HITACHI
Inspire the Next

ノンストップデータベース

HiRDB Version 10 バッチ高速化機能

解説・手引・操作書

3020-6-567-10

前書き

■ 対象製品

●適用 OS：AIX V7.1，AIX V7.2

P-1M62-35A1 HiRDB Server Version 10 10-06

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)

P-8462-35A1 HiRDB Server Version 10 10-06

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

●適用 OS：Windows Server 2016，Windows Server 2019，Windows Server 2022，Windows 10 Pro (x64)，Windows 10 Enterprise (x64)，Windows 11

P-2962-91A4 HiRDB Server Version 10 10-06

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

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

■ 発行

2022 年 10 月　3020-6-567-10

■ 著作権

変更内容

変更内容(3020-6-567-10)　HiRDB Version 10 10-06

追加・変更内容	変更個所
HP-UX に関する説明を削除しました。	－

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(3020-6-567)　HiRDB Version 10 10-00

追加・変更内容
HiRDB を 10-00 にバージョンアップしました。

変更内容(3020-6-368-50)　HiRDB Version 9 09-66，HiRDB Accelerator Version 8 08-03

追加・変更内容
マニュアルの体裁を変更しました。

変更内容(3020-6-368-40)　HiRDB Version 9 09-65，HiRDB Accelerator Version 8 08-03

追加・変更内容
HiRDB のサポートプラットフォームに次の OS を追加しました。 • AIX V7.2 • Windows Server 2016

はじめに

このマニュアルは，HiRDB Accelerator Version 10 を使用したバッチ高速化機能の運用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

バッチ高速化機能を使って HiRDB Version 10（以降，HiRDB と表記します）を運用する方々を対象にしています。

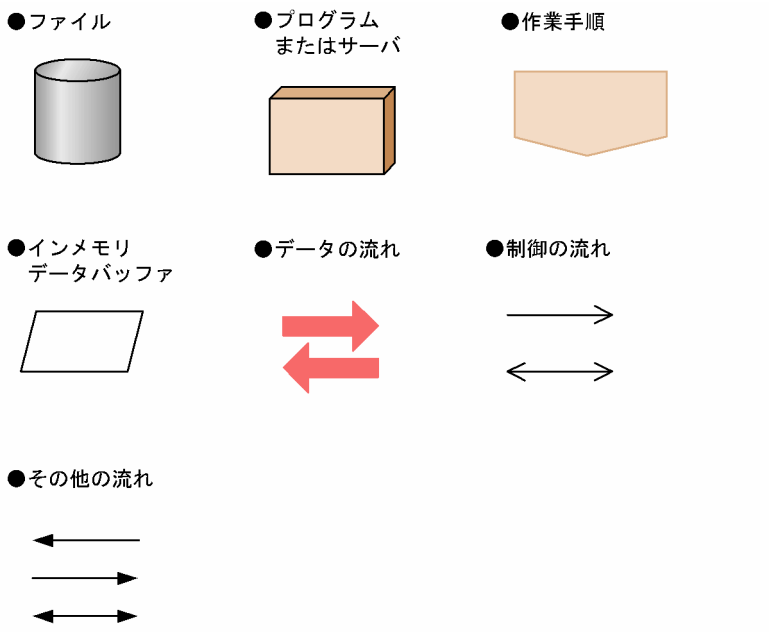
このマニュアルの記述は，次に示す知識があることを前提にしています。

- UNIX，または Windows のシステム管理の基礎的な知識
- HiRDB のシステム管理の基礎的な知識

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。

■ 図中で使用する記号

このマニュアルの図中で使用する記号を次のように定義します。



目次

1 バッチ処理の高速化

この章では，大量のデータを扱うバッチ処理の高速化を実現するインメモリデータ処理について説明します。

1.1　バッチ処理の高速化が必要とされる理由

企業や組織が保有するデータは増加の一途をたどっています。データ量の増加に伴い，バッチ処理に掛かる時間は長くなる一方です。さらに，業務のサービス時間は延長傾向にあるため，バッチ処理を実行できる時間が短くなってきています。また，これまでメインフレームでやっていたような大量のバッチ処理を，オープンシステムでも実施するという需要が高まっています。このような理由から，オープンシステムでのバッチ処理の高速化が企業や組織の課題となっています。

HiRDB では，大量のデータを扱うバッチ処理を高速化するために，RD エリア内の全データをメモリ上に一括して読み込み，バッチ処理の実行中はメモリ上のデータだけを更新し，ディスク上のデータは更新しません。バッチ処理が完了したあとにメモリ上の更新データを一括してディスクに書き込みます。バッチ処理中はディスク入出力が発生しないため，その分バッチ処理に掛かる時間を短縮できます。

このバッチ高速化を実現するための処理方式を**インメモリデータ処理**といいます。

例えば，次のようなケースに当てはまる場合は，インメモリデータ処理を適用してバッチ処理の高速化を検討してください。

ケース 1
: バッチ処理を実行できる時間の範囲が午前 0 時から午前 6 時までの 6 時間と決まっていて，バッチ処理が完了するまで 4 時間掛かります。バッチ処理の途中でエラーが発生した場合，決められた時間内にバッチ処理を完了できないおそれがあります。

ケース 2
: データ量の増加によって，3 か月後のバッチ処理時間が 12 時間以上になると予想され，翌日のオンライン業務に影響が出ると考えられます。

ケース 3
: 将来，メインフレームからデータを移行する予定があります。しかし，データの移行に掛かる時間を試算した結果，移行予定期間内にデータの移行が終わりそうにありません。

1.2　インメモリデータ処理の仕組み

ここでは，インメモリデータ処理の仕組みについて説明します。

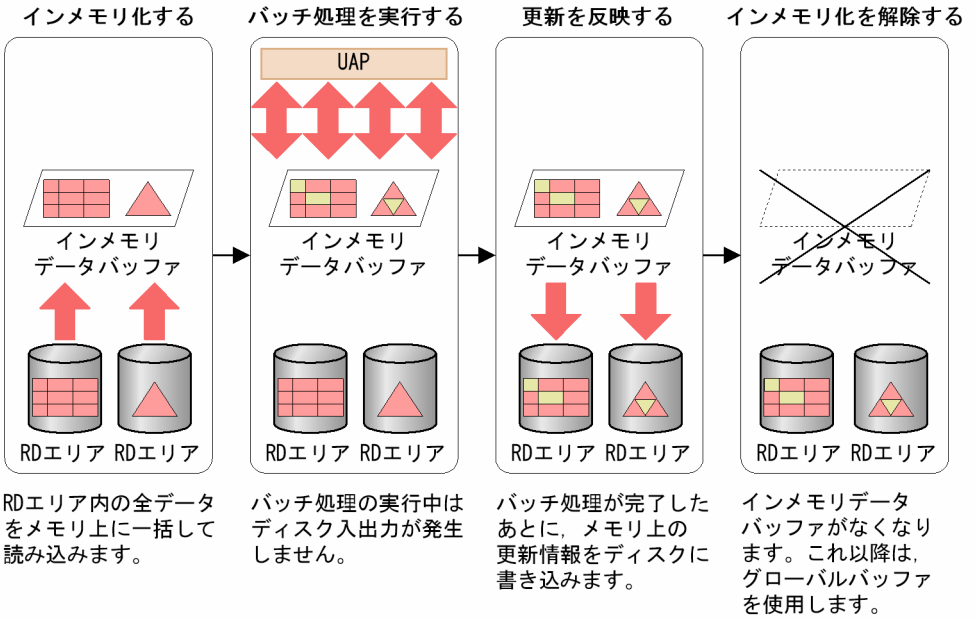
なお，インメモリデータ処理を実行するには HiRDB Accelerator が必要です。

1.2.1　インメモリデータ処理の概要

インメモリデータ処理では，RD エリア内の全データをメモリ上に一括して読み込み，バッチ処理の実行中はメモリ上のデータだけを更新し，ディスク上のデータは更新しません。バッチ処理が完了したあとにメモリ上の更新データを一括してディスクに書き込みます。バッチ処理中はディスク入出力が発生しません。ディスク入出力が発生するのは，RD エリア内のデータをメモリ上に読み込むときと，メモリ上の更新データをディスクに書き込むときの 2 回だけです。

インメモリデータ処理の概要を次に示します。

図 1-1　インメモリデータ処理の概要



[説明]

RD エリア内の全データをメモリ上に一括して読み込むことを**インメモリ化**といいます。インメモリ化は RD エリアごとに行います。インメモリ化している RD エリアを**インメモリ RD エリア**といいます。また，インメモリデータ処理で使用するデータバッファを**インメモリデータバッファ**といいます。

ポイント

インメモリデータ処理を実行する場合は，グローバルバッファを使用しないでインメモリデータバッファを使用します。インメモリデータバッファは，HiRDB が動的に共用メモリ上に確保するため，HiRDB 管理者がインメモリデータバッファを定義する必要はありません。

バッチ処理の実行中はインメモリデータバッファ上のデータだけが更新されて，ディスク上のデータは更新されません。バッチ処理が完了したあとにコマンド（pdhold コマンド）を実行して，インメモリデータバッファ上の更新情報を一括してディスクに書き込みます。シンクポイント時もディスクへの書き込みが発生しません。なお，ログ取得モードであれば，インメモリデータ処理中でも，通常と同じようにデータベースの更新ログが取得されます。障害が発生した場合，そのログを使って，データベースを最新の状態に回復できます。

1.2.2　インメモリデータバッファの優位点

ここでは，グローバルバッファと比較したときのインメモリデータバッファの優位点について説明します。性能面ではインメモリデータバッファの方が優れていますが，障害発生時の対処方法がグローバルバッファに比べてやや複雑になります。

インメモリデータバッファとグローバルバッファのそれぞれの優位点を次に説明します。

(1)　インメモリデータバッファが優位な点

インメモリデータバッファが優位な点を次に示します。

- ディスク入出力回数が少ない

  グローバルバッファを使用している場合はディスク入出力が定期的に発生しますが，インメモリデータバッファを使用している場合は，ディスク入出力が発生するのはインメモリ化した時点とディスクへのデータ書き込み時点の 2 回だけです。シンクポイント時にもディスクへの書き込みが発生しません。そのため，インメモリデータバッファを使用する方が，グローバルバッファを使用したときに比べてディスク入出力回数が少なくなります。

- バッファの管理処理が少ない

  グローバルバッファを使用している場合は次に示すバッファ管理処理が発生しますが，インメモリデータバッファを使用している場合はこれらのバッファ管理処理が発生しません。

  - キャッシュバッファチェーン（バッファへのポインタの集合）の管理処理
  - LRU リストの管理処理
  - 更新バッファの管理処理

  インメモリデータバッファを使用した方が，これらの管理処理が不要になり，その分性能が向上します。また，トランザクションの同時実行性も向上します。

- チューニングが不要

  グローバルバッファの場合はヒット率を意識してグローバルバッファのチューニングをする必要があります。一方，インメモリデータバッファの場合はヒット率を意識する必要がないため，チューニングが必要ありません。そのため，チューニングに掛かるコストを削減できます。

(2) グローバルバッファが優位な点

グローバルバッファが優位な点を次に示します。

- HiRDB が異常終了したときのデータの回復方法

  グローバルバッファを使用している場合は，HiRDB が異常終了しても，再開始時に最新の状態に HiRDB が回復します。インメモリデータバッファを使用している場合は，HiRDB が異常終了して再開始しても，HiRDB は自動回復しません。この場合，HiRDB 管理者がバックアップとログを使用して最新の状態に回復するか，またはインメモリ化時点以降の処理を再実行する必要があります。

1.2.3 インメモリデータ処理に関する用語

インメモリデータ処理に関する用語を次の表に示します。

表 1-1 インメモリデータ処理に関する用語

用語	説明
インメモリデータ処理	RD エリア内の全データをメモリ（インメモリデータバッファ）上に常駐させて，ディスク入出力回数を抑える処理方式のことです。
インメモリデータバッファ	インメモリデータ処理で使用するデータバッファのことです。
インメモリ化	インメモリデータ処理で，RD エリア内の全データをメモリ上に一括して読み込み，常駐させることです。
インメモリ RD エリア	インメモリデータ処理の対象となっている RD エリアのことです。この RD エリア内の全データがメモリ上に常駐します。
DB 同期状態	インメモリ RD エリア内のデータとインメモリデータバッファ上のデータの同期が取れている状態のことです。
DB 非同期状態	インメモリ RD エリア内のデータとインメモリデータバッファ上のデータの同期が取れていない状態のことです。
RD エリア障害	インメモリ RD エリアに障害が発生した状態です。
バッファ障害	インメモリデータバッファに障害が発生した状態です。

1.3　インメモリデータ処理を適用すると効果が期待できるケース

次の場合にインメモリデータ処理を適用すると効果が期待できます。

- **処理時間の長いバッチ処理を実行している場合**

  大量のデータ更新を行う，処理時間の長いバッチ処理にインメモリデータ処理を適用すると，効果が期待できます。また，ログレスモードを適用すると，さらに処理時間を短縮できます。

  バッチ処理に掛かる時間が長いほど，ディスク入出力回数に差が出るため，インメモリデータ処理を適用したときの効果が期待できます。

- **複数のバッチ処理を連続して実行している場合**

  複数のバッチ処理を連続して実行する場合にインメモリデータ処理を適用すると効果が期待できます。

  バッチ処理を連続して行うほど，ディスク入出力回数に差が出るため，インメモリデータ処理を適用したときの効果が期待できます。

- **グローバルバッファの排他競合が原因で性能が上がらない場合**

  グローバルバッファの排他競合が原因で性能が上がらない場合に，インメモリデータ処理を適用すると性能向上が期待できます。

  グローバルバッファを使用する場合（インメモリデータ処理を使用しない場合），参照または更新処理の発生時，グローバルバッファ上にデータがキャッシュされているかどうかを検索する処理（キャッシュサーチ処理）が行われます。このとき，グローバルバッファの全ページに対して排他が掛かるため，ほかの参照または更新処理は排他待ちとなります。インメモリデータ処理の場合は，インメモリデータバッファ上に全データがキャッシュされているため，キャッシュサーチ処理が発生しません。グローバルバッファの全ページに対する排他制御がなくなるため，同時実行性の向上が見込まれ，その分性能向上が期待できます。例えば，同じグローバルバッファを割り当てている複数の RD エリアを更新するバッチ処理などで排他競合が多発している場合に，インメモリデータ処理を適用すると性能向上が期待できます。

  なお，排他競合の発生頻度については，統計解析ユティリティのグローバルバッファに関する統計情報で確認できます。

- **メインフレームからデータを移行する場合**

  メインフレームの膨大なデータを HiRDB に移行する場合（データを HiRDB のデータベースにデータロードする場合）に，インメモリデータ処理を適用すると効果が期待できます。

> **参考**
>
> インメモリデータ処理はユティリティにも適用できます。データロードや表の再編成など，処理時間の長いユティリティを実行する場合にも，効果が期待できます。

1.4　インメモリデータ処理を適用しても効果が期待できないケース

次のような場合にインメモリデータ処理を適用しても効果が期待できません。

- **処理時間の短いバッチ処理を実行している場合**

  インメモリデータ処理では，RD エリア内の全データをインメモリ化します。そのため，RD エリアのデータ量に比例して，インメモリ化に掛かる時間や，インメモリデータバッファ上のデータをディスクに書き込む時間が長くなります。これらの処理に掛かる時間とバッチ業務の処理時間を考慮した上で，インメモリデータ処理の適用を検討してください。

  例えば，処理時間が 30 分のバッチ業務にインメモリデータ処理を適用したとします。インメモリデータ処理の適用によってバッチ業務の処理時間は 20 分と短くなりましたが，インメモリ化の処理に 10 分，インメモリデータバッファ上のデータをディスクに書き込む時間が 10 分掛かると，合計 40 分掛かり，インメモリデータ処理を適用する前より処理時間が長くなってしまいます。

  > **参考**
  >
  > ここの説明で使用している時間はあくまで例です。業務の内容によって各処理に掛かる時間は変わります。

- **RD エリア内の全データをグローバルバッファ上にキャッシュしている場合**

  RD エリア内の全データをグローバルバッファ上にキャッシュしている場合と，インメモリデータ処理を適用した場合では，あまり性能に差がありません。

1.5　インメモリデータ処理が適用できないケース

1.5.1　インメモリデータ処理が適用できないケースの説明

ここでは，インメモリデータ処理が適用できないケースについて説明します。

(1)　インメモリデータ処理を適用できない RD エリア

ユーザ用 RD エリア，ユーザ LOB 用 RD エリア，およびリスト用 RD エリアに対してだけ，インメモリデータ処理を適用できます。次に示す RD エリアについては，インメモリデータ処理を適用できません。

- 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
- レジストリ用 RD エリア
- レジストリ LOB 用 RD エリア

なお，ユーザ用 RD エリアおよびユーザ LOB 用 RD エリアについても，オープン契機が SCHEDULE 属性の RD エリアの場合はインメモリ化できません。また，ユーザ用 RD エリアが共用 RD エリアの場合もインメモリ化できません。

(2)　リアルタイム SAN レプリケーション（ログ同期方式を除く）を使用している場合

全同期方式，全非同期方式，またはハイブリッド方式のリアルタイム SAN レプリケーションを使用している場合は，インメモリデータ処理を適用できません。

ログ同期方式のリアルタイム SAN レプリケーションの場合は，インメモリデータ処理を適用できます。ただし，ログ適用サイトのデータベースの更新ができなくなるため，ログレスモードは使用しないでください。

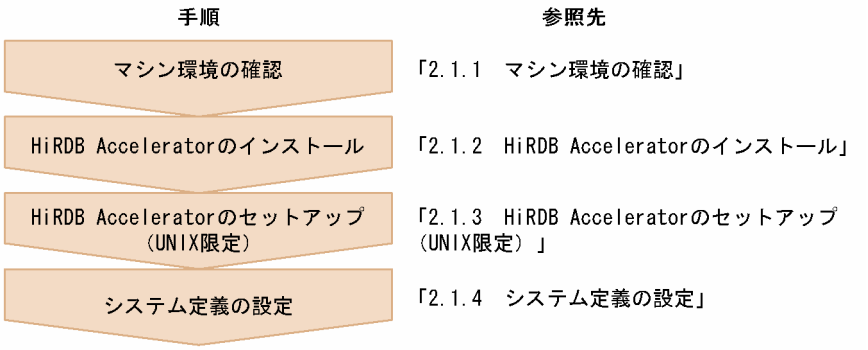
2 インメモリデータ処理を実行するための環境を作る

この章では，インメモリデータ処理を実行するために必ず行う準備と，処理性能を向上させるために考えておくことについて説明します。

2.1　必ず行う準備

ここでは，インメモリデータ処理を実行するために必ず行う準備について説明します。準備の手順を次の図に示します。

図 2-1　必ず行う準備



なお，これ以降の説明は，HiRDB のインストールおよびセットアップは完了していて，すでに HiRDB を運用していることを前提としています。

2.1.1　マシン環境の確認

インメモリデータ処理を実行するためには，マシン環境がここで説明する条件を満たしている必要があります。マシン環境について確認してください。

(1)　前提プラットフォーム

インメモリデータ処理は次のプラットフォームで実行できます。

- AIX（64 ビットモード）
- Linux (EM64T)
- Windows (x64)

(2)　メモリ容量の確認

インメモリデータ処理を実行するためのメモリ容量が十分にあるか確認してください。インメモリデータ処理に必要なメモリ容量は，インメモリ化する RD エリアの容量に，HiRDB がインメモリデータバッファを管理するための領域を加えた値になります。

インメモリデータ処理で必要なメモリ所要量については，マニュアル「HiRDB システム導入・設計ガイド」を参照してください。

(3) OS のオペレーティングシステムパラメタの確認（UNIX 限定）

次に示す設定をするオペレーティングシステムパラメタの値を見直してください。

- 共用メモリセグメントの合計サイズ
- システム上の共用メモリセグメントの最大数
- 1 プロセス当たりの共用メモリセグメントの最大数

OS の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，マニュアル「HiRDB システム導入・設計ガイド」または OS のマニュアルを参照してください。

2.1.2 HiRDB Accelerator のインストール

ここでは，HiRDB Accelerator のインストールおよびアンインストールの手順について説明します。

(1) インストール手順

HiRDB Accelerator のインストール手順を次に示します。

Windows 版の場合

1. pdstop コマンドで HiRDB を正常終了します。
2. [コントロールパネル］の［サービス］で HiRDB のサービスを停止します。
3. HiRDB Accelerator をインストールします。Administrators 権限を持つユーザがインストールを実行してください。手順はインストーラの指示に従ってください。

   なお，HiRDB/パラレルサーバの場合は，すべてのサーバマシンに HiRDB Accelerator をインストールしてください。系切り替え機能を使用している場合は予備系のサーバマシンにも，HiRDB Accelerator をインストールしてください。

   また，マルチ HiRDB 構成のときは，セットアップ識別子の指定が必要です。セットアップ識別子は pdntenv コマンドで確認できます。

UNIX 版の場合

1. pdstop コマンドで HiRDB を正常終了します。
2. 日立 PP インストーラを実行して HiRDB Accelerator をインストールします。

   なお，HiRDB/パラレルサーバの場合は，すべてのサーバマシンに HiRDB Accelerator をインストールしてください。

   また，系切り替え機能を使用している場合は予備系のサーバマシンに，リアルタイム SAN レプリケーションを使用している場合はリモートサイトのサーバマシンにも，HiRDB Accelerator をインストールしてください。

注意事項

HiRDB をバージョンアップした場合（例えば，バージョン 08-03 から 09-00 にバージョンアップした場合など）は，HiRDB Accelerator を再インストールする必要があります。

(2) アンインストール手順

HiRDB Accelerator のアンインストール手順を次に示します。

Windows 版の場合

1. pdstop コマンドで HiRDB を正常終了します。
2. [コントロールパネル] の [サービス] で HiRDB のサービスを停止します。
3. [コントロールパネル] の [アプリケーションの追加と削除] から HiRDB Accelerator をアンインストールします。

UNIX 版の場合

1. pdstop コマンドで HiRDB を正常終了します。
2. pdopsetup コマンドで HiRDB Accelerator のセットアップを解除します。

   なお，HiRDB/パラレルサーバの場合は，すべてのサーバマシンで pdopsetup コマンドを実行してください。

   ```
   pdopsetup -d -k acl $PDDIR
   ```

   $PDDIR には，HiRDB 運用ディレクトリを指定してください。
3. 日立 PP インストーラを実行して HiRDB Accelerator をアンインストールします。

2.1.3 HiRDB Accelerator のセットアップ（UNIX 限定）

UNIX 版の場合，HiRDB Accelerator のセットアップをしてください。セットアップは，pdopsetup コマンドで実行します。

なお，HiRDB/パラレルサーバの場合，すべてのサーバマシンで pdopsetup コマンドを実行する必要があります。

2.1.4 システム定義の設定

インメモリデータ処理を実行するには，次に示すオペランドを指定してください。

- pd_max_resident_rdarea_no

インメモリ化する RD エリアの最大数を指定します。このオペランドに 1 以上の値を指定すると，インメモリデータ処理を実行できます。

OS がページ固定に対応している Windows で，このオペランドに 1 以上の値を指定した場合は，SHMMAX オペランドに Windows の Large Page のページサイズに切り上げた値を指定してください。

Windows がページ固定に対応しているかどうか，および Large Page のページサイズについては，pdntenv -os コマンドで確認してください。

- pd_max_resident_rdarea_shm_no

  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の最大数を指定します。

注意事項

インメモリ化する RD エリアには，pdbuffer オペランドでグローバルバッファが割り当てられている必要があります。グローバルバッファが割り当てられていない RD エリアは，インメモリ化できません。

2.2　インメモリデータ処理をより効果的に実行するために考えておくこと

ここでは，インメモリデータ処理をより効果的に実行するために検討するとよいポイントについて説明します。あらかじめインメモリデータ処理に適したシステム構成にしておくことで，処理性能の向上が期待できます。

2.2.1　どの RD エリアをインメモリ化するか

インメモリデータ処理をより効果的に実行するには，表格納 RD エリアだけではなく，インデクス格納 RD エリアも一緒にインメモリ化してください。

特に，次に示す RD エリアはまとめてインメモリ化してください。

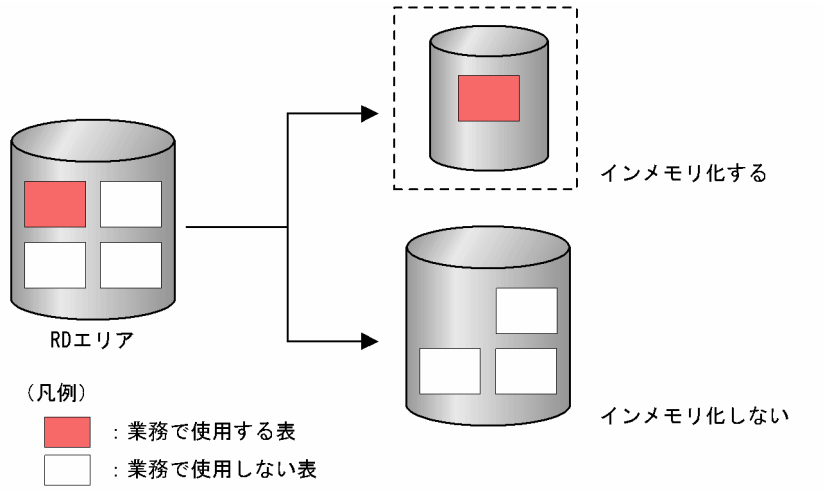
- 表に定義されているインデクスを格納している RD エリア
- 表を横分割している場合，その横分割表が格納されているすべての RD エリア

2.2.2　RD エリア内の表構成にむだがないか

インメモリ化する RD エリアに，業務ではアクセスしない表が混在している場合，そのような表は別の RD エリアに格納することをお勧めします。これによって，メモリ領域がむだに確保されることを防ぎます。

RD エリア内に使用しない表が混在している場合の対処例を次の図に示します。

図 2-2　RD エリア内に使用しない表が混在している場合の対処例



2.2.3　一括指定ができるRDエリア名になっているか

インメモリデータ処理に使用するコマンド（pdmemdbコマンド，pdholdコマンド）は，RDエリア名一括指定ができます。RDエリア名一括指定ができるように，RDエリア名の末尾をそろえておくなどの工夫をすると便利です。

（例）

インメモリ化するRDエリア名の末尾を次のようにそろえます。

RDDATA01_mem，RDDATA02_mem，RDIDX01_mem，RDIDX02_mem

上記四つのRDエリアをまとめてインメモリ化する場合，次のように指定できます。

```
pdmemdb -k stay -r “*_mem”
```

運用コマンドでのRDエリア名一括指定については，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。

2.2.4　ログレスモードを適用できるか

インメモリデータ処理でバッチ業務を実行する場合，ログレスモードで実行すると，より処理時間を短縮できます。バッチ業務を実行する場合は，ログレスモードの適用を検討してください。

3 インメモリデータ処理を適用する

この章では，業務の種類に応じたインメモリデータ処理の適用例について説明します。

3.1　基本的な運用の流れ

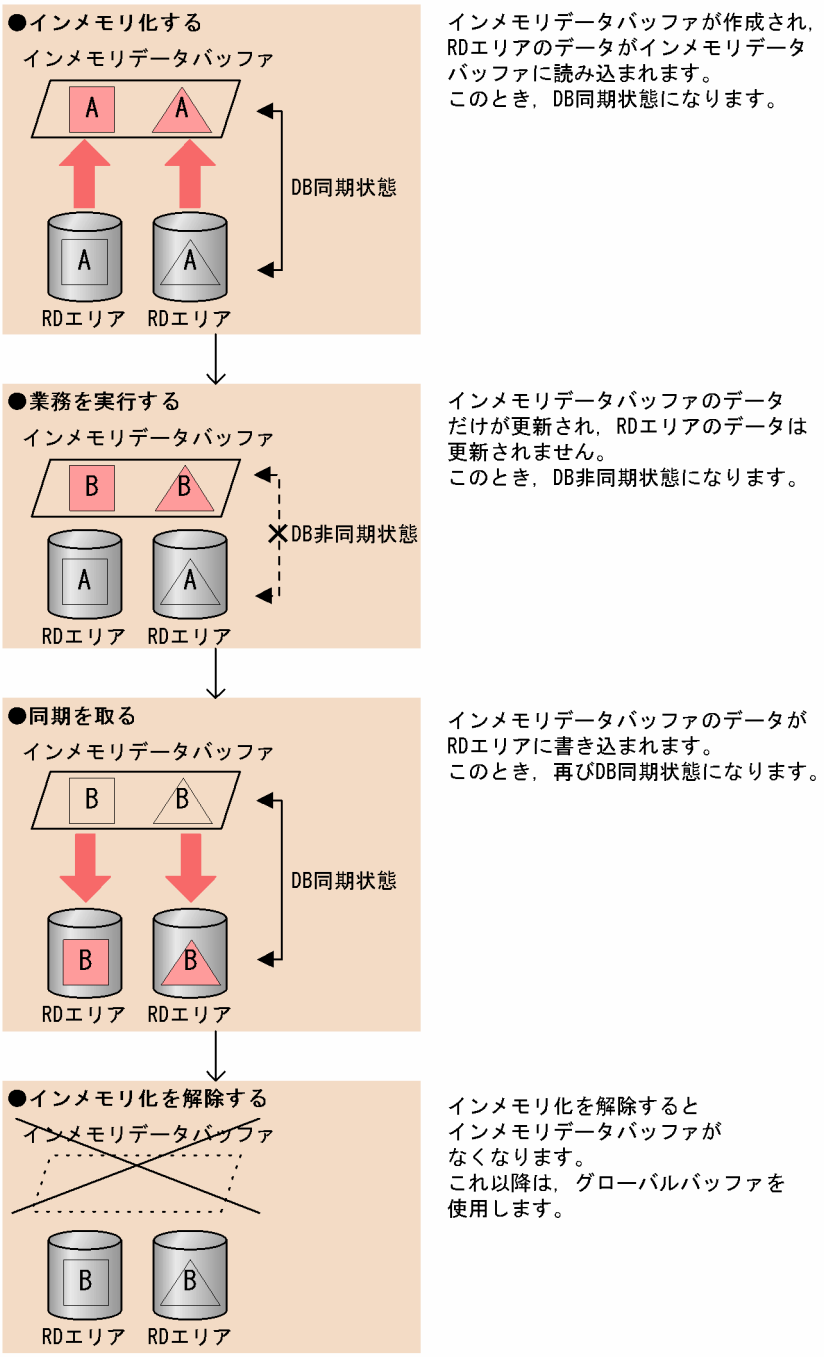
ここでは，インメモリデータ処理を実行するときの基本的な運用の流れについて説明します。

運用には，大きく分けて次の四つのステップがあります。

1. RD エリアをインメモリ化する
2. 業務を実行する
3. インメモリデータバッファと RD エリアの同期を取る
4. インメモリ化を解除する

基本的な運用の流れを次の図に示します。

図 3-1　基本的な運用の流れ



ポイント

RD エリアのデータとインメモリデータバッファのデータが同じ場合を **DB 同期状態**といい，異なる場合を **DB 非同期状態**といいます。

ステップ 2.で業務を実行している間は，インメモリデータバッファ上のデータだけが更新され，RD エリアのデータは更新されないため，DB 非同期状態になります。インメモリデータバッファと RD エリアの状態は，pddbls コマンドで確認できます。詳細は，「インメモリデータバッファの利用状況を確認するには」を参照してください。

注意事項

ステップ 1.のインメモリ化は，RD エリアのデータ量に比例して時間が掛かります。その間はシンクポイントが取得されないため，注意してください。

この基本的な運用の流れを基に，3.2 ではバッチ業務に適用する場合の手順，3.3 ではオンライン業務に適用する場合の手順について説明します。

3.2　バッチ業務に適用する場合

3.2.1　バッチ業務に適用する場合の運用

ここでは，インメモリデータ処理をバッチ業務に適用する場合の運用について，例題を使って説明します。

(1)　例題の条件

この例題での条件を次に示します。

- 二つのバッチ業務を連続して実行します。
- バッチ業務でアクセスする RD エリアは，RDDATA01（表格納 RD エリア）および RDIDX01（インデクス格納 RD エリア）です。
- ログレスモードでバッチ業務を実行します。

運用の流れを次の図に示します。

図 3-2　バッチ業務に適用する場合の運用の流れ

参考

初期データロードにもインメモリデータ処理を適用できます。その際，長時間シンクポイントを取得できなくなるのを防ぐため，同期点指定のデータロードの適用を検討してください。同期点指定の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。

初期データロードにインメモリデータ処理を適用する場合の手順は，バッチ業務に適用する場合の手順と同じです。(2)(b)および(d)のバッチ業務を pdload コマンドによる初期データロードに置き換えてお読みください。

なお，初期データロードの場合，(2)(a)のバックアップの取得は必要ありません。

(2)　コマンド実行手順

コマンドの実行手順を次に示します。

(a)　RD エリアをインメモリ化する

1. RDDATA01 および RDIDX01 を閉塞クローズ状態にします。

```
pdhold -r RDDATA01,RDIDX01 -c
```

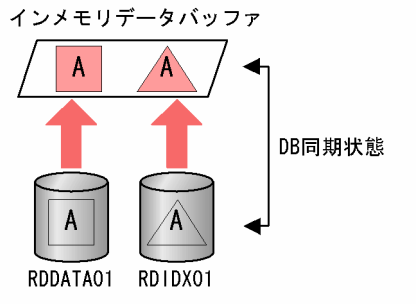
2. システムログファイルをスワップします。

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

3. RDDATA01 および RDIDX01 をインメモリ化します。

```
pdmemdb -k stay -r RDDATA01,RDIDX01
```

このとき，インメモリデータバッファと RD エリアは，DB 同期状態になります。



なお，ここでは，複数の RD エリアをまとめてインメモリ化しています。この場合の注意事項については，「複数の RD エリアを一度にインメモリ化する場合の注意事項」を参照してください。

4. RDDATA01 および RDIDX01 の閉塞クローズを解除します。

```
pdrels -r RDDATA01,RDIDX01 -o
```

5. RDDATA01 および RDIDX01 のバックアップを取得します。

```
pdcopy -m C:¥rdarea¥mast¥mast01 -M r
-r RDDATA01,RDIDX01
-b C:¥pdcopy¥backup01 -p C:¥pdcopy¥list01
```

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。

ポイント

ここでは，RD エリアの閉塞クローズを解除してから参照可能モード（-M r）でバックアップを取得しています。通常と異なり，参照可能モードでバックアップを取得している間も更新ができます。このため，バックアップの取得中にバッチ業務を実行できます。インメモリ RD エ

リアのバックアップ取得の仕組みについては，「バックアップを取得するときに気をつけること」を参照してください。

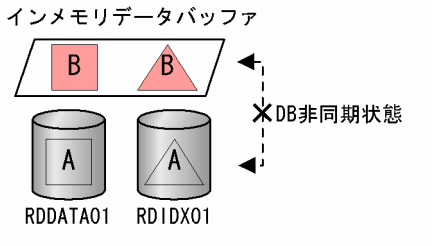
注意事項

インメモリ化するときの RD エリアの閉塞は，pdhold -c コマンドで行ってください。pdhold -c コマンドで閉塞クローズ状態にすることによって，ほかのトランザクションが対象 RD エリアにアクセスできないようにします。

(b) 一つ目のバッチ業務を実行する

1. バッチ業務を実行します。

インメモリデータバッファの内容が更新され，DB 非同期状態になります。



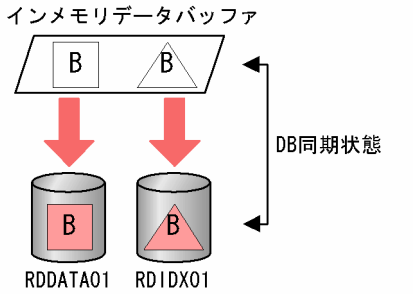
(c) インメモリデータバッファと RD エリアの同期を取る

1. インメモリデータバッファと RD エリアの同期を取ります。

```
pdhold -r RDDATA01,RDIDX01 -b
```

一つ目のバッチ業務が終わったら，インメモリデータバッファのデータを RD エリアに書き込み，インメモリデータバッファと RD エリアの同期を取ります。これによって，DB 同期状態になります。

pdhold -b コマンドを実行すると，インメモリデータバッファのデータを RD エリアに書き込みます。



2. システムログファイルをスワップします。

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

3. RDDATA01 および RDIDX01 の閉塞を解除します。

```
pdrels -r RDDATA01,RDIDX01
```

4. RDDATA01 および RDIDX01 のバックアップを取得します。

```
pdcopy -m C:¥rdarea¥mast¥mast01 -M r
-r RDDATA01,RDIDX01
-b C:¥pdcopy¥backup02 -p C:¥pdcopy¥list02
```

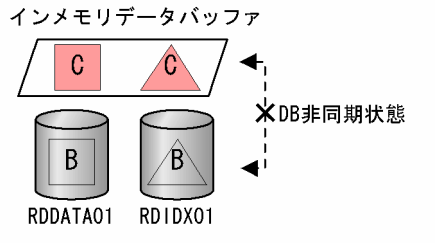
ポイント

ログレスモードで業務を実行しているため，一つ目のバッチ業務が終わったら，必ずインメモリデータバッファと RD エリアの同期を取ります。ここで同期を取らないと，二つ目のバッチ業務を実行中にバッファ障害が発生した場合，一つ目のバッチ業務から再実行しなくてはなりません。

(d) 二つ目のバッチ業務を実行する

1. バッチ業務を実行します。

インメモリデータバッファの内容が更新され，DB 非同期状態になります。



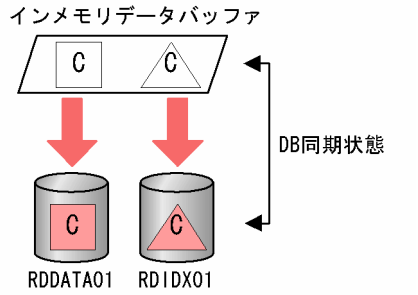
(e) インメモリデータバッファと RD エリアの同期を取る

1. インメモリデータバッファと RD エリアの同期を取ります。

```
pdhold -r RDDATA01,RDIDX01 -c
```

二つ目のバッチ業務が終わったら，インメモリデータバッファのデータを RD エリアに書き込み，インメモリデータバッファと RD エリアの同期を取ります。これによって，DB 同期状態になります。

pdhold -c コマンドを実行すると，インメモリデータバッファのデータを RD エリアに書き込みます。



注意事項

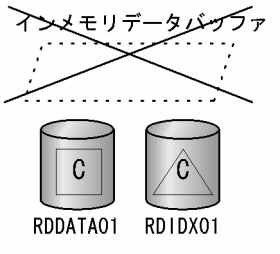
ここで，pdhold -c コマンドを使用しているのは，(f)でインメモリ化を解除するためです。インメモリ化を解除するときの RD エリアの閉塞は，pdhold -c コマンドで行います。pdhold -c コマンドで閉塞クローズ状態にすることによって，ほかのトランザクションが対象 RD エリアにアクセスできないようにします。

(f) インメモリ化を解除する

1. RDDATA01 および RDIDX01 のインメモリ化を解除します。

```
pdmemdb -k rels -r RDDATA01,RDIDX01
```

インメモリデータバッファがなくなります。これ以降は，グローバルバッファを使用します。



2. システムログファイルをスワップします。

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

3. RDDATA01 および RDIDX01 のバックアップを取得します。

```
pdcopy -m C:¥rdarea¥mast¥mast01 -M r
-r RDDATA01,RDIDX01
-b C:¥pdcopy¥backup03 -p C:¥pdcopy¥list03
```

ここでは前の手順と異なり，RD エリアの閉塞クローズを解除する前にバックアップを取得してください。バックアップを取得する前に RD エリアの閉塞クローズを解除すると，ほかの UAP からの更新が発生する場合があります。更新が発生したあとで RD エリアに障害が発生した場合，ログレスモードでバッチ業務を実行しているため，その更新内容を回復することができません。

4. RDDATA01 および RDIDX01 の閉塞を解除します。

```
pdrels -r RDDATA01,RDIDX01 -o
```

3.3　オンライン業務に適用する場合

3.3.1　オンライン業務に適用する場合の運用

ここでは，インメモリデータ処理をオンライン業務に適用する場合の運用について，例題を使って説明します。

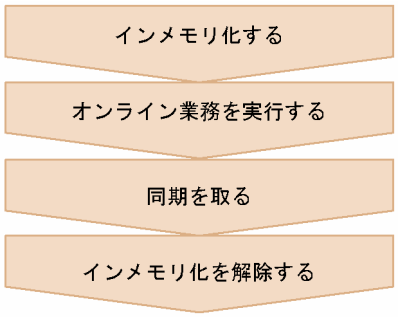
(1)　例題の条件

この例題での条件を次に示します。

- オンライン業務でアクセスするRDエリアは，RDDATA01（表格納RDエリア）およびRDIDX01（インデクス格納RDエリア）です。

運用の流れを次の図に示します。

図 3-3　オンライン業務に適用する場合の運用の流れ



(2)　コマンド実行手順

コマンドの実行手順を次に示します。

(a)　RDエリアをインメモリ化する

1. RDDATA01およびRDIDX01を閉塞クローズ状態にします。

```
pdhold -r RDDATA01,RDIDX01 -c
```

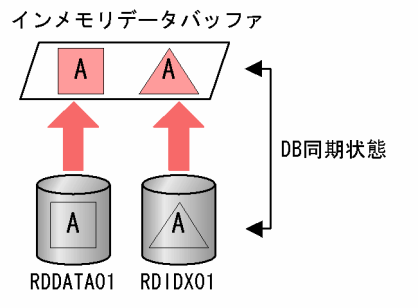
2. システムログファイルをスワップします。

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

3. RDDATA01およびRDIDX01をインメモリ化します。

```
pdmemdb -k stay -r RDDATA01,RDIDX01
```

このとき，インメモリデータバッファとRDエリアは，DB同期状態になります。



なお，ここでは，複数の RD エリアを一度にインメモリ化しています。この場合の注意事項については，「複数の RD エリアを一度にインメモリ化する場合の注意事項」を参照してください。

4. RDDATA01 および RDIDX01 の閉塞クローズを解除します。

```
pdrels -r RDDATA01,RDIDX01 -o
```

5. RDDATA01 および RDIDX01 のバックアップを取得します。

```
pdcopy -m C:¥rdarea¥mast¥mast01 -M r
-r RDDATA01,RDIDX01
-b C:¥pdcopy¥backup01 -p C:¥pdcopy¥list01
```

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。

ポイント

ここでは，RD エリアの閉塞クローズを解除してから参照可能モード（-M r）でバックアップを取得しています。通常と異なり，参照可能モードでバックアップを取得している間も更新ができます。このため，バックアップの取得中にオンライン業務を実行できます。インメモリ RD エリアのバックアップ取得の仕組みについては，「バックアップを取得するときに気をつけること」を参照してください。

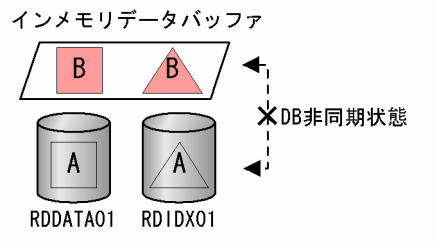
注意事項

インメモリ化するときの RD エリアの閉塞は，pdhold -c コマンドで行ってください。pdhold -c コマンドで閉塞クローズ状態にすることによって，ほかのトランザクションが対象 RD エリアにアクセスできないようにします。

(b) オンライン業務を実行する

1. オンライン業務を実行します。

これによって，インメモリデータバッファの内容が更新され，DB 非同期状態になります。



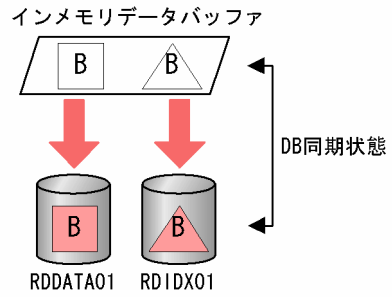
(c) インメモリデータバッファと RD エリアの同期を取る

1. インメモリデータバッファと RD エリアの同期を取ります。

```
pdhold -r RDDATA01,RDIDX01 -c
```

オンライン業務が終わったら，インメモリデータバッファのデータを RD エリアに書き込み，インメモリデータバッファと RD エリアの同期を取ります。これによって，DB 同期状態になります。

pdhold -c コマンドを実行すると，インメモリデータバッファのデータを RD エリアに書き込みます。



注意事項

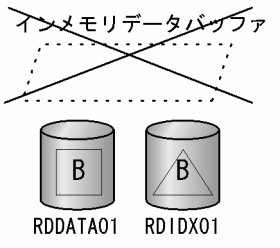
ここで，pdhold -c コマンドを使用しているのは，(d)でインメモリ化を解除するためです。インメモリ化を解除するときの RD エリアの閉塞は，pdhold -c コマンドで行います。pdhold -c コマンドで閉塞クローズ状態にすることによって，ほかのトランザクションが対象 RD エリアにアクセスできないようにします。

(d) インメモリ化を解除する

1. RDDATA01 および RDIDX01 のインメモリ化を解除します。

```
pdmemdb -k rels -r RDDATA01,RDIDX01
```

インメモリデータバッファがなくなります。これ以降は，グローバルバッファを使用します。



2. RDDATA01 および RDIDX01 の閉塞を解除します。

```
pdrels -r RDDATA01,RDIDX01 -o
```

3.4　テスト業務に適用する場合

3.4.1　テスト業務に適用する場合の運用

ここでは，インメモリデータバッファの更新内容を破棄したり，RD エリアのデータをインメモリデータバッファに再読み込みしたりするなどの運用方法を，インメモリデータ処理をテスト業務に適用する場合の応用例を使って説明します。

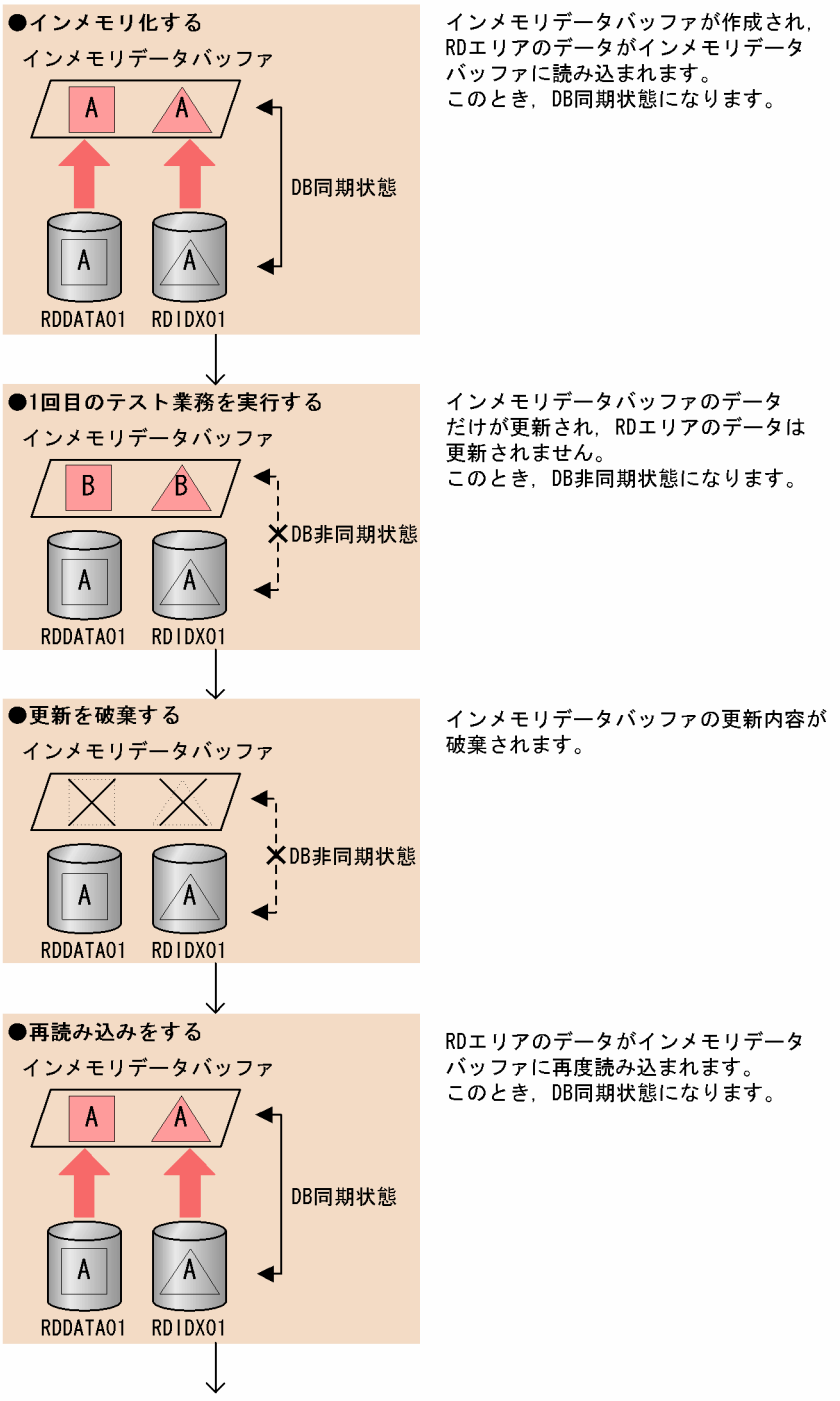
(1)　例題の条件

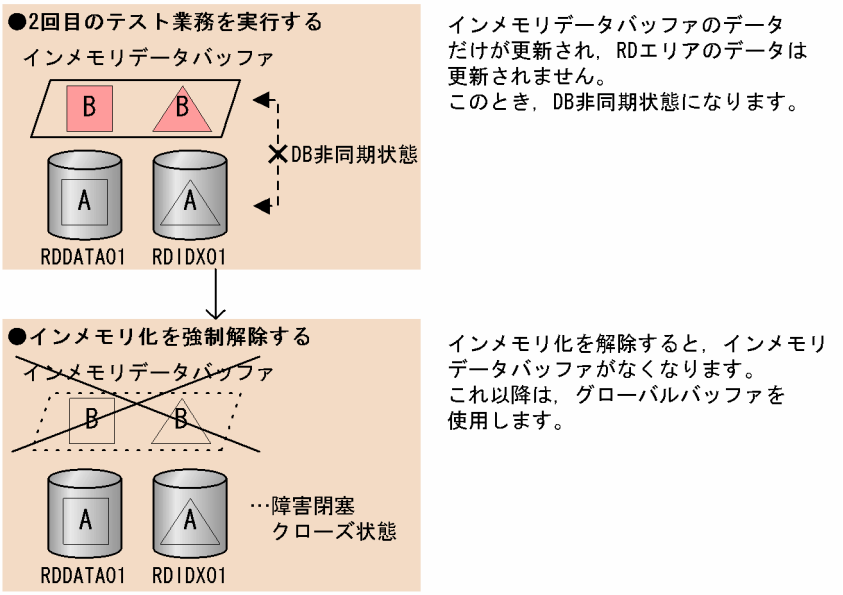
この例題での条件を次に示します。

- テスト業務を連続して 2 回実行します。
- テスト業務でアクセスする RD エリアは，RDDATA01（表格納 RD エリア）および RDIDX01（インデクス格納 RD エリア）です。
- テスト業務の更新内容は，RD エリアに書き込まないで破棄します。

運用の流れを次の図に示します。

図 3-4　テスト業務に適用する場合の運用の流れ





(2)　コマンド実行手順

コマンドの実行手順を次に示します。

(a)　RD エリアをインメモリ化する

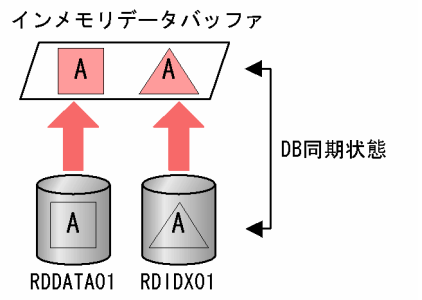
1. RDDATA01 および RDIDX01 を閉塞クローズ状態にします。

```
pdhold -r RDDATA01,RDIDX01 -c
```

2. RDDATA01 および RDIDX01 をインメモリ化します。

```
pdmemdb -k stay -r RDDATA01,RDIDX01
```

このとき，インメモリデータバッファと RD エリアは，DB 同期状態になります。



なお，ここでは，複数の RD エリアをまとめてインメモリ化しています。この場合の注意事項については，「複数の RD エリアを一度にインメモリ化する場合の注意事項」を参照してください。

注意事項

インメモリ化するときの RD エリアの閉塞は，pdhold -c コマンドで行ってください。pdhold -c コマンドで閉塞クローズ状態にすることによって，ほかの UAP が対象 RD エリアにアクセスできないようにします。

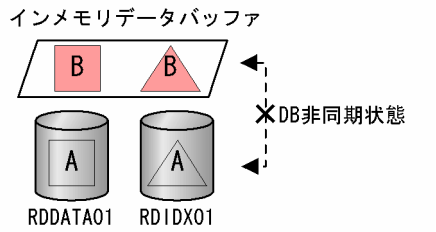
3. RDDATA01 および RDIDX01 の閉塞クローズを解除します。

```
pdrels -r RDDATA01,RDIDX01 -o
```

(b) テスト業務を実行する

1. 1 回目のテスト業務を実行します。

インメモリデータバッファの内容が更新され，DB 非同期状態になります。



(c) インメモリデータバッファ上の更新内容を破棄する

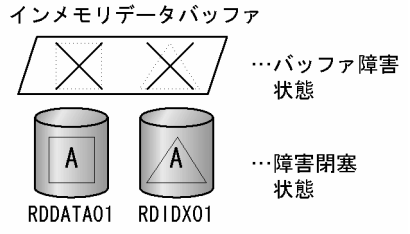
1. RDDATA01 および RDIDX01 を閉塞状態にします。

```
pdhold -r RDDATA01,RDIDX01
```

2. インメモリデータバッファ上のデータを破棄して更新を無効にします。

```
pdmemdb -k cancel -r RDDATA01,RDIDX01
```

このとき，インメモリデータバッファはバッファ障害状態に，RD エリアは障害閉塞状態になります。



(d) RD エリアのデータを再読み込みする

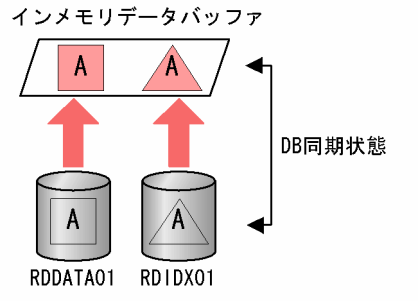
1. RDDATA01 および RDIDX01 をクローズ状態にします。

```
pdclose -r RDDATA01,RDIDX01
```

2. RD エリアのデータをインメモリデータバッファへ再読み込みします。

```
pdmemdb -k reload -r RDDATA01,RDIDX01
```

これによって，テスト業務実行前の状態に戻ります。このとき，インメモリデータバッファと RD エリアは，DB 同期状態になります。



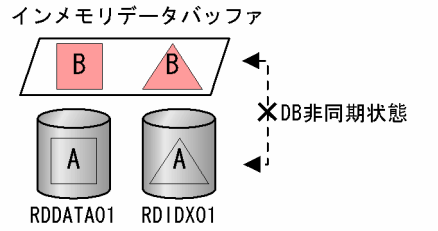
3. RDDATA01 および RDIDX01 の閉塞を解除します。

```
pdrels -r RDDATA01,RDIDX01
```

(e) テスト業務を実行する

1. 2 回目のテスト業務を実行します。

インメモリデータバッファの内容が更新され，DB 非同期状態になります。



(f) インメモリ化を強制解除する

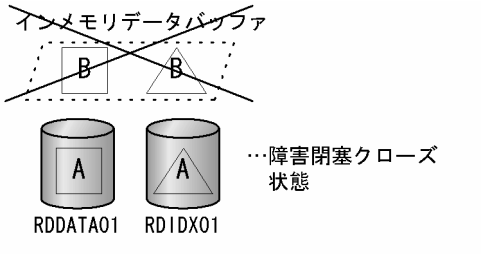
1. RDDATA01 および RDIDX01 を閉塞状態にします。

```
pdhold -r RDDATA01,RDIDX01
```

2. RDDATA01 および RDIDX01 のインメモリ化を強制解除します。

```
pdmemdb -k rels -r RDDATA01,RDIDX01 -d
```

これによって，インメモリデータバッファはなくなり，RD エリアは障害閉塞クローズ状態になります。なお，更新情報は RD エリアに反映されません。



3. RDDATA01 および RDIDX01 の閉塞を解除します。

```
pdrels -r RDDATA01,RDIDX01 -o
```

3.5　複数の RD エリアを一度にインメモリ化する場合の注意事項

複数の RD エリアをインメモリ化する場合，一度にまとめてインメモリ化する方法と，個別にインメモリ化する方法があります。運用が煩雑になるのを防ぐため，関連する RD エリアはまとめてインメモリ化することをお勧めしますが，その場合は次のことに注意してください。

3.5.1　インメモリ化に失敗したとき

複数の RD エリアのうち，一つでもインメモリ化に失敗した場合は，一緒に指定したそのほかの RD エリアもインメモリ化しません。この場合は，pdmemdb -k stay コマンドを再実行してください。

なお，HiRDB/パラレルサーバの場合は，バックエンドサーバごとにこの仕組みが適用されます。

（例）

pdmemdb -k stay -r RD01,RD02,RD03,RD04

RD01 と RD02 はバックエンドサーバ 1 で，RD03 と RD04 はバックエンドサーバ 2 で管理されています。RD02 のインメモリ化に失敗した場合，RD01 もインメモリ化しません。RD03 と RD04 はインメモリ化します。

注意事項

インメモリ化時だけではなく，インメモリ化の解除，再読み込み，インメモリデータバッファの破棄を行うときにも，この仕組みが適用されます。pdmemdb コマンドで複数の RD エリアを操作したときに，一部の RD エリアに対する処理でエラーが発生すると，すべての RD エリアの処理を無効化します。

3.5.2　セグメントの割り当てについて

インメモリデータバッファもグローバルバッファと同様に，共用メモリセグメント上に割り当てられます。

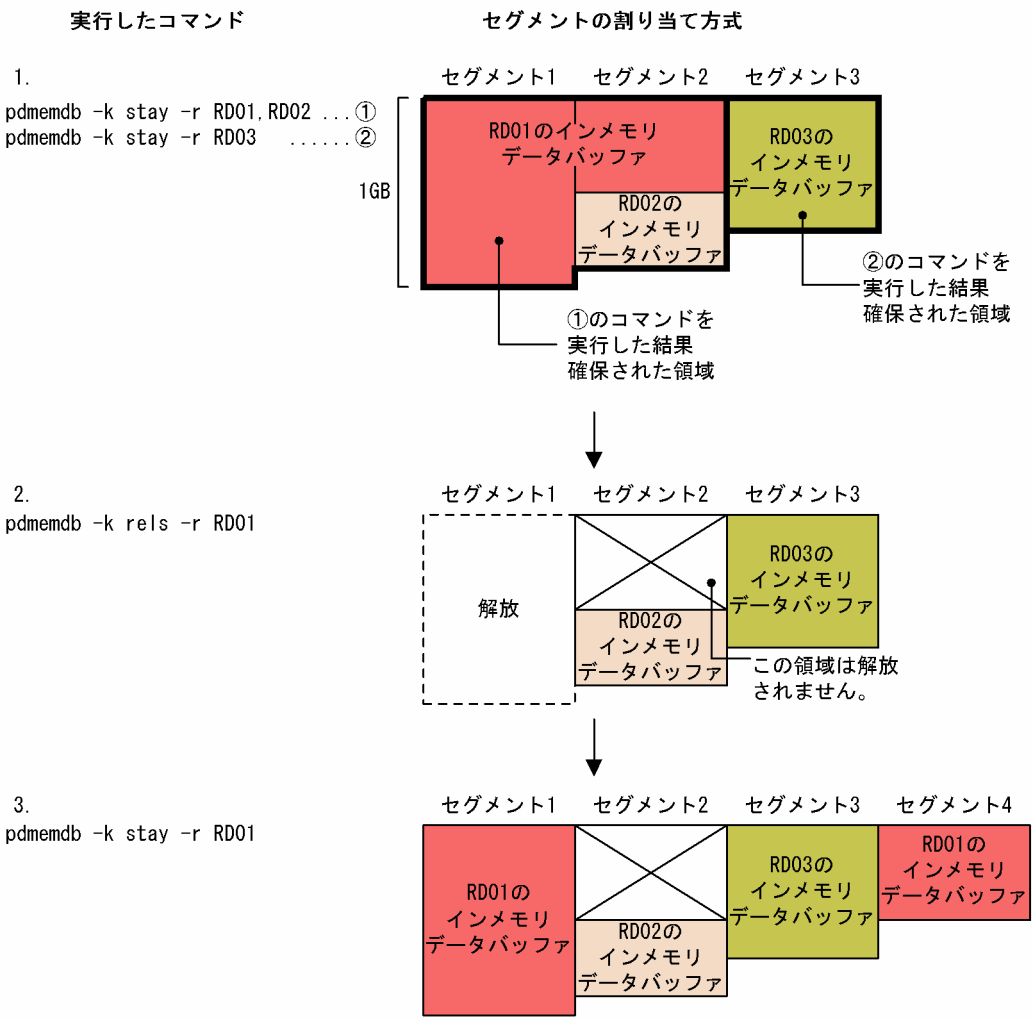
インメモリデータバッファに割り当てられる共用メモリセグメントを効率的に使用するために，複数の RD エリアをまとめてインメモリ化した場合は，インメモリ化を解除するときもまとめて解除してください。共用メモリセグメントの割り当ての仕組みを次に示す例を使って説明します。

（例）

- 共用メモリセグメントの最大サイズ：1GB
- RD エリア（RD01）のサイズ：1.5GB
- RD エリア（RD02）のサイズ：0.4GB
- RD エリア（RD03）のサイズ：0.7GB

この条件下での共用メモリセグメントの割り当ての仕組みを次の図に示します。

図 3-5　共用メモリセグメントの割り当ての仕組み



[説明]

1. RD01 と RD02 をまとめてインメモリ化し，それとは別に RD03 をインメモリ化しました。

   RD01 のインメモリデータバッファはセグメント 1 およびセグメント 2 に割り当てられます。RD02 のインメモリデータバッファは RD01 と同じセグメント 2 に割り当てられます。

   RD03 のインメモリデータバッファは RD01，RD02 とは別のセグメント（セグメント 3）に割り当てられます。

2. RD01 のインメモリ化を解除すると，セグメント 1 は解放されて利用できますが，セグメント 2 の RD01 のインメモリデータバッファに割り当てられた領域は解放されません。そのため，このメモリ領域がむだになります。

3. RD01 を再度インメモリ化すると，新しいセグメント（セグメント 1 とセグメント 4）が割り当てられます。

ポイント

メモリ領域がむだになるのを防ぐため，複数の RD エリアをまとめてインメモリ化した場合は，インメモリ化の解除もまとめて行うようにしてください。

4 運用時に気をつけること

インメモリデータ処理を実行する場合に，そのほかの機能や運用で注意が必要なときがあります。この章では，そのような注意事項について説明します。

4.1　バックアップを取得するときに気をつけること

ここでは，インメモリ RD エリアのバックアップを取得するときの注意点について説明します。

なお，インメモリ RD エリアのバックアップを取得する場合，バックアップ取得モードは参照・更新不可能モード（-M x），または参照可能モード（-M r）のどちらかです。更新可能モード（-M s）でのバックアップ取得はできません。

また，インメモリ RD エリアの差分バックアップは取得できません。

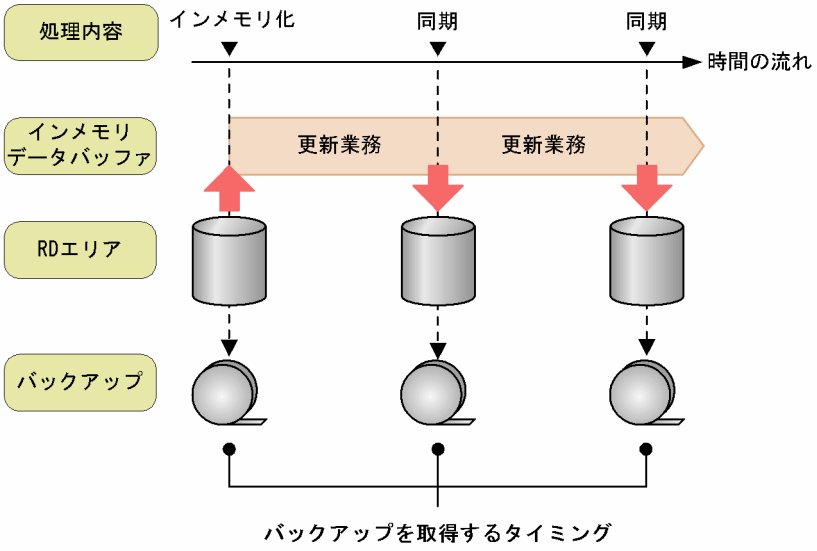
4.1.1　バックアップを取得するタイミング

バックアップは，次のタイミングで取得してください。

- RD エリアをインメモリ化したあと
- インメモリデータバッファと RD エリアの同期を取ったあと

バックアップを取得するタイミングを次の図に示します。

図 4-1　バックアップを取得するタイミング



RD エリアとインメモリデータバッファの両方に障害が発生した場合，pdrstr コマンドで RD エリアを回復します。前記で説明した時点でバックアップを取得しないと，インメモリ化時点または同期時点（例えば，バッチ業務の開始前の時点）に RD エリアを回復するときの手順が煩雑になります。

また，バックアップを取得するときは，インメモリ RD エリア（表格納 RD エリア）だけではなく，関連する RD エリア（インデクス格納 RD エリアや，ユーザ LOB 用 RD エリア）も一緒にバックアップを取得してください。関連する RD エリアと一緒にバックアップを取得しておかないで，インメモリ RD エリアだけをバックアップ取得時点に回復すると，関連する RD エリアとデータの不整合が発生するおそれがあります。

ポイント

バックアップを取得するときは，次のことに注意してください。

- DB 同期状態のときにバックアップを取得する
- 関連する RD エリアはまとめてバックアップを取得する

4.1.2　参照可能モード（-M r）でバックアップを取得する場合

参照可能モードでバックアップを取得する場合，インメモリデータ処理を実行していないときの運用と異なる点について説明します。

(1)　同時に更新業務を実行できる

参照可能モードでバックアップを取得する場合，同時に更新業務を実行できます（RD エリアのバックアップを取得しながら，インメモリデータバッファの内容を更新できます）。

(2)　バックアップ取得の仕組み

参照可能モードでバックアップを取得する場合，通常はグローバルバッファの更新情報が RD エリアに書き込まれてから，バックアップが取得されます。これによって，最新の状態のバックアップが取得できます。

しかし，インメモリ RD エリアのバックアップを参照可能モードで取得する場合，インメモリデータバッファの更新情報は RD エリアに書き込まれません。そのため，いったんインメモリデータバッファと RD エリアの同期を取り，インメモリデータバッファの更新情報を RD エリアに書き込んでからバックアップを取得してください。

バックアップ取得の手順

参照可能モードでバックアップを取得する場合の手順を次に示します。

1. pdhold -b コマンドでインメモリデータバッファと RD エリアの同期を取ります。
2. pdlogswap コマンドでシステムログファイルをスワップします。
3. pdrels コマンドで RD エリアの閉塞を解除します。
4. pdcopy -M r コマンドで 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。
5. 更新業務を開始します。

参考

「インメモリデータバッファと RD エリアの同期を取る」に，運用の流れの中でのバックアップの取得手順が記載されています。

ポイント

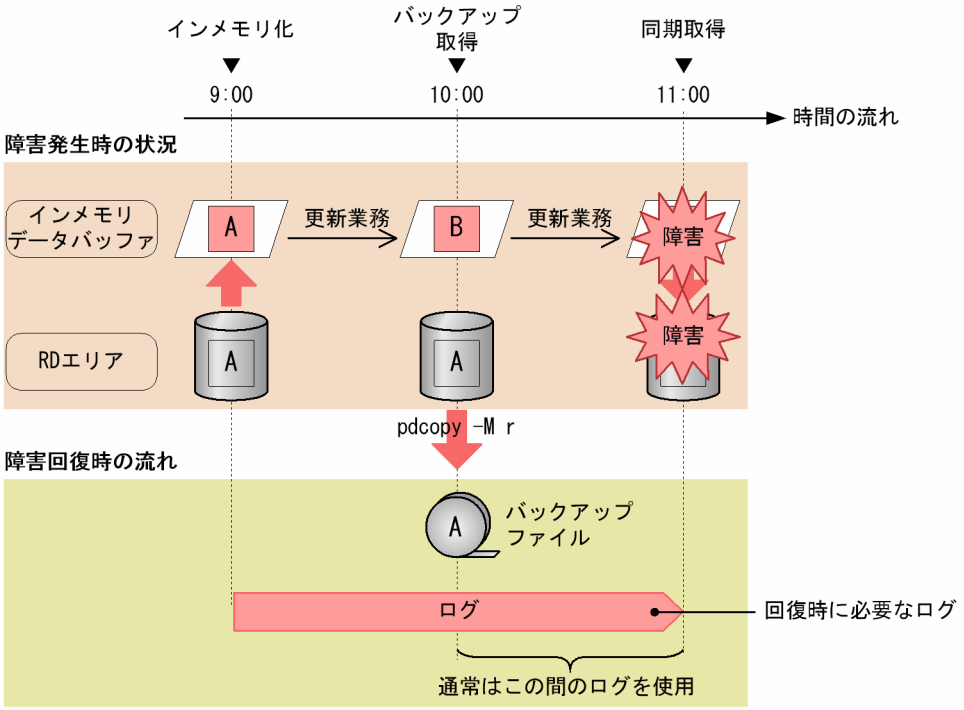
通常はバックアップとバックアップ取得時点以降のログがあれば，RD エリアを最新の状態に回復できますが，DB 非同期状態のときにバックアップを取得した場合，バックアップ取得以前のログも必要になります。

RD エリアを最新の状態に回復するには，インメモリ化した時点（または前回の同期取得時点）からのログが必要になります。DB 非同期状態でバックアップを取得した場合の例を次に示します。

DB 非同期状態でバックアップを取得した場合の例

DB 非同期状態でバックアップを取得した場合の例を次の図に示します。

図 4-2　DB 非同期状態でバックアップを取得した場合の例



[障害発生時の状況]

9:00 に RD エリアをインメモリ化し，その後インメモリデータバッファの内容を更新しました。DB 非同期状態のまま，10:00 に参照可能モードで RD エリアのバックアップを取得しました。その後，再び更新業務を行ったあと，更新内容を RD エリアに書き込むときに，インメモリデータバッファと RD エリアに障害が発生しました。

［障害回復時の流れ］

この場合，バックアップを取得した時刻は 10:00 ですが，バックアップファイルの内容は 9:00 の時点のままになっています。RD エリアを最新（11:00）の状態に回復するには，10:00 に取得したバックアップファイルと 9:00 からのログが必要になります。

通常では，バックアップ取得時点より古いログは使用しません。通常と同じ運用ができるように，インメモリ RD エリアのバックアップを参照可能モードで取得する場合は，インメモリデータバッファと RD エリアの同期を取ったあと（DB 同期状態のとき）に行うことをお勧めします。

参考

参照・更新不可能モード（-M x）でバックアップを取得する場合は，必ず RD エリアを pdhold -c コマンドで閉塞クローズ状態にするため，インメモリデータバッファの更新情報が RD エリアに書き込まれて DB 同期状態になります。したがって，RD エリアを最新の時点に回復する場合は，バックアップとバックアップ取得時点以降のログを使うという通常と同じ考えで RD エリアを回復できます。

4.2　HiRDB を終了するときに気をつけること

インメモリ RD エリアがある場合は，HiRDB の正常終了，および計画停止ができません。HiRDB を停止する場合は，インメモリ化を解除してから停止するか，強制終了してください。

なお，HiRDB を強制終了した場合，インメモリデータバッファ上の更新情報は RD エリアに反映されません。また，再開始後，インメモリ化は解除された状態になります。

注意事項

RD エリアは障害閉塞状態になります。pddbls コマンドで RD エリアの状態を確認してから，pdrels コマンドで障害閉塞状態を解除してください。

4.3　データベース構成変更ユティリティを使用するときに気をつけること

インメモリ RD エリアに対しては，データベース構成変更ユティリティを使った次の操作ができません。次の操作をする場合は，インメモリ化を解除してから実行してください。

- RD エリアの拡張
- RD エリアの再初期化
- RD エリアの削除
- RD エリアの移動
- RD エリアの属性変更
- RD エリアのレプリカ定義（インナレプリカ機能）
- RD エリアの構成情報複写（インナレプリカ機能）
- RD エリアの統合（インナレプリカ機能）

4.4　インナレプリカ機能を使用しているときに気をつけること

インメモリ RD エリアのレプリカ RD エリアは作成できません。また，レプリカ RD エリアをインメモリ化することもできません。オリジナル RD エリアはインメモリ化できますが，オリジナル RD エリアをインメモリ化したまま，更新可能なオンライン再編成を実行することはできません。更新可能なオンライン再編成をする場合は，オリジナル RD エリアのインメモリ化を解除してから実行してください。

注意事項

インメモリ RD エリアに追い付き状態管理表を作成しないでください。インメモリ RD エリアに追い付き状態管理表を格納した場合，HiRDB が異常終了するなどしてインメモリデータバッファの更新情報がなくなると，追い付き反映ができなくなることがあります。

ポイント

- レプリカ RD エリアは常にインメモリ化できません。
- pdorbegin コマンドの実行から pdorend コマンドの実行の間はオリジナル RD エリアもインメモリ化できません。
- インメモリ化している間は pdorbegin コマンドを実行できません。

4.5　系切り替え機能を使用しているときに気をつけること

インメモリ RD エリアがある場合に系切り替えが発生すると，インメモリデータバッファがなくなってしまうため，更新情報が RD エリアに反映されません。これは，計画系切り替えや，スローダウンによる系切り替えの場合も同じです。そのため，計画系切り替えは，インメモリ化を解除してから実行してください。

なお，インメモリ RD エリアがある場合に系切り替えが発生したとき，またはインメモリ化を解除しないで計画系切り替えを実行したとき，インメモリ化していた RD エリアは障害閉塞状態になります。この場合は pdrels コマンドで閉塞を解除してから，再度インメモリ化してください。

注意事項

- 系が切り替わると，インメモリデータバッファのデータがなくなり，更新情報が RD エリアに反映されません。また，インメモリ化していた RD エリアは障害閉塞状態になります。
- 1：1 スタンバイレス型系切り替え機能を使用している場合，代替 BES 下の RD エリアのオープン契機は常に SCHEDULE 属性になります。この SCHEDULE 属性を変えることができないため，代替 BES 下の RD エリアをインメモリ化することはできません。

4.6　その他の注意事項

4.6.1　その他の注意事項の説明

ここでは，その他の注意事項について説明します。

(1)　システム用 RD エリアを更新するような操作を行う場合

インメモリ RD エリアに対して，システム用 RD エリアを更新するような操作（例えば，CREATE TABLE や，ALTER TABLE など）を行わないでください。インメモリ RD エリアに対して，システム用 RD エリアを更新するような操作を行うと，HiRDB が異常終了するなどの障害が発生したときに，システム用 RD エリアと同期を合わせて回復する必要があるため，回復作業が難しくなります。

システム用 RD エリアを更新するような操作については，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。

(2)　セキュリティ監査機能を使用している場合

監査証跡表を格納している RD エリアをインメモリ化しないでください。インメモリ化している間に HiRDB が強制終了または異常終了した場合，インメモリデータバッファ上の更新情報が RD エリア（監査証跡表）に反映されません。

監査証跡表も，ほかの表と同様にデータベース回復ユティリティを使用すれば，更新情報を反映してインメモリ RD エリアを最新の状態に回復できますが，セキュリティ上の観点から，インメモリデータ処理の適用はお勧めしません。

また，監査証跡表の自動データロード機能は更新前ログ取得モードで動作するため，自動データロード機能を使用している場合は，監査証跡表に更新情報を反映できなくなるおそれがあります。

(3)　改竄防止表を定義している場合

改竄防止表を格納している RD エリアをインメモリ化しないでください。インメモリ化している間に HiRDB が強制終了または異常終了した場合，インメモリデータバッファ上の更新情報が RD エリア（改竄防止表）に反映されません。

改竄防止表も，ほかの表と同様にデータベース回復ユティリティを使用すれば，更新情報を反映してインメモリ RD エリアを最新の状態に回復できますが，セキュリティ上の観点から，インメモリデータ処理の適用はお勧めしません。

(4)　システム構成変更コマンドを実行する場合

インメモリ RD エリアがある場合は，システム構成変更コマンド（pdchgconf）を実行できません。システム構成変更コマンドを実行する場合は，インメモリ化を解除してから行ってください。

(5) 修正版 HiRDB の入れ替えを行う場合

インメモリ RD エリアがある場合は，修正版 HiRDB の入れ替え（pdprgrenew）を実行できません。修正版 HiRDB の入れ替えを行う場合は，インメモリ化を解除してから行ってください。

(6) RD エリアの自動増分機能を使用する場合

インメモリ RD エリアには，RD エリアの自動増分は適用されません。ただし，インメモリ化を解除すれば，自動増分が適用されます。そのため，インメモリ化を解除したあとに再度 RD エリアの容量不足が発生した場合は，自動的に RD エリアの容量が拡張されます。

(7) アンロードレスシステムログ運用を適用している場合

バックアップ対象 RD エリアにインメモリ RD エリアがある場合は，ログポイント情報ファイルを取得できません。

(8) ローカルバッファを割り当てた場合

インメモリ RD エリアにローカルバッファを割り当てた場合でも，その RD エリアに対するアクセスはインメモリデータバッファを介して行われます。ローカルバッファは使用されません。

(9) 更新可能バックアップ閉塞を実行する場合

インメモリ RD エリアに対しては，更新可能バックアップ閉塞（pdhold -b -u）を実行できません。

5 トラブルシュート

この章では，インメモリデータ処理中に障害が発生した場合の回復方法について説明します。

5.1　障害回復の基本的な考え方

5.1.1　障害の種類と回復方法を決めるときの基準

ここでは，インメモリデータ処理中に発生した障害を回復するときの，基本的な考え方について説明します。

注意事項

HiRDB が異常終了して再開始した場合，異常終了する前までの更新情報は RD エリアに反映されません（HiRDB が自動的に最新の状態に回復することはありません）。また，インメモリデータバッファはなくなり，インメモリ化が解除された状態になります。

(1)　障害の種類

インメモリデータ処理中に障害が発生した場合，インメモリデータバッファの状態と RD エリアの状態を確認する必要があります。

インメモリデータ処理中の障害には，次の 3 種類があります。

RD エリア障害

RD エリアに障害が発生している状態です。

RD エリアの障害を取り除いたあと，インメモリデータバッファ上のデータを RD エリアに書き込むことで回復できます。インメモリデータバッファ上のデータは最新であるため，RD エリアも最新の状態に回復できます。

バッファ障害

インメモリデータバッファに障害が発生している状態です。

最新の状態に回復する場合は，いったんインメモリ化を解除してから，アンロードログファイルを使って pdrstr コマンドで RD エリアを最新の状態に回復し，再度インメモリ化します。または，インメモリ化したままアンロードログファイルを使って pdrstr コマンドで RD エリアを最新の状態に回復し，その内容をインメモリデータバッファに再読み込みします。

同期取得時点に回復する場合は，いったんインメモリ化を解除してから，RD エリアを再度インメモリ化します。または，RD エリアの内容をインメモリデータバッファに再読み込みします。

RD エリア障害かつバッファ障害

インメモリデータバッファと RD エリアの両方に障害が発生している状態です。

最新の状態に回復する場合は，バックアップファイルとアンロードログファイルを使って pdrstr コマンドで RD エリアを最新の状態に回復し，その内容を再度インメモリ化します。

同期取得時点に回復する場合は，バックアップファイルを使って pdrstr コマンドで RD エリアを同期取得時点に回復し，その内容を再度インメモリ化します。

これらの障害状態については，pddbls -M コマンドで確認できます。詳細については，「障害が発生したときに最初に確認すること」を参照してください。

ポイント

バッファ障害で，かつ RD エリアが障害閉塞クローズ状態のときだけ，データベース回復ユティリティ（pdrstr）を実行できます。RD エリア障害が発生している場合（RD エリア障害かつバッファ障害を含む）は，一度インメモリ化を解除してから回復する必要があります。

(2) 回復方法を決めるときの基準

バッファ障害が発生している場合，回復方法は次の条件によって異なります。

- どの時点に回復するか（最新の状態か，または同期取得時点か）
- 同期取得時点に回復する場合，関連 RD エリアがあるか

詳細については，図「回復手順の参照先を決めるフローチャート」を参照してください。

5.2　障害が発生したときに最初に確認すること

障害が発生したときは，まず pddbls -M コマンドを実行して障害発生個所を確認します。pddbls -M コマンドの実行結果を次の図に示します。

図 5-1　pddbls -M コマンドの実行結果

```
STATE OF RDAREA
   RDAREA              ID       STATUS                TYPE
                                OPNMODE               MEMORY-STATUS
     :                  :         :                    :
   RDDATA01            8        HOLD                  USER
                                INITIAL               Y(OBST-MEM)
   RDDATA02            9        OPEN                  USER
                                INITIAL               Y(OBST-DB)
   RDDATA03            10       CLOSE HOLD            USER
                                INITIAL               Y(OBST-ALL)
```

（凡例）

：RDエリアとインメモリデータバッファの状態

MEMORY-STATUS に表示される RD エリアとインメモリデータバッファの状態から障害発生個所を確認します。RD エリアとインメモリデータバッファの状態を次の表に示します。

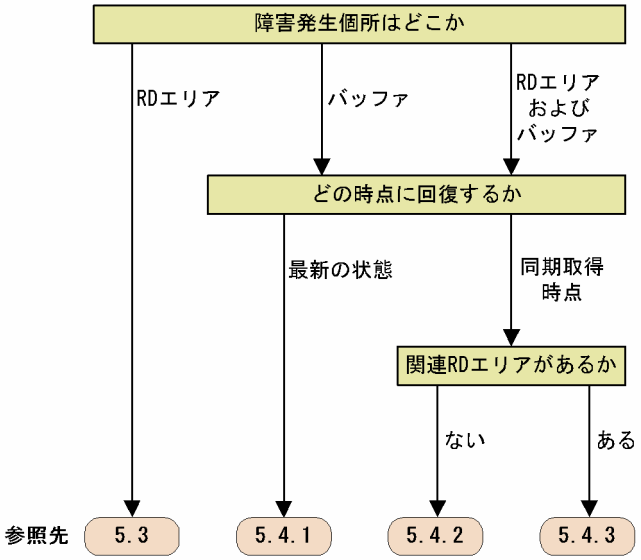
表 5-1　RD エリアとインメモリデータバッファの状態

MEMORY-STATUS の表示	状態	意味
Y(SYNC)	正常	DB 同期状態
Y(ASYNC)	正常	DB 非同期状態
Y(OBST-MEM)	障害発生	バッファ障害状態
Y(OBST-DB)	障害発生	RD エリア障害状態
Y(OBST-ALL)	障害発生	RD エリア障害状態かつバッファ障害状態
N	正常	インメモリ化していない

バッファ障害が発生している場合は，データベースをどの時点に回復するか（最新の状態か，または同期取得時点か）を決めます。また，関連 RD エリアがあるかどうかについても確認してください。回復手順の参照先を決めるフローチャートを次の図に示します。

5.3 以降では，それぞれの回復方法について説明しています。どのパターンに該当するか，次のフローチャートで確認してください。

図 5-2　回復手順の参照先を決めるフローチャート



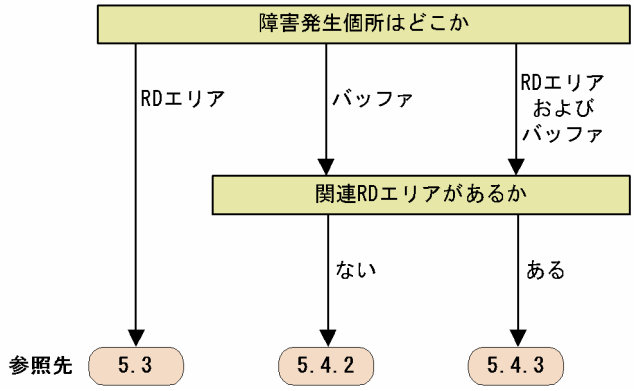
ポイント

- RD エリア障害だけの場合は，常に最新の状態に回復できます。
- 最新の状態に回復する場合は，関連 RD エリアの有無を意識する必要はありません。
- 同期取得時点とは，DB 同期状態のときを指します（インメモリ化した時点，または RD エリアとインメモリデータバッファの同期を取った時点）。

ログレスモードでバッチ業務を実行中に障害が発生した場合

ログレスモードでバッチ業務を実行中に障害が発生した場合は，次のフローチャートに従ってください。回復手順の参照先を決めるフローチャート（ログレスモードでバッチ業務を実行した場合）を次の図に示します。

図 5-3　回復手順の参照先を決めるフローチャート（ログレスモードでバッチ業務を実行した場合）



ポイント

- RD エリア障害だけの場合は，インメモリデータバッファ上に最新のデータがあるため，再度インメモリデータバッファとインメモリ RD エリアの同期を取ることで，常に最新の状態に回復できます。
- バッファ障害が発生している場合（RD エリア障害かつバッファ障害を含む）は，同期取得時点から業務を再実行する必要があります。
- 同期取得時点とは，DB 同期状態のときを指します（インメモリ化した時点，または RD エリアとインメモリデータバッファの同期を取った時点）。

5.3　RD エリア障害が発生した場合の回復手順

5.3.1　RD エリア障害が発生した場合の回復手順の例題

ここでは，RD エリア障害が発生した場合の回復手順について，例題を使って説明します。

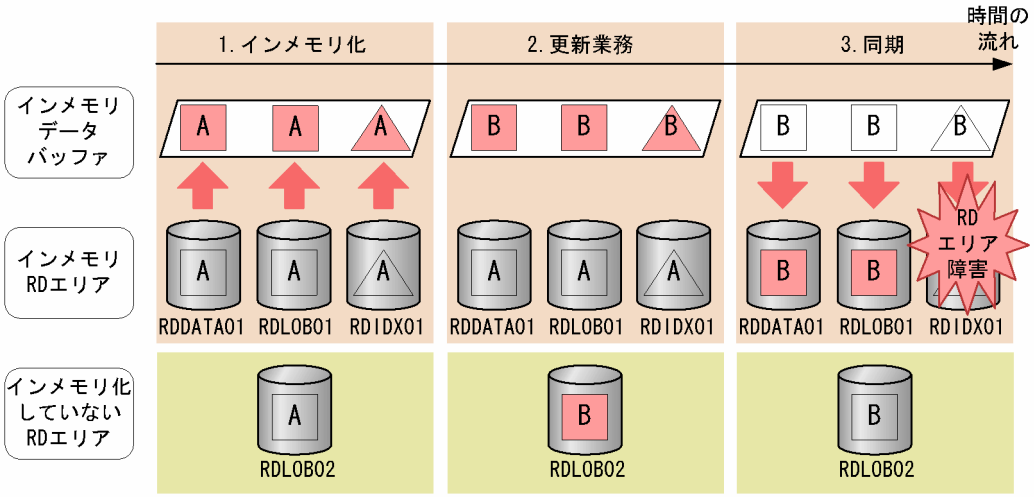
(1)　障害発生時の状況

RDDATA01（表格納 RD エリア），RDLOB01（LOB データ格納 RD エリア）および RDIDX01（インデクス格納 RD エリア）をインメモリ化しました。RDLOB02（LOB データ格納 RD エリア）はインメモリ化していません。

また，RDDATA01，RDLOB01 および RDIDX01 は異なる HiRDB ファイルシステム領域に作成されているとします。

障害発生時の状況を次の図に示します。

図 5-4　障害発生時の状況（RD エリア障害が発生した場合）



[説明]

1. RDDATA01，RDLOB01 および RDIDX01 をインメモリ化しました。
2. 更新業務を実行し，インメモリデータバッファと RDLOB02 のデータを更新しました。
3. pdhold -b コマンドでインメモリデータバッファと RD エリアの同期を取りました。このとき，RDDATA01 および RDLOB01 への書き込みは正常にできましたが，RDIDX01 が書き込みエラーによって RD エリア障害になりました。

このときのインメモリデータバッファと各 RD エリアの状態を次に示します。

- インメモリデータバッファの状態：正常（最新の状態）
- RDDATA01 の状態：正常（最新の状態）

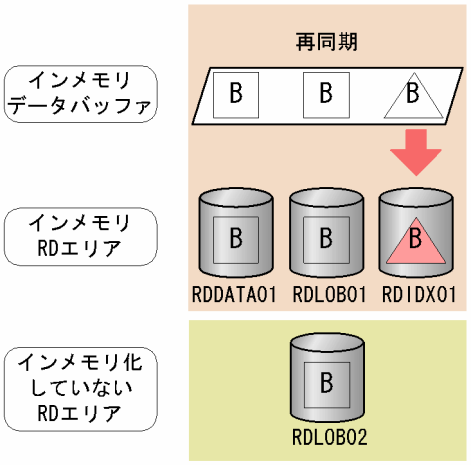
- RDLOB01 の状態：正常（最新の状態）
- RDIDX01 の状態：RD エリア障害
- RDLOB02 の状態：正常（最新の状態）

(2)　障害回復の手順

インメモリデータバッファ上に最新のデータがあるため，インメモリデータバッファと RDIDX01 の同期を再度取り，RDIDX01 を最新の状態に回復します。

障害回復の手順を次の図に示します。

図 5-5　障害回復の手順（RD エリア障害が発生した場合）



コマンドの実行手順を次に示します。

1. サーバプロセスをリフレッシュします。

```
pdpfresh -s BES1
```

ディスク交換を行う場合にこの操作を実行してください。この操作を実行しないとディスクを切り離すことができません。

2. RDDATA01 および RDLOB01 の閉塞を解除します。

```
pdrels -r RDDATA01,RDLOB01
```

pdhold -b コマンドを実行したため，RDDATA01 および RDLOB01 が参照可能バックアップ閉塞状態になっています。閉塞を解除して，RDDATA01 および RDLOB01 に対する更新業務を続行します。

3. RDIDX01 の障害を取り除きます。

ディスク障害が発生した場合は，マニュアル「HiRDB システム運用ガイド」の「ディスク障害が発生したときの対処方法」を参照して対処してください。

4. インメモリデータバッファと RDIDX01 の同期を取ります。

```
pdhold -r RDIDX01 -b
```

インメモリデータバッファ上のデータが RDIDX01 に書き込まれました。これによって，RDIDX01 は最新の状態に回復されました。

5. RDIDX01 の閉塞を解除します。

```
pdrels -r RDIDX01
```

参考

RD エリア障害の回復を行う場合は，バックアップファイルを使用しません。

(3) 障害回復時のインメモリデータバッファおよびインメモリ RD エリアの状態遷移

障害回復時，インメモリデータバッファの状態およびインメモリ RD エリアの状態は次の表のように遷移します。表の項番は，「障害回復の手順」の実行手順の番号と対応しています。なお，表中の実行コマンドは，一部オプションを省略しています。

表 5-2 障害回復時のインメモリデータバッファおよびインメモリ RD エリアの状態遷移

項番	実行コマンド	障害が発生した RD エリア		障害が発生していない RD エリア	
		RDIDX01		RDDATA01，RDLOB01	
		コマンド実行後のバッファの状態※1	コマンド実行後の RD エリアの状態※2	コマンド実行後のバッファの状態※1	コマンド実行後の RD エリアの状態※2
－	－	RD エリア障害状態	障害閉塞かつオープン状態	DB 同期状態	参照可能バックアップ閉塞状態
1	pdpfresh	↓	↓	↓	↓
2	pdrels -r RDDATA01,RDLOB01	↓	↓	↓	オープン状態
3	－	↓	↓	↓	↓
4	pdhold -r RDIDX01 -b	DB 同期状態	参照可能バックアップ閉塞状態	↓	↓
5	pdrels -r RDIDX01	↓	オープン状態	↓	↓

（凡例）

↓：状態の変化なし

－：該当しない

注※1 バッファの状態とは，インメモリデータバッファの状態のことです。

注※2 RD エリアの状態とは，インメモリ RD エリアの状態のことです。

5.4　バッファ障害が発生した場合の回復手順

ここでは，バッファ障害が発生した場合の回復手順について，例題を使って説明します。また，インメモリデータバッファと RD エリアの両方に障害が発生している場合も，基本的な回復手順は同じであるため，ここで一緒に説明します。

なお，ここで説明する回復手順は，インメモリ化するときにバックアップを取得していることを前提としています。バックアップを取得するタイミングについては，「バックアップを取得するタイミング」を参照してください。

5.4.1　最新の状態に回復する場合

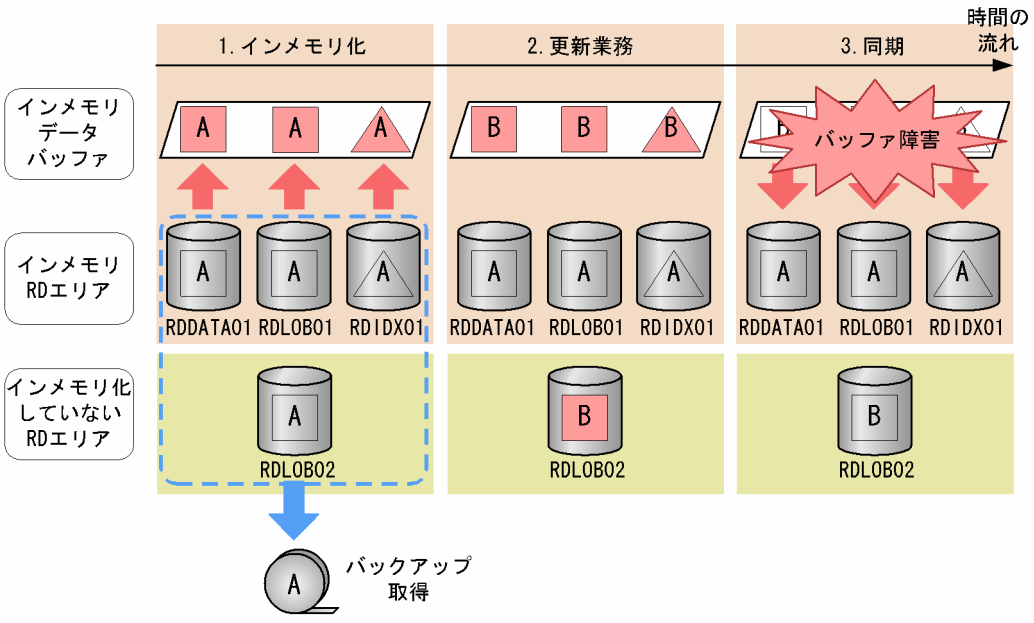
最新の状態に回復する場合について説明します。

(1)　障害発生時の状況

RDDATA01（表格納 RD エリア），RDLOB01（LOB データ格納 RD エリア）および RDIDX01（インデクス格納 RD エリア）をインメモリ化しました。RDLOB02（LOB データ格納 RD エリア）はインメモリ化していません。

障害発生時の状況を次の図に示します。

図 5-6　障害発生時の状況（バッファ障害を最新の状態に回復する場合）



[説明]

1. RDDATA01，RDLOB01 および RDIDX01 をインメモリ化しました。また，関連する RD エリア RDLOB02 と一緒にバックアップを取得しました。

2. 更新業務を実行し，インメモリデータバッファと RDLOB02 のデータを更新しました。
3. pdhold -b コマンドでインメモリデータバッファと RD エリアの同期を取りました。このとき，バッファ障害になりました。

このときのインメモリデータバッファと各 RD エリアの状態を次に示します。

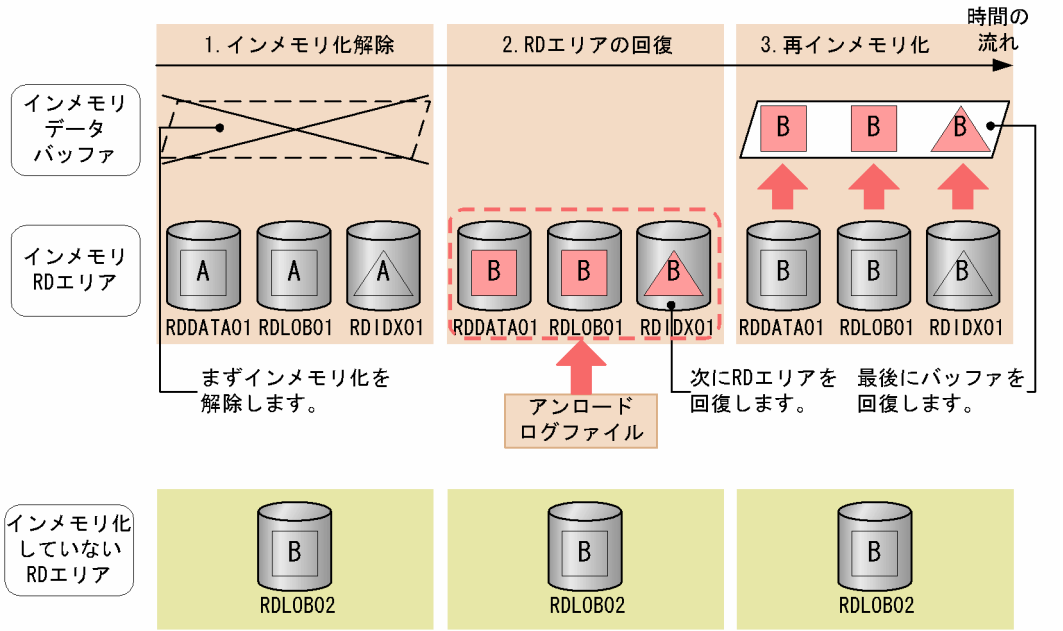
- インメモリデータバッファの状態：バッファ障害
- RDDATA01 の状態：障害閉塞状態（同期取得時（インメモリ化時）の状態）
- RDLOB01 の状態：障害閉塞状態（同期取得時（インメモリ化時）の状態）
- RDIDX01 の状態：障害閉塞状態（同期取得時（インメモリ化時）の状態）
- RDLOB02 の状態：正常（最新の状態）

(2) 障害回復の手順

pdrstr コマンドで RDDATA01，RDLOB01 および RDIDX01 を最新の状態に回復します。そのあとに，RDDATA01，RDLOB01 および RDIDX01 を再度インメモリ化します。

障害回復の手順を次の図に示します。

図 5-7 障害回復の手順（バッファ障害を最新の状態に回復する場合）



[説明]

1. RDDATA01，RDLOB01 および RDIDX01 のインメモリ化を解除します。
2. アンロードログファイルを使って RDDATA01，RDLOB01 および RDIDX01 を最新の状態に回復します（RDDATA01，RDLOB01 および RDIDX01 のデータはバックアップ取得時点と同じ状態であるため，回復時にバックアップファイルは使用しません）。

   ただし，RD エリア障害とバッファ障害の両方が発生している場合は，バックアップファイルとアンロードログファイルを使って RDDATA01，RDLOB01 および RDIDX01 を回復します。

3. RDDATA01 および RDIDX01 のデータを再度インメモリ化します。

コマンドの実行手順を次に示します。

1. RDDATA01，RDLOB01 および RDIDX01 をクローズ状態にします。

```
pdclose -r RDDATA01,RDLOB01,RDIDX01
```

インメモリ化を強制解除するには，RD エリアの状態が障害閉塞クローズ状態になっている必要があるため，ここで RDDATA01，RDLOB01 および RDIDX01 をクローズしています。

2. RDDATA01，RDLOB01 および RDIDX01 のインメモリ化を強制的に解除します。

```
pdmemdb -k rels -r RDDATA01,RDLOB01,RDIDX01 -d
```

3. 現用のシステムログファイルを調べます。

```
pdlogls -d sys
```

4. システムログファイルをスワップします。

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

5. 現用だったファイルの内容をアンロードします。

```
pdlogunld -d sys -g log01 -o C:¥usr¥hirdb¥pdlogunld¥unldlog01
```

6. RDDATA01，RDLOB01 および RDIDX01 をアンロードログファイルから最新の状態に回復します。

```
pdrstr -m C:¥rdarea¥mast¥mast01 -d C:¥usr¥hirdb¥pdlogunld
-p C:¥usr¥hirdb¥pdrstr¥list¥list01
-w C:¥tmp¥sortwork -r RDDATA01,RDLOB01,RDIDX01
```

7. RDDATA01，RDLOB01 および RDIDX01 の障害閉塞クローズを解除します。

```
pdrels -r RDDATA01,RDLOB01,RDIDX01 -o
```

再度インメモリ化するには，RD エリアの状態がコマンド閉塞クローズ状態になっている必要があるため，ここでいったん RDDATA01，RDLOB01 および RDIDX01 の障害閉塞クローズを解除しています。

8. RDDATA01，RDLOB01 および RDIDX01 をコマンド閉塞クローズ状態にします。

```
pdhold -r RDDATA01,RDLOB01,RDIDX01 -c
```

9. RDDATA01，RDLOB01 および RDIDX01 を再度インメモリ化します。

```
pdmemdb -k stay -r RDDATA01,RDLOB01,RDIDX01
```

これによって，インメモリデータバッファは最新の状態に回復されました。

10. RDDATA01，RDLOB01 および RDIDX01 の閉塞クローズを解除します。

```
pdrels -r RDDATA01,RDLOB01,RDIDX01 -o
```

これによって，業務が再開できるようになりました。

RD エリア障害かつバッファ障害の場合

RD エリア障害とバッファ障害の両方が発生している場合は，手順 6.でバックアップファイルとアンロードログファイルから RD エリアを最新の状態に回復してください。それ以外の手順については，バッファ障害の場合と同じです。

(3) 障害回復時のインメモリデータバッファおよびインメモリ RD エリアの状態遷移

障害回復時，インメモリデータバッファの状態およびインメモリ RD エリアの状態は次の表のように遷移します。表の項番は，「障害回復の手順」の実行手順の番号と対応しています。なお，表中の実行コマンドは，一部オプションを省略しています。

表 5-3　障害回復時のインメモリデータバッファおよびインメモリ RD エリアの状態遷移

項番	実行コマンド	コマンド実行後のバッファの状態※1	コマンド実行後の RD エリアの状態※2
			RDDATA01，RDLOB01，RDIDX01
－	－	バッファ障害状態	障害閉塞かつオープン状態
1	pdclose -r RDDATA01,RDLOB01,RDIDX01	↓	障害閉塞かつクローズ状態
2	pdmemdb -k rels -r RDDATA01,RDLOB01,RDIDX01 -d	未使用状態	↓
3	pdlogls -d sys	↓	↓
4	pdlogswap -d sys -w	↓	↓
5	pdlogunld -d sys	↓	↓
6	pdrstr -r RDDATA01,RDLOB01,RDIDX01	↓	↓
7	pdrels -r RDDATA01,RDLOB01,RDIDX01 -o	↓	オープン状態
8	pdhold -r RDDATA01,RDLOB01,RDIDX01 -c	↓	コマンド閉塞かつクローズ状態
9	pdmemdb -k stay -r RDDATA01,RDLOB01,RDIDX01	DB 同期状態	↓
10	pdrels -r RDDATA01,RDLOB01,RDIDX01 -o	↓	オープン状態

（凡例）

↓：状態の変化なし

－：該当しない

注※1　バッファの状態とは，インメモリデータバッファの状態のことです。

注※2　RD エリアの状態とは，インメモリ RD エリアの状態のことです。

5.4.2　同期取得時点に回復する場合（関連 RD エリアなし）

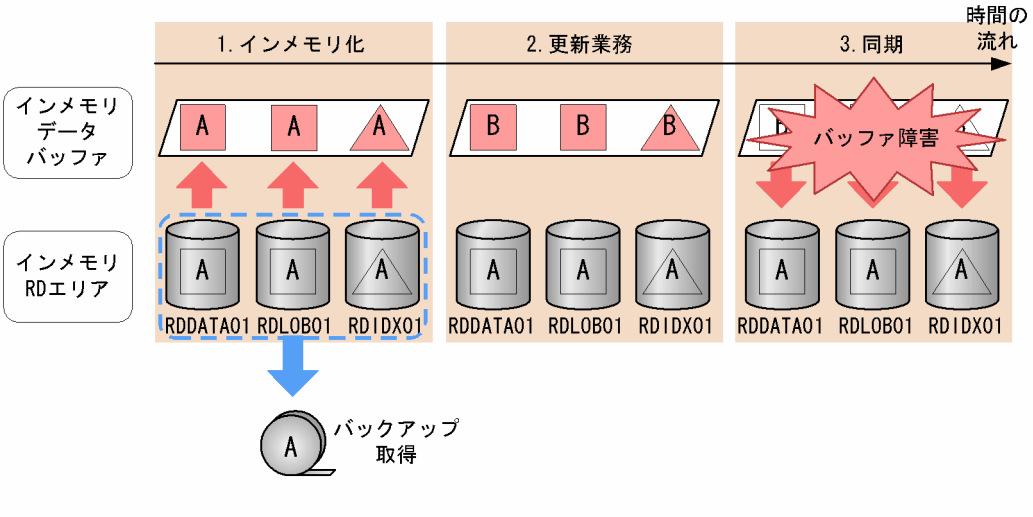
同期取得時点に回復する場合について説明します。

(1)　障害発生時の状況

RDDATA01（表格納 RD エリア），RDLOB01（LOB データ格納 RD エリア）および RDIDX01（インデクス格納 RD エリア）をインメモリ化しました。

障害発生時の状況を次の図に示します。

図 5-8　障害発生時の状況（バッファ障害を同期取得時点に回復する場合（関連 RD エリアなし））



[説明]

1. RDDATA01，RDLOB01 および RDIDX01 をインメモリ化しました。また，このとき，バックアップを取得しました。
2. 更新業務を実行し，インメモリデータバッファ上のデータを更新しました。
3. pdhold -b コマンドでインメモリデータバッファと RD エリアの同期を取ろうとしましたが，バッファ障害によって，RDDATA01，RDLOB01 および RDIDX01 との同期が取れませんでした。

このときのインメモリデータバッファと各 RD エリアの状態を次に示します。

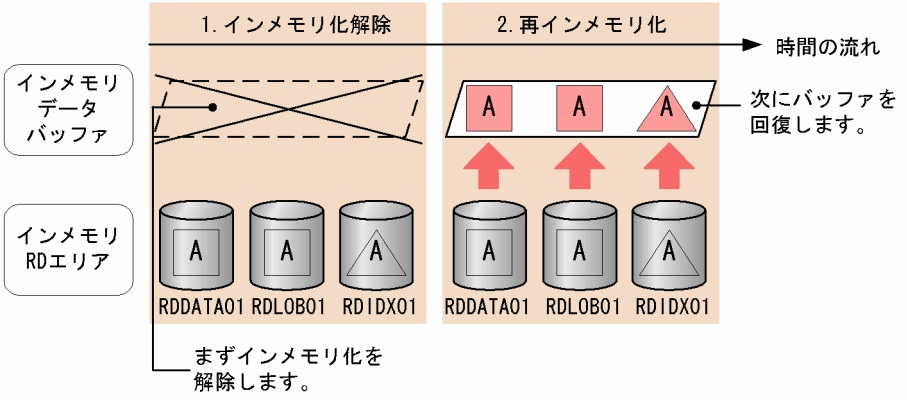
- インメモリデータバッファの状態：バッファ障害
- RDDATA01 の状態：障害閉塞状態（同期取得時（インメモリ化時）の状態）
- RDLOB01 の状態：障害閉塞状態（同期取得時（インメモリ化時）の状態）
- RDIDX01 の状態：障害閉塞状態（同期取得時（インメモリ化時）の状態）

(2)　障害回復の手順

RDDATA01，RDLOB01 および RDIDX01 中のデータは同期取得時（インメモリ化時）のままです。RDDATA01，RDLOB01 および RDIDX01 のデータを再度インメモリ化してインメモリデータバッファを同期取得時点に回復します。

障害回復の手順を次の図に示します。

図 5-9　障害回復の手順（バッファ障害を同期取得時点に回復する場合（関連 RD エリアなし））



［説明］

1. RDDATA01，RDLOB01 および RDIDX01 のインメモリ化を解除します。
2. RDDATA01，RDLOB01 および RDIDX01 を再度インメモリ化します。
   ただし，RD エリア障害とバッファ障害の両方が発生している場合は，バックアップファイルを使って RDDATA01，RDLOB01 および RDIDX01 を回復したあとにインメモリ化します。

コマンドの実行手順を次に示します。

1. RDDATA01，RDLOB01 および RDIDX01 をクローズ状態にします。

```
pdclose -r RDDATA01,RDLOB01,RDIDX01
```

インメモリ化を強制解除するには，RD エリアの状態が障害閉塞クローズ状態になっている必要があるため，ここで RDDATA01，RDLOB01 および RDIDX01 をクローズしています。

2. RDDATA01，RDLOB01 および RDIDX01 のインメモリ化を強制的に解除します。

```
pdmemdb -k rels -r RDDATA01,RDLOB01,RDIDX01 -d
```

3. RDDATA01，RDLOB01 および RDIDX01 の障害閉塞クローズを解除します。

```
pdrels -r RDDATA01,RDLOB01,RDIDX01 -o
```

再度インメモリ化するには，RD エリアの状態がコマンド閉塞クローズ状態になっている必要があるため，ここでいったん RDDATA01，RDLOB01 および RDIDX01 の障害閉塞クローズを解除しています。

4. RDDATA01，RDLOB01 および RDIDX01 をコマンド閉塞クローズ状態にします。

```
pdhold -r RDDATA01,RDLOB01,RDIDX01 -c
```

5. RDDATA01，RDLOB01 および RDIDX01 を再度インメモリ化します。

```
pdmemdb -k stay -r RDDATA01,RDLOB01,RDIDX01
```

これによって，インメモリデータバッファは同期取得時点の状態に回復されました。

6. RDDATA01，RDLOB01 および RDIDX01 の閉塞クローズを解除します。

```
pdrels -r RDDATA01,RDLOB01,RDIDX01 -o
```

これによって，業務が再開できるようになりました。

RD エリア障害かつバッファ障害の場合

RD エリア障害とバッファ障害の両方が発生している場合は，手順 2.でインメモリ化を解除したあと，バックアップファイルから RD エリアを同期取得時点に回復してください。それ以外の手順については，バッファ障害の場合と同じです。

(3) 障害回復時のインメモリデータバッファおよびインメモリ RD エリアの状態遷移

障害回復時，インメモリデータバッファの状態およびインメモリ RD エリアの状態は次の表のように遷移します。表の項番は，「障害回復の手順」の実行手順の番号と対応しています。なお，表中の実行コマンドは，一部オプションを省略しています。

表 5-4 障害回復時のインメモリデータバッファおよびインメモリ RD エリアの状態遷移

項番	実行コマンド	コマンド実行後のバッファの状態※1	コマンド実行後の RD エリアの状態※2
			RDDATA01，RDLOB01，RDIDX01
－	－	バッファ障害状態	障害閉塞かつオープン状態
1	pdclose -r RDDATA01,RDLOB01,RDIDX01	↓	障害閉塞かつクローズ状態
2	pdmemdb -k rels -r RDDATA01,RDLOB01,RDIDX01 -d	未使用状態	↓
3	pdrels -r RDDATA01,RDLOB01,RDIDX01 -o	↓	オープン状態
4	pdhold -r RDDATA01,RDLOB01,RDIDX01 -c	↓	コマンド閉塞かつクローズ状態
5	pdmemdb -k stay -r RDDATA01,RDLOB01,RDIDX01	DB 同期状態	↓
6	pdrels -r RDDATA01,RDLOB01,RDIDX01 -o	↓	オープン状態

（凡例）

↓：状態の変化なし

－：該当しない

注※1　バッファの状態とは，インメモリデータバッファの状態のことです。

注※2　RD エリアの状態とは，インメモリ RD エリアの状態のことです。

5.4.3　同期取得時点に回復する場合（関連 RD エリアあり）

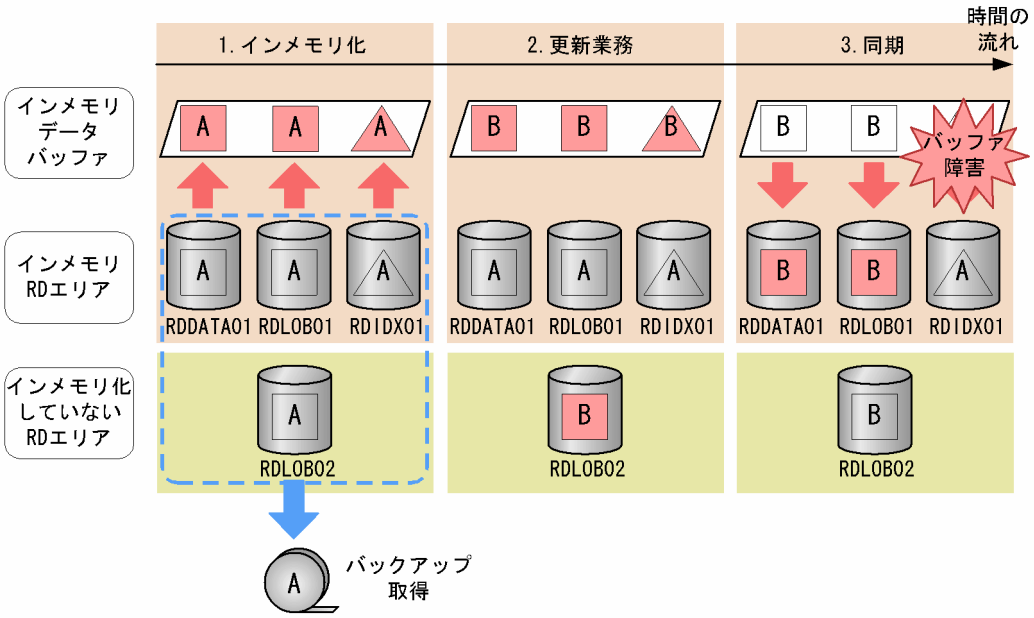
同期取得時点に回復する場合について説明します。関連する RD エリアがあるため，それらも一緒に同期取得時点に戻します。

(1)　障害発生時の状況

RDDATA01（表格納 RD エリア），RDLOB01（LOB データ格納 RD エリア）および RDIDX01（インデクス格納 RD エリア）をインメモリ化しました。RDLOB02（LOB データ格納 RD エリア）はインメモリ化していません。

障害発生時の状況を次の図に示します。

図 5-10　障害発生時の状況（バッファ障害を同期取得時点に回復する場合（関連 RD エリアあり））



[説明]

1. RDDATA01，RDLOB01 および RDIDX01 をインメモリ化しました。また，このとき，関連する RD エリア RDLOB02 も一緒にバックアップを取得しました。
2. 更新業務を実行し，インメモリデータバッファと RDLOB02 のデータを更新しました。

3. pdhold -b コマンドでインメモリデータバッファと RD エリアの同期を取りました。このとき，RDDATA01 および RDLOB01 への書き込みは正常にできましたが，RDIDX01 への書き込み中にバッファ障害になりました。

このときのインメモリデータバッファと各 RD エリアの状態を次に示します。

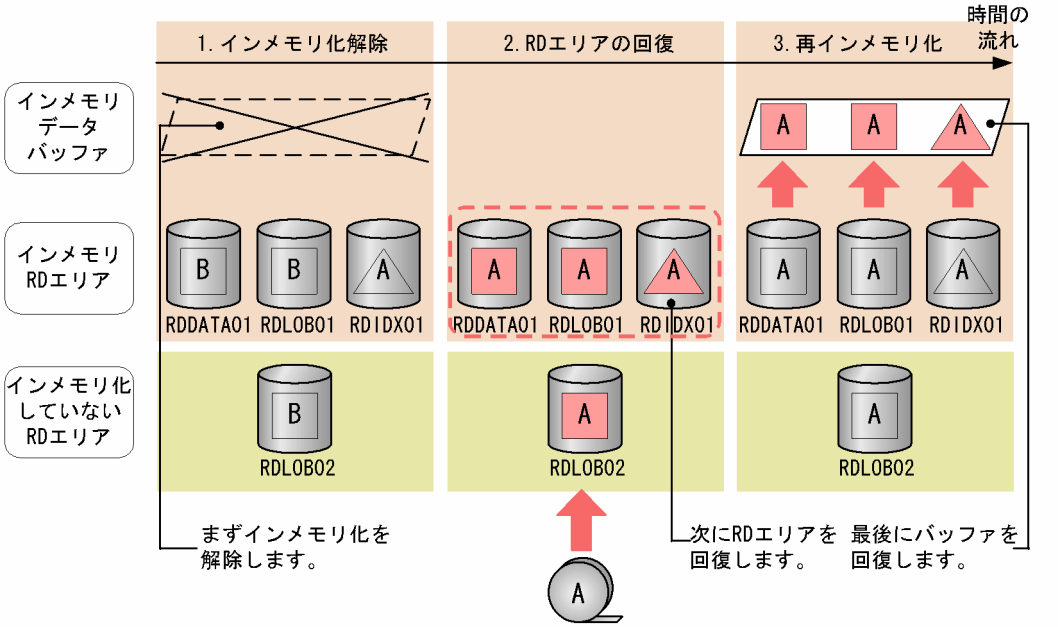
- インメモリデータバッファの状態：バッファ障害
- RDDATA01 の状態：正常（最新の状態）
- RDLOB01 の状態：正常（最新の状態）
- RDIDX01 の状態：障害閉塞状態（同期取得時（インメモリ化時）の状態）
- RDLOB02 の状態：正常（最新の状態）

(2) 障害回復の手順

pdrstr コマンドで RDDATA01，RDLOB01，RDIDX01 および RDLOB02 を同期取得時点（インメモリ化時点）に回復します。そのあとに，RDDATA01，RDLOB01 および RDIDX01 のデータをインメモリデータバッファに再度インメモリ化します。

障害回復の手順を次の図に示します。

図 5-11 障害回復の手順（バッファ障害を同期取得時点に回復する場合（関連 RD エリアあり））



[説明]

1. RDDATA01，RDLOB01 および RDIDX01 のインメモリ化を解除します。
2. バックアップファイルを使って RDDATA01，RDLOB01，RDIDX01 および RDLOB02 を同期取得時点に回復します。
3. RDDATA01，RDLOB01 および RDIDX01 を再度インメモリ化します。

コマンドの実行手順を次に示します。

1. RDDATA01 および RDLOB01 の閉塞を解除します。

```
pdrels -r RDDATA01,RDLOB01
```

pdhold -b コマンドを実行したため，RDDATA01 および RDLOB01 が参照可能バックアップ閉塞状態になっています。インメモリ化を解除するにはコマンド閉塞クローズ状態になっている必要があるため，いったん参照可能バックアップ閉塞を解除しています。

2. RDDATA01，RDLOB01 および RDLOB02 をコマンド閉塞クローズ状態にします。

```
pdhold -r RDDATA01,RDLOB01,RDLOB02 -c
```

3. RDDATA01 および RDLOB01 のインメモリ化を解除します。

```
pdmemdb -k rels -r RDDATA01,RDLOB01
```

4. RDIDX01 をクローズ状態にします。

```
pdclose -r RDIDX01
```

インメモリ化を強制解除するには，RD エリアの状態が障害閉塞クローズ状態になっている必要があるため，ここで RDIDX01 をクローズしています。

5. RDIDX01 のインメモリ化を強制的に解除します。

```
pdmemdb -k rels -r RDIDX01 -d
```

6. RDDATA01，RDLOB01，RDIDX01，および RDLOB02 をバックアップファイルから同期取得時点に回復します。

```
pdrstr -m C:¥rdarea¥mast¥mast01 -b C:¥usr¥hirdb¥pdcopy¥backup01
-p C:¥usr¥hirdb¥pdrstr¥list¥list01 -r RDDATA01,RDLOB01,RDIDX01,RDLOB02
```

7. RDIDX01 の障害閉塞クローズを解除します。

```
pdrels -r RDIDX01 -o
```

再度インメモリ化するには，RD エリアの状態がコマンド閉塞クローズ状態になっている必要があるため，ここでいったん RDIDX01 の障害閉塞クローズを解除しています。

8. RDIDX01 をコマンド閉塞クローズ状態にします。

```
pdhold -r RDIDX01 -c
```

9. RDDATA01，RDLOB01 および RDIDX01 を再度インメモリ化します。

```
pdmemdb -k stay -r RDDATA01,RDLOB01,RDIDX01
```

これによって，インメモリデータバッファは同期取得時点の状態に回復されました。

10. RDDATA01，RDLOB01，RDIDX01，および RDLOB02 の閉塞クローズを解除します。

```
pdrels -r RDDATA01,RDLOB01,RDIDX01,RDLOB02 -o
```

これによって，業務が再開できるようになりました。

(3) 障害回復時のインメモリデータバッファ，インメモリ RD エリアおよびインメモリ化していない RD エリアの状態遷移

障害回復時，障害回復時のインメモリデータバッファ，インメモリ RD エリアおよびインメモリ化していない RD エリアの状態は次の表のように遷移します。表の項番は，「障害回復の手順」の実行手順の番号と対応しています。なお，表中の実行コマンドは，一部オプションを省略しています。

表 5-5 障害回復時のインメモリデータバッファ，インメモリ RD エリアおよびインメモリ化していない RD エリアの状態遷移

項番	実行コマンド	正常に書き込みできた RD エリア		正常に書き込みできなかった RD エリア		インメモリ化していない RD エリア
		RDDATA01，RDLOB01		RDIDX01		RDLOB02
		コマンド実行後のバッファの状態※1	コマンド実行後の RD エリアの状態※2	コマンド実行後のバッファの状態※1	コマンド実行後の RD エリアの状態※2	コマンド実行後の RD エリアの状態
−	−	DB 同期状態	参照可能バックアップ閉塞状態	バッファ障害状態	障害閉塞かつオープン状態	オープン状態
1	pdrels -r RDDATA01,RDLOB01	↓	オープン状態	↓	↓	↓
2	pdhold -r RDDATA01,RDLOB01,RDLOB02 -c	↓	コマンド閉塞かつクローズ状態	↓	↓	コマンド閉塞かつクローズ状態
3	pdmemdb -k rels -r RDDATA01,RDLOB01	未使用状態	↓	↓	↓	↓
4	pdclose -r RDIDX01	↓	↓	↓	障害閉塞かつクローズ状態	↓
5	pdmemdb -k rels -r RDIDX01 -d	↓	↓	未使用状態	↓	↓
6	pdrstr -r RDDATA01,RDLOB01,RDIDX01,RDLOB02	↓	↓	↓	↓	↓
7	pdrels -r RDIDX01 -o	↓	↓	↓	オープン状態	↓

項番	実行コマンド	正常に書き込みできたRDエリア		正常に書き込みできなかったRDエリア		インメモリ化していないRDエリア
		RDDATA01，RDLOB01		RDIDX01		RDLOB02
		コマンド実行後のバッファの状態※1	コマンド実行後のRDエリアの状態※2	コマンド実行後のバッファの状態※1	コマンド実行後のRDエリアの状態※2	コマンド実行後のRDエリアの状態
8	pdhold -r RDIDX01 -c	↓	↓	↓	コマンド閉塞かつクローズ状態	↓
9	pdmemdb -k stay -r RDDATA01,RDLOB01,RDIDX01	DB同期状態	↓	DB同期状態	↓	↓
10	pdrels -r RDDATA01,RDLOB01,RDIDX01,RDLOB02 -o	↓	オープン状態	↓	オープン状態	オープン状態

（凡例）

↓：状態の変化なし

－：該当しない

注※1　バッファの状態とは，インメモリデータバッファの状態のことです。

注※2　RDエリアの状態とは，インメモリRDエリアの状態のことです。

6 こんなときどうする

この章では，インメモリデータ処理を実行する場合に知っておくとよい操作について説明します。

6.1　インメモリデータバッファと RD エリアの同期を取るには

ここでは，インメモリデータバッファと RD エリアの同期を取る方法，およびそのタイミングについて説明します。

6.1.1　同期を取る方法

インメモリデータバッファと RD エリアの同期を取る場合，ほかのトランザクションが対象 RD エリアを参照できるように，通常は pdhold -b コマンドを使います。

また，pdhold -b 以外のコマンドでも，インメモリデータバッファと RD エリアの同期が取れます。インメモリデータバッファと RD エリアの同期を取るコマンドを次に示します。

- pdhold -b
- pdhold -b -w
- pdhold -c
- pdclose

なお，どのコマンドを使用しても，同期の取得に掛かる処理時間は同じです。

注意事項

- 同期取得中の更新トランザクションに対する処理は，使用するコマンドによって異なります。このことを考慮して同期取得に使用するコマンドを決定してください。または，同期を取るときは更新業務を停止してください。
- コマンド実行後は RD エリアの状態が変わります（バックアップ閉塞，または閉塞かつクローズ状態）。そのため，業務を続行する場合は，pdrels コマンドで閉塞を解除する必要があります。
- pdhold -b コマンドの場合，リスト用 RD エリアは指定できません。

6.1.2　同期を取るタイミング

バッファと RD エリアの同期を取るタイミングは，主に次の二つが考えられます。

- **バックアップを取得する前**
  バックアップを取得する前には，バッファと RD エリアの同期を取ることをお勧めします。
- **バッチ業務を実行したあと**
  バッチ業務を実行したあとには，バッチ実行後のインメモリデータバッファ障害に備えて，バッファと RD エリアの同期を取ることをお勧めします。

6.2　インメモリデータバッファの利用状況を確認するには

インメモリデータバッファの利用状況は，pddbls -M コマンドを実行することで確認できます。コマンドの実行例と実行結果を次に示します。

[コマンド実行例]

```
pddbls -r ALL -M
```

[実行結果]

```
STATE OF RDAREA
    RDAREA              ID      STATUS              TYPE
                                OPNMODE             MEMORY-STATUS
    RDMAST              1       OPEN                MAST
                                INITIAL             N
    RDDIR               2       OPEN                DDIR
                                INITIAL             N
    RDDICT              3       OPEN                DDIC
                                INITIAL             N
    RDDATA01            4       HOLD(BU I)          USER
                                INITIAL             Y(SYNC)
    RDIDX01             5       HOLD(BU I)          USER
                                INITIAL             Y(SYNC)
    RDLOB01             6       HOLD(BU I)          USER
                                INITIAL             N
    RDDATA02            7       OPEN                USER
                                INITIAL             Y(ASYNC)
    RDIDX02             8       OPEN                USER
                                INITIAL             Y(ASYNC)
    RDDATA03            9       OPEN                USER
                                INITIAL             N
    RDIDX03             10      OPEN                USER
                                INITIAL             N
```

（凡例）

：インメモリRDエリア

：RDエリアとインメモリデータバッファの状態

[説明]

インメモリデータバッファの利用状況は，MEMORY-STATUS の値で確認します。

Y：インメモリ化しています。

N：インメモリ化していません。

また，RD エリアとインメモリデータバッファの状態は，次の六つのどれかになります。

MEMORY-STATUS の表示	意味
Y(SYNC)	DB 同期状態
Y(ASYNC)	DB 非同期状態
Y(OBST-MEM)	バッファ障害状態
Y(OBST-DB)	RD エリア障害状態
Y(OBST-ALL)	RD エリア障害状態かつバッファ障害状態
N	インメモリ化していない

6.3.1　確認方法

(1)　インメモリデータバッファの名称を調べる

インメモリデータバッファの統計情報を確認するには，まず，HiRDB が決定したインメモリデータバッファの名称を調べます。

インメモリデータバッファの名称は，pdbufls コマンドの実行結果で確認できます。実行結果にインメモリデータバッファの名称を表示させるには，-M オプションを指定します。コマンドの実行例と実行結果を次に示します。

[コマンド実行例]

```
pdbufls -k def -M
```

[実行結果]

```
DEFINE OF GLOBAL BUFFER
 EDIT TIME 2007-10-18 16:36:23
 BUFFNAME         SVID      TYPE  SIZE   NUM     WRATIO  RDAREA/INDEX NAME
                            PRMAX PRNUM  CSIZE   MAPS
 gbufsys          DB01      R      4k        20   20      "RDMAST"
                                                          "RDDIR"
                                                          "RDDICT"
                               0  32   *******   **
 gbufdata01       DB01      R      4k       256   20      "RDDATA01"
                               0  32   *******   **
 gbufidx01        DB01      R      4k      4925   20      "RDIDX01"
                               0  32   *******   **
 gbuflob01        DB01      R      4k      4925   20      "RDLOB01"
                               0  32   *******   **
 _pdidbuf0000001  DB01      M      4k       256   20      "RDDATA01"
                               0  32   *******   **
 _pdidbuf0000002  DB01      M      4k      4925   20      "RDIDX01"
                               0  32   *******   **
```

「M」は，インメモリデータバッファであることを示します。

インメモリデータバッファの名称が表示されます。

(2)　インメモリデータバッファの統計情報を確認する

(1)で調べたバッファの名称を pdbufls コマンドに指定して，インメモリデータバッファの統計情報を確認できます。

インメモリデータバッファの統計情報は，pdbufls コマンドに-M オプションを指定することで確認できます。コマンドの実行例を次に示します。

[コマンド実行例]

```
pdbufls -k sts -M -a _pdidbuf0000001
```

コマンドの実行結果については，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。

6.4　強制的にインメモリ化を解除するには

6.4.1　解除方法

(1)　前提条件

障害が発生した場合や，テスト環境の場合は，インメモリ化を強制的に解除することがあります。インメモリ化を強制的に解除する場合は，最初にインメモリデータバッファの状態を確認してください。インメモリデータバッファの状態に合わせて，RD エリアの状態を次の表に示すとおりにコマンドで変更してください。その後，(2)に示す方法でインメモリ化を解除します。

表 6-1　インメモリ化を強制的に解除するときの RD エリアの状態

インメモリデータバッファの状態	コマンド実行後の RD エリアの状態
DB 非同期状態	コマンド閉塞かつオープン状態
バッファ障害状態	障害閉塞かつクローズ状態
RD エリア障害状態	コマンド閉塞かつオープン状態
バッファ障害状態かつ RD エリア障害状態	障害閉塞かつクローズ状態

(2)　実行方法

強制的にインメモリ化を解除するには，pdmemdb -k rels -d コマンドを使用します。コマンドの実行例を次に示します。

[コマンド実行例]

```
pdmemdb -k rels -r RDDATA01 -d
```

これによって，インメモリデータバッファがなくなります。また，RD エリアは障害閉塞状態になります。なお，この場合の障害閉塞は，pdrels コマンドで解除できます。

注意事項

pdhold -s コマンドを実行した場合にも，RD エリアのインメモリ化が強制的に解除され，インメモリデータバッファがなくなります。このとき，RD エリアは同期化閉塞状態になります。

6.5　インメモリデータバッファ上のデータを破棄するには

6.5.1　インメモリデータバッファ上のデータを破棄する前提条件と実行方法

(1)　前提条件

インメモリデータバッファ上のデータを破棄するには，インメモリデータバッファが DB 非同期状態で，RD エリアがコマンド閉塞かつオープン状態である必要があります。

(2)　実行方法

インメモリデータバッファのデータを破棄するには，pdmemdb -k cancel コマンドを使用します。コマンドの実行例を次に示します。

[コマンド実行例]

```
pdmemdb -k cancel -r RDDATA01
```

これによって，インメモリデータバッファはバッファ障害状態に，RD エリアは障害閉塞状態になります。なお，この場合の障害閉塞は，pdmemdb -k reload コマンドを実行してインメモリデータバッファにデータを再読み込みすることで解除できます。

6.6 RD エリアのデータをインメモリデータバッファに再読み込みするには

6.6.1 RD エリアのデータをインメモリデータバッファに再読み込みする前提条件と実行方法

(1) 前提条件

RD エリアのデータをインメモリデータバッファに再読み込みするには，インメモリデータバッファがバッファ障害状態で，RD エリアがコマンド閉塞かつクローズ状態である必要があります。

(2) 実行方法

RD エリアのデータをインメモリデータバッファに再読み込みするには，pdmemdb -k reload コマンドを使用します。コマンドの実行例を次に示します。

[コマンド実行例]

```
pdmemdb -k reload -r RDDATA01
```

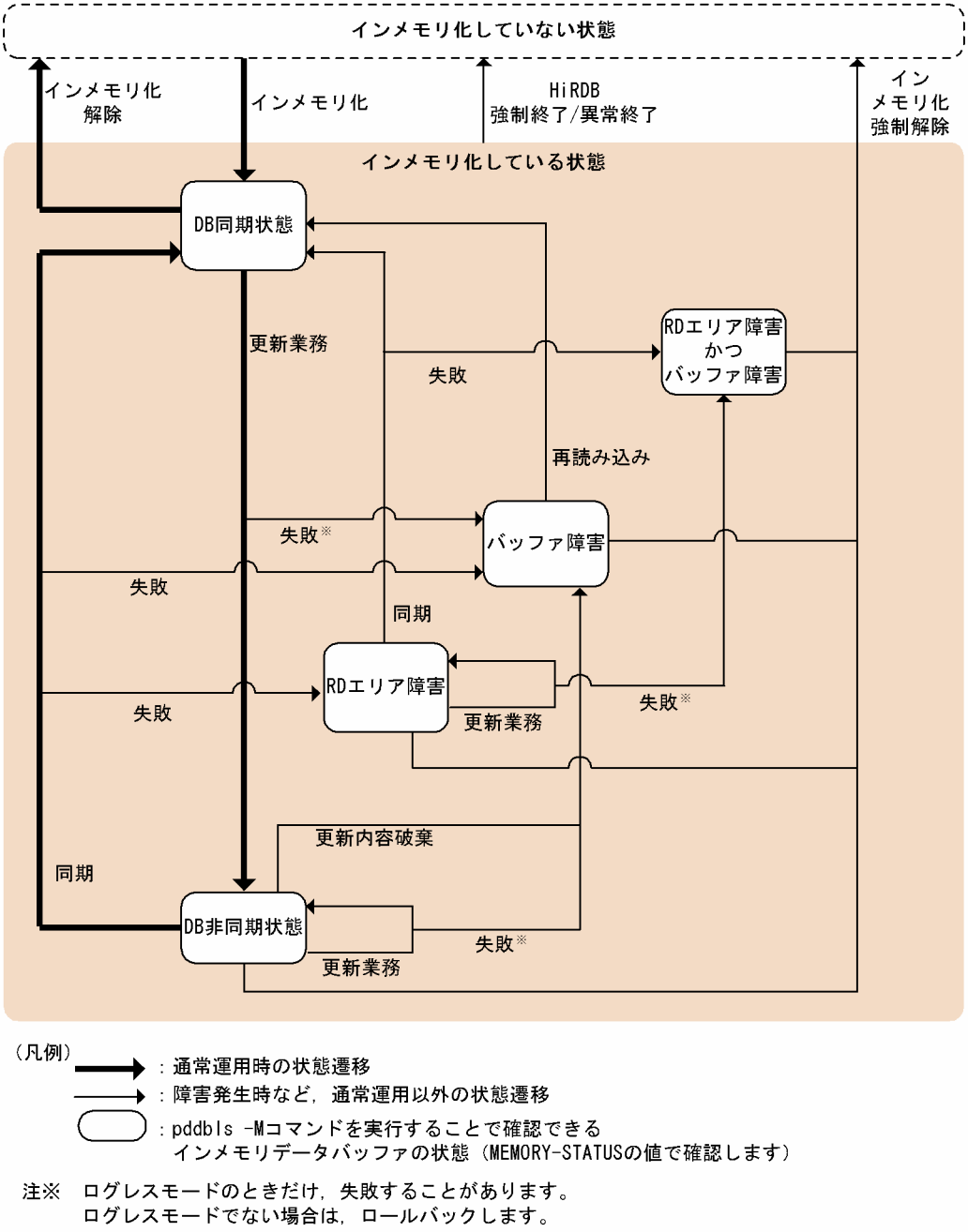
RD エリアのデータを再読み込みしたため，RD エリアとインメモリデータバッファは DB 同期状態になります。

付録

付録 A　インメモリデータ処理の状態遷移図

インメモリデータ処理の状態遷移を次の図に示します。インメモリデータ処理の実行中は，インメモリデータバッファの状態が次に示す図のように遷移します。障害発生時には，コマンドを使って，インメモリデータバッファの状態を変更し，正常な状態に戻す必要があります。そのため，インメモリデータバッファの状態遷移を正しく理解しておく必要があります（この図は状態遷移を正しく理解するためにお使いください)。

図 A-1　インメモリデータ処理の状態遷移



（凡例）
- ：通常運用時の状態遷移
- ：障害発生時など，通常運用以外の状態遷移
- ：pddbls -Mコマンドを実行することで確認できる
  インメモリデータバッファの状態（MEMORY-STATUSの値で確認します）

注※　ログレスモードのときだけ，失敗することがあります。
ログレスモードでない場合は，ロールバックします。

付録 B　インメモリデータ処理に関するコマンド一覧

インメモリデータ処理に関するコマンド一覧を次の表に示します。

表 B-1　インメモリデータ処理に関するコマンド一覧

コマンド	インメモリデータ処理での操作
pdmemdb -k stay	RD エリアをインメモリ化します。
pdmemdb -k rels	インメモリ化を解除します。
pdmemdb -k rels -d	インメモリ化を強制的に解除します。
pdmemdb -k cancel	インメモリデータバッファの更新内容を破棄します。
pdmemdb -k reload	RD エリアのデータをインメモリデータバッファに再読み込みします。
pdhold -b pdhold -b -w pdhold -c pdclose	RD エリアとインメモリデータバッファの同期を取ります。
pddbls -M	RD エリアとインメモリデータバッファの状態を表示します。
pddbls -D	インメモリ化した時刻，および RD エリアとインメモリデータバッファの同期を取った時刻を表示します。
pdbufls -k def -M	インメモリデータバッファの情報を表示します。
pdbufls -k sts -M	インメモリデータバッファの統計情報を表示します。

付録 C　pdmemdb コマンド実行時の前提条件

pdmemdb コマンドを実行する前に，インメモリデータバッファと対象 RD エリアの状態を確認してください。正しい状態になっていないと pdmemdb コマンドを実行できません。pdmemdb コマンド実行時の前提条件（インメモリデータバッファと対象 RD エリアの状態）を次の表に示します。

表 C-1　pdmemdb コマンド実行時の前提条件（インメモリデータバッファと対象 RD エリアの状態）

pdmemdb コマンドの-k オプションの指定		コマンド実行時の前提条件		コマンド実行後の状態	
		インメモリデータバッファの状態	RD エリアの状態	インメモリデータバッファの状態	RD エリアの状態
stay		未使用状態	コマンド閉塞かつクローズ状態※	DB 同期状態	コマンド閉塞かつクローズ状態
reload		バッファ障害状態	障害閉塞かつクローズ状態		
rels	-d オプション指定なし	DB 同期状態	コマンド閉塞かつクローズ状態	未使用状態（インメモリ化の解除）	コマンド閉塞かつクローズ状態
	-d オプション指定あり	DB 非同期状態	コマンド閉塞かつオープン状態		障害閉塞かつクローズ状態
		バッファ障害状態	障害閉塞かつクローズ状態		
		RD エリア障害状態	コマンド閉塞かつオープン状態		
		バッファ障害状態かつ RD エリア障害状態	障害閉塞かつクローズ状態		
cancel		DB 非同期状態	コマンド閉塞かつオープン状態	バッファ障害状態	障害閉塞かつオープン状態

注※

オンライン再編成閉塞中からコマンド閉塞かつクローズ状態に遷移した場合は，pdmemdb -k stay コマンドを実行できません。

索引