オプション

複数ドメインのシステム構成例を次の図に示します。

図 21–7　複数ドメインのシステム構成例



この場合の pdunit -x の指定例を次に示します。

- FQDN 指定の場合
  pdunit -x hirdb01.pl1.ne.jp -u unt1 -d "運用ディレクトリ名" -p ポート番号 …
  pdunit -x hirdb02.pl1.ne.jp -u unt2 -d "運用ディレクトリ名" -p ポート番号 …
  pdunit -x hirdb03.pl2.ne.jp -u unt3 -d "運用ディレクトリ名" -p ポート番号 …

21.3　ファイアウォールや NAT が設置されている場合の設定

HiRDB サーバと HiRDB クライアント間又は HiRDB サーバのユニット間に，ファイアウォールや NAT が設置されている場合の HiRDB の環境設定について説明します。

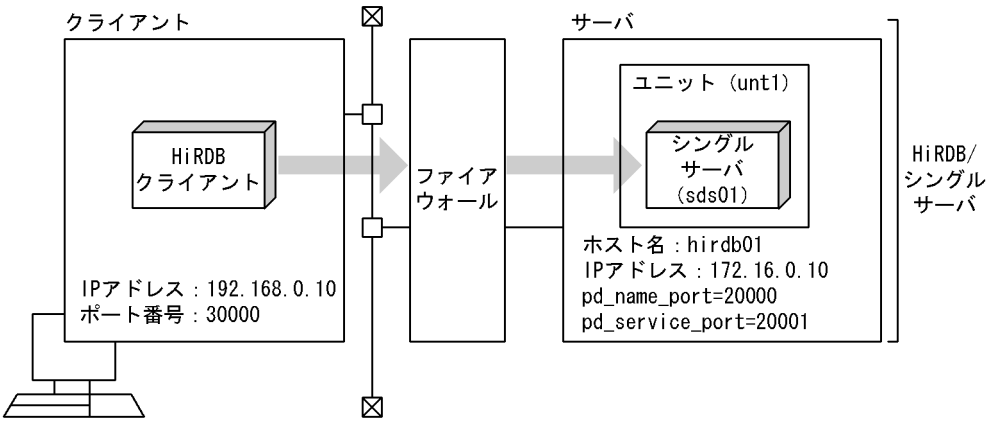
21.3.1　HiRDB/シングルサーバ側にファイアウォールを設置した場合

次の図のように HiRDB/シングルサーバ側にファイアウォールが設置され，そのファイアウォールが次のように設定されているとします。

ファイアウォールの設定

- 方向：受信
- 透過させる IP アドレス：172.16.0.10
- 透過させるポート番号：20000，20001

図 21-8　ファイアウォールが HiRDB/シングルサーバ側に設置されているネットワーク構成例



この場合，サーバマシン及びクライアントマシンの設定は次のようになります。ファイアウォールを設置する場合，次のどれかのオペランドを指定する必要があります。

- pd_service_port オペランド
- pd_scd_port オペランド
- pdunit オペランドの-s オプション

ファイアウォールだけ設置する場合，クライアント環境定義（PDSERVICEPORT オペランド）を指定する必要はありません。

サーバマシンの設定

- システム共通定義ファイル
  set pd_name_port= 20000

set pd_service_port= 20001
pdunit -x hirdb01 -u unt1
pdstart -t SDS -s sds01 -u unt1

クライアントマシンの設定

- クライアント環境定義
  PDHOST　hirdb01
  PDNAMEPORT　20000
  PDCLTRCVPORT　30000※
  PDCONTYPE　ACTIVE※
- hosts ファイル
  172.16.0.10　hirdb01

注※　クライアント側にファイアウォールがある場合は，PDCLTRCVPORT を指定するか，PDCONTYPE に ACTIVE を指定してください。クライアントとサーバの間に NAPT（IP マスカレード）のように，グローバル IP アドレスとローカル IP アドレスを 1 対複数で変換するような機能を設定している場合は，PDCONTYPE に ACTIVE を指定してください。

21.3.2　HiRDB/シングルサーバ側にファイアウォールと NAT を設置した場合

次の図のように HiRDB/シングルサーバ側にファイアウォールと NAT が設置され，それらが次のように設定されているとします。

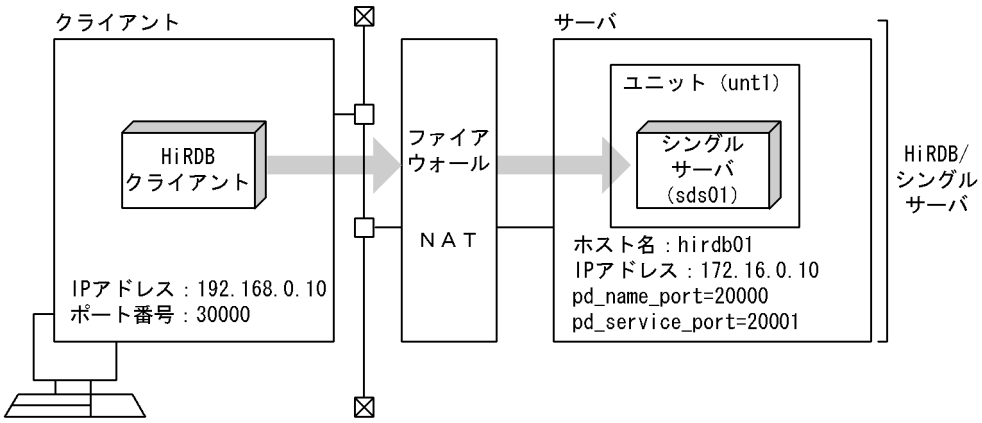
ファイアウォールの設定

- 方向：受信
- 透過させる IP アドレス：172.16.0.10
- 透過させるポート番号：20000，20001

NAT によるアドレス変換

128.1.1.1←→172.16.0.10

図 21-9　ファイアウォールと NAT が HiRDB/シングルサーバ側に設置されているネットワーク構成例



この場合，高速接続機能を使用する設定にしてください。サーバマシン及びクライアントマシンの設定は次のようになります。

サーバマシンの設定

- システム共通定義ファイル
  set pd_name_port = 20000
  set pd_service_port = 20001
  pdunit -x hirdb01 -u unt1
  pdstart -t SDS -s sds01 -u unt1

クライアントマシンの設定

- クライアント環境定義
  PDHOST　hirdb01
  PDNAMEPORT　20000
  PDSERVICEGRP　sds01
  PDSERVICEPORT　20001
  PDSRVTYPE　WS※1
  PDCLTRCVPORT　30000※2
  PDCONTYPE　ACTIVE※2
- hosts ファイル
  128.1.1.1　hirdb01

注※1　Linux 上で HiRDB サーバを動作させる場合は「PC」にしてください。

注※2　クライアント側にファイアウォールがある場合は，PDCLTRCVPORT を指定するか PDCONTYPE に ACTIVE を指定してください。クライアントとサーバの間に NAPT（IP マスカレード）のように，グローバル IP アドレスとローカル IP アドレスを 1 対複数で変換するような機能を設定している場合は，PDCONTYPE に ACTIVE を指定してください。

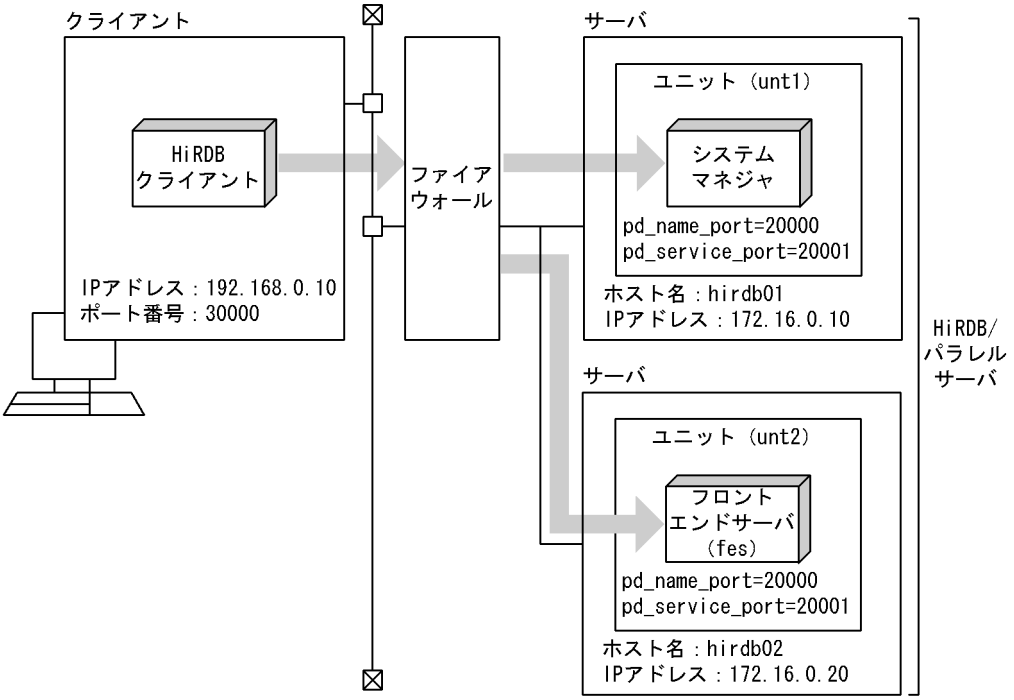
21.3.3 HiRDB/パラレルサーバ側にファイアウォールを設置した場合

次の図のように HiRDB/パラレルサーバ側にファイアウォールが設置され，そのファイアウォールが次のように設定されているとします。

ファイアウォールの設定

- 方向：受信
- 透過させる IP アドレス：172.16.0.10，172.16.0.20
- 透過させるポート番号：20000，20001

図 21-10 ファイアウォールが HiRDB/パラレルサーバ側に設置されているネットワーク構成例



この場合，サーバマシン及びクライアントマシンの設定は次のようになります。ファイアウォールを設置する場合，次のどれかのオペランドを指定する必要があります。

- pd_service_port オペランド
- pd_scd_port オペランド
- pdunit オペランドの-s オプション

ファイアウォールだけ設置する場合，クライアント環境定義（PDSERVICEPORT オペランド）を指定する必要はありません。

サーバマシンの設定

- システム共通定義ファイル

  set pd_name_port = 20000

set pd_service_port = 20001
pdunit -x hirdb01 -u unt1
pdunit -x hirdb02 -u unt2
pdstart -t MGR -u unt1
pdstart -t FES -s fes -u unt2

クライアントマシンの設定

- クライアント環境定義
  PDHOST　hirdb01
  PDNAMEPORT　20000
  PDCLTRCVPORT　30000※
  PDCONTYPE　ACTIVE※
- hosts ファイル
  172.16.0.10　hirdb01
  172.16.0.20　hirdb02

注※　クライアント側にファイアウォールがある場合は，PDCLTRCVPORT を指定するか PDCONTYPE に ACTIVE を指定してください。クライアントとサーバの間に NAPT（IP マスカレード）のように，グローバル IP アドレスとローカル IP アドレスを 1 対複数で変換するような機能を設定している場合は，PDCONTYPE に ACTIVE を指定してください。

21.3.4　HiRDB/パラレルサーバ側にファイアウォールと NAT を設置した場合

次の図のように HiRDB/パラレルサーバ側にファイアウォールと NAT が設置され，それらが次のように設定されているとします。

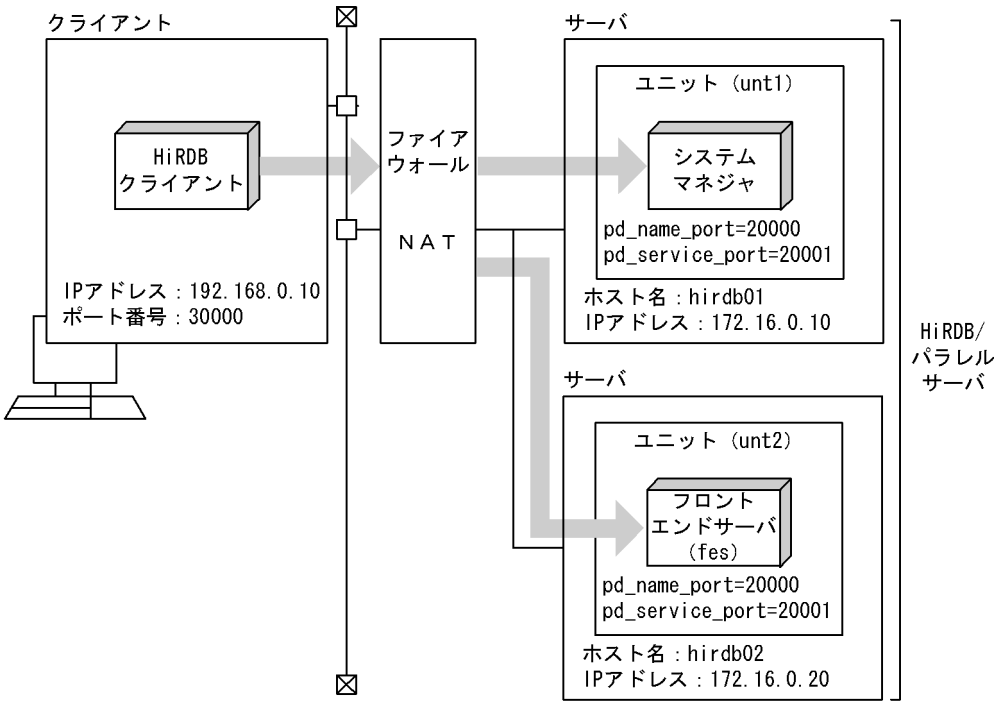
ファイアウォールの設定

- 方向：受信
- 透過させる IP アドレス：172.16.0.10，172.16.0.20
- ポート番号：20000，20001

NAT によるアドレス変換

128.1.1.1←→172.16.0.10
128.1.1.2←→172.16.0.20

図 21-11　ファイアウォールと NAT が HiRDB/パラレルサーバ側に設置されているネットワーク構成例



この場合，高速接続機能を使用する設定にしてください。サーバマシン及びクライアントマシンの設定は次のようになります。

サーバマシンの設定

- システム共通定義ファイル

  set pd_name_port = 20000

  set pd_service_port = 20001

  pdunit -x hirdb01 -u unt1

  pdunit -x hirdb02 -u unt2

  pdstart -t MGR -u unt1

  pdstart -t FES -s fes -u unt2

クライアントマシンの設定

- クライアント環境定義

  PDHOST　hirdb01

  PDNAMEPORT　20000

  PDSERVICEGRP　fes

  PDSERVICEPORT　20001

  PDFESHOST　hirdb02

  PDSRVTYPE　WS※1

  PDCLTRCVPORT　30000※2

PDCONTYPE　ACTIVE※2

- hosts ファイル

  128.1.1.1　hirdb01

  128.1.1.2　hirdb02

注※1　Linux 上で HiRDB サーバを動作させる場合は「PC」にしてください。

注※2　クライアント側にファイアウォールがある場合は，PDCLTRCVPORT を指定するか PDCONTYPE に ACTIVE を指定してください。クライアントとサーバの間に NAPT（IP マスカレード）のように，グローバル IP アドレスとローカル IP アドレスを 1 対複数で変換するような機能を設定している場合は，PDCONTYPE に ACTIVE を指定してください。

21.3.5　HiRDB サーバのユニット間にファイアウォールを設置した場合

HiRDB サーバはすべてのユニット間で通信できる必要があります。次の図のように HiRDB サーバのユニット間にファイアウォールを設置する場合，ファイアウォールでユニット間の通信に使用するポートを透過させる必要があります。

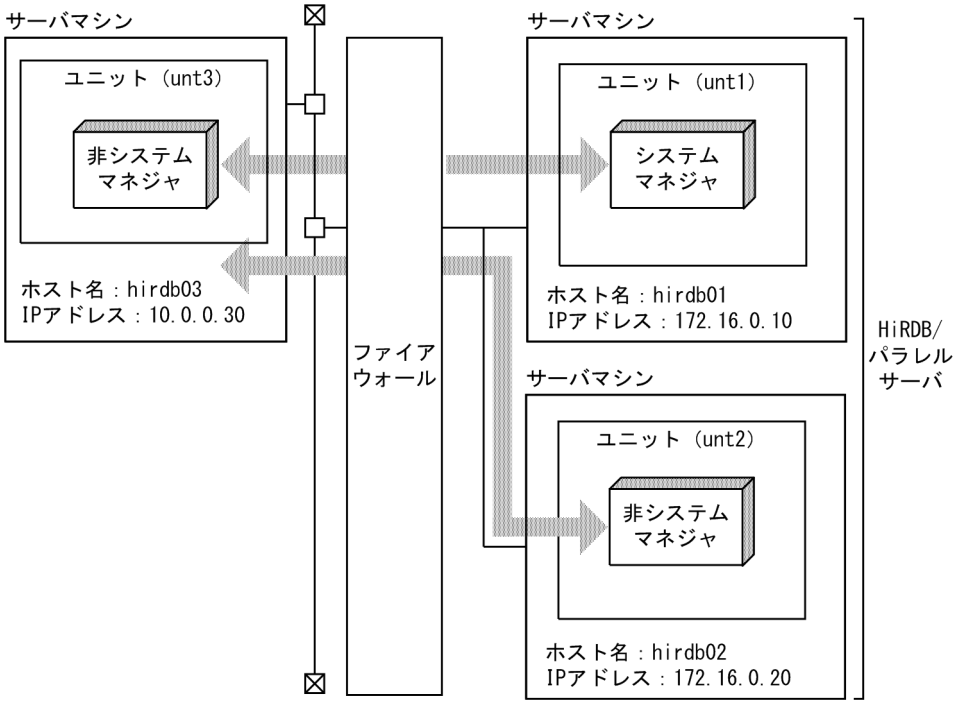
ファイアウォールの設定

- 方向：送信，受信
- 透過させる IP アドレス：172.16.0.10，172.16.0.20，10.0.0.30
- 透過させるポート番号：OS の自動割り当てポート，リモートシェル実行環境で使用するポート※，HiRDB のオペランドで指定したポート

  注※

  リモートシェル実行環境の詳細は，「リモートシェル実行環境の設定」を参照してください。

図 21-12　ファイアウォールが HiRDB サーバのユニット間に設置されているネットワーク構成例



次の場合，HiRDB サーバはユニット間の通信に OS の自動割り当てポートを使用するため，ファイアウォールで OS の自動割り当てポートを透過させる必要があります。

- ユティリティ及び運用コマンドを実行する場合
- pd_registered_port オペランドを指定しない場合
- pd_registered_port オペランドへ指定したポート番号を使用できなかった場合

次の場合，HiRDB サーバはユニット間の通信にリモートシェル又はセキュアシェルを使用するため，ファイアウォールでリモートシェル実行環境が使用するポートを透過させる必要があります。リモートシェル実行環境の詳細は，「リモートシェル実行環境の設定」を参照してください。

- ユティリティ及び運用コマンドを実行する場合
- IP アドレスを引き継がない系切り替え機能を使用する場合

次のオペランドへ指定したポートを HiRDB サーバはユニット間の通信に使用するため，ファイアウォールでオペランドへ指定したポートを透過させる必要があります。

サーバマシンの設定

- システム共通定義ファイル
  pd_name_port オペランド
  pd_service_port オペランド
  pd_scd_port オペランド
  pd_trn_port オペランド

pd_mlg_port オペランド
pd_alv_port オペランド
pd_registered_port オペランド
pdunit オペランドの-p オプション
pdunit オペランドの-s オプション
pdunit オペランドの-t オプション
pdunit オペランドの-m オプション
pdunit オペランドの-a オプション

21.4　HiRDB が使用するポート数

HiRDB の通信処理では，pd_registered_port オペランドの指定がない場合，OS が自動的に割り当てる通信ポート番号を使用します。使用する通信ポート数は，pd_max_users オペランドの値やバックエンドサーバ数の増加に伴って増加します。ポート数が不足すると，処理が中断したり，他プログラムの通信処理に影響を与えたりします。

なお，pd_registered_port オペランドで HiRDB が通信処理で使用するポート番号の範囲を指定（HiRDB 予約ポート機能）できます。HiRDB 予約ポート機能については，「HiRDB 予約ポート機能」を参照してください。

21.4.1　ユニットが使用する通信ポート数の見積もり

HiRDB のユニットが使用する通信ポート数の目安を次に示します。

(1)　HiRDB/シングルサーバの場合

計算式

```
pd_max_usersの値×4＋1000
```

(2)　HiRDB/パラレルサーバの場合

マルチフロントエンドサーバ構成の場合は，フロントエンドサーバごとに f，F を決定し，算出したポート数の合算となります。フロントエンドサーバごとのポート数の目安を算出する計算式を次に示します。

計算式

```
 {b× [k× (B＋F) ＋1]
＋f× (k×B＋D＋2) ＋d× (F＋1) }
×pd_max_usersの値＋1000
```

b：ユニット内のバックエンドサーバ数

B：ユニット外のバックエンドサーバ数

f：ユニット内のフロントエンドサーバ数
　フロントエンドサーバごとに 1，又は 0 のどちらかの値になります。
　ユニット内にフロントエンドサーバがある場合：1
　ユニット外にフロントエンドサーバがある場合：0

F：ユニット外のフロントエンドサーバ数
　フロントエンドサーバごとに 1，又は 0 のどちらかの値になります。
　ユニット外にフロントエンドサーバがある場合：1

ユニット内にフロントエンドサーバがある場合：0

d：ユニット内のディクショナリサーバ数

D：ユニット外のディクショナリサーバ数

k：2÷3

21.4.2 注意事項

- 実行する SQL 文によっては，目安以上のポート数が必要となる場合があります。
- OS が自動的に割り当てるポート数で足りない場合は，pd_registered_port オペランドで指定し直してください。
- ポート番号は，HiRDB が解放しても OS がすぐに解放しない場合（TIME_WAIT 状態）があります。そのため，目安以上のポート数を一時的に使用する場合があります。TIME_WAIT 状態によるポート数不足を回避する方法については，「ポート数不足を回避する方法」を参照してください。
- システム内のバックエンドサーバ数が多いシステム構成の場合，計算した値が OS のポート番号の上限値や pd_registered_port に指定したポート番号を超えるおそれがあります。その場合の対策については，「ユニット数又はサーバ数が多いシステムを構築する場合の考慮点」を参照してください。

21.4.3 計算例

(例 1)

1 ユニット構成の HiRDB/パラレルサーバ（FES，DS，BES が 5 個）で，pd_max_users ＝ 1000 とした場合

```
 {5× [2÷3× (0＋0) +1]
＋1× (2÷3×0＋0＋2) ＋1× (0＋1) }
×1000＋1000＝9000
```

9000 個のポート数が目安となります。

(例 2)

ユニット構成のパラレル（ユニット 1（FES，DS，BES が 2 個），ユニット 2（BES が 3 個））で，pd_max_users ＝ 1000 とした場合

ユニット 1：

```
 {2× [2÷3× (3＋0) ＋1]
＋1× (2÷3×3＋0＋2) ＋1× (0＋1) }
×1000＋1000＝12000
```

ユニット 2：

```
 {3×［2÷3×（2＋1）＋1］
＋0×（2÷3×2＋1＋2）＋0×（1＋1）}
×1000＋1000＝10000
```

それぞれに 12000 個，10000 個のポート数が目安となります。

21.4.4　ポート数不足を回避する方法

次に示すような運用をする場合，大量の TCP ポートが TIME_WAIT 状態となり，システム全体の TCP ポートが不足し，トランザクションがエラーとなったり，HiRDB が異常終了したりすることがあります。

- ユティリティ及びコマンドを連続実行するとき（HiRDB/パラレルサーバの場合）
- UAP の同時接続数が多く，その UAP が次のどれかに該当するトランザクションの実行を繰り返すとき（HiRDB/パラレルサーバの場合）
  - 複数の表を操作する SQL を実行するトランザクション
  - フロータブルサーバを使用するトランザクション
  - 回復不要 FES を使用する更新トランザクション

このような運用をする場合，次に示す設定を行い，TCP ポートが不足しないようにしてください。

なお，ここで説明するオペレーティングシステムパラメタや設定方法は，使用している OS のバージョン，及びカーネルのバージョンごとに異なります。使用している OS のマニュアルを参照し，ここで説明している指定値の目安で示した値を該当するオペレーティングシステムパラメタに設定してください。

(1)　TCP ポートの枯渇を回避する設定

pd_registered_port オペランドで HiRDB が通信処理で使用するポート番号の範囲を拡大（HiRDB 予約ポート機能）すると TCP ポートの枯渇を回避できます。HiRDB 予約ポート機能については，「HiRDB 予約ポート機能」を参照してください。

(2)　TCP ポートを TIME_WAIT 状態に保つ時間を短くする設定（AIX 限定）

TCP ポートを TIME_WAIT 状態に保つ時間を短くするには，次に示すオペレーティングシステムパラメタを指定してください。

- オペレーティングシステムパラメタ：tcp_timewait
- 指定値の目安：1
- パラメタ設定コマンドの例：no コマンド

(3) TCP ポートの OS 自動割り当てポートの範囲を拡大する設定

TCP ポートの OS 自動割り当てポートの範囲を拡大するには，次に示すオペレーティングシステムパラメタを指定してください。

●Linux の場合

- オペレーティングシステムパラメタ：ip_local_port_range
- 指定値の目安：サーバマシン内で使用する予約ポートと重複しない範囲で，OS 自動割り当てポートの範囲を拡大してください。
- オプション設定ファイルの例：/proc/sys/net/ipv4/ip_local_port_range

●AIX の場合

- オペレーティングシステムパラメタ：tcp_ephemeral_high 及び tcp_ephemeral_low
- 指定値の目安：サーバマシン内で使用する予約ポートと重複しない範囲で，OS 自動割り当てポートの範囲を拡大してください。
- パラメタ設定コマンドの例：no コマンド

21.5　HiRDB で指定するポート番号

21.5.1　HiRDB で指定するポート番号の一覧

ポート番号を指定するオペランドを次の表に示します。

表 21-2　HiRDB で指定するポート番号の一覧

設定箇所	指定する環境変数又はオペランド	説明
クライアント環境定義	PDNAMEPORT	HiRDB のポート番号
	HiRDB_PDNAMEPORT	
	PDSERVICEPORT	高速接続用のポート番号
	PDFESHOST	フロントエンドサーバがあるユニットのポート番号
	PDCLTRCVPORT	クライアントの受信ポート番号
	PDASTPORT	HiRDB Control Manager - Agent のポート番号
システム共通定義	pd_name_port	HiRDB のポート番号
	pdunit -p	
	pd_service_port	スケジューラプロセスのポート番号
	pd_scd_port	
	pdunit -s	
	pd_trn_port	トランザクションサーバプロセスのポート番号
	pdunit -t	
	pd_mlg_port	メッセージログサーバプロセスのポート番号
	pdunit -m	
	pd_alv_port	ユニット監視プロセスのポート番号
	pdunit -a	
	pd_registered_port	HiRDB 予約ポート機能で使用するポート番号
ユニット制御情報定義	pd_service_port	スケジューラプロセスのポート番号
	pd_registered_port	HiRDB 予約ポート機能で使用するポート番号

21.5.2　ポート番号の指定方法

ここでは，ポート番号の指定方法について，それぞれ説明します。

(1) クライアント環境定義

マニュアル「HiRDB UAP 開発ガイド」の「クライアント環境定義（環境変数の設定）」を参照してください。

(2) システム共通定義及びユニット制御情報定義

システム共通定義及びユニット制御情報定義のオペランドの指定有無と使用するポート番号について説明します。

オペランドに指定したポート番号を固定したい場合，表中の「使用するポート番号」を参照して，そのポート番号を使用する組み合わせでオペランドを指定してください。例えば，「(a)HiRDB のポート番号」で，HiRDB/パラレルサーバの場合，pd_name_port オペランドに指定したポート番号に固定したいとき，pdunit -p は指定を省略してください。

(a) HiRDB のポート番号

HiRDB のポート番号は，pd_name_port オペランド及び pdunit オペランドの-p オプションに指定します。これらのオペランドの指定有無と使用するポート番号について説明します。

HiRDB/シングルサーバの場合

pd_name_port	pdunit -p	使用するポート番号
あり	あり	pd_name_port の指定
	なし	
なし	あり	20000
	なし	

ユティリティ専用ユニットを配置するときに使用するポート番号を次に示します。

指定箇所		使用するポート番号			
システム共通定義		シングルサーバ側		ユティリティ専用ユニット側	
pd_name_port	pdunit -p	シングルサーバがあるユニット	ユティリティ専用ユニット	シングルサーバがあるユニット	ユティリティ専用ユニット
あり	あり	pd_name_port の指定	pdunit -p の指定	pdunit -p の指定	pd_name_port の指定
	なし	pd_name_port の指定			
なし	あり	20000	pdunit -p の指定	pdunit -p の指定	20000
	なし	20000	20000	20000	20000

HiRDB/パラレルサーバの場合

pd_name_port	pdunit -p	使用するポート番号
あり	あり	pdunit -p の指定
	なし	pd_name_port の指定
なし	あり	pdunit -p の指定
	なし	20000

(b) スケジューラプロセスのポート番号

スケジューラプロセスのポート番号は，次のオペランドに指定します。

- pd_service_port オペランド
- pd_scd_port オペランド
- pdunit オペランドの-s オプション

これらのオペランドの指定有無と使用するポート番号を次に示します。

指定箇所				使用するポート番号
システム共通定義			ユニット制御情報定義	
pd_service_port	pd_scd_port	pdunit -s	pd_service_port	
あり	あり	あり	あり	pdunit -s の指定
			なし	
		なし	あり	pd_scd_port の指定
			なし	
	なし	あり	あり	pdunit -s の指定
			なし	
		なし	あり	ユニット制御情報定義の pd_service_port の指定
			なし	システム共通定義の pd_service_port の指定
なし	あり	あり	あり	pdunit -s の指定
			なし	
		なし	あり	pd_scd_port の指定
			なし	
	なし	あり	あり	pdunit -s の指定
			なし	

指定箇所				使用するポート番号
システム共通定義			ユニット制御情報定義	
pd_service_port	pd_scd_port	pdunit -s	pd_service_port	
		なし	あり	ユニット制御情報定義の pd_service_port の指定
			なし	※

注※

システム共通定義又はユニット制御情報定義に pd_registered_port オペランドを指定した場合，pd_registered_port オペランドに指定した範囲内のポート番号を使用します。pd_registered_port オペランドを指定しない場合，OS が自動的に割り当てたポート番号を使用します。

(c) トランザクションサーバプロセス，メッセージログサーバプロセス，及びユニット監視プロセスのポート番号

トランザクションサーバプロセス，メッセージログサーバプロセス，及びユニット監視プロセスのポート番号は，次に示すオペランドに指定します。

- トランザクションサーバプロセス

  pd_trn_port オペランド及び pdunit オペランドの-t オプション
- メッセージログサーバプロセス

  pd_mlg_port オペランド及び pdunit オペランドの-m オプション
- ユニット監視プロセス

  pd_alv_port オペランド及び pdunit オペランドの-a オプション

これらのオペランドの指定有無と使用するポート番号を次に示します。

pd_trn_port 又は pd_mlg_port 又は pd_alv_port	pdunit -t 又は -m 又は -a	使用するポート番号
あり	あり	pdunit -t，-m，又は-a の指定
	なし	pd_trn_port 又は pd_mlg_port 又は pd_alv_port の指定
なし	あり	pdunit -t，-m，又は-a の指定
	なし	※

注※

システム共通定義又はユニット制御情報定義に pd_registered_port オペランドを指定した場合，pd_registered_port オペランドに指定した範囲内のポート番号を使用します。pd_registered_port オペランドを指定しない場合，OS が自動的に割り当てたポート番号を使用します。

(d)　HiRDB 予約ポート機能で使用するポート番号

HiRDB 予約ポート機能で使用するポート番号は，pd_registered_port オペランドに指定します。このオペランドの指定有無と使用するポート番号を次に示します。

指定箇所		使用するポート番号
システム共通定義	ユニット制御情報定義	
pd_registered_port	pd_registered_port	
あり	あり	ユニット制御情報定義の指定
	なし	システム共通定義の指定
なし	あり	ユニット制御情報定義の指定
	なし	－

（凡例）－：該当しません。

21.5.3　ポート番号の重複に関する注意事項

HiRDB システム定義（表「HiRDB で指定するポート番号の一覧」のシステム共通定義及びユニット制御情報定義）で指定するポート番号の重複についての注意事項を次に示します。

- 同じユニットでは，各オペランドに異なるポート番号を指定してください。
- HiRDB/パラレルサーバで，一つのサーバマシン上で複数のユニットを起動する場合，それぞれのユニットの各オペランドには異なるポート番号を指定してください。
- 一つのサーバマシン上で，複数の HiRDB を起動する場合，それぞれの HiRDB の各オペランドには異なるポート番号を指定してください。
- 一つのサーバマシン上に HiRDB クライアントと HiRDB サーバがある場合，各オペランドには，クライアント環境定義の PDCLTRCVPORT と異なるポート番号を指定してください。
- 次の条件を満たすようにポート番号を指定してください。
  - ほかのプログラムプロダクトが使用しているポート番号と異なる
  - /etc/services ファイル（NIS 又は DNS 環境の場合はそれぞれに定義した場所）に登録されたポート番号と異なる
  - OS が自動的に割り当てるポート番号※の範囲に含まれない

    注※　OS が自動的に割り当てるポート番号の範囲は OS によって異なります。
- HiRDB のポート番号の指定を省略した場合，HiRDB のポート番号に 20000 が仮定されます。そのため，各オペランドには 20000 を指定しないでください。

21.6　HiRDB 予約ポート機能

HiRDB 予約ポート機能とは，pd_registered_port オペランドでポート番号の範囲を指定して，特定の通信ポート番号の範囲で通信できるようにする機能です。この機能によって，次のようなことが防げます。

- HiRDB 以外のプログラムが OS の自動割り当てポート番号を多数使用した通信処理をしている場合，通信ポート番号不足が発生して処理が中断される
- HiRDB が大量の通信ポート番号を使用し，他プログラムの通信処理に影響を与える
  確保したポートは HiRDB が解放しても，ある一定時間は再利用されません。そのため，短時間にポートを使用する処理が大量に発生した場合，ポート番号不足が発生する可能性があります。

HiRDB が大量のポート番号を使用する運用をする場合は，「ポート数不足を回避する方法」を参照してください。

なお，HiRDB 予約ポート機能を使用する場合の定義例，及び注意事項については，マニュアル「HiRDB システム定義」の pd_registered_port オペランドの説明を参照してください。

21.6.1　HiRDB 予約ポート数の見積もり

(1)　統計情報からの見積もり

統計情報から HiRDB の通信ポート番号の使用数を計れます。統計解析ユティリティの「システムの稼働に関する統計情報」で，次の情報から HiRDB の通信ポート数を算出します。なお，これらの詳細は，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。

- HiRDB 予約ポート使用数
- HiRDB 予約ポートオーバー時の OS 自動割り当てポート使用数

計算式

```
HiRDB予約ポート使用数＋HiRDB予約ポートオーバー時のOS自動割り当てポート使用数＋100※
```

注※　予備として加算しています。

(2)　推奨値の見積もり

推奨値は，「ユニットが使用する通信ポート数の見積もり」の計算式で求められます。なお，この値は目安です。運用上は「統計情報からの見積もり」の値を使用してください。

付録

付録 A　HiRDB の最大値・最小値

付録 A.1　システム構成に関する最大値と最小値

HiRDB のシステム構成に関する最大値と最小値を次の表に示します。

表 A-1　HiRDB のシステム構成に関する最大値と最小値

項　目	最小値	最大値	単位
1 ユニットに作成できるサーバ数	1	34	個
シングルサーバ数	1	1	個
システムマネジャ数	1	1	個
フロントエンドサーバ数	1	1,024	個
ディクショナリサーバ数	1	1	個
バックエンドサーバ数	1	16,382	個
総 RD エリア数	3	8,388,592	個
マスタディレクトリ用 RD エリア数	1	1	個
データディレクトリ用 RD エリア数	1	1	個
データディクショナリ用 RD エリア数	1	59	個
解析情報表及び運用履歴表を格納するデータディクショナリ用 RD エリア	1	1	個
ユーザ用 RD エリア数	0	8,388,589	個
データディクショナリ LOB 用 RD エリア数	0	2	個
ユーザ LOB 用 RD エリア数	0	8,388,325	個
レジストリ用 RD エリア数	0	1	個
レジストリ LOB 用 RD エリア数	0	1	個
リスト用 RD エリア	0	8,388,588	個
1RD エリア中の HiRDB ファイル数	1	16	個
1RD エリア中の実表数と順序数生成子数の合計	0	500	個
1RD エリア中のインデクス数	0	500	個
1RD エリア中のリスト数	0	50,000	個
総 HiRDB ファイル数	1	134,217,728	個
同時アクセス可能実表数	4	32,000	個

項　目				最小値	最大値	単位
最大同時接続数				1	HiRDB/シングルサーバの場合：3,000 HiRDB/パラレルサーバの場合：2,000※1	個
HiRDB で作成可能なユーザ数※2				1	無制限	個
同時リスト所有可能ユーザ数				0	32,767	個
1 ユーザ当たりのリスト作成数				0	32,767	個
1 トランザクション当たりの作業表作成数				0	512	個
1 サーバ当たりのグローバルバッファ数※3				1	2,000,000	個
HiRDB ファイルシステム領域長	AIX 版	ラージファイル未使用	通常ファイル	1	2,047	メガバイト
			キャラクタ型スペシャルファイル	1	2,047	メガバイト
		ラージファイル使用	通常ファイル（JFS）	1	65,411	メガバイト
			通常ファイル（JFS2）	1	1,048,575	メガバイト
			キャラクタ型スペシャルファイル	1	1,048,575	メガバイト
	Linux 版	ラージファイル未使用	通常ファイル	1	2,047	メガバイト
			キャラクタ型スペシャルファイル	1	2,047	メガバイト
		ラージファイル使用	通常ファイル	1	1,048,575	メガバイト
			キャラクタ型スペシャルファイル	1	1,048,575	メガバイト

注※1

マルチフロントエンドサーバ構成の場合，フロントエンドサーバ数×pd_max_users 数の上限が 2,000 になります。

注※2

1 ユーザ当たりディクショナリ表（SQL_USERS）を 1 行消費するため，データディクショナリ用 RD エリアの容量に依存します。

注※3

ただし，システム全体では 2,147,483,647 が上限となります。

付録 A.2　データベースに関する最大値と最小値

データベースに関する最大値と最小値を次の表に示します。

表 A-2　データベースに関する最大値と最小値

項　目		最小値	最大値	単位
文字データの長さ（定義長）	CHAR	1	30,000	字（バイト）
	VARCHAR	1	32,000	字（バイト）
各国文字データの長さ（定義長）	NCHAR	1	15,000	字
	NVARCHAR	1	16,000	字
混在文字データの長さ（定義長）	MCHAR	1	30,000	バイト
	MVARCHAR	1	32,000	バイト
パック形式 10 進数の精度	DECIMAL 又は NUMERIC	1	38	けた
パック形式 10 進数の位取り	DECIMAL 又は NUMERIC	0	38	けた
時刻印データの小数秒精度	TIMESTAMP	0	6	けた
BLOB データの長さ		1	2,147,483,647	バイト
BINARY データの長さ（定義長）		1	2,147,483,647	バイト
表中の列数		1	30,000	列
表中のインデクス数		0	255	個
インデクスの構成列数		1	64	列
クラスタキーの構成列数		1	64	列
表を分割格納する RD エリア数		1	4,096	個
表を分割配置する BES 数		1	4,096	個
横分割表を定義時の格納条件に指定する定数の総数（格納条件を省略した場合も 1 と数える）		1	15,000	個
ビュー表の基になる表数		1	128	個
ビュー表の列数		1	30,000	列
主キーの構成列数		1	64	列
外部キーの構成列数		1	64	列
1 表中の外部キー数		0	255	個
一つの主キーを参照する外部キー数		0	255	個
1 表中に指定できる検査制約定義数		0	254	個

項　目		最小値	最大値	単位
1 表中に定義できる検査制約中で指定した論理演算子 AND，OR，及び検査制約定義の数の合計（CASE 式の WHEN 探索条件中の論理演算子の AND 及び OR は除きます）		0	254	個
識別子の長さ （表識別子，列名，データ型識別子，インデクス型識別子，属性名，ルーチン識別子，相関名，インデクス識別子，カーソル名，SQL 文識別子，RD エリア名，埋込み変数名，標識変数名，パスワード，制約名，条件名，SQL 変数名，問合せ名，トリガ識別子，文ラベル，ループ変数名，ホスト識別子，リスト名，SQL パラメタ名，接続制約名）		1	30	バイト
FIX 表の行長		1	30,000	バイト
手続き又は関数の SQL パラメタ数		0	30,000	個
繰返し列数		2	30,000	個
IP アドレスによる接続制限の登録数		0	10,000	個
1 サーバで生成するインデクス情報ファイル数	AIX 版	1	8,092※1	個
	Linux 版	1	8,092※1	個
次に示すユティリティがメッセージに表示できる処理行数 • pdload • pdrorg • pdrbal		0	4,294,967,295※2	件

注※1

pd_max_open_fds オペランドの指定値，プラグイン使用の有無，サーバ内の RD エリア数などによって最大値は変動します。表の値は pd_max_open_fds オペランドに最大値を指定した場合です。なお，プラグインインデクスの遅延一括作成に HiRDB ファイルシステム領域を使用する場合，最大値は 4,096 になります。

注※2

4,294,967,295 件以上のデータを処理した場合，表示行数が一度 0 にリセットされ，再度 1 からカウントアップします。

付録 A.3　HiRDB ファイル名に関する最大値と最小値

HiRDB ファイル名に関する最大値と最小値を次の表に示します。

表 A-3　HiRDB ファイル名に関する最大値と最小値（単位：文字）

HiRDB ファイル種別	HiRDB ファイルに指定できるファイル名の長さ		パス名の長さ※の最大値
	最小値	最大値	
システムステータスファイル	1	30	167
サーバステータスファイル	1	30	167
システムログファイル	1	30	167
シンクポイントダンプファイル	1	30	167
アンロードログファイル	1	30	167
監査証跡ファイル	16	－	167
作業表用ファイル	25	－	167
RD エリア構成ファイル	1	30	167
バックアップファイル	1	30	167
アンロードデータファイル	1	30	167
インデクス情報ファイル	30	－	167
差分バックアップ管理ファイル	1	30	167

（凡例）－：該当しません。最小値の欄に記載されている値が固定値です。

注※

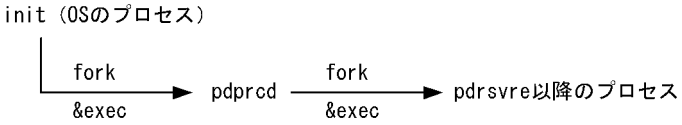
パス名の長さは，「HiRDB ファイルシステム領域/HiRDB ファイル」の長さです。

付録 B　HiRDB のプロセス一覧

ここでは，HiRDB で起動するプロセスについて説明します。

付録 B.1　HiRDB/シングルサーバで起動するプロセス

HiRDB で起動するプロセスの構造を次に示します。



HiRDB/シングルサーバで起動するプロセスを次の表に示します。

表 B-1　HiRDB/シングルサーバで起動するプロセス（システムサーバ）

プロセス名		説明	プロセス数	サーバ名称
マニュアルでの表記	プロセス名称			
プロセスサーバプロセス	pdprcd	HiRDB 関連プロセスの管理	1	_prc
後処理プロセス	pdrsvre	プロセス異常終了時の後始末処理	pd_process_terminator の指定値が fixed の場合： pd_process_terminator_max の値 pd_process_terminator の指定値が resident の場合：3～HiRDB が規定したプロセス数 pd_process_terminator の指定値が nonresident の場合：0～異常終了したプロセス数	_admrsvr
サーバモード系切り替え用 HiRDB 起動プロセス	pdstart2d	クラスタソフトウェアと連動した HiRDB プロセスの起動制御 • ユーザサーバホットスタンバイを行わないスタンバイ型系切り替え • ユーザサーバホットスタンバイ • 高速系切り替え	1（オンライン時はなし）	_pdstrt2
メッセージログサーバプロセス	pdmlgd	メッセージ出力制御	1	_mlg

プロセス名		説明	プロセス数	サーバ名称
マニュアルでの表記	プロセス名称			
システムマネジャプロセス	pdrdmd	ユニット起動・停止制御，接続ユーザ管理 （なお，ネームサービス，又はノードマネジャと表記している場合があります）	1	_rdm
ステータスサーバプロセス	pdstsd	ユニット用ステータスファイルの入出力制御	1	_sts0
スケジューラプロセス	pdscdd	シングルサーバプロセスへのトランザクションの割り当て （なお，ロックサーバと表記している場合があります）	1	_scd
トランザクションサーバプロセス	pdtrnd	トランザクション制御	1	_trn
トランザクション回復プロセス	pdtrnrvd	トランザクションの決着・回復制御	1～ダウンした pdsds 数※1	_trnrcv
監査証跡管理サーバプロセス	pdaudd	監査証跡管理	pd_aud_file_name の指定がある場合は 1，ない場合は 0	_aud
監査証跡自動データロード制御プロセス	pdaudld	自動データロード用 pdload の起動制御	pd_aud_file_name の指定があり，かつ pd_aud_auto_loading に Y を指定している場合は 1，それ以外は 0	_audld
トラブルシュート情報取得プロセス	pdprfd	トラブルシュート機能の制御	1	－
ログサーバプロセス	pdlogd	システムログ取得制御，ログ関連プロセス制御	1	_logN
デファードライトプロセス	pd_buf_dfw	DB 格納ディスクへのバックグラウンドライト	1	1dfwN
非同期 READ プロセス	pd_ios_ard	非同期 READ 機能	pd_max_ard_process の値	1ardN

プロセス名		説明	プロセス数	サーバ名称
マニュアルでの表記	プロセス名称			
デファードライト処理用並列 WRITE プロセス	pd_buf_awt	デファードライト処理の並列 WRITE 機能	pd_dfw_awt_process の値	1awtN
REDO プロセス	pd_rcv_rd	全面リラン時の DB の前進復帰	MIN（接続ディスク数，pd_max_recover_process の値） **接続ディスク数**：RD エリアを定義したキャラクタ型スペシャルファイル数 pd_rdarea_open_attribute_use に Y を指定している場合は pd_max_recover_process の値※2	2rrnM
ログスワッパプロセス	pdlogswd	システムログ関連ファイルの割り当て・解放・入出力管理，シンクポイントダンプ取得	1	_logNs
デッドロック監視プロセス	pdlckmnd	排他制御処理の分散を行っている場合のデッドロック検知	pd_lck_pool_partition に 2 以上を指定し，かつ pd_lck_deadlock_check に Y を指定すると 1，それ以外は 0	_lckmnN

表 B–2　HiRDB/シングルサーバで起動するプロセス（ユーザサーバ）

プロセス名		説明	プロセス数	サーバ名称
マニュアルでの表記	プロセス名称			
シングルサーバプロセス	pdsds	SQL 処理	pd_max_users の値＋ pd_max_reflect_process_count の値※3	サーバ名※4

表 B–3　HiRDB/シングルサーバで起動するプロセス（ユティリティサーバ）

プロセス名※5		説明	プロセス数	サーバ名称
マニュアルでの表記	プロセス名称			
pdinit 制御プロセス	pdinitd	初期設定ユティリティ実行プロセス	1	0minit0
pdcopy バックアップ出力プロセス	pdcopyb	バックアップファイル出力	多重度数×コマンド同時実行数	0bcpy?0※6

プロセス名※5		説明	プロセス数	サーバ名称
マニュアルでの表記	プロセス名称			
pdcopy データベース読み込みプロセス	pdcopyr	データベース読み込み	コマンド同時実行数	0rcopy0
pdrstr バックアップ読み込みプロセス	pdrstrb	バックアップファイル読み込み	コマンド同時実行数	0brstr0
pdrstr マスタディレクトリ用 RD エリア読み込みプロセス	pdrstrm	マスタディレクトリ用 RD エリア読み込み	コマンド同時実行数	0mrstr0
pdrstr アンロードログ読み込みプロセス	pdrstrl	アンロードログファイル読み込み	コマンド同時実行数	0lrstr0
pdrstr データベース書き込みプロセス	pdrstrr	データベース書き込み	コマンド同時実行数	0brstr0
pdrstr マスタディレクトリ用 RD エリア書き込みプロセス	pdrstrw	マスタディレクトリ用 RD エリア書き込み	コマンド同時実行数	0wrstr0
pdload 制御プロセス	pdloadm	データロード制御	pdload コマンド同時実行数	0mload0
pdrorg 制御プロセス	pdrorgm	DB 再編成制御（アンロード，リロード，インデクス再編成・再作成，空きページ解放，及びグローバルバッファ常駐化）	pdrorg，pdreclaim，pdpgbfon コマンド同時実行数	0mrorg0
pdgetcst 制御プロセス	pdgcstm	最適化情報収集	pdgetcst コマンド同時実行数	0mgcst0
pddbst 制御プロセス	pddbst1	DB 状態解析制御	pddbst コマンド同時実行数	0mdbst0
pdexp 制御プロセス	pdexpm	ディクショナリ搬出入制御	1	0mexp0

プロセス名※5		説明	プロセス数	サーバ名称
マニュアルでの表記	プロセス名称			
pdplgexe 制御プロセス	pdplgexm	プラグインユティリティ実行制御	プラグインが提供するコマンド同時実行数	0mplge0
pdorend 制御プロセス	pdorendm	オンライン再編成 追い付き反映制御	1	0more0
pdorend 反映プロセス	pdorendl	オンライン再編成 追い付き反映処理	多重度数（pdorend コマンドの-m オプション指定値数）	0lore?0※7

（凡例）

－：該当しません。

注

- サーバ名称の xxxN はユーザサーバ数によって 1，2，…ユニット最大サーバ数と増加します。
- サーバ名称の xxxM は定義によって 2～11 と増加します。
- サーバ名称はメッセージ出力やコマンド情報出力などで使用されます。

注※1

ダウンした pdsds 数が二つ以上発生した場合に，ダウンした数と同数まで増加します。ダウンした pdsds のトランザクションが決着するに従ってプロセス数は減少し，回復対象となるトランザクションがなくなると 1 に戻ります。上限値を次に示します。

MIN（⌊pd_trn_rcvmsg_store_buflen の値÷72⌋，（pd_max_users の値＋ pd_max_reflect_process_count の値）×2 ＋ 7）

注※2

REDO プロセスは，HiRDB 開始時に起動され，起動が完了すれば停止します。

注※3

ユティリティサーバの場合にはプロセス数は 0 となります。計算後に 2000 を超える場合には 2000 までが起動されるプロセスとなります。また，pd_process_count が指定されている場合には起動プロセス数は指定値分となります。pd_process_count を超えるアクセス要求があった場合は，同時処理するために同時アクセス数のプロセスを pd_max_users の値＋ pd_max_reflect_process_count の値まで起動します。

注※4

システム共通定義の pdstart オペランドの-s オプションで指定するサーバ名です。

注※5

対応するコマンドが実行中のときだけプロセスが起動され，コマンドが終了すればプロセスは停止します。

注※6

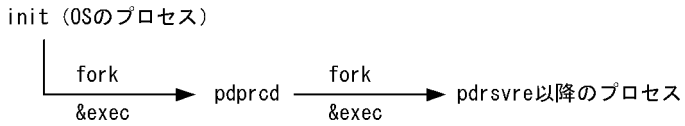
「?」は，バックアップ出力プロセスの多重度数（pdcopy の-f オプションで指定した制御文ファイル中の-b オプション指定値数）によって，0，1，…f と増加します。

注※7

「?」は，オンライン再編成反映プロセスの多重度数（pdorend コマンドの-m オプション指定値数）によって，1，2，…8 と増加します。

付録 B.2　HiRDB/パラレルサーバで起動するプロセス

HiRDB で起動するプロセスの構造を次に示します。



HiRDB/パラレルサーバで起動するプロセスを次の表に示します。

表 B-4　HiRDB/パラレルサーバで起動するプロセス（システムサーバ）（1/2）

プロセス名		説明	プロセス数	サーバ名称	サーバごとのプロセスの起動有無	
マニュアルでの表記	プロセス名称				MGR	非MGR
プロセスサーバプロセス	pdprcd	HiRDB 関連プロセスの管理	1	_prc	○	○
後処理プロセス	pdrsvre	プロセス異常終了時の後始末処理	pd_process_terminator の指定値が fixed の場合：pd_process_terminator_max の値 pd_process_terminator の指定値が resident の場合：3～HiRDB が規定したプロセス数 pd_process_terminator の指定値が nonresident の場合：0～異常終了したプロセス数	_admrsvr	○	○
サーバモード系切り替え用 HiRDB	pdstart2d	クラスタソフトウェアと連動した HiRDB プロセスの起動制御 • ユーザサーバホットスタンバイを行わないスタンバイ型系切り替え	1（オンライン時はなし）	_pdstrt2	○	○

プロセス名		説明	プロセス数	サーバ名称	サーバごとのプロセスの起動有無	
マニュアルでの表記	プロセス名称				MGR	非MGR
起動プロセス 1		• ユーザサーバホットスタンバイ • 高速系切り替え • 1：1 スタンバイレス型系切り替えの実行系				
サーバモード系切り替え用 HiRDB 起動プロセス 2	pdstart2a	クラスタソフトウェアと連動した HiRDB プロセスの起動制御 • 1：1 スタンバイレス型系切り替えの待機系	1（オンライン時はなし）	_pdst2a1	×	○
サーバモード系切り替え用 HiRDB 起動プロセス 3	pdsvstartd	クラスタソフトウェアと連動した HiRDB プロセスの起動制御 • 影響分散スタンバイレス型系切り替えの待機系	pd_ha_agent に activeunits を指定している場合は 1，それ以外は 0	_pdsvstd	×	○
トラブルシュート情報取得プロセス	pdprfd	トラブルシュート機能の制御	1	−	○	○
メッセージログサーバプロセス	pdmlgd	メッセージ出力制御 (pd_mlg_msg_log_unit に local を指定している場合に起動します)	1	_mlg	○	○
システムマネジャ (MGR ユニット) プロセス	pdrdmd	ユニット起動・停止制御，接続ユーザ管理 (なお，ネームサービス，又はノードマネジャと表記している場合があります)	1	_rdm	○	×
ノードマネジャ (非 MGR ユニット) プロセス	pdndmd	ユニット起動・停止制御，接続ユーザ管理 (なお，ネームサービスと表記している場合があります)	1	_ndm	×	○

プロセス名		説明	プロセス数	サーバ名称	サーバごとのプロセスの起動有無	
マニュアルでの表記	プロセス名称				MGR	非MGR
ステータスサーバプロセス	pdstsd	ユニット用ステータスファイルの入出力制御	1	_sts0	○	○
スケジューラプロセス	pdscdd	バックエンドサーバプロセス，ディクショナリサーバプロセス，及びフロントエンドサーバプロセスの割り当て（なお，ロックサーバと表記している場合があります）	1	_scd	○	○
トランザクションサーバプロセス	pdtrnd	トランザクション制御	1	_trn	○	○
トランザクション回復プロセス	pdtrnrvd	トランザクションの決着・回復制御	FES の場合： 1～ダウンした pdfes 数※1 DS の場合： 1～ダウンした pddic 数※1 BES の場合： 1～ダウンした pdbes 数※1	_trnrcv	○	○
監査証跡管理サーバプロセス	pdaudd	監査証跡管理	pd_aud_file_name の指定がある場合は 1，ない場合は 0	_aud _auz※2	○	○
監査証跡自動データロード制御プロセス	pdaudld	自動データロード用 pdload の起動制御	監査証跡管理サーバプロセスの条件に合わせて pd_aud_auto_loading に Y を指定している場合は 1，N 又は指定していない場合は 0	_audld	○	×
ユニット監視プロセス	pdrdma	HiRDB ユニットが稼働しているかどうかの監視をおこなう	1（ユニット数が 1 の場合は 0）	_rdmck	○	×

表 B-5　HiRDB/パラレルサーバで起動するプロセス（システムサーバ）（2/2）

プロセス名		説明	プロセス数	サーバ名称	サーバごとのプロセスの起動有無		
マニュアルでの表記	プロセス名称				BES	DS	FES
ログサーバプロセス	pdlogd	システムログ取得制御，ログ関連プロセス制御	1	_logN _lozN※2	○	○	○
デファードライトプロセス	pd_buf_dfw	DB 格納ディスクへのバックグラウンドライト	1	1dfwN 1dfzN※2	○	○	×
非同期 READ プロセス	pd_ios_ard	非同期 READ 機能	pd_max_ard_process の値	1ardN 1arzN※2	○	○	×
デファードライト処理用並列 WRITE プロセス	pd_buf_awt	デファードライト処理の並列 WRITE 機能	pd_dfw_awt_process の値	1awtN 1awzN※2	○	○	×
REDO プロセス	pd_rcv_rd	全面リラン時の DB のロールフォワード	MIN（接続ディスク数，pd_max_recover_process の値） **接続ディスク数：**RD エリアを定義したキャラクタ型スペシャルファイル数 pd_rdarea_open_attribute_use の値が Y の場合は pd_max_recover_process の値※3	2rrnM 2rrzM※2	○	○	×
ログスワッパプロセス	pdlogswd	システムログ関連ファイルの割り当て・解放・入出力管理，シンクポイントダンプ取得	1	_logsN _lozsN※2	○	○	○
デッドロック監視プロセス	pdlckmnd	排他制御処理の分散を行っている場合のデッドロック検知	**FES の場合：** pd_fes_lck_pool_partition に 2 以上を指定し，かつ pd_lck_deadlock_check に Y を指定した場合は 1，それ以外は 0 **DS，BES の場合：** pd_lck_pool_partition に 2 以上を指定し，かつ pd_lck_deadlock_check に Y を指定した場合は 1，それ以外は 0	_lckmnN _lckmzN※2	○	○	○

表 B-6　HiRDB/パラレルサーバで起動するプロセス（ユーザサーバ）

プロセス名		説明	プロセス数	サーバ名称	サーバごとのプロセスの起動有無		
マニュアルでの表記	プロセス名称				BES	DS	FES
バックエンドサーバプロセス	pdbes	データベースへのアクセス	MAX（pd_max_users の値, pd_max_bes_process の値）	サーバ名※4	○	×	×
ディクショナリサーバプロセス	pddic	ディクショナリ表の一括管理	MAX（pd_max_users の値, pd_max_dic_process の値）	サーバ名※4	×	○	×
フロントエンドサーバプロセス	pdfes	SQL 処理, バックエンドサーバへの指示	pd_max_users の値	サーバ名※4	×	×	○

表 B-7　HiRDB/パラレルサーバで起動するプロセス（ユティリティサーバ）

プロセス名		説明	プロセス数	サーバ名称	サーバごとのプロセスの起動有無		
マニュアルでの表記	プロセス名称				BES	DS	FES
pdinit 制御プロセス	pdinitd	初期設定ユティリティ実行プロセス	1	0minit0	×	○	×
pdinit 実行プロセス	pdinitb	初期設定ユティリティ BES 側実行プロセス	1～2	0sinit0	○	×	×
pdcopy バックアップ出力プロセス	pdcopyb	バックアップファイル出力（pdcopy に指定したバックアップ出力先（-b オプション）に起動します）	多重度数×pdcopy コマンド同時実行数	0bcpy?0※5	×	×	×
pdcopy データベース読み込みプロセス	pdcopyr	データベース読み込み	複写対象サーバ数※6	0rcopyN	○	○	×
pdrstr バックアップ読み込みプロセス	pdrstrb	バックアップファイル読み込み（pdrstr に指定したバック	pdrstr コマンド同時実行数	0brstr0	×	×	×

プロセス名		説明	プロセス数	サーバ名称	サーバごとのプロセスの起動有無		
マニュアルでの表記	プロセス名称				BES	DS	FES
		アップ読み込み先（-b オプション）に起動します）					
pdrstr マスタディレクトリ用 RD エリア読み込みプロセス	pdrstrm	マスタディレクトリ用 RD エリア読み込み※7	pdrstr コマンド同時実行数	0mrstr0	×	○	×
pdrstr アンロードログ読み込みプロセス	pdrstrl	アンロードログファイル読み込み※8	pdrstr コマンド同時実行数	0lrstr0	×	×	×
pdrstr データベース書き込みプロセス	pdrstrr	データベース書き込み	回復対象サーバ数※9	0brstrN	○	○	×
pdrstr マスタディレクトリ用 RD エリア書き込みプロセス	pdrstrw	マスタディレクトリ用 RD エリア書き込み※10	1	0wrstr0	×	○	×
pdload 制御プロセス	pdloadm	データロード制御	pdload コマンド同時実行数	0mload0	○	○	○
pdrorg 制御プロセス	pdrorgm	DB 再編成制御（アンロード，リロード，インデクス再編成・再作成，空きページ解放，及びグローバルバッファ常駐化）	pdrorg,pdreclaim,pdpgbfon コマンド同時実行数	0mrorg0	○	○	○
pdrbal 制御プロセス	pdrbalm	リバランス制御	pdrbal コマンド同時実行数	0mrbal0	×	○	×
pdgetcst 制御プロセス	pdgcstm	最適化情報収集	pdgetcst コマンド同時実行数	0mgcst0	×	○	×

プロセス名		説明	プロセス数	サーバ名称	サーバごとのプロセスの起動有無		
マニュアルでの表記	プロセス名称				BES	DS	FES
pddbst 制御プロセス	pddbst1	DB 状態解析制御 （MGR ユニットで起動します）	pddbst コマンド同時実行数	0mdbst0	×	×	×
pdexp 制御プロセス	pdexpm	ディクショナリ搬出入制御 （搬出ファイルがあるユニットに起動します）	1	0mexp0	×	×	×
pdplgexe 制御プロセス	pdplgexm	プラグインユティリティ実行制御	プラグインが提供するコマンド同時実行数	0mplge0	×	○	×
pdorend 制御プロセス	pdorendm	オンライン再編成 追い付き反映制御	1	0more0	○	×	×
pdorend 反映プロセス	pdorendl	オンライン再編成 追い付き反映処理	多重度数（pdorend コマンドの-m オプション指定値数）	0lore?0※11	○	×	×

（凡例）

○：プロセスが起動します。

×：プロセスが起動しません。

－：該当しません。

注

- サーバ名称の xxxN はユーザサーバ数によって 1，2，…ユニット最大サーバ数と増加します。
- サーバ名称の xxxM は定義によって 2～（ユニット最大サーバ数×11）と増加します。

注※1

ダウンした pdbes，pdfes，pddic 数が二つ以上発生した場合に，ダウンした数と同数まで増加します。ダウンした pdbes，pdfes，pddic のトランザクションが決着するに従ってプロセス数は減少し，回復対象となるトランザクションがなくなると 1 に戻ります。上限値を次に示します。

FES の場合：（pd_max_users の値＋ pd_max_reflect_process_count の値）×2 ＋ 7

DS の場合：（pd_max_dic_process の値＋ pd_max_reflect_process_count の値）×2 ＋ 7

BES の場合：（pd_max_bes_process の値＋ pd_max_reflect_process_count の値）×2 ＋ 7

なお，ユニット当たりの上限値は↓pd_trn_rcvmsg_store_buflen の値÷72↓です。ユニット内の FES，DS，BES ごとに求めた値の総和がこれより大きい場合でも，ユニット当たりの最大プロセス数はこの値になります。

注※2

1：1 スタンバイレス型系切り替え構成の場合で，代替 BES ユニット用に起動するときの名称です。

注※3

REDO プロセスは，HiRDB 開始時に起動され，起動が完了すれば停止します。

注※4

システム共通定義の pdstart オペランドの-s オプションで指定するサーバ名です。

注※5

「?」は，バックアップ出力プロセスの多重度数（pdcopy の-f オプションで指定した制御文ファイル中の- b オプション指定値数）によって，0，1，…f と増加します。

注※6

pdcopy に指定した複写対象（-r，-s，-u，-a オプション）の RD エリアが属するサーバ数分起動します。

注※7

pdrstr に指定したバックアップファイル（-b オプション）があるホストと，ディクショナリサーバがあるホストが異なる場合に起動します。また，pdrstr に指定したバックアップファイル（-b オプション）の出力先ホストと，ディクショナリサーバがあるホストが異なる場合に起動します。

注※8

pdrstr でアンロードログファイル（-l オプション）又はディレクトリ（-d オプション）を指定した場合に，回復対象が 2 サーバ以上あるとき，若しくはアンロードログファイルが格納されているホストと回復対象の RD エリアが属するサーバがあるホストとが異なるときに起動します。

注※9

pdrstr に指定した回復対象（-r，-s，-u，-c，-a オプション）の RD エリアが属するサーバ数分起動します。

注※10

回復対象にマスタディレクトリ用 RD エリアを指定し，バックアップファイルがあるホストとディクショナリサーバがあるホストが異なる場合に起動します。

注※11

「?」は，オンライン再編成反映プロセスの多重度数（pdorend コマンドの-m オプション指定値数）によって，1，2，…8 と増加します。

付録 C　Q&A

HiRDB システムの構築に関する事例を Q&A 形式でまとめています。

付録 C.1　HiRDB/Developer's Kit に関する質問

質問

HiRDB/Developer's Kit はどのようなときに必要ですか？

お答えします

HiRDB サーバがあるマシンで UAP を作成する場合は，HiRDB サーバに HiRDB/Developer's Kit の機能が含まれているので必要ありません。HiRDB サーバがあるマシンとは別のマシンで UAP を開発する場合に必要です。

また，HiRDB サーバと異なるプラットフォームの UAP を作成する場合にも必要です。

付録 C.2　データベース定義ユティリティ（pddef）の実行に関する質問

質問

データベース定義ユティリティ（pddef）で CREATE TABLE を実行したのですが，何も実行されませんでした。なぜですか？

お答えします

pddef の制御文でセミコロン（;）の後ろに空白が入っていませんか？空白が入っていると，その SQL 文は実行されません（この例では，CREATE TABLE 以降の指定は省略しています）。

（誤）CREATE TABLE ;△ △：空白

（正）CREATE TABLE ;

付録 C.3　表の最大容量に関する質問

質問

HiRDB の表の最大容量はどのくらいですか？

お答えします

一つの表は最大 4,096 個の RD エリアに分割格納でき，一つの RD エリアは最大 16 個の HiRDB ファイルで構成できます。

また，一つの HiRDB ファイルは最大約 2 ギガバイトであるため，1 表の最大容量は次のようになります。

- 1 表の最大容量

4096×16×2 ギガバイト＝約 128 テラバイト

なお，HiRDB ではラージファイルをサポートしています。ラージファイルを使用すると，一つの HiRDB ファイルの最大容量が 64 ギガバイトになるため，1 表の最大容量は次のようになります。

- 1 表の最大容量
  4096×16×64 ギガバイト＝約 4 ペタバイト

付録 C.4　OpenTP1 との XA インタフェースに関する質問

質問

OpenTP1 と HiRDB の連携時，参照するだけの SQL であれば，そのトランザクションのコミットは XA インタフェースを通らないように思えますが，どうでしょうか？

お答えします

参照するだけの SQL であっても，コミット時には必ず XA インタフェースを経由して HiRDB に処理が渡されます。ただし，更新 SQL があるときのコミットに比べると，走行するステップ数が少なくなります。

付録 C.5　FIX 表の性能に関する質問

質問

FIX 表とそうでない表（非 FIX 表）の性能差はどれくらいありますか？

お答えします

操作対象列数や操作対象行数によって性能差は変わるため一概には言えませんが，1 列の大量更新で FIX 指定をした場合の実行時間が，FIX 指定をしなかった場合に比べ，約 2/3 になった事例があります。

FIX 指定の方が性能劣化しないため，次に示す条件を満たす場合は FIX 指定にしてください。

- 可変長の列がない
- NULL 値を持つ列がない

付録 C.6　重複キーインデクスに関する質問

質問

重複したキーに対してもインデクスを定義できますか？

定義できるなら，それによって何か困ることが生じるのでしょうか？

お答えします

インデクスは定義できます（非 UNIQUE 属性）。ただし，大量の重複キー（201 件以上）があるインデクスは特殊な格納構造になってアクセスするインデクスページが増えるため，性能上好ましくありません。

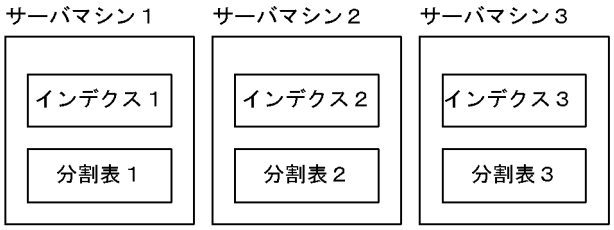
付録 C.7　横分割表のインデクス定義に関する質問

質問

サーバマシン間に分割された表に対してインデクスを定義する場合，どのようにインデクスを配置すればよいですか？

お答えします

次のように分割表単位にインデクスも分割してください。



付録 C.8　pdsetup -d コマンドに対する応答に関する質問

質問

pdsetup -d コマンドを入力すると KFPS00036-Q メッセージが出力され，応答を要求してきましたが y と n のどちらを応答したらよいですか？

お答えします

y を応答した場合，HiRDB の実行に必要なファイル及びディレクトリを削除します。その場合，次回 pdsetup コマンドを実行したときに，実行に必要なファイルをインストールディレクトリからコピーします。n を応答した場合は，ファイル及びディレクトリは削除しません。

次のような場合に y を応答してください。

- 現在運用している HiRDB をインストールした HiRDB に入れ替える場合（HiRDB をバージョンアップする場合は pdstop コマンドで正常終了する必要があります）
- HiRDB 管理者用の認可識別子を変更する場合
- HiRDB 運用ディレクトリ下のファイル，ディレクトリの所有者，又はファイルモードを誤って変更したり，削除したりした場合

備考

y を応答すると，HiRDB 運用ディレクトリとインストールディレクトリが異なる場合，HiRDB 運用ディレクトリ下のロード一式を削除します。そして次回の pdsetup コマンド実行でロード一式をイン

ストールディレクトリからコピーすることになります。このためコマンド実行に多少の時間が掛かります。

付録 C.9　シンクポイントダンプの運用に関する質問

質問

シンクポイントダンプファイルの有効保証世代数とはどのようなことですか？

お答えします

シンクポイントダンプファイルには，全面回復処理に備えてシステムログファイルをどの位置から読み始めればよいかなどの情報をシンクポイントごとに取得します。シンクポイントダンプファイルに格納されている位置情報が示すシステムログファイル以降のシステムログファイルは，全面回復処理で使用する可能性があるため，上書き禁止状態にして保護しています。

有効保証世代数とは，「シンクポイントダンプファイルの何世代分の位置からシステムログファイルを上書き禁止にして保護するか」ということです。つまり，有効保証世代数が 1 の場合は，最新のシンクポイントダンプファイルが示すシステムログファイル以降が上書き禁止状態になります。有効保証世代数が 2 の場合は，最新のシンクポイントダンプファイルの 1 世代前のシンクポイントダンプファイルが示すシステムログファイル以降が上書き禁止状態になります。

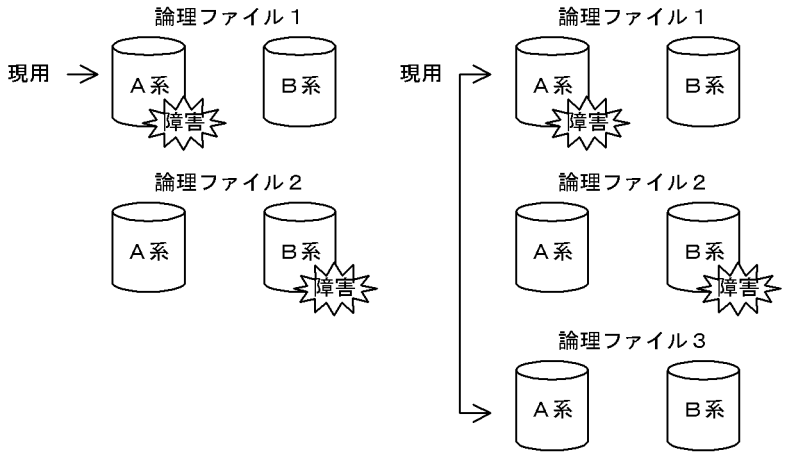
付録 C.10　ステータスファイルに関する質問

(1)　ステータスファイルの運用に関する質問（ステータスファイルの二重化）

質問

ステータスファイルの二重化のペアはどのように構成するのですか？

次のように論理ファイル 1 の A 系と論理ファイル 2 の B 系が障害の場合，論理ファイル 2 の A 系と論理ファイル 1 の B 系でペアを組むことはありますか？



お答えします

異なる論理ファイル間でペアを組むことはありません。A 系，B 系が両方とも正常な論理ファイルにスワップします。A 系，B 系の両方とも正常な論理ファイルが一つもない場合は，pd_sts_singleoperation オペランドの指定に従ってユニットを異常終了するか，又は片系運転で処理を続行します。

(2) ステータスファイルの運用に関する質問（障害発生時）

質問

A 系，B 系の両方とも正常な論理ファイルが一つもない（A 系，B 系のどちらかが障害）場合の処理方式を決める次に示すオペランドがあります。

- pd_syssts_singleoperation = stop | continue（ユニット用ステータスファイルの場合）
- pd_sts_singleoperation = stop | continue（サーバ用ステータスファイルの場合）

stop と continue，どちらを指定すればよいですか？

お答えします

stop を指定すると，HiRDB（HiRDB/パラレルサーバの場合はユニット）が異常終了します。continue を指定すると，ステータスファイルを片系運転します。

ステータスファイルは，全面回復処理のための情報を記録している重要なファイルです。continue を指定して片系運転中にステータスファイルが障害になると，両系障害でユニットが異常終了します。そして，現用ファイルが両系ともアクセスできないため，全面回復処理ができなくなります。したがって，次のような考え方で判断してください。

- HiRDB の異常終了よりも，全面回復処理の保証を重視する場合は stop を指定します。
- とにかく HiRDB を途中で停止したくない（最悪の場合，全面回復処理はあきらめ，データベースをバックアップ時点まで戻したりデータロードし直す）場合は continue を指定します。

(3) ステータスファイルの運用に関する質問（ステータスファイルの定義）

質問

論理ステータスファイルは 1～7 個定義できますが，ディスクに余裕がありません。どの程度用意するのが現実的ですか？

お答えします

ディスクを障害回復して再使用可能になるまでの安全性を考えると，論理ステータスファイルは三つ（二重化×3=6 ファイル）以上用意することをお勧めします。

ディスクに余裕がなければ，論理ステータスファイルは二つ（二重化×2=4 物理ファイル）が妥当です。この場合，障害発生時には速やかにステータスファイルを回復してください。

論理ステータスファイルを一つしか用意しないと，ステータスファイルの障害発生時にデータベースをバックアップから回復する必要があります。

(4) ステータスファイルの運用に関する質問（ステータスファイルの配置）

質問

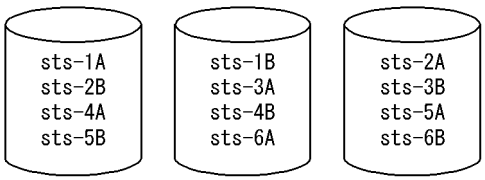
ステータスファイルはどのようにディスクに配置すればよいですか？

お答えします

基本は，複数の物理ステータスファイルを同じディスクに配置しないことです。同じディスクに配置すると，二重化や論理ファイルの複数化の意味がなくなります。つまり，論理ファイルを二つ定義している場合は四つのディスクに分散配置し，論理ファイルを三つ定義している場合は六つのディスクに分散配置します。

なお，次のように，物理ステータスファイルをリング状に配置すると，少ないディスク数で信頼性を確保できます。

論理ファイル数 6 の場合の配置例



このように配置すると，1 ボリュームに障害が発生しても，2 世代の両系ファイルが無事のままです。

付録 C.11　作業表用 HiRDB ファイルシステム領域の最大使用量に関する質問

質問

作業表用ファイルを作成する HiRDB ファイルシステム領域（バックエンドサーバ定義の pdwork オペランド）の最大使用量，最大使用ファイル数，最大使用増分回数を知りたいのですが，手段がありますか？

お答えします

pdfstatfs コマンドで取得できます。

pdfstatfs -d 作業表用 HiRDB ファイルシステム領域名

「-d」は，HiRDB ファイルシステム領域に割り当てた領域の最大使用量，最大使用ファイル数，最大使用増分回数を表示するオプションです。出力の「peak capacity」が最大使用量，「peak file count」が最大使用ファイル数，「peak expand count」が最大使用増分回数です。

なお，上記の最大使用量は，pdfstatfs コマンドでクリアできます。

pdfstatfs -c 作業表用 HiRDB ファイルシステム領域名

「-c」は，HiRDB ファイルシステム領域に割り当てた領域の最大使用量，最大使用ファイル数，最大使用増分回数を 0 にするオプションです。

注意事項

pdfstatfs コマンドの「-c」オプションは，HiRDB ファイルシステム領域の使用目的が「WORK」又は「UTL」の場合に有効です。

なお，HiRDB ファイルシステム領域の使用目的は，pdfmkfs コマンドの-k オプションで指定します。

付録 C.12　pdstart コマンドに関する質問

(1)　pdstart コマンドで HiRDB が開始しない場合

質問

pdstart コマンドで HiRDB が開始しないで，Psp4017 で-prc がアボートしてしまうのはなぜですか？

お答えします

次に示す原因が考えられます。

- HiRDB が正しくセットアップされていない
- /dev/HiRDB/pth ディレクトリにアクセス権限がないか又は存在しない

HiRDB を正しくセットアップし直してください。セットアップした後，/dev/HiRDB/pth のアクセス権限を再確認してください。

(2)　pdstart コマンドで特定のユニットが開始しない場合

質問

pdstart コマンドを実行しましたが，システムマネジャ以外のユニットが開始しないのはなぜですか？

システムマネジャ以外のユニットで OS の ps コマンドを実行すると，pdprcd しかありません（HiRDB のほかのプロセスが起動されていません）。

お答えします

開始できなかったユニットのシステム共通定義を確認してください。pdunit 又は pdstart オペランドの指定値が，システムマネジャがあるユニットのシステム共通定義と一致していない可能性があります。

開始できなかったユニットのシステム共通定義を修正した後に，pdstart -u コマンドでユニットを再開始してください。

(3)　pdstart コマンドで HiRDB の開始が遅い場合

質問

pdstart コマンドを入力して，コマンドが KFPS05078-I Unable to recognize HiRDB initialization Completion で終了しましたが，すべてのユニットの開始に時間が掛かるのはなぜですか？

1 時間～2 時間ぐらいで開始しました。

お答えします

1. KFPS00608-W（-314）メッセージが複数出力されている場合は，すべてのユニットのシステム共通定義を確認してください。pdunit 及び pdstart オペランドで指定したホスト名がすべて一致しているか，正しいホスト（存在するホスト）名が指定されているかを確認してください。

2. HiRDB で指定したホスト，ネットワークがすべて開始完了（稼働中）になっているかを確認してください。

(4) pdstart コマンドがエラーリターンする場合（reason code=SETUP）

質問

pdstart コマンドが KFPS01801-E メッセージ（reason code=SETUP）を出力してエラーリターンしてしまいます。なぜですか？

お答えします

次に示す原因が考えられます。

1. 環境変数 PDDIR に設定した HiRDB 運用ディレクトリを pdsetup コマンドで OS に登録していない
2. 前提になる製品がインストールされていない
3. カーネルのセマフォ不足によってプロセスサーバプロセスが起動できない

次に示す対策をしてください。

1. HiRDB 運用ディレクトリを pdsetup コマンドで OS に登録してください。
2. 前提製品をインストールしてください。
3. システムで定義するセマフォの使用数を大きくしてください。この場合，リブートしないとその値は有効にならないので注意してください。

(5) pdstart コマンドがエラーリターンする場合（reason code=TIMEOUT）

質問

pdstart コマンドが KFPS01861-E メッセージ（reason code=TIMEOUT）を出力してエラーリターンしてしまいます。なぜですか？

お答えします

次に示す原因が考えられます。

1. ユニット開始に予想以上に時間が掛かってしまった
2. サーバ共通定義又は各サーバ定義の指定に誤りがある

次に示す対策をしてください。

1. pd_start_time_out オペランドの指定値を大きくしてから，pdstart コマンドを再入力してください。
2. syslogfile に出力されている HiRDB のメッセージを参照して，誤っている定義を修正してください。その後，pdsetup -d コマンドで KFPS00036-Q 応答メッセージに y を応答し，HiRDB を OS から削除し，再度 pdsetup コマンドを実行してください。

(6) pdstart コマンドでユニットが開始できない場合

質問

pdstart を実行したところ，セマフォ操作（semop,semctl）に失敗したことを示すメッセージ KFPS01815-E（errno=11，13，22）が出力されてユニットが開始できません。なぜですか？

お答えします

次に示す原因が考えられます。

1. HiRDB をそのマシンにインストールしていない
2. pdsetup コマンドで HiRDB を OS に登録していない
3. インストールディレクトリを共有ディスク上に配置している

次に示す対策をしてください。

1. そのマシンを再起動した後に pdsetup -d コマンドで HIRDB を OS から削除してください。その後，pdsetup コマンドを再実行してください。
2. HiRDB をそのマシンにインストールしている場合は，pdsetup コマンドを実行してください。
3. インストールディレクトリは自ノードのローカルディスクに置いてください。

備考

HiRDB をインストールしないで，ほかの環境のロードをコピーして使用できません。

付録 C.13 データベース定義ユティリティ（pddef）がエラーになる場合

質問

データベース初期設定ユティリティ（pdinit）は実行できましたが，データベース定義ユティリティ（pddef）がエラーになって実行できません。どんな原因が考えられますか？

お答えします

次に示す原因が考えられます。

- **応答なしや接続エラーになる場合は，環境変数の設定漏れが考えられます。**

  PDHOST，PDNAMEPORT の設定値を確認してください。

  PDHOST：

  HiRDB サーバがあるホスト名を指定します。システム共通定義の pdstart オペランドに指定したホスト名を指定します。

  PDNAMEPORT：

  システム共通定義の pd_name_port オペランドに指定したポート番号を指定します。

- **コネクトエラーとなる場合，環境変数 PDUSER の設定値の不正が考えられます。**

  pdinit の実行直後は，DBA 権限を持つ認可識別子が一つだけ存在します。認可識別子及びパスワードを次のように環境変数 PDUSER に指定してください。

Bourne シェルの場合：

PDUSER='"認可識別子"/"パスワード"'

export PDUSER

C シェルの場合：

setenv PDUSER '"認可識別子"/"パスワード"'

注意事項

1. pdinit 実行直後の認可識別子とパスワードについては，マニュアル「HiRDB コマンドリファレンス」の「データベース初期設定ユティリティ（pdinit)」の「オプション」を参照してください。
2. 環境変数 PDUSER を設定するときは，認可識別子とパスワードを引用符で囲み，その外側をアポストロフィで囲んでください。これは，ほかの HiRDB のユティリティを実行する場合や，UAP を実行する場合も同じです。

付録 C.14　CREATE TABLE 文の LOB 列定義に関する質問

質問

CREATE TABLE 文の列定義で，LOB 列の最大長を指定した場合（例：300 メガバイト）と指定しなかった場合（省略値 2 ギガバイト）では，HiRDB サーバと HiRDB クライアントのメモリ所要量やデータ転送量にどのような違いがありますか？

お答えします

LOB 列の最大長を指定しても指定しなくても，メモリ所要量やデータ転送量は同じです。LOB 列の検索又は更新時に使用するメモリ所要量やデータ転送量は，列定義の最大長ではなく，検索又は更新時の実長と埋込み変数の定義長に依存します。なお，格納するバイナリデータサイズを制限したい場合は，LOB 列の最大長で制限できます。

付録 C.15　ウィルス対策ソフトに関する質問

質問

ウィルス対策ソフトをインストールしたところ，UAP が HiRDB に接続できなくなりました。ウィルス対策ソフトのファイアウォールが原因のようです。ファイアウォールの除外リストに，どのポート番号を除外指定すればよいでしょうか？

お答えします

「HiRDB で指定するポート番号の一覧」に HiRDB で使用するポート番号の一覧があります。この一覧の中で，次に示すポート番号を除外指定してください。

なお，ファイアウォールを設置した場所によって，除外指定するポート番号が異なります。

●HiRDB サーバ側にファイアウォールを設置した場合

システム共通定義及びユニット制御情報定義で指定する次に示すポート番号を除外指定してください。

- HiRDB のポート番号
- スケジューラプロセスのポート番号

これらのポート番号を指定するオペランドを指定していない場合は，オペランドを指定して，除外指定してください。

●クライアント側にファイアウォールを設置した場合

クライアント環境定義で指定する次に示すポート番号を除外指定してください。

- クライアントの受信ポート番号

このポート番号を指定するクライアント環境定義を指定していない場合は，クライアント環境定義を指定して，除外指定してください。

付録 C.16　NFS の使用に関する質問

質問

NFS（ネットワークファイルシステム）は使用できますか？

お答えします

HiRDB システム定義ファイルの共用化機能で NFS を使用できます。

なお，HiRDB 運用ディレクトリ，HiRDB ファイルシステム領域としては使用できません。

また，コマンド・ユティリティの入出力ファイルについても，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。

索引

I

J

K

L

N

O

P

Q

R

S

T

U

W

X

あ

い

う

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

と

に

ね

の

は

ひ

ふ

へ

ほ

ま

み

め

も

ゆ

よ

ら

り

る

れ

ろ

わ