図2-8：相互系切り替えの例

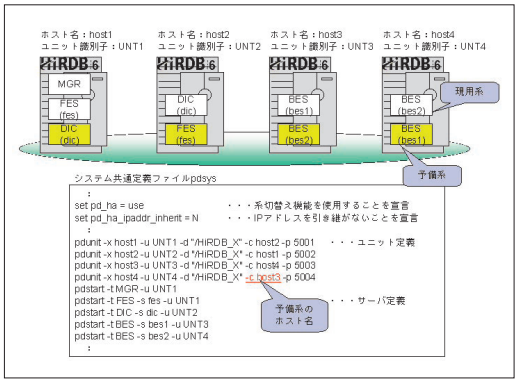
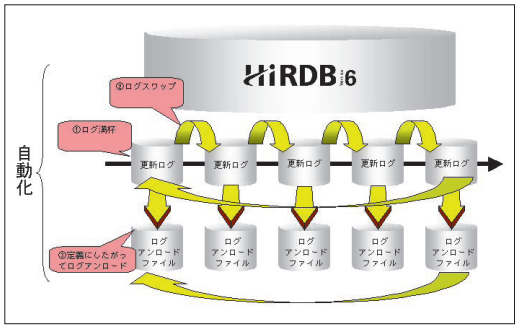


図2-9：システムログ自動アンロード



切り替えが発生したときに、実行系から待機系に情報を引き継ぐために使用される。

系切り替え機能を使用する場合、次のシステム構成が考えられる。

・1：1系切り替え構成
・2：1系切り替え構成
・相互系切り替え構成

障害検知と系切り替えはクラスタソフトウェアで行う。系切り替え後の回復処理はHiRDBで行う。HiRDBは各種クラスタソフトウェアに対応しているが、中でもHAモニタ（HP-UX、HI-UX/WE2）では高速な系切り替えを実現でき、AIXもサポート予定である。その他、対応しているクラスタソフトウェアは次の通りである。

・MC/ServiceGuard（HP-UX）
・Sun Cluster（Solaris）、Veritas Cluster Server（Solaris）
・Microsoft Cluster Server（Windows NT®、Windows® 2000）
・DNCWARE Cluster Perfect（Turbolinux）

なお、HACMP（AIX）とDNCWARE Cluster Perfect（Red Hat Linux）も近日サポートの予定だ。

■満杯となったシステムログを自動アンロード

HiRDB Version 6のシステムログ自動アンロードとは、満杯となったシステムログをHiRDBが自動的にアンロードし、使用可能なログファイルにスワップする機能である。

システムログ自動アンロード機能を使用するには、HiRDBの定義ファイルにシステムログのアンロード先となる領域を定義しておくだけでよい。あるシステムログファイルが満杯となり、別のシステムログファイルにスワップする、いわゆるアンロード待ち状態となった時点で、指定された領域へ自動的にアンロードを行う。

この際、従来のように統合システム運用管理「JP1」などでシステムを監視したり、監視用シェルを組み込んだりしなくても、システムログの再利用を自動化できる。これによりシステム管理者の負担を軽減できる。

■効率性の高い差分バックアップ

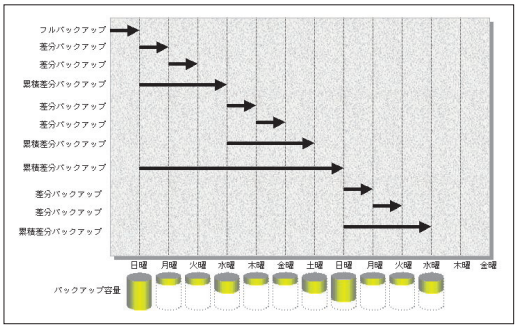
大規模なデータベースでは、データベース全体のデータ量に比して、日々のデータ更新量がさほどでもないというケースが多い。これに対し、大規模になればデータベース全体を毎日バックアップすることは時間的にも作業的にも非効率的であり、翌朝の通常の業務に間に合わなくなるといった事態にもつながりかねない。

差分バックアップ機能とは、このような場合でも最適なバックアップが取得できるよう、前回取得したバックアップから変更された箇所だけをバックアップする機能である。差分バックアップの方法としては、以下の2種類がある。

・前回バックアップした時点からのデータベースの更新差分のみバックアップする。
・前回の全体バックアップ以降のデータベース更新の累積差分をバックアップする。

差分バックアップによって、バックアップ時間を大幅に短縮できるため、従来は時間的制約により実行できな

図2-10：バックアップ運用スケジュールの例



かった大規模データベースのバックアップができるようになり、運用の自由度が高まる（デイリーバックアップなど）。

毎日取得している差分バックアップをマージし、累積差分バックアップを作成することもできる。例えば毎日同一のページを更新しているような場合、同一ページの更新情報をマージすれば、個々に差分バックアップファイルを作成するよりも、より容量の小さなバックアップファイルを作成することができる。これを累積差分バックアップという。

この累積差分バックアップファイルは、容量が小さくなることから回復にかかる時間を短縮することが可能となる。以下に、データベース静止化機能と組み合わせた差分バックアップ方式について説明する。説明を簡単にするために、バックアップ取得時、データベース静止化によりバックアップ参照閉塞モードにするものとする。

まず、差分バックアップ運用開始のため、フルバックアップを取得する。このときのバックアップ対象RDエリアのLSN（Log Sequence Number）を取得しておく。続いて、差分バックアップを取得するため、再度バックアップ参照閉塞状態にする。ここで差分バックアップを

図2-11：DB静止化と組み合わせた差分バックアップ

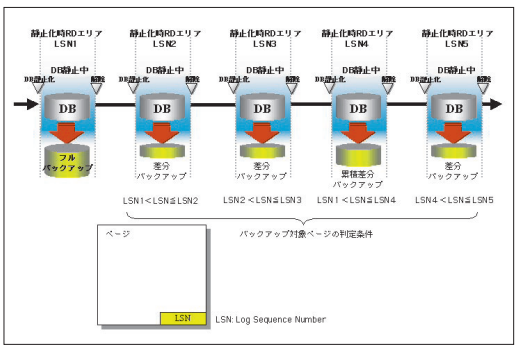
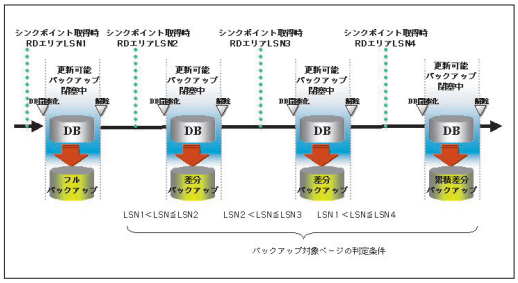


図2-12：更新可能な状態での差分バックアップ



取得する場合、前回のバックアップ取得時のLSNと今回のバックアップ参照閉塞時のLSNとを用い、バックアップ対象RDエリア中の使用中ページのLSNが範囲内のものをバックアップ対象とすることで差分バックアップが取得できる。もし、累積差分バックアップの場合は、前回取得したフルバックアップ時のLSNを用いればよい。

次に、更新可能な状態での差分バックアップ方式について説明する。

この場合、更新可能バックアップ閉塞状態にする。更新可能バックアップ閉塞状態で差分バックアップを取得する場合、直前のシンクポイント時のLSNを起点としたバックアップを取得する。累積差分バックアップの場合、前回取得したフルバックアップまたは累積差分バックアップ取得時のシンクポイント時のLSNから変更のあったページについてバックアップを取得する。

■可用性を高めるRDエリア容量の自動拡張

データ量の増加によりRDエリアの容量が不足した場合に、指定された大きさの容量で業務を停止せずに動的に拡張できる。自動拡張機能を使うには、データベース初期設定（pdinit）ユーティリティにおけるRDエリアの定義（create rdarea文）、データベース構成変更（pdmod）ユーティリティにおけるRDエリア変更（alter rdarea文）などで指定する。拡張する単位は、RDエリアを構成するセグメント数となる。以下は、既定義RDエリアに自動拡張指定をする場合の例である。

```
alter rdarea AREA1 extension use 100 segments;
```

これにより、データ量の増加が予想されるシステムや、連続運転が必要なシステムなどで、運用の負荷を減らすことができHiRDBの可用性をさらに高めることができる。また、RDエリア容量不足への対処方法としては、データベース構成変更（pdmod）ユーティリティによる

RDエリアの拡張（RDエリアを構成するファイルを最大16個まで拡張可能）を利用することもできる。

3 多様なビジネスアプリケーションの実現

3.1 最新のネットビジネスアプリケーションのための技術基盤

3.1.1 XMLによる柔軟なシステム間連携

■XML-RDBデータマッピングによる柔軟なデータ連携

従来HiRDBでは、XMLをHiRDB表のBLOB（Binary Large Object）型カラムに格納して全文検索や概念検索を行うというように、XMLを文書というコンテンツとして利用しやすくする取り組みをしてきた。

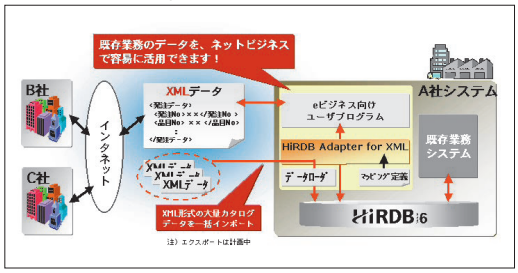
HiRDB Version 6では、さらにシステム間の柔軟なデータ連携を強化するために、新たに製品ラインナップに「HiRDB Adapter for XML」を加えた。HiRDB Adapter for XMLでは、XML文書のタグとHiRDB表のカラムとの間で双方向のデータ交換を行うためのアプリケーションプログラムの作成が可能となる。つまり、XML文書を解析してタグを分解し、HiRDB表のカラムに格納してリレーショナルデータとして操作したり、HiRDB表データをXML文書として取り出すことができるのである。

XMLは単なるHTMLの拡張ではなく、複雑なデータ構造を表現できるタグ付き言語である。HiRDB Adapter for XMLでは、XMLタグとHiRDB表のカラムとでマッピング定義に従った双方向データ交換やXML文書の添付ファイルの取込み等を行うための高レベルなAPI（C++、Java）を提供する。XMLタグとHiRDB表カラムとは、N：Mマッピングが可能であり、自由度の高いデータ交換を可能にしている。これを実現しているのがSAXベースの高速パーサである。さらにはHiRDBのデータベース作成（pdload）ユーティリティと連携し、大量のXML文書の一括インポートができる。

以下に、HiRDB Adapter for XMLの機能概要をまとめておく。

- XML文書とデータベースの双方向データ交換
- XMLタグ：表カラム＝N：Mのマッピング
- XML文書：表＝1：Nのマッピング
- 添付ファイルの読み込み（BLOB格納、外部リンク）
- データローダ連携による複数XML文書の一括インポート
- DTDに従った構造及び記述モレのチェック
- 標準データ変換（例：和暦文字列を日付型への変換）
- コールバックI/F（ユーザ独自のデータ変換、事前処理など）
- C++、Javaのクラスライブラリ形式のスレッドセーフAPI
- 1：1マッピング向け簡易コマンド（プログラミングレス）

図3-1：HiRDB Adapter for XML



3.1.2 Javaによる柔軟なビジネスロジック開発

■アプリケーションサーバとの親和性の向上

国内で初めてJ2EEブランディングを取得したAPサーバである「Cosminexus」でのサポートDBMSとして、HiRDB Version 6はCTS（Conformance Test Suite）をパスした。これにより、CosminexusでのJSP ServletやEJBのビジネスロジックから従来以上にHiRDBにアクセスしやすくなっている。また、J2EEサーバクラスタリングの組み合わせにより、拡張性と可用性の高いシステムを実現できる。HiRDBとCosminexusとの連携により、柔軟性が高く高信頼なシステムの構築を可能にする。Cosminexusのほかには、日本BEA社「WebLogic」との連携も可能であり、フロントエンド基盤のバリエーションを継続的に拡張していく。

■複雑なビジネスロジックをサポートするJavaストアドルーチン

HiRDBのストアドプロシジャおよびストアドファンク

図3-2：Cosminexus

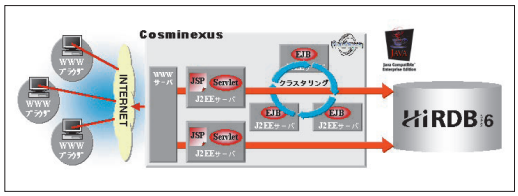
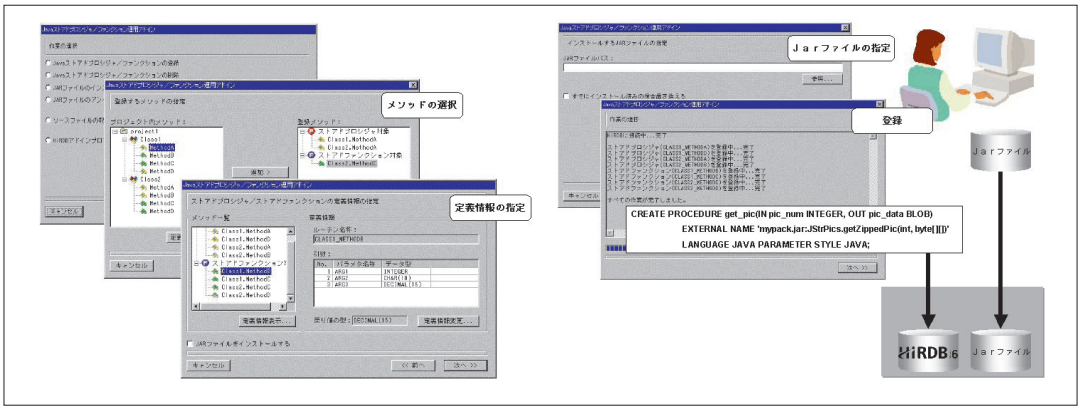


図3-3：JBuilderによるJavaストアドルーチンの登録



ションの記述言語としてJavaを利用することができる。ストアドプロシジャの目的は、データベースにアクセスする処理をデータに近いところ、すなわちDBサーバ側で実行することで、クライアントアプリケーションとの通信回数を減らしてレスポンス時間を向上することにある。従来はSQL構文しか使えなかったため、記述レベルに限界があったが、HiRDB Version 6では、ストアドプロシジャおよびストアドファンクションをJavaでも記述できるようにし、複雑なビジネスロジックをサーバ側で実行できるようになる。WebアプリケーションではJavaを使用する機会が多いので、同じ言語を使用してこれらのルーチンを開発できることから、開発者の負担も軽減できる。

Javaストアドルーチンの開発には、スタンダードなJava統合開発環境であるBorland Software社「JBuilder」を利用できる。したがって、Javaアプリケーションの開発スタイルをそのままに、Javaストアドルーチンを作成、デバッグすることが可能となっている。さらには、開発したJavaストアドルーチンのHiRDBサーバへの配布を支援する専用ウィザードも提供しており、JBuilderにアドインして利用できる。このウィザードを利用すると、ストアドルーチンの登録など、運用操作を簡単に行える。

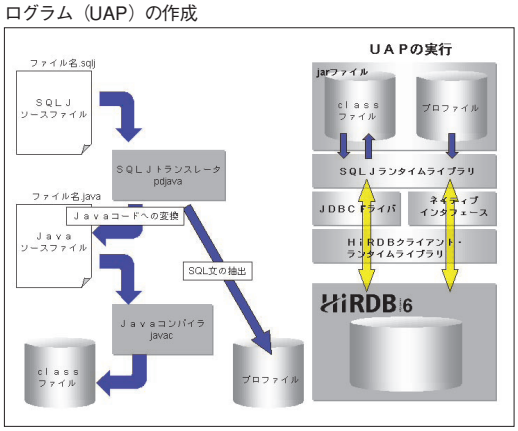
■コーディングを省力化するJava埋め込みSQL

HiRDB Version 6は、ISO SQL Part 10（SQL/OLB）に準拠したSQL埋め込みJava機能（SQLJ）を提供する。これにより、Javaのプログラム中にSQLを直接記述できるようになる。

これまでJavaのプログラムからSQLを発行するには、JDBCインタフェースを使用した複数ステップに渡るコーディングが必要だった。HiRDB Version 6では、SQLをJavaプログラム中に直接記述できるため、コーディングが簡素になる。また、HiRDBのネイティブSQLJランタイムを使用することにより、パフォーマンスも向上する。

SQLJはSQLJトランスレータとSQLJランタイムライブラリで構成される。SQLJトランスレータ(pdjava)は、SQLJソースプログラムを解析して、SQL文をSQLJランタイムを通じてデータベースにアクセスする標準的なJavaのコードに置き換える。このとき、SQLトランスレータはJavaソースファイルと、SQLの情報を格納したプロファイルを生成する。プロファイルには、SQLの文字列、パラメタの個数、各パラメタのタイプとモード、出力する列に関する情報が出力される。プロファイルは、SQLJランタイムライブラリによって参照される。プロファイルの実体は、java.sql.runtime.profileクラスのインスタンスである。ユーザは生成されたJavaソースファ

図3-4：SQLJで記述したJava言語ユーザーアプリケーションプログラム（UAP）の作成



イルをJavaコンパイラでコンパイルして実行jarファイルを生成する。SQLJランタイムライブラリは、コンパイルされたJavaソースファイルを実行するときに使用される。

3.2 ｜データウェアハウスのための技術基盤

データウェアハウスにおいては、アドホックな結合検索や集計処理が主となる。ここでは、HiRDB Version 6のDBMSエンジンとしての基本機能である「内結合・外結合」ならびに「ハッシュジョイン」について説明する。さらに、大規模なデータウェアハウスではヘテロ環境となることが珍しくない。そうした様々な外部データとの連携に対するHiRDBの解決法について説明する。最後に、データウェアハウスから切り出してきたデータマートを十分に活用するときに必要となるOLAP分析ツール「HITSENSER5」を紹介する。

3.2.1 データウェアハウスを支えるコア技術

■傾向分析やサマリ作成に大きな効果を発揮する内結合・外結合

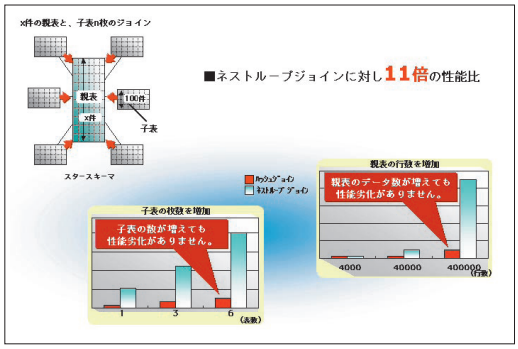
HiRDB Version 6では、データウェアハウス向けの機能としてSQLを強化した。例えば、内結合と外結合をSQL文に自由に組み合わせて指定できるようになっている。また、複雑な結合をビュー表として定義し、さらにそのビュー表に対して内結合・外結合の演算を実施することができる。この結果として得られるリレーショナルに対して集合関数を適用すれば、データウェアハウスで重要な傾向分析やサマリの作成に大きな効果を発揮することが可能である。

■結合検索を高速化するハッシュジョイン

HiRDB Version6では、結合処理の方法として従来のネステッドループ結合、ソートマージジョインに加えて新たにハッシュジョインという結合処理方法をサポートした。ハッシュジョインによって、スタースキーマ構造でのアドホックな結合検索でも高速に実行できるようになった。

ハッシュジョインは、件数の多い表（親表）に件数の少ない多数の表（子表）を結合するようなスタースキーマでの結合検索に特に適していて、結合する子表数の増加や親表の件数の増加による性能の劣化が少ない。従来はあらかじめ結合キーにインデクスを作成することで、

図3-5：ハッシュジョインとネストループジョインとの性能比較モデル



結合検索を高速化してきた。しかし、ハッシュジョインを使用すれば、結合キーのインデクスが不要となるため、インデクスに必要なディスク容量を削減でき、データベースのメンテナンスの負荷も軽減できる。

HiRDBでは、以下の4種類のハッシュジョインの処理方式を用意している。

・一括ハッシュジョイン方式

内表から作成したハッシュ表を、すべて作業表バッファ領域に展開してハッシュジョインする方式である。すべてがバッファ領域に展開されているので不要なI/Oが発生せず、高速に処理することができる。

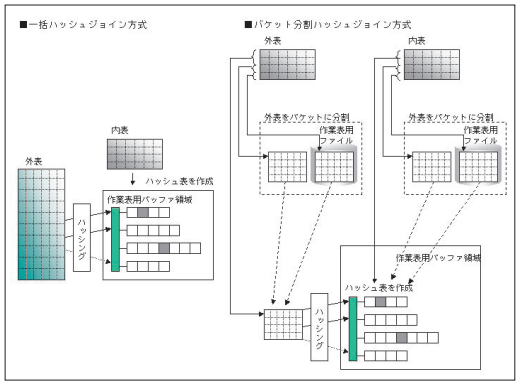
・バケット分割ハッシュジョイン方式

内表、外表をバケットに分割し、内表の一部を作業表用バッファ領域に展開し、残りを作業表用ファイルに退避する。バケットとは、表の結合列の値でハッシングして、複数の小さな表に分割することをいう。結合処理は、作業表用バッファ領域に展開された内表の一部で行う。まず、内表からハッシュ表を作成し、外表から1行ずつ取り出して内表から作成したハッシュ表と突き合わせて結合を行う。作業表バッファ領域にある表同士の結合が終わった時点で、作業表用ファイルから外表、内表のバケットを作業表用バッファ領域に展開し、同様に結合処理を行う。そして、表全体を作業表用バッファ領域に展開して結合した時点で終了となる。この方式は、バッファ領域が少ない場合もハッシュジョインを実施することができるという特長がある。

・連続ハッシュジョイン方式

3表以上の検索に適用する。まず、最も外側の表以外

図3-6：一括ハッシュジョイン方式とバケット分割ハッシュジョイン方式



の表からハッシュ表を作成し、作業表用バッファ領域に展開する。次に、外表から1行取り出してハッシングし、内表から作成したハッシュ表と突き合わせて結合する。結合条件を満たす場合には、結合結果でハッシングしてハッシュ表と突き合わせて結合する。最後の行まで結合し終わるか、または条件が偽になった時点で、最も外側の表まで戻り、次の行を取り出して同様に結合処理を繰り返す。結合途中で内表の結合キー値が重複している箇所がある場合には、そこまで戻って結合処理を繰り返す。重複キー値の処理がすべて終了したら、最も外側の表まで戻り、次の行を取り出して同様に結合処理を繰り返す。この方式では、ハッシュ表をすべてバッファ領域に展開してハッシュジョインをするため、高速に処理することが可能となる。また、最も外側の表だけが大きい場合にも高速に処理できるという特長がある。

・断続ハッシュジョイン方式

3表以上の検索に適用する。まず、最初の結合の内表からハッシュ表を作成し、作業表用バッファ領域に展開

図3-7：連続ハッシュジョイン方式

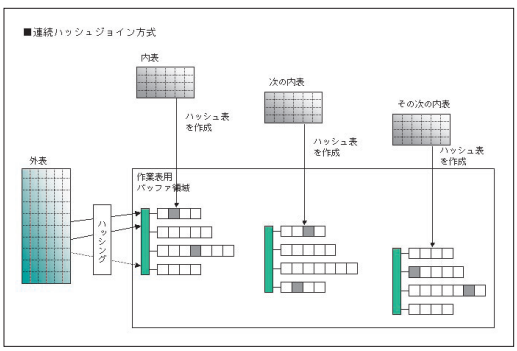
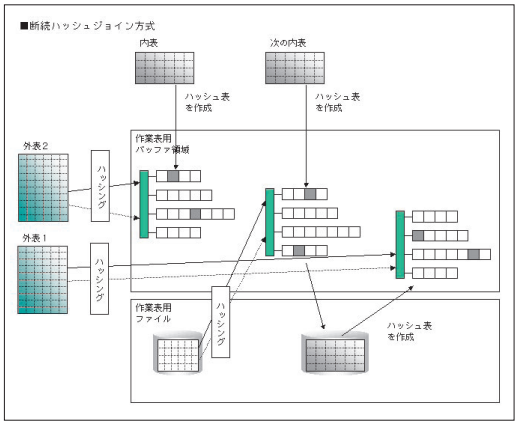


図3-8：断続ハッシュジョイン方式



する。次に、外表を1行ずつ取り出して外表の結合列の値でハッシングし、内表から作成したハッシュ表と突き合わせて結合する。外表からすべての行を取り出して結合し終わったら、次の結合処理に移る。結合結果が外表になる場合と、内表になる場合とで処理が変わる。結合処理が外表になる場合は、次の結合の内表からハッシュ表を作成し、結合結果から1行ずつ取り出して、内表から作成したハッシュ表と突き合わせて結合する。処理結果が内表になる場合は、結合結果からハッシュ表を作成し、外表から1行ずつ取り出して、結合結果から作成したハッシュ表と突き合わせて結合する。この方式は、バッファ領域が少ない場合であっても3表以上のハッシュジョインができるという特長がある。

3.2.2 外部データとの連携

■ Oracle8iとのデータ連携

データウェアハウスで分析しようとするデータは既に構築されたシステムに格納されていることが多い。したがって、新しいデータとの突き合わせを行うためには、こうした既存の異なるDBMSで管理しているデータとの連携が必要になる。これを実現するのが「HiRDB Data extractor/ Datareplicator」である。

HiRDB Dataextractorは基幹データベースから条件抽出して部門データベースに反映させるツールであり、表の初期作成やデータの総入れ替えにも利用することができる。一方のHiRDB Datareplicatorは基幹データベースへの更新情報を部門データベースに反映させるツールであり、基幹と部門の遅延同期を実現する。

その連携対象DBMSのメニューとして、Oracle8iが加わった。これにより、Oracle8iで構築されている既存