SURE 静岡大学学術リポジトリ
Shizuoka University REpository

リレーショナルデータベースにおけるアクセス方式に関する研究

メタデータ	言語: ja 出版者: 静岡大学 公開日: 2012-01-17 キーワード (Ja): キーワード (En): 作成者: 中村, 仁之輔 メールアドレス: 所属:
URL	https://doi.org/10.14945/00006378

理工学研究科；中
GD
K
352
静岡大学附属図書館
0003503547 R

静 岡 大 学　　博 士 論 文

リレーショナルデータベースにおける
アクセス方式に関する研究

平成 14 年 12 月

大学院理工学研究科

設計科学専攻

中村　仁之輔

静岡大学図書

論文要旨

リレーショナルデータベース管理システム(DBMS)は，統一的なデータ形式でデータの定義，蓄積，操作を可能とし，アプリケーションプログラム(AP)の開発，およびシステム運用の効率化をもたらすことにより，システムのパッケージとして重要なものとなっている．DBMS の適用分野も，ビジネス分野から，ネットワークサービス分野へとひろがりつつあり，各種分野での DBMS の実現が求められている．

ビジネス分野の DBMS において，大量データからの検索時間の短縮は，大きな課題であり，解決のアプローチとして，検索機構をハードウェアによって実現するデータベースマシンの研究がある．データベースマシンは，検索処理やソート処理を専用マシン化する付加プロセッサ方式が一般的であり，このハードウェアを，扱うデータの特性に適した形態で利用できる必要がある．リレーショナルデータベースにおいては，検索時，複数のアクセスパスの中から最適なアクセスパスを選択する最適化が重要な課題であり，各種研究が行われている．しかし，データベースマシンでの最適化については，従来の方法では，ハードウェアを十分に利用できない可能性があり，データベースマシン向けの最適化が必要となる．

また，ネットワークサービスにおいては，電話等の接続処理を高速にかつ大量に処理する必要があることから，超高トラヒックな処理が必要となる．従来のビジネス向け DBMS に比べて，レスポンス，スループットとも，1 桁以上の性能向上が必要であり，また，24 時間無中断でのサービアス継続が可能な高可用性が求められる．

本論文は，リレーショナル DBMS の適用領域の拡大に向け，大容量データの検索時間短縮の課題とネットワークサービス向け DBMS 実現の課題を解決するため，データベースマシン向けの最適化方式ならびにメモリ常駐型 DBMS の実現方式を提案し，その実装，評価を行うものである．

大量データからの非定型な検索処理の高速化を目的として開発した，付加プロセッサ方式のデータベースプロセッサ RINDA(Relational Database Processor)は，リレーショナルデータベース用の内容検索とソート処理を実現するハードウェアから構成されている．RINDA システムは，ハードウェアである RINDA とそれを制御する RINDA 制御プログラ

ムより構成される．RINDA を有効に利用して，高速化を達成するためには，検索処理に適合したアクセスパス，関係演算処理を選択する最適化が重要となる．従来の DBMS で実現されている最適化は，SQL の解析時に最適化を行う静的最適化方式であり，問合せ処理途中のテーブルの大きさ等を予測した結果に基づいて最適化を行うため，精度の向上には限界があった．この問題に対し，RINDA システムでは，演算の途中結果のサイズ等に基づき次のアクセスパスおよび関係演算処理を決定する動的最適化方式を実現している．RINDA においては，効率的な処理を実現するために，一度に複数ブロックが処理される．RINDA 制御プログラムでは，この特性に合わせ，複数の行を一時的な格納空間である一時テーブルの形で引き継ぎながら処理する一時テーブル方式を実現している．一時テーブル方式では，一時テーブルのサイズ等を処理途中で正確に知ることができ，この時点で最適なアクセスパスおよび関係演算アルゴリズムを決定することができる．このように，途中結果の情報を利用した動的最適化方式により RINDA を有効に利用することができ，処理速度を向上させることが出来る．本論文では，結合処理，副問合せに対する動的最適化方式を提案し，結合処理の性能評価の結果，動的最適化方式により，RINDA を有効に利用できていることを示す．

次に，ネットワークサービス分野向け DBMS として開発した，メモリ常駐型のリレーショナルデータベース管理システム RENA（Real-time Database Management System）の実現方式を示す．この分野では，従来のビジネス処理向け DBMS に比べ，扱うデータ量が少なく，データ操作も簡易である．しかし，DB アクセス処理においては，従来の 10 倍以上の高性能が求められている．また，24 時間無中断でサービスを提供する必要があるため，DB バックアップ処理や DB 保守処理についても，サービス無中断で提供する高可用性が求められている．RENA はこれらの要求にこたえて開発した DBMS である．メモリデータベースは，従来の DK ベースの DBMS に比べて，メモリ上での処理が主体となるため，従来とは異なる技術が必要となる．RENA は，メモリデータベースに必要な，アーキテクチャ，格納構造，アクセス法，言語処理，並行制御，リカバリ制御，DB 保守を実装したものである．RENA は，新たに提案したメモリセル型記憶構造，リアルタイム SQL 仕様，コンパイルド SQL 方式，最適化方式，並行制御におけるファジーロック方式，サービス無中断 DB 保守方式技術と，既存技術である，改良拡張ハッシングによるインデッ

クスアクセス法，スピンロック方式による並行制御方式，ファジーチェックポイント方式によるオンライン・コンカレント・リカバリ制御方式を組み合わせて実現している．本論文では，RENA で実現したメモリデータベース方式に基づいた高性能化方式，および高可用性を実現するサービス無中断 DB 保守方式の設計，実現方法を述べる．また，従来 DBMS との性能比較を行い，CPU 処理コストおよびスループットが従来 DBMS に比べて格段に優れていることを示す．

目次

第 1 章

序論

1.1 研究の背景と目的

データベース管理システム（DBMS）は，統一的な形式でデータの定義・蓄積・操作を可能とし，アプリケーションプログラムの開発，およびシステム運用の効率化をもたらすことにより，システムの中核パッケージとして重要なものとなっている．特に，リレーショナルデータベースは，表形式のインタフェースであり，アプリケーションプログラム内への記述が簡易な点から，幅広い分野に普及しつつある．リレーショナルデータベースは，ビジネス処理分野を中心に普及しているが，ビジネス分野だけでなく，意思決定支援システム等，大量データの検索処理が必要な分野から，ネットワークサービス等，高トラヒック処理が必要な分野へと，適用領域の拡大が求められている．

大量のデータから必要なデータを検索する場合，リレーショナルデータベースにおいては，高速化が大きな課題である．特に，複数のテーブルを取り扱う場合の処理に多くの処理コストがかかることから，高速化は重要な課題となっている．高速化のアプローチには，ソフトウェアで実現するものと，処理の一部をハードウェアにて実現するアプローチがある．ソフトウェアにおいては，複数テーブルの演算である結合処理や副問合せ等に関し，各種方式が提案され，さらに，これらの各種方式の中から最短時間で実行できる方式を選択する最適化方式により，高速化が図られている．しかし，大量データからの検索の場合，I/O ごとにブロックを読み込み，その中から必要なデータを探索する，あるいは，ソート処理をソフトウェアで実行する等の処理が必要であり，処理時間の短縮には，限界がある．そのため，処理の一部をハードウェアで実現するデータベースマシンによる方法が研究されている．データベースマシンの研究では，ハードウェア化の対象により，各種研究が行われているが，筆者らは，非定型検索処理の高速化を達成するため，付加プロセッサ方式のデータベースプロセッサ RINDA(Relational database processor)を提案している．データベースプロセッサ RINDA を使用した RINDA システムの実現にあたっては，RINDA

ハードウェアを有効に利用することが重要であり，それを実現する最適化が大きな課題となっている．

リレーショナルデータベース処理の高速化を目指して開発したデータベースプロセッサ RINDA は，内容検索を主に実行する CSP（内容検索プロセッサ）と，ソート処理を主に分担する ROP（関係演算プロセッサ）から構成され，これらを使用してデータベースの検索処理の高速化を達成している[46][47]．CSP，ROP の制御は，RINDA 制御プログラムで行われ，その制御方式は，CSP，ROP の処理結果である複数の行を一時的な格納空間である一時テーブルの形で引き継ぎながら処理する一時テーブル方式で実現している[48]．従来の最適化方式は，SQL 解析時にアクセスパスおよび関係演算アルゴリズムを決定する方式であり，I/O 回数が最も少ないと想定されるものを選択する予測に基づく方法のため，精度が低い問題があった．一方，RINDA の一時テーブル方式においては，実行時にアクセスパスおよび関係演算アルゴリズムを決定する動的最適化方式を実現可能であり，より精度の高い最適化により，CSP，ROP を有効に利用できる．本論文では，結合処理，副問合せに関して，CSP，ROP を利用した関係演算方式を示し， CSP，ROP の処理結果情報から，それらの関係演算方式の選択を実行時に行う動的最適化方式を示す．最後に，結合処理に関して，実装評価を行い，RINDA を使用しない場合に比べて，格段に性能が向上していることと，CSP，ROP を有効に利用できていることを示す．

DBMS の適用分野も，ビジネス分野から，ネットワークサービス分野へと広がりつつあり，これらの分野においても，利用可能な DBMS の実現が求められている．

ネットワークサービス分野の場合，たとえば，高度電話サービス等の実現においては，サービス変更，追加の容易なソフトウェア構成が必要なことから，データ管理に DBMS を適用する必要性が出てきている．この分野においては，扱うデータ量は少なく，データの操作も簡易であるが，超高トラヒックな処理が必要であり，レスポンス，スループットとも，従来のビジネス向け DBMS と比べて，一桁以上の性能向上（高性能）が必要である．また，24 時間無中断でサービスを継続する必要があるため，従来は計画停止などを必要とした DB 保守処理やバックアップ処理などについても，オンライン処理を中断することなく行うことが必要である．

一方，近年のメモリコストの減少により，全 DB をメモリ上に常駐させるメモリ常駐データベース（メモリデータベース）の実現が可能となってきている．メモリデータベースでは，従来の磁気ディスク(DK)にデータベースが格納される DBMS(DK-DBMS)で必要であった I/O が不要となり，メモリ上でのみの処理でアクセスが可能となるため，高性能化を図ることができる．

以上の背景から，ネットワーク系のサービスに適用可能な DBMS として，メモリ DB 技術を主体とし，高性能かつ高可用な特徴をもつ RENA（Real-time Database Management System）を開発した．本論文では，RENA で実現したメモリデータベース方式に基づいた高性能化方式，および高可用性を実現するサービス無中断 DB 保守方式の設計および実現方式を示す．

RENA の実現にあたっては，メモリデータベース技術の実装が必要であり，新たに提案した技術と既存技術を組み合わせて本格的な DBMS を実現している．具体的には，新たに提案した技術として，メモリセル型記憶構造，言語処理におけるサブセット SQL 仕様，実行コードを直接リンクするコンパイルド SQL 方式，メモリ上のデータ配置特性と SQL 発行順序特性に着目した最適化方式，並行制御におけるファジーロック方式，ならびに，新旧テーブル切替え方式によるサービス無中断 DB 保守方式を提案・実装している．既存技術としては，改良拡張ハッシングによるインデックスアクセス法，スピンロック方式による並行制御方式，ファジーチェックポイント方式によるオンライン・コンカレント・リカバリ制御方式を実装している．本論文では，これらの実現方式を示すとともに，性能評価を行った結果，従来 DK-DBMS に比べて CPU 処理コストおよびスループットが約 10 倍以上優れていることを示す．

1.2 論文の構成

本論文は，5 章より構成される．

1 章ではリレーショナルデータベースのアクセス方式に関し，研究の背景とその目的を述べ，論文の構成を述べる．

2 章では，従来の研究を概観し，本研究の特長を述べる．具体的には，2.1 でリレーショ

ナルデータベースの動向を，2.2 でデータベースマシンの研究動向と最適化の研究動向を，2.3 でメモリデータベース技術の研究動向とメモリデータベースシステムの研究動向を述べ，2.4 で本研究の特長を述べる．

3 章では，データベースプロセッサ RINDA の最適化方式を提案し，実装評価結果を述べる．具体的には，3.2 で RINDA の関係演算処理を詳細に概説し，3.3 で RINDA の最適化方式全体を提案し，静的最適化方式と新たに提案した動的最適化方式から構成されることを示す．3.4 で RINDA における静的最適化方式を述べ，3.5 で結合処理に関する動的最適化方式を，3.6 で副問合せに関する最適化方式を述べる．3.7 で結合処理に関する動的最適化方式の実装評価結果を示し，動的最適化方式により，RINDA ハードウェアを有効に利用できていることを明らかにする．

4 章では，ネットワークサービス向けのメモリ常駐型 DBMS である RENA の実現方式と実装評価結果を示し，従来 DBMS に比べて，一桁以上の性能向上を達成していることを示す．具体的には，4.2 でメモリデータベースに関する従来の研究を概観し，4.3 でネットワークサービスの DBMS に関する要求条件を明らかにし，4.4 では，4.3 で示した要求条件を満足する DBMS としてメモリ常駐型 DBMS を提案し，そのアーキテクチャを示す．4.5 で RENA の高速アクセスの核となる記憶構造とアクセス制御方式を，4.6 で SQL のサブセット化，コンパイルド SQL ならびに最適化の言語処理方式を，4.7 でスループット向上のためのトランザクション間の並行制御方式を，4.8 で 24 時間無中断向けのリカバリを実現するオンライン・コンカレント・リカバリ方式を示し，4.9 で 24 時間 DB 保守を実現するサービス無中断 DB 保守方式を示す．4.10 では，ネットワークサービスにおける CPU 処理コスト評価とスループット評価を行い，従来 DK-DBMS に比べて格段に性能が優れていることを示す．

5 章では，本研究の成果のまとめを示す．

第 2 章

従来の研究の概観と研究の特長

2.1 リレーショナルデータベースの動向

データベース処理において，ネットワーク構造型データベースは，情報処理システムの発展過程において，定型的な業務処理のシステム化に大いに貢献してきた．それは，ISO および JIS で標準化されたデータベース言語 NDL[53][57]に代表されるように，ネットワーク型データベースへのアクセス経路を親レコード型と子レコード型の関連(親子集合型)として，データ構造に表現できるため，効率のよいデータベースシステムを実現できることによる．しかし，ネットワーク構造型データベースは，アクセス経路を正確に設計する必要がある点，アクセス経路の変更時にアプリケーションプログラムに影響がある点，アクセス経路から外れたデータ操作の非効率さなどの問題点がある．情報処理システムが一般化し，業務の専門家であるが，情報処理の非専門家によるデータベース設計や非定型的な業務処理へのデータベースの適用が求められるに伴い，ネットワーク型データベースの上記の問題点の解決が必要になってきた．

上記の問題点を解決する手段として，リレーショナルデータベース[17]が注目されてきた．

リレーショナルデータベースでは，アプリケーションはデータ構造をすべて表の形で取り扱い，その表の任意の行と列の集合に対してデータ操作を行うことができる．そのためのデータ操作は，ISO および JIS で標準化されたデータベース言語 SQL[54][58]に見られるように高水準言語インタフェースが使われている．この標準化された高水準言語インタフェースを用いることによりリレーショナルデータベースは使いやすく，柔軟性が高い特徴を備えるにいたっている．

しかし，リレーショナルデータベースでは，アプリケーションプログラムとの言語インタフェースが高水準であるが故に，データベース管理システムの処理負荷が大きい欠点を持つ．リレーショナルデータベースの適用分野をビジネス分野だけでなく，ネットワーク

サービスに広げていくためには，上記の性能課題を解決する必要がある．

ビジネス分野においては，大型アプリケーションで本格的に利用されるようになるにつれ，検索処理速度が問題になってきた．大量データに対する複雑な検索では，処理速度は急速に悪化する．また，インデックスを利用しない全件検索（サーチ処理）や複数のテーブルを共通な列の値で結合する処理（結合演算処理）なども時間がかかる．

この性能改善に対しては，①DBMS ソフトウェア技術によるアプローチと，②ハードウェアアーキテクチャによるアプローチがある．後者においては，データベースマシンによるアプローチが一般的である．

ネットワークサービス分野においては，ビジネス分野に比べて，10 倍以上の高性能と 24 時間サービス無中断の高可用性が求められている．ネットワークサービスの処理特性を見ると，DB 容量が小さく，処理パターンが簡易であるという特長がある．一方，近年，メモリコストが低下してきており，DK へのアクセスが不要となることにより，高性能化を図ることができるメモリデータベースによるアプローチが実施されている．

以降では，データベースマシンの研究動向と最適化の研究動向を，また，メモリデータベースの研究動向を示す．

2.2 データベースマシン関連の研究動向

2.2.1 データベースマシンの研究動向

1970 年代の初めに，汎用計算機からデータベース処理の機能を分離し，データベース処理専用の計算機であるデータベースマシンにその機能を分担させるという概念が提案された[95]． データベースマシンに関する提案は数多くなされており，その流れについては，文献[45]，[101]に詳述されている．

実用のデータベースマシンは，実現している機能とホスト計算機との接続形態により，以下の２つに分類できる．

（ａ）バックエンドプロセッサ方式

この方式のデータベースマシンは，ホスト計算機からデータベースの処理指令を受ける

と，ホスト計算機とは独立に，自己完結的にデータベース処理を行い，ホスト計算機へ処理結果を転送する．

（b）付加プロセッサ方式

この方式のデータベースマシンは，ホスト計算機上の DBMS のデータベース処理機能の一部を専用ハードウェア化して，DBMS と協調して動作する．ホスト計算機に接続する形態は 2 種類に分類され，専用ハードウェア化する処理機能が異なる．

（i）I/O 強化型

ホスト計算機とディスク装置などのデータベース格納装置との間に接続され，サーチ処理の高速化をねらうものである．ディスク制御装置の中に付加される形態もある．

（ii）CPU 強化型

データベース格納装置とは独立にホスト計算機に接続され，CPU 処理の高速化をねらうものである．専用ハードウェア化する機能は，ソート処理や結合演算処理である．CPU の内部に接続する形態と I/O インタフェースを介して接続する形態がある．

上記の分類に従って，データベースマシンの開発動向を述べる．

（1） バックエンドプロセッサ方式

大型汎用計算機用の DBC/1012[99]，ミニコン用の Server シリーズ[13][93]，ワークステーション用の HDM[79]，パソコン用の DB-1/XP[82]がある．これらのシステムは，汎用プロセッサの上にデータベース処理に最適化した専用 OS を開発し，周辺装置やシステム構成を専用化している．

（2）付加プロセッサ方式

汎用計算機に付加する CAFS[44]，IDP[63][64]，オフコンに付加する GREO[2]がある．

CAFS は，I/O 強化型の付加プロセッサで，ディスク装置中のファイルとして格納されたデータのサーチ処理とふるい落とし機能を専用化したものであり，検索用のハードウェアを組み込んだ専用のディスク制御装置として実現されている．GREO は，CPU 強化型の付加プロセッサで，主記憶上に読み上げられたデータに対するソートとインデックスの生成処理機能を専用化したものであり，ハードウェアのマージソータと 3 個のマイクロプロセッサにより実現されている．また，IDP も，CPU 強化型の付加プロセッサで，主記憶

上のデータに対するソートと基本的な関係演算機能を専用化したものであるが，GREOとは異なり，汎用計算機に内蔵されたベクトルプロセッサをデータベース処理用に拡張することによって実現されている．

一般に，付加プロセッサ方式は，汎用計算機の性能ネックとなっている機能のみを専用ハードウェア化するため，小さな追加コストで大きな性能向上が実現できるという利点がある．

筆者らが開発したRINDA(Relational Database Processor)は，リレーショナルデータベースに対するインデックスの有効利用が困難な非定型処理を高速化することを目的とする付加プロセッサ方式のプロセッサである．具体的には，非定型処理を実行する際に汎用計算機にとって負担が重いサーチ，ソート等の処理を専用ハードウェアが分担して高速に実行する．これにより，従来のシステムではオンライン・リアルタイムでの処理が困難であった大規模データベースに対する非定型処理を最高100倍にまで高速化することを可能にした．

RINDAは汎用計算機のDIPSシリーズ[83][104]と入出力インタフェースにより接続される，DIPS上のソフトウェアのデータベース管理システム（DEIMS-5：Denden Information Management System-5）によって制御される．以下，RINDAの接続された汎用計算機をホスト計算機，データベース管理システムのうち，RINDAを制御するサブシステムをRINDA制御プログラムと呼ぶ．RINDAが実現する機能は，非定型処理のみであり，その他の機能，例えばデータベースの生成処理や，定型処理は従来通りデータベース管理システムによって実現される．アプリケーションプログラムを作成するためのインタフェースはSQL[54][58]に準拠したものであり，アプリケーションプログラムからRINDAを意識する必要はない．また，ホスト計算機およびデータベースが格納されるディスクとディスク制御装置は汎用のものがそのまま使用できるため，既存ユーザの設備を無駄にすることなくRINDAを導入することが可能である．

2.2.2 最適化の研究動向

リレーショナルデータベースのソフトウェア処理での最適化方式には，代表的なコスト評

価方式がある [11][92]. 最適化方式の研究は，System R の開発の中で，研究されたもので，商用化されたDBMSの元となったものである．このコスト評価方式の特長は，DBの状態情報（各テーブルの行数，インデックスキー値の種類数等）を元に，I/O 回数が最小と予想されるアクセスパスを選択することにある．DB の状態情報は，一定の周期あるいはアドホックに更新される．本方式は，予測に基づくため，必ずしも最適なパスが選択されるとは限らない．特に，大量データを検索する場合，データの絞り込み状態により，アクセスパスの選択方法は異なるが，絞り込み効果を予測に基づいて算出するため，精度が低い問題がある．また，従来の方式は，ソフトウェア処理向けの最適化しか実現されておらず，RINDA のように，ソフトウェアとハードウェアの処理が混在している最適化は実現されていない．

INGRES で提案された初期の最適化[103]は，質問処理実行時に，アクセスプランを構築しそれを実行する，一種の動的最適化であった．しかし，System R の最適化が[92]が公表されたのち，大部分の研究は，言語解析時に最適化を行う，静的最適化に焦点が当てられた．その後，静的最適化のアプローチは，予測に基づく方法のため，限界があることが認識され，最適化の実行を質問処理の実行直前に行う方法が提案されてきた．

文献[19]では，質問処理実行直前に，準最適と想定されるアクセスパスに再最適化が行われるオプチマイザーにより，質問処理の実行プランが生成される方法が述べられている．最適化実行時オプチマイザーで使用される統計情報は，データベース内に実行プランと一緒に格納されている．質問処理実行時，システムカタログから得られる統計情報が，実行プランと一緒に格納された統計情報と比較される．双方の差が大きい場合，実行前に，再最適化が行われる．この方法では，質問処理実行直前に再最適化が1回行われるのみである．また，統計情報の収集は行われず，質問処理途中での実行プランの変更は行われない．

文献[3]，[4]では，競合モデルによる動的な実行プランの決定方法が示されている．このアプローチでは，競合する複数の実行プランが実行され，その後，その中で有効なプランが明らかになり，他の準最適なプランの実行が中止される．このアプローチは，特別なテーブルスキャンや特殊な結合演算にのみ利用されるもので，すべての質問処理への適用ができない．

文献[1]では，質問スクランブリングと呼ばれるいくつかの再最適化が示されている．しかし，この方法は，分散DB向けの特殊な向けのものである．分散DBのアクセス時，データの到着が遅い場合，動的に再スケジュールするものであり，分散DBの資源状態に依存した場合に有効なものである．

文献[59]では，実行時に質問処理の実行プランを変更する再最適化方式が示されている．本アプローチは，最初の最適化時に，質問処理の実行プランすべてを生成しておき，実行時に統計情報を収集しながら，その結果を参照して，残りの実行プランを再度最適化するものである．

文献[56]でも，同様に再最適化方式が提案されている．本アプローチは，最初の最適化時に，実行プランを生成するが，その際，実行プランを生成するための情報が不足している場合，部分的な実行プランを作成しておくものである．実行時には，統計情報を収集しながら，残された処理に対して，最適化を実行するものである．しかし，最初に実行プランが生成される場合は，文献[59]と同様な処理を行うもので，基本的には，文献[59]と同様なアプローチである．

文献[59]，[56]ともに，ソフトウェア処理を対象にしたものであり，実行時の統計情報の収集もソフトウェアで処理を実行している．

2.3 メモリデータベースの研究動向

メモリコストの低下に伴い，主記憶メモリ上にデータベースを配置したメモリデータベース(MMDB)の研究が進展している．MMDBは，従来のDKベースのDBMS(DK-DBMS)に比べて，DKへのアクセスが不要となることから，レスポンスタイム，スループットが向上できる特長がある．

メモリデータベースは，主に，通常のビジネス分野やテレコム系の業務向けに研究が行われている[24][31]．また，プロセス制御やレーダ追跡等の時間保証が必要なリアルタイムシステム向けに，リアルタイムデータベースの研究が行われている[94]．

MMDBは，データベースのデータがすべてメモリ上で操作されることから，サイズ上の制限があるが，フリーダイヤルに代表されるネットワークサービス分野等，データ量が

少ない分野での利用が可能である.

データ量により，MMDB と DK-DBMS を使い分ける方法もあり，IMS では，Fast Path[33]がメモリ上で用いられ，残りのデータが DK で管理されている．文献[96]では，マルチレベルのデータベースシステムに関するいくつかの課題が議論されている.

MMDB を実現するには,従来の DK-DBMS とは異なる技術が必要になってくる.以下，MMDB 実現技術の研究動向と MMDB システムの研究動向を述べる.

2.3.1 メモリデータベース技術の研究動向

（1）データ構造

メモリデータベースでは，データ構造において，効率的なポインタ利用が可能である．リレーショナルデータベースの行をデータ値に対するポインタの集合として表現可能である[84][102]．また，可変長データに対し，ポインタを利用した方法がある[73][88].

（2）アクセス法

メモリデータベースにおいては，バイナリサーチツリー[81]や B-tree[8][18]のようなブロックを意識した格納構造向けのインデックス構造では効率が悪い．メモリデータベース向けに，各種のインデックス構造が提案されている[20][39][40][71][102]．これらの方法には，ハッシングとツリー構造がある．ハッシングは，ユニーク・キーアクセス向けであり，ツリー構造は，範囲検索やキーに対する複数タプルアクセス向けである.

メモリデータベースに適したツリー構造として，T-tree が提案されている[71]．この方式は，DK-DBMS の B-tree に相当するもので，範囲検索やキーに対する複数タプル向けに有効な方式である.

ユニーク・キーアクセス向けのハッシングに関しては，スタティックハッシング[6]とダイナミックハッシングがあるが，データ量に応じて動的にハッシュ空間の変更が可能なダイナミックハッシングの研究に重点がおかれており，各種の方法が提案されている[5][7][15][25][28][52][74].

ダイナミックハッシングは，ディレクトリ構造を用いる手法と用いない手法に分類され

る．ディレクトリを用いる手法としては，拡張ハッシング[26]が代表的である．拡張ハッシングは，ハッシュ表を各ページへのポインタとして利用するもので，ディレクトリ構造部のサイズが対象レコード数に応じて変化する．ディレクトリ構造により各種の手法が提案されている[27][40][77][78]．その中でも，平野らによって提案されている改良拡張ハッシング[40]は，ユニークキー向けの同時走行性を向上させる手法であり，実用性が高い．ディレクトリを用いない手法としては，線形ハッシング[76]が代表的である．線形ハッシングは，拡張ハッシングと異なり，ハッシュ関数値が直接ページ番号に対応する手法であり，キー値が増加することにより，レコード格納空間が拡張される．線形ハッシングについては，あふれ空間の実現方法により，各種方法が提案されている[65][66][67][68][69][70].

（3）言語処理

メモリデータベースの言語処理では，主に最適化の研究が行われており，結合等の複雑な演算を伴う最適化が研究されている[10][20][72][84][102]．最適化の方法は，DK-DBMS が I/O 回数を最小にするアクセスパスを選択するのに対し，CPU 処理コストを最小にするアクセスパスを選択する方法が採られている．しかし，言語仕様に着目したアプローチや実行時にメモリデータベースの領域をアプリケーションから直接アクセスする方法はない．また，言語処理においては，実行時の性能向上のため，プリコンパイル方式が主流である[12]．この方式では，プリコンパイル時に，SQL の解析，最適化を行い，その結果の DBMS へのアクセスコードを生成し，そのコードを DK に格納しておく．実行時には，そのコードを呼び出すことにより，実行時に SQL 解析，最適化を省略できるため，処理の高速化を図ることができる．この方式では，標準 SQL を実装すると，多くの処理単位（数式処理，結合処理，カーソル管理等）を管理しておき，その中から必要な処理単位をダイナミックリンクする必要がある．そのため，実行時の制御用の処理コストがかかる．また，DBMS の処理単位の実行空間は，アプリケーションの領域には含まれない．

（4）並行制御

並行制御は，一般的に，ロック制御による方法が主流である．MMDB において，ロック制御を利用する場合，DK-DBMS に比べて，ロック時間が短くなる．ロック競合を避け

るため，ロックの粒度を小さくするのが一般的であるが，MMDB では，ロック時間が短いため，ロック粒度を大きくするアプローチがある[73]．また，トランザクションをシリアルに実行することにより，ロック制御のコストを削減するアプローチがある[30]，[75]．実システムでのロック制御の実装は，MMDB においても，メモリ上のロック対象に対して行われる．DK-DBMS では，ある時点でロックされている資源に対するエントリを含むハッシュテーブルを使って実装されており，DK 上の資源自体には，ロック情報を含んでいない．MMDB では，ロック情報を資源上にビットレベルで格納しておく方法が採られている．IMS では，各資源（レコード等）にロック有無とウェイト有無のフラグを持つ方法が実装されている．この方法は，ロック状態を現状のフラグに設定するのみのため，ロックの管理用テーブルが不要となる．

以上のように，MMDB では，ロックの粒度を下げてロック競合を小さくし，また，資源内にロック管理情報を保持する方法が採用されている．ただし，ロック制御ではウェイト制御が行われており，そのオーバヘッド削減が課題である．

（5）リカバリ制御

DK 障害に対しては，バックアップを取得しておき，トランザクションのログを保存しておく方法が一般的である[34]．MMDB においては，コミット時の処理時間短縮が課題であり，いくつかの解決法が提案されている．

ログの一部をメモリ上の別領域に保持しておき，それを別プロセスで順に書き出す方法がある[20][30][36][73]．この方法は，ログ書き出しのボトルネックは解消しないが，レスポンスは向上する．次に，ロックリリースの時点でコミットしたとみなすプリコミットによる方法がある[20][33]．この方法は，レスポンスは向上しないが，ブロッキングの待ちを解消できる．さらに，ログのボトルネックを解消する方法に，グループコミットがある[20][33]．この方法は，複数のコミットをメモリ上に保有しておき，まとめて DK に書き出すものである．

メモリデータベースのバックアップは，通常，DK 等の永続記憶装置に取得される．大部分のシステムでは，障害からのリカバリのため，バックアップまたはチェックポイントにログをかぶせる方法を採っている[36][73][75][87][88]．チェックポイント取得時は，ト

ランザクション処理への影響を最小にする必要がある．チェックポイント取得時，トランザクションとの整合性をとるためには，トランザクションとの同期を取る必要があるが，ファジーチェックポイント方式では，同期が不要であり，トランザクションへの影響を抑えることができる[36][85].

DK-DBMS 向けではあるが，データベースリカバリの高速化に向けて，頻繁に更新されるデータのみ，RAM に格納する方法も提案されている[60].

（6）データベース保守

メモリデータベースにおいても，DB 再編成，DB 再構成に関しては，DK-DBMS と同様，オンライン中にサービスを停止せず実現する方法は提案されていない．TimesTen[100]では，レプリケーションを利用した DB 保守方式が実装されているが，サービス無中断での DB 保守は実現できていない．

2.3.2 メモリデータベースシステムの研究動向

メモリデータベース管理システムに関しては，提案レベル，プロトタイプレベル，商用レベルのものがある．提案レベルのものには，MM-DBMS，MARS，HALO が，プロトタイプレベルのものには，OBE，TPK，System M が，商用レベルのものには，IMS/VS Fast Path，Times Ten がある．

（1） OBE

メインメモリデータベースマネージャは，IBM の Office-By-Example(OBE)データベースプロジェクト[10][102]にて実装された．このシステムは，IBM370 アーキテクチャ上で動作するもので，非定型検索向けに焦点が当てられている．データ構造は，リレーションはタプルのリンクリスト構造であり，アトリビュート値に対するポインタのアレイ構造となっている．インデックスは，アトリビュート値に対応するタプルポインタのアレイ構造である．結合演算は，ネステッドループ方式であり，インデックスは，結合演算実行時に on-the-fly で生成される．言語処理は，CPU 処理コストの削減に焦点が当てられている．

（2）MM-DBMS

MM-DBMS は，Wisconsin 大学で設計されたリレーショナルデータベース向けのシステム[72][73]であり，ポインタを利用したデータ構造を採っている．インデックス構造は，非順序データアクセス向けのリニアハッシングと順序データアクセス向けの T-tree[71]である．リカバリのため，メモリ上は，大き目のブロック単位に分割され，このブロック単位で DK へのバックアップが行われる．ログは，別プロセッサにて取得される．チェックポイント処理中は，DK 上の一貫性を保持するため，ロックが設定される．並行制御は，2フェーズロック方式で実現され，ロック単位は，リレーション単位である．

（3）IMS/VS Fast Path

IMS/VS Fast Path は，メモリ常駐データをサポートする IBM 社の商用プロダクトである[33]．このシステムでは，DK 常駐のデータもサポートされ，それぞれのデータベースがメモリ上か DK 上のどちらかに格納される．コミット処理は，グループコミットが実装され，ロック単位は，レコード単位である．データベースのタイプはネットワーク型である．

（4）MARS

MARS MMDB は，メモリ常駐データに対し，高速なトランザクション処理を可能とする 2 種類のプロセッサ用に開発された[22][23]．MARS システムは，データベースプロセッサとリカバリプロセッサから構成され，双方のプロセッサは，メモリ上に直接アクセスを行う．リカバリプロセッサは，ログ取得とバックアップ処理時に，DK にアクセスする．チェックポイントは，ファジーダンプであり，並行制御は，リレーション単位の 2 フェーズロック方式である．

（5）HALO

Hardware Logging(HALO)は，トランザクションログ取得のために開発された装置である[30]．

（6）TPK

TPK はマルチプロセッサのメインメモリトランザクション処理システムのプロトタイプである[75]．TPK は，debit/credit タイプのトランザクションの高速実行向けに開発され，ユニークレコードが対象の単純なデータモデルをサポートしている．

並行制御は，各トランザクションがシリアルに実行されるため，特に実装されていない．データベースは，2 世代（Primary，Secondary）あり，Primary は，全トランザクションのリードとアップデートをサポートしている．全更新レコードのコピーはログに取得される．コミット制御は，トランザクション毎のログの DK 書き込み回数を削減するため，グループコミットを実装している．

実行スレッドは，ログレコードのコピーをチェックポイントプロセスのキューに配置する．チェックポイント機構は，このレコードを読み出し，Secondary データベースの更新を反映する．Secondary データベースは，チェックポイントを完全にするために，周期的に DK 上にコピーされる．Secondary データベースは，チェックポイントとチェックポイント操作中の実行スレッドとの不整合を解消するために実装されている．

（7）System M

System M は，TPK プロトタイプと同様に，非定型処理よりもトランザクション系に向けて開発されたテストベッドである．本システムは，単純なレコード指向データモデルを採用している．

System M は，Mach オペレーティングシステム上の複数のサーバとして実装されている[88]．メッセージサーバは，トランザクション要求を受け付け，結果をクライアントに返却する．トランザクションサーバは，トランザクションの実行，データベースの更新ならびにログデータの生成を行う．ログサーバはメモリ上のログを DK に書き込み，チェックポイントサーバは，DK 内のバックアップデータベースを更新する．

TPK と異なり，並行制御機構を実装しているが，同時走行トランザクションの数を少なくするアプローチを採っている．並行制御は，2 フェーズロック方式を，コミット制御は，効率的なログ処理のために，プレコミットとグループコミット双方を実装している．

MM-DBMS と同様，Primary データベースは固定長のセグメントに分割され，その単位で，バックアップディスクに格納される．インデックスレコードは，バックアップデータベースには格納されず，障害時の復旧時，バックアップレコードを元に生成される．Syatem M は，リカバリ方式の比較実験のため，複数のチェックポイント方式とログ取得方式が実装されている．チェックポイント方式は，ファジーチェックポイント方式と完全チェックポイント方式が，ログ取得方式は，物理ログと論理ログが実装されている．

（8）Times Ten

Times Ten[100]は，最近発表された本格的な商用 DBMS である．本システムは，DK-DBMS と接続しており，SQL はフル仕様で実現されている．また，アプリケーションプログラムとのインタフェースは，ODBC，JDBC が実装されている．インデックス構造は，ユニーク・キーの＝条件向けにハッシュが，範囲検索向けと複数タプル向けに T-tree が実装されている．2 フェーズコミット，チェックポイントによりリカバリ機構が実装され，可用性向上のために，レプリケーション機構が実装されている．言語処理の高速化のためのストアードプロシージャは，DK-DBMS に依存している．

2.4 研究の特長

本節では，RINDA 最適化研究とメモリ常駐型リレーショナル DBMS 研究の特長を示す．

2.4.1 RINDA 最適化研究の特長

本研究は，非定型処理に向けて開発した，付加プロセッサ方式の RINDA に関し，そのハードウェアである，CSP，ROP の機能を有効に利用することにより，性能向上を達成できる最適化方式を明らかにするものである．

2.2.2 に示すように，最適化については，主に，ソフトウェア処理向けの静的最適化と動的最適化が提案されている．

しかし，RINDA においては，ソフトウェア処理とハードウェア処理の並存問題の解決

が必要である．RINDA はデータベース処理全体のうち非定型処理のみをサポートしており，定型処理は具備していない．そのため，汎用計算機上のデータベース管理システムがRINDA の制御を行うアーキテクチャを採用している．即ち，従来の DBMS に RINDA 制御プログラムを付加し，ソフトウェアによる処理（ソフト処理）と RINDA による処理（RINDA 処理）を統合した DBMS の開発を行った．その結果，同一のデータベースに対して,①各利用者が両処理を混在させた実行,②両処理が混在した複数利用者の同時実行，の両方を実現し，利用者は RINDA 処理による非定型処理とソフト処理による定型処理を共通インタフェースで利用可能となった．

両処理を統合した DBMS では，利用者は RINDA の使用有無を意識せずに SQL を記述できることが望ましい．RINDA は機能上，検索処理のみをサポートしている．また，検索処理であっても，インデックスが有効に利用できる場合，例えばユニークなキーを用いた 1 行検索では，ソフト処理の方が高速になる．従って，利用者に RINDA の使用有無を意識させず，かつ高速のデータベース処理を提供するためには，機能上，性能上から両処理の最適なアクセスパスの選択が重要となる．このソフト処理と RINDA 処理の最適化方式に関しては，文献[41]にて解決案を示している．しかし，RINDA 処理における最適化については，RINDA のハードウェアを有効に利用するために，研究課題がある．

最適化に関しては，最適化方式の代表的なものに，System R のコスト評価方式による静的最適化方式がある[92]．この方式は，データベースの状態に関する統計情報，例えば，テーブル中の行数，列の値の分布等を利用するものであるが，一般に一つの問合せは複数のオペレーションからなり，オペレーション間で引き渡される中間結果を予測することには，限界があり，精度向上が図りにくい問題があった．

動的最適化方式のアプローチとして，実行時に動的に質問処理の実行プランを変更する再最適化方式がある[59][56]．文献[59]は，最適化時に，質問処理の実行プランすべてを生成しておき，実行時に統計情報を収集しながら，その結果を参照して，残りの実行プランを再度最適化するものである．また，文献[56]は，最適化時に，実行プランを生成する情報が不足している場合，部分的な実行プランを作成し，実行時に，統計情報を収集しながら，残された処理に対して，最適化を実行するものである．しかし，最初に実行プランが生成できる場合には，文献[56]と同様な処理を行うもので，基本的には，文献[56]と同様

なアプローチである．統計情報の収集は，文献[59]，[56]ともにソフトウェアで処理を実行している．

上記のように，すでに提案されている動的最適化方式では，基本的にソフトウェア処理向けであり，実行時に再度実行プランを生成するためのオーバヘッドがかかる，また，統計情報の収集をソフトウェアで実行しているため，実行時の負荷がかかり，処理速度の低下を招いている．本研究の RINDA 最適化においては，RINDA ハードウェアの実行結果を処理途中で参照することができるため，この情報を利用し，実行時に一度だけ，アクセスパスの選択を行う，精度の高い動的最適化を実現している．また，動的最適化においては，動的最適化時に参照する統計情報が重要であるが，RINDA では，RINDA ハードウェアの機能を利用しており，統計情報の収集なしでの動的最適化を実現可能である．

2.4.2 メモリ常駐型リレーショナル DBMS 研究の特長

ネット･ワークサービス向けに開発したメモリ常駐型リレーショナル DBMS である RENA（Real-time database management system）は，本格的なメモリ常駐型の商用システムである．以下，RENA の設計で採用または提案した技術の位置付けを述べる．

（1） メモリデータベースシステムとしての位置付け

RENA の適用分野は，ネットワークサービスであり，主にショートトランザクション向けのシステムである．この特性を備えた，システムとしては，System M と Times Ten が候補となる．System M は，リアルタイム OS である MachOS 向けに開発された実験システムであり，2 フェーズコミット，ファジーチェックポイントによるリカバリ手法，固定サイズのセグメント分割，インデックスバックアップなしによる新たな研究が行われている．しかし，ネットワークサービスでは，単一レコードアクセスの高速化が必要であり，さらに，サービス無中断での DB 保守が求められるため，ユニーク・キーアクセスの高速化と DB 保守方法の改善が必要となる．

Times Ten は，本格的な商用メモリ DBMS であり，DK-DBMS との接続，SQL，ODBC，JDBC インタフェース，ハッシュインデックス，T-tree インデックス，2 フェーズコミッ

ト，チェックポイント，レプリケーションを実現している．しかし，SQLをフル仕様で実現しており，ユニーク・キーアクセスに向けた処理速度の高速化には限界がある．また，ストアードプロシージャ等の定型処理に向けの高速化手法は，DK-DBMSに依存しており，高速化の余地がある．また，DB 保守については，レプリケーションを利用した手法を実現しているが，サービス無中断でのDB保守は実現できていない．

RENAは，これらのシステムと比較し，ユニーク・キーアクセスの高速化を改良拡張ハッシュで，並行制御はスピンロック方式で，リカバリ制御はファジーチェックポイント方式で，言語処理はサブセットSQLとコンパイルドSQL方式で，さらに，サービス無中断のDB保守方式は，新旧テーブルの切替え方式で実装した実システムであり，高速性，可用性の向上を可能としている．

（2） メモリデータベース技術の位置付け

（a）データ構造

メモリ上で，固定長データは配列構造で，可変長データに対しては，ポインタを利用した手法を採用している．ファジーチェックポイント方式を採用しているため，メモリ空間をブロック単位に分割する必要はない．

（b）アクセス法

ユニーク・キーアクセス向けの手法として，改良拡張ハッシングを採用している．拡張ハッシングのアルゴリズムの提案は行われているが，商用システムでの実装の例はなく，初めての実装である点に特徴がある．

（c）言語処理

言語処理では，ユニーク・キーアクセスが主体である点に着目し，SQLのサブセット仕様を新たに提案している．また，言語の実行時処理として，従来システムで採用されているプリコンパイル方式をベースに，実行処理コストを大幅に削減できるコンパイルドSQL方式を提案している．最適化に関しては，メモリ上のデータ配置特性とSQLの発行順序特性を利用した新しい最適化を提案している．これらの各手法は，RENAで新たに提案した方式であり，処理速度の向上に寄与している．

（d）並行制御

メモリデータベースの並行制御は，メモリ上の資源（レコード等）上にロック管理情報を保有する方式が主体であり，RENA においても，同様にレコード上にロック管理情報を保有している．また，ロックの単位は，メモリ上での処理速度が速いため，リレーション単位が多いが，RENA では，同時走行性をさらに向上させるため，レコード単位を採用している．ロック制御は，ウェイトによる CPU 処理コストを低減させるため，スピンロックを採用しているが，さらなる性能向上のため，インデックスのロック制御において，ファジーロック処理により，不要なロック処理の削減を図っている点に特徴がある．

（e）リカバリ制御

リカバリ制御は,すでに提案されているファジーチェックポイント方式を採用しており，ログ取得量の削減を図っている．

（f）データベース保守

データベースの再編成，再構成を実施するデータベース保守においては，従来は，DB アクセスを一旦停止し，その間にデータベース保守を行う方法が採られており，サービス無中断でのデータベース保守方式はない．RENA では，メモリデータベースの保守方式として，新旧テーブルの切替え方式を提案し，実装している．本方式では，オンラインサービスを停止することなく，データベース保守が可能であり，24 時間サービスの継続が可能である．

上記のように，本研究は，既存の技術を実装しただけでなく，新たに提案した技術を実装し，本格的な DBMS の実現技術を明らかにしたものである．

第 3 章

データベースプロセッサ RINDA の最適化方式

3.1 はじめに

リレーショナルデータベース処理の高速化を目指して開発したデータベースプロセッサ RINDA(Relational Database Processor)は，内容検索を主に実行する CSP（内容検索プロセッサ）と，ソート処理を主に分担する ROP（関係演算プロセッサ）から構成され，これらを使用してデータベースの非定型検索処理の高速化を達成している[29][37][46][47][50][51][89][90][91]．CSP，ROP の制御は，RINDA 制御プログラムで行われ，その制御方式は，CSP，ROP の処理結果である複数の行を一時時的な格納空間である一時テーブルの形で引き継ぎながら処理する一時テーブル方式で実現している[48][49]．本論文では，RINDA 制御プログラムで実現した実行時にアクセスパスおよび関係演算アルゴリズムを決定する動的最適化方式を示す．

従来開発されている最適化方式の代表的なものに，System R のコスト評価方式による静的最適化方式がある[92]．この方式は，データベースの状態に関する統計情報，例えば，テーブル中の行数，列の値の分布等を利用するものであるが，一般に一つの問合せは複数のオペレーションからなり，オペレーション間で引き渡される中間結果を予測することには，限界があり，精度向上が図りにくい問題があった．

一方，RINDA では，CSP，ROP の処理結果が，次の処理に移る前に参照できる利点があることから，実行時にアクセスパスおよび関係演算アルゴルズムを選択する動的最適化方式を用いることができる．動的最適化方式では，正確な情報に基づいて次のアクセスパスおよび関係演算アルゴリズムを決定できるため，予測に基づいて事前にアクセスパスおよび関係演算アルゴリズムを決定する静的最適化方式に比べて，処理速度を向上させることができる．

動的最適化方式は，複数のテーブルを処理する結合処理，副問合せ処理に対して適用される．質問処理実行時には，事前に CSP，ROP で処理された結果に情報（行数等）に基

づき，次のアクセスパスおよび関係演算アルゴリズムを決定する．このためのオーバヘッドは，ファイルへのI/Oはなく，単にメモリ上の情報の判定とアクセス条件の変換であり，全体の処理に比べてほとんど無視できる．

動的最適化方式のアプローチとして，実行時に動的に質問処理の実行プランを変更する再最適化方式がある[59][56]．文献[59]は，最適化時に，質問処理の実行プランすべてを生成しておき，実行時に統計情報を収集しながら，その結果を参照して，残りの実行プランを再度最適化するものである．また，文献[56]は，最適化時に，実行プランを生成する情報が不足している場合，部分的な実行プランを作成し，実行時に，統計情報を収集しながら，残された処理に対して，最適化を実行するものである．しかし，最初に実行プランが生成できる場合には，文献[59]と同様な処理を行うもので，基本的には，文献[59]と同様なアプローチである．統計情報の収集は，文献[59]，[56]ともにソフトウェアで処理を実行している．一方，RINDA においては，実行プランの生成時には，動的最適化の対象については，最適化を行わず，実行時のみに行うため，再最適化の必要性がない．また，統計情報の収集は，RINDAの機能を利用しており，処理速度を向上させることができる．

本論文では，利用頻度の高い結合処理に関する動的最適化方式とその実測評価結果を示すとともに，副問合せに関する最適化方式を示す．

以下，3.2でRINDAの関係演算処理方式を説明し，3.3でRINDAの最適化方式の概要を，3.4で静的最適化方式を，3.5で結合処理時の動的最適化方式を，3.6で副問合せの最適化方式について示し，最後に3.7で結合処理時の評価結果を示す．

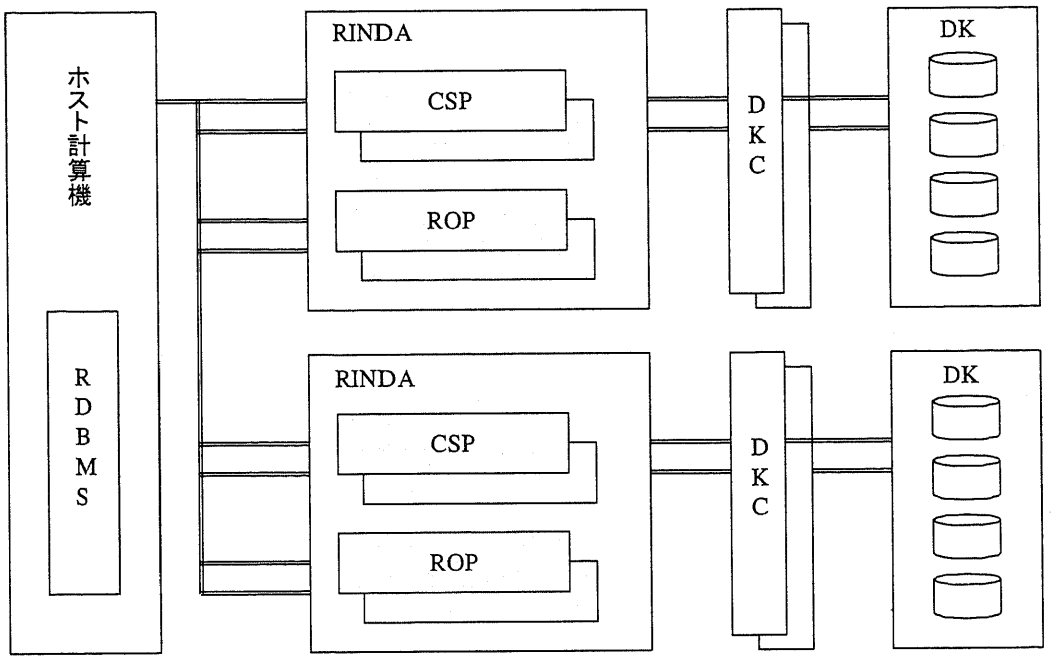
3.2 RINDAの関係演算処理

3.2.1 ハードウェアの概要

RINDAで新たに開発したハードウェアの各機能とシステム構成を示す.表3.1にCSP, ROPの機能の概要を示す. CSP, ROPは, ホスト計算機とチャネルで結合されるもので, ホスト計算機から制御を行う. システム構成を図3.1に示す.

表3.1 RINDAハードウェアの主要機能

機能			説明
CSP	述語判定	比較述語	<列><比較演算子><定数> の条件判定.
		IN述語	<列>[NOT] IN <定数リスト> の条件判定.
		LIKE述語	<列>[NOT] LIKE <パタン> の条件判定.
		NULL述語	<列> IS [NOT] NULL の条件判定.
	検索条件判定		述語のAND/ORによる任意の論理式判定.
	出力列の抽出		上記検索条件を満たす行から任意の列の出力.
	集合関数		上記検索条件を満たす行数のカウント(COUNT(*)).
ROP	ソート		指定された列の値(ソートキーと呼ぶ)による昇順または降順への行の並べ替え. ソートキーは任意の順序の複数列により構成可能.
	ふるい落とし		ソートキーの値による2つの表からの結合可能性のない行の削除. 一方または両方の表からの除去が可能.
	重複計数/除去		ソートキーの値が重複する行数のカウント, および2番目以後の行の除去.



CSP:内容検索プロセッサ, DKC:ディスク制御装置
ROP:関係演算プロセッサ, DK :ディスク装置
RDBMS:関係データベース管理システム

図 3.1 RINDA システムの概要

3.2.2 関係演算処理

RINDA の関係演算は，CSP，ROP とホスト計算機の CPU で分担して実現される．また，CSP，ROP とホスト計算機とは，一度に複数の行が格納されたページ単位で転送が行われる．ホスト計算機上には，CSP，ROP の入出力データを蓄積する入出力バッファが用意され，行は常にこのバッファを経由して，CSP，ROP との転送が行われる．このバッファは，検索処理の中間結果を格納しておく一時的なテーブルであり，データベースの永続的なデータが格納されたテーブル（以下，基本テーブルと呼ぶ）とは，別に管理されている．中間結果のサイズが大きい場合には作業用のディスクを併用する．

この一時テーブルを併用した基本的な演算は，以下の手順で実行される．

（1）CSP を用いて基本テーブル全体を一括してサーチし，その結果をすべてホスト計算機上の一時テーブルに蓄積する．

（2）以後は，一時テーブルを入出力単位として CSP／ROP によるソート，結合演算等を演算単位に逐次実行する．

代表的な処理の流れを図 3.2 に示す．

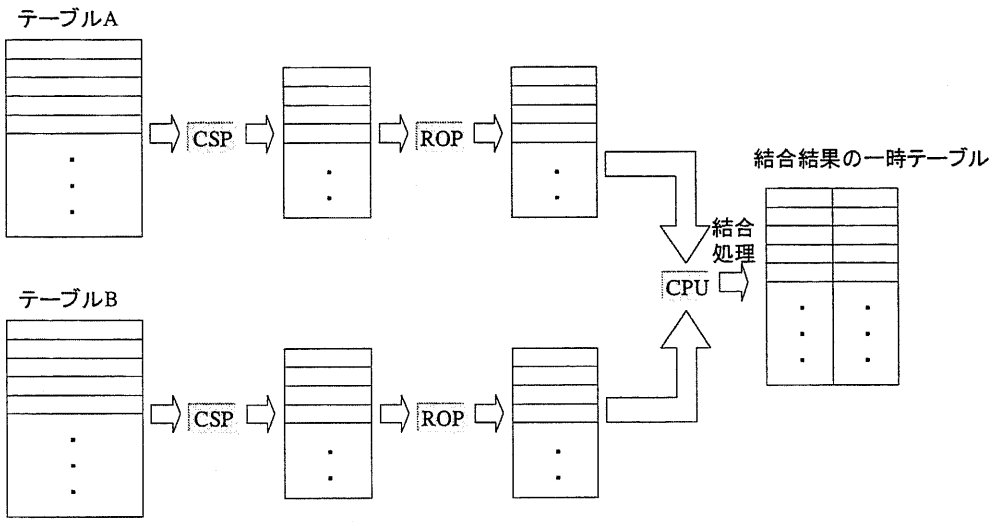


図 3.2　RINDA の関係演算処理方式

3.3 RINDA最適化方式

3.3.1 従来の静的最適化方式の問題点

（1）最適化が低精度

従来の代表的な静的最適化方式であるコスト評価方式は，I/O回数をベースとした実行時間予測を行い，最も実行時間が少ないと想定されるアクセスパスおよび関係演算アルゴリズムを選択している．静的最適化方式の場合，実行時間の予測に用いられる途中結果の行数等が予測よって求められており，実行時間が正確に見積もれないという問題がある，

（2）ハードウェアの有効利用が不可能

結合処理等では，CSP アクセスにより検索された結果の行数が少ない場合，次の基本テーブルへのアクセスにおいて，IN述語を利用してCSPによる条件検索が可能であるが，静的最適化方式では，行数が正確に予測できないため，CSPを有効に利用できない．

3.3.2 RINDA最適化の着眼点

3.2で示したRINDAの処理方式の特徴は，CSP，ROPの出力結果の情報（行数等）を知ることができるため，次の処理に利用する情報を正確に知ることができる点である．この点に着目し，RINDA では，複数のテーブルを操作する関係演算（結合処理，副問合せ処理）において，事前にアクセスした結果情報を利用して次のアクセスパスをおよび関係演算アルゴリズムを決定する動的最適化方式を実現している．動的最適化方式により，正確な情報によるアクセスパスおよび関係演算アルゴリズム選択が可能となり，ハードウェアを有効に利用でき，全体の処理時間短縮が実現できる．

3.3.3 RINDA最適化方式の概要

RINDA最適化は，すべての処理を動的最適化で行う訳ではなく，最初の基本テーブル

へのアクセスパス選択および残りのテーブルへのアクセス順序の決定は，静的最適化で行い，実行時のオーバヘッドを抑えている．静的最適化と動的最適化の基本的な分担を以下に示す．

（1）静的最適化

SQL の言語解析時に行われるもので，以下の処理を行う．

（a）検索対象の基本テーブルが一つの場合のアクセスパスの決定．

（b）複数テーブルを対象とする場合の基本テーブルへのアクセス順序ならびに最初の基本テーブルへのアクセスパスおよび関係演算アルゴリズムの決定．

（2）動的最適化

SQL 実行時に行われるもので，複数テーブルを対象とする場合の 2 番目以降の基本テーブルへのアクセスパスおよび関係演算アルゴリズムを決定する．

3.4 静的最適化方式

静的最適化では3.3.3で示したように，一つの基本テーブルへのアクセスパスの決定と複数テーブルへのアクセス時のアクセス順序を決定する．

RINDAにおける静的最適化は以下の順序で行われる．

（1）RINDA処理とソフト処理の決定

RINDA制御プログラムは，図3.3に示すように，ソフト処理とRINDA処理とから構成される．ソフト処理は，更新系の処理とユニーク・キーを利用した1行検索等で利用され，RINDA処理は，大量データの検索等で利用される[41]．

最適化は，まず，共通最適化にて，RINDA処理とソフト処理の選択が行われる．共通最適化は，機能判定と性能判定の順で実行される．機能判定は，SQLの種類により，定義・更新用SQLはソフト処理とし，検索用SQLを性能判定に引き継ぐ．性能判定では，以下の判定基準に基づき，性能上最適なアクセスパスを決定する．

（a）原則は，RINDA処理とし，明らかに処理コストが小さいと判定できる場合のみをソフト処理とする．

（b）処理コストとして，I/O回数を評価する．

（c）インデックスのI/O回数は，通常，インデックスはメモリ上に常駐されることから，除外する．

（d）共通最適化はインデックス定義情報，ソフト処理アクセス法，およびSQLから得られる情報のみを用いて行う．

本論文では，RINDA処理が選択された場合の最適化を示す．

（2）テーブルアクセス順序の決定

作成される一時テーブルの大きさと個数を最小にすることが性能上重要であり，検索結果の行数が少ない（一時テーブルの大きさが小さい）と想定されるテーブルの順に結合順序ならびに副問合せ順序を決定する．

副問合せにおいては，主問合せ側の限定述語を比較述語，IN述語に変換する．変換方

式については，3.6.3 副問合せ最適化方式に示す．

（3）最初の基本テーブルへのアクセスパスの決定

条件式に基づき，最初の基本テーブルにアクセスする CSP の検索命令を決定する．

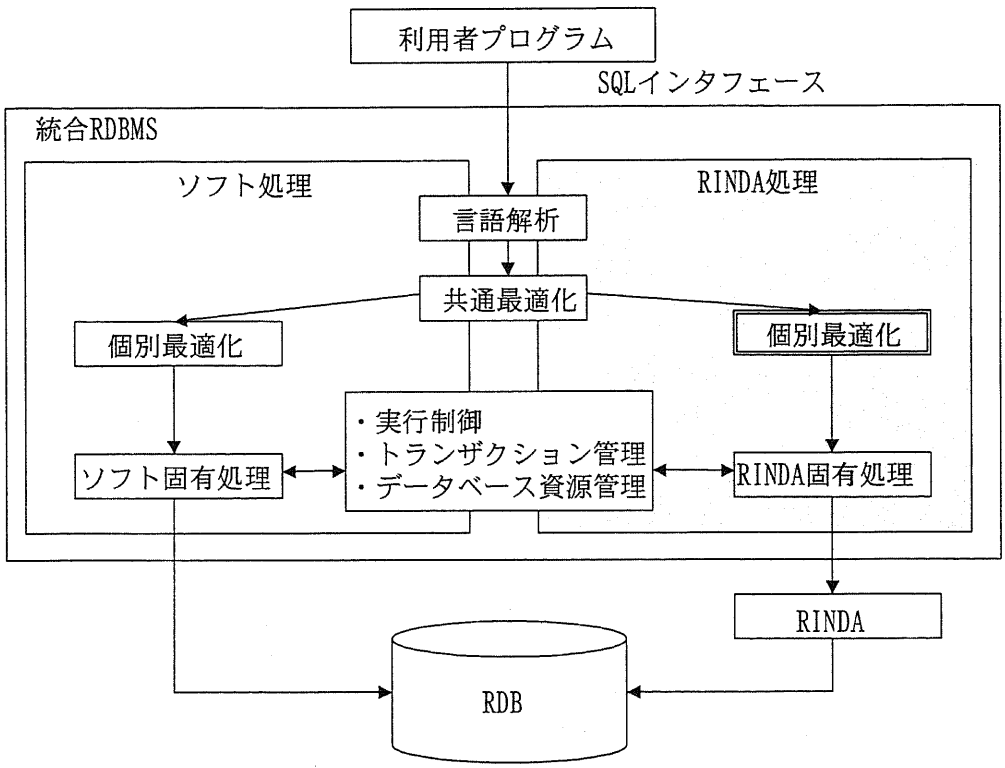


図3.3　DBMSのソフトウェア構成

3.5 結合処理動的最適化方式

3.5.1 結合処理方式

RINDA の結合処理は，CSP，ROP の特性を活かし，以下に示す結合処理方式を実現する．

3.5.1.1 結合処理の基本手順

RINDA の結合処理は，以下の基本手順で実行する．（図 3.4）

[STEP1]:最初の基本テーブルへ CSP によりアクセスし,結果を一時テーブルに保存する．

[STEP2]：次の基本テーブルへ CSP によりアクセスし，結果を一時テーブルに保存する．

[STEP3]：一時テーブル同士の結合処理を行い，結果を新たな一時テーブルに保存する．

[STEP4]：次の基本テーブルがなければ処理を終了する．次の基本テーブルがあれば，STEP3 で作成した一時テーブルを STEP1 で作成した一時テーブルとみなし，STEP2 に戻る．

本基本手順においては，CSP，ROP の特性は，以下のように利用できる．

（1）上記の STEP2 において，CSP の述語判定機能を利用する．すなわち，直前にアクセスした結果の一時テーブルの結合キー値を比較述語，IN 述語の定数,定数リストに展開した条件によりアクセスし，検索結果を絞り込むことにより，結合処理対象を削減でき，処理時間を短縮できる．

（2）STEP3 において，結合処理時に，ROP のふるい落とし機能を利用して結合処理対象を削減することにより処理時間を短縮できる．

以下，CSP，ROP の有効利用法を示す．

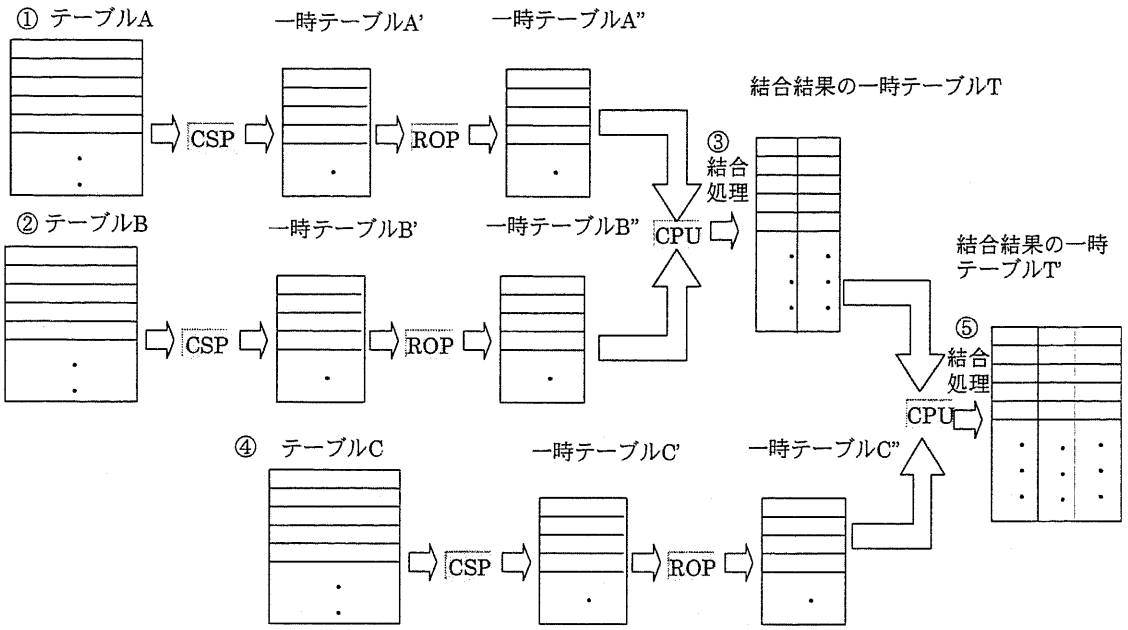


図3.4　RINDA結合演算の基本手順

3.5.1.2 CSP アクセスでの条件式変換方式

2 番目以降の基本テーブルへのアクセス時，CSP の条件は，その基本テーブルに係わる条件式のみでのアクセスだけではなく，直前にアクセスした結果の一時テーブルの内容を考慮して，新たな条件を生成してアクセスすることにより，より絞り込み効果が高くなる．結合処理の場合，以下の 3 種類のアクセスが考えられる．

（1）基本テーブルに関する結合キーに対する条件を除いた条件式によるアクセス．この場合，結合条件以外の条件を満足する行が検索され，結合キー値を利用した絞り込みは期待できない．

（2）結合以外の条件に，結合キーに関する比較述語（列名　比較述語　定数）を付加した条件によるアクセス．結合キーに対する値がただ一つの場合に適用でき，絞り込み効果が最も期待できる．

（3）結合以外の条件に，結合キーに対する IN 述語（列名　IN　（定数，定数，…））を付加したアクセス．IN 述語の指定できる条件内に結合キー値が収まる場合に有効であり，（2）と同様，絞り込み効果が期待できる．

3.5.1.3 一時テーブル間の結合処理方式

二つの一時テーブル間の結合処理は，基本的に，ROP のふるい落とし機能を使用するが，2 つの一時テーブルの行数，サイズにより，ROP のふるい落としの効果が異なる．そのため，ROP のふるい落とし機能の使用法別に以下の 3 種類の結合方式を実現している．

（A）ネステッドループ結合方式

一方のテーブルの一つの結合キー値に対し，もう一方のテーブルの結合キーの先頭から順に比較を行い，次に，最初のテーブルの次の結合キー値についても同様，もう一方のテーブルの結合キーの先頭から順に比較を行う．これを最初のテーブルのすべての結合キーについて行う．

本方式は，ROP を利用せず，CPU 上のみで処理を行う方法であり，一方のテーブルの行数が 1 行か，双方の行数が少ない時に有効である．

（B）ソートマージ結合方式

一般に知られているソートマージ結合方式のうち，ソート時に，ROP のふるい落とし

機能を利用して結合対象行を減少させてマージ結合を行う方式であり，以下の２種類がある．

（Ｂ－１）片ハッシュソートマージ結合方式

以下の処理手順で実行される．

[STEP1]：一方の一時テーブル（T1）の行すべてを，ROP に設定モードで送信する．

[STEP2]：ROP では，ソートキー値（結合キー値）に基づいて，ハッシュ関数により値を求め，その値をビットアレイに記憶しておく．同時に結合キー値に基づくソートを行い，結果を CPU に返す．CPU では，ソートされた結果を一時テーブル（T11）に保存する．

[STEP3]：もう一方の一時テーブル（T2）の行すべてを，ROP に参照モードで送信する．

[STEP4]：ROP では，前回と同様ソートキー値（結合キー値）に基づいてハッシュ関数による計算を行い，設定ソート処理で記憶していた値と同じ行のみを残す．同時に，残された行について，結合キー値に基づきソートを行い，結果を CPU に返す．CPU では，ソートされた結果を一時テーブル（T21）に保存する．

[STEP5]：一時テーブル T11，T21 の行を結合キー値でマージ比較を行い，最終結果を得る．

片ハッシュソートマージ結合方式の処理イメージを図 3.5 に示す．

本方式は，一方のテーブルの絞り込みが行われるため，２つのテーブルのうち一方の行数が少ない場合，少ない側のテーブルを設定ソートし，多い側のテーブルを参照ソートすることにより，絞り込み効果が高くなる．また，ROP については，一度の転送で処理できる量には限界があるため，この限界を超える場合（オーバフローが生じる場合），複数回の ROP への転送制御が必要になる．そのため，少ない側のテーブルの大きさが ROP でオーバフローを生じない範囲である場合に有効である．

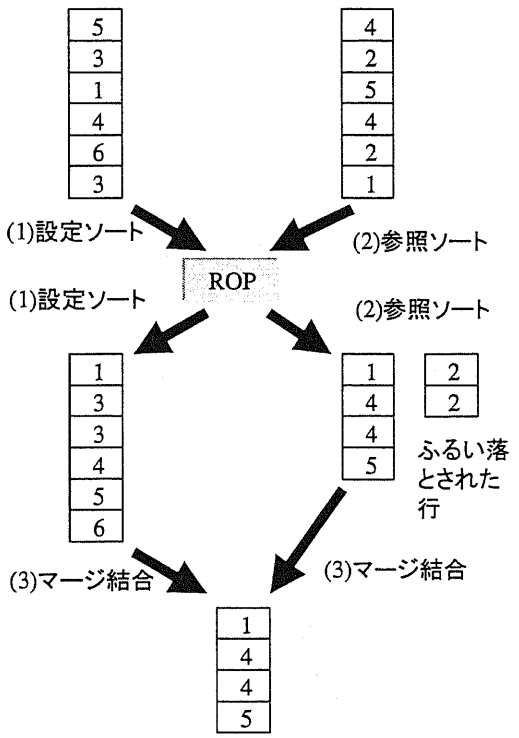


図 3.5　片ハッシュソートマージ結合方式

（Ｂ－２）両ハッシュソートマージ結合方式

以下の処理手順で実行される．

[STEP1]：一時テーブル T1 の全ての行を，ROP の設定モードで送信する．ROP では，結合キー値に基づいて，ハッシュ関数により値を求め，その値をビットアレイに記憶しておく．

[STEP2]:もう一方の一時テーブル T2 の全ての行を,ROP に参照設定モードで送信する．

[STEP3]：ROP では，結合キー値に基づいてハッシュ関数による計算を行い，設定時に記憶していたものと同じ行のみを残す．同時に，残された行について，結合キー値に基づきソートを行うとともに，新たなビットアレイの設定を行い，結果を CPU に返す．CPU では，結果を一時テーブル T21 に保存する．

[STEP4]：再度一時テーブル T1 の全ての行を．ROP に参照モードで送信する．

[STEP5]：ROP では，結合キー値に基づきハッシュ関数による計算を行い，参照設定ソート時の値と比較しながら，ふるい落としを行い，ハッシュ値が一致した行のみを残す．同時に残された行について，結合キー値によるソートを行い，結果を CPU に返す．CPU では，結果を一時テーブル T11 に保存する．

[STEP6]：一時テーブル T11，T21 の行を結合キー値でマージ比較を行い，最終結果を得る．

両ハッシュソートマージ結合処理方式の処理イメージを図 3.6 に示す．

本方式は，両方のテーブル中の行を絞り込めるため，両方のテーブル中の行数が多い場合に有効である．この時，最初の設定ソートでは，行数が少ないテーブルを使用することにより，参照設定ソートの処理結果の行数が少なくなり，処理時間を短縮できる．

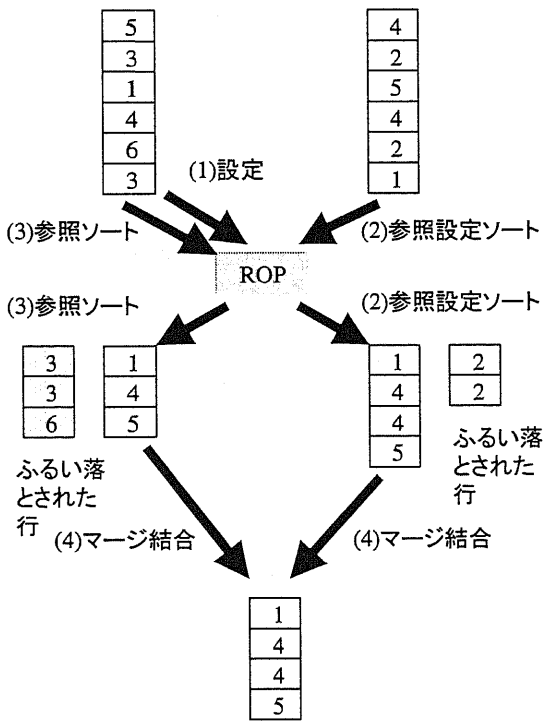


図 3.6　両ハッシュソートマージ結合方式

3.5.2 動的最適化方式

動的最適化では，2 番目以降にアクセスする基本テーブルへのアクセス条件の決定と，アクセスした結果の一時テーブル同士の結合処理方式を実行時に決定し，処理時間の短縮を図る．そのため，以下の考えに基づいてアルゴリズムを考えた．

（1）基本テーブルへのアクセスは，直前にアクセスされた結果の一時テーブル内の結合キー値の数により，より絞り込む効果の高いアクセス条件に条件式変換を行った条件で行う．これにより，アクセスした結果の一時テーブルの行数を少なくでき，処理時間の短縮が可能となる．

（2）結合処理についても，CPU 上での演算時間を最小にできる方法を選択する．選択の判定は，各結合方式の特徴に合わせて，一時テーブル内の行数と一時テーブルのサイズにより行う．

上記の考え方に基づいた動的最適化アルゴリズムを以下に示す．

[STEP1]：1 番目の基本テーブル B1 に CSP によりアクセスし，検索結果を一時テーブル T1 に保存する．

[STEP2]：T1 の行数を参照し，表 3.2 に従って，2 番目の基本テーブル B2 へのアクセス条件を決定する．

[STEP3]：B2 へのアクセス結果を一時テーブル T2 に保存する．

[STEP4]：T1 と T2 の行数，サイズを参照し，表 3.3 に従って結合方式を決定する．

[STEP5]：STEP4 で決定された結合方式に従って結合処理を行い，結果を一時テーブル T3 に保存する．

[STEP6]：全ての基本テーブルの結合が終われば，処理を終了する．結合対象の基本テーブルが残っていれば，次の結合対象の基本テーブルを B2 と，T3 を T1 と読みかえて，STEP2 に戻る．

動的最適化の実行例を図 3.7 に示す．

表 3.2　2 番目のテーブルへのアクセス方式決定方法

一時テーブルの行数	アクセス方式
1	他の検索条件に、結合キーに対する比較述語(列名 比較演算子 定数)を付加してテーブルにアクセスする
IN述語内の定数数の制限内	他の検索条件に、結合キーに対するIN述語(列名　IN (定数, 定数, ・ ・ ・))を付加してテーブルにアクセスする
上記以外	結合キーに対する条件を除いた条件でテーブルにアクセスする

表3.3　結合方式判定方法

判定条件	結合方式
C(T1)=1 OR C(T2)=1 OR (C(T1)*C(T2)<=n1 AND (S(T1)<=n2 AND S(T2)<=n2))	ネステッドループ結合方式
MIN(S(T1),S(T2))<O(ROP) AND (C(T1)/C(T2)<=n3 OR C(T2)/C(T1)<=n3)	片ハッシュソートマージ結合方式 ・C(T1)>C(T2) 設定ソート:T2, 参照ソート:T1 ・C(T1)<=C(T2) 設定ソート:T1, 参照ソート:T2
上記以外	両ハッシュソートマージ結合方式 ・C(T1)>C(T2) 設定:T2, 参照設定ソート:T1, 参照ソート:T2 ・C(T1)<=C(T2) 設定:T1, 参照設定ソート:T2, 参照ソート:T1

C(T1):一時テーブルT1の行数, C(T2):一時テーブルT2の行数, n1,n2,n3:定数
S(T1):一時テーブルT1のページ数, S(T2):一時テーブルT2のページ数
Min(T1,T2):T1,T2のいずれか小さい方, O(ROP):ROPのオーバフロー条件

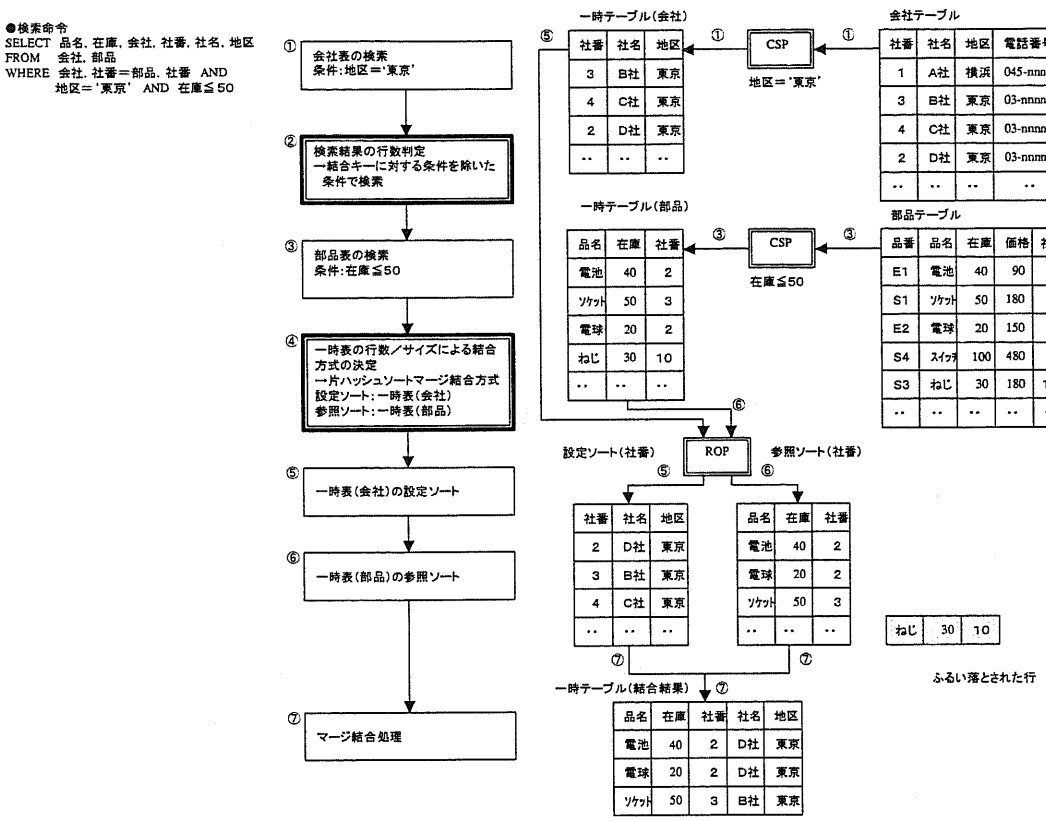


図 3.7　結合処理の動的最適化例

3.6 副問合せ最適化方式

3.5 では，結合処理の動的最適化方式を示したが，複数テーブルを扱う処理に副問合せがある．副問合せにおいても，CSP，ROP を有効に利用する動的最適化を実現できる．本章では，副問合せの従来方式の問題点と，RINDA での動的最適化による解決方法を示す．

3.6.1 従来方式の問題点

従来の DBMS はソフトウェアにより限定述語を以下のように処理した．（図 3.8）

（1）副問合せ側のテーブルを検索し，結果を作業域に格納する．

（2）主問合せ側のテーブルで条件を満足する各行について，作業域から副問合せ結果を1行ずつ取り出し限定述語の条件を満足するか判定する．

この方法では，副問合せ結果の要素値すべてに対して，比較演算子の条件を満足するかどうかを判定するため，主問合せ側のテーブルの行数を M，副問合せ結果の要素値の件数を N とすると，限定述語の判定時間は M*N のオーダとなり，処理時間が長くなるという問題があった．

●検索命令
SELECT *
FROM 会社
WHERE 地区＝'東京' AND 社番 ＝ANY (SELECT 社番
FROM 会社
WHERE 在庫＜＝50)

●処理の流れ
①部品テーブルの検索
条件:在庫＜＝50
②会社テーブルの検索
条件:地区＝'東京'
を満足する各行について社番が作業域に存在するか比較

●データの流れ

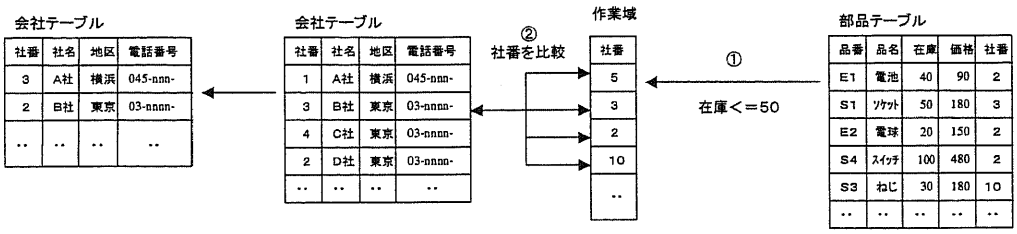


図3.8　従来方式による副問合せ処理方式

3.6.2 RINDA 利用の着眼点

限定述語の判定には，列の値（主問合せのテーブル）と複数の値（副問合せの結果）との比較が必要である．このため，RINDA の以下の機能を利用することにより，高速に処理することが可能である．

（1）CSP の IN 述語判定機能を用いて，列の値と複数の値を比較する．

（2）主問合せの列と副問合せ結果の比較回数を減らすためには，両者をソートしておくことが有効であり，ROP のソート機能を利用する．

3.6.3 副問合せ最適化方式

限定述語を含む問合せの場合には，副問合せ側のテーブル，主問合せ側のテーブルの順で検索を行う．後者については，SQL 解析時または実行時に CSP またはソフトウェアで行うかを判断する．以下に，RINDA にて実現している副問合せ処理方式を示す．

（1） 比較述語変換方式

表 3.4 に示すように比較述語に変換し（SQL 解析時），副問合せ結果の複数の要素の値をその最大値，または最小値に置換し（実行時），主問合せ側のテーブルの検索条件に組み込み検索を行う．

表 3.4　限定述語の変換方式

No	限定述語	変換後の比較述語
1	>ANY　副問合せ	>MIN
2	>ALL　副問合せ	>MAX
3	<ANY　副問合せ	<MAX
4	<ALL　副問合せ	<MIN
5	>=ANY　副問合せ	>=MIN
6	>=ALL　副問合せ	>=MAX
7	<=ANY　副問合せ	<=MAX
8	<=ALL　副問合せ	<=MIN

（注）MIN：副問合せ結果の要素の最小値

MAX：副問合せ結果の要素の最大値

（2） IN 述語変換方式

限定述語を IN 述語に変換し（SQL 解析時），副問合せ結果の要素を IN 述語に展開し（実行時），主問合せ側のテーブルの検索条件に組み込み検索を行う．（図 3.9）

=ANY　副問合せ　　→　IN( )

<>ANY　副問合せ　　→　NOT　IN( )

ここで，( )の中は，副問合せ結果の要素値が列挙されていることを示す．

（3） ソートマージ方式

（a）限定述語を除いた検索条件で主問合せ側のテーブルを検索し，一時テーブル T2 に格納する．

（b）副問合せの結果が格納されている一時テーブル T1 と T2 をソートし，ソート結果をマージすることにより，限定述語を満足する T2 の行を出力する．

（4） ネステッドループ方式

(a) 限定述語を除いた検索条件で主問合せ側のテーブルを検索し，一時テーブル T2 に格納する．

(b) 副問合せの結果が格納されている一時テーブル T1 と T2 をネステッドループにより比較し，限定述語を満足する T2 の行を出力する．

上記の副問合せ処理方式の選択方法を表 3.5 に，副問合せ動的最適化の例を図 3.9 に示す．

表 3.5　副問合せ処理方式選択法

比較演算子	限定子	判断条件	判断時期	副問合せ処理方式
>,<,>=,<=	ANY,ALL		SQL 解析時	比較述語変換方式
=,<>	ANY	副問合せ結果を IN 述語に展開できる	実行時	IN 述語変換方式
=	ANY	副問合せ結果を IN 述語に展開できない	実行時	ソートマージ方式
上記以外			SQL 解析時	ネステッドループ方式

以上のように，副問合せにおいても，比較述語変換方式と RINDA ハードウェアの機能を利用する IN 述語変換法方式，ソートマージ方式を，動的最適化を含む最適化方式により選択することにより，処理速度を向上することができる．

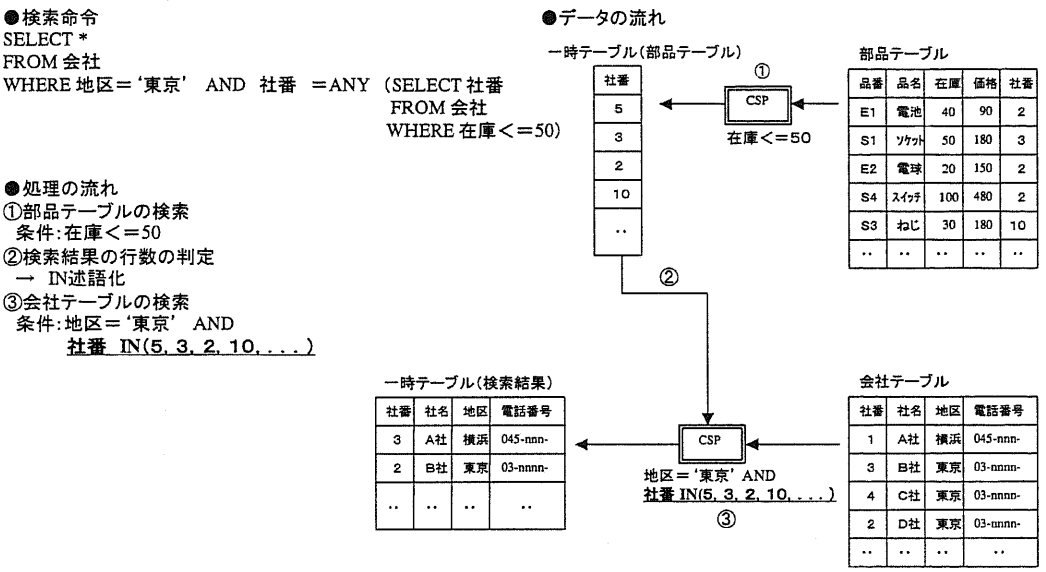


図3.9　副問合せ動的最適化方式の例

3.7 性能評価

RINDAを使用しない場合（RINDAなし）と使用する場合（RINDA有り）との性能比較およびRINDAにおける動的最適化の効果を確認するために，結合処理に関する性能実測を行った．以下，評価モデルと評価結果を示す．

3.7.1 評価モデル

（1）ＤＢモデル

拡張ウィスコンシンＤＢモデル[21] を使用した．テーブルは行数が 1,000，10,000，100,000 の 3 種類で，行長は 208 バイトである．また，結合キー長は 4 バイトである．

（2）問合せモデル

拡張ウィスコンシン DB モデルの結合処理に関する問合せをもとに，表 3.6 に示す問合せ処理を実測した．表 3.6 の中で，モデルＡは，RINDA なしの場合 RINDA 有りの場合との性能比較を行うもので，2 つのテーブル間と 3 つのテーブル間の結合処理の性能を実測するものである．なお，RINDA なしの場合，性能向上効果を明らかにするため，インデックスは使用していない．モデルＢは，RINDA の動的最適化の効果を確認するもので，2 つのテーブル間の結合処理において，出力件数を 10 件から 100,000 件に変化させた場合の性能を実測するためのものである．

表 3.6　性能評価に用いた SQL 文

モデル		問合せ種別	SQL文
モデルA	M-A11	A sel B	SELECT * FROM TENKA A, TENKB B WHERE A.UNIQUE1=B.UNIQUE1 AND A.UNIQUE1<1000
	M-A12		SELECT * FROM HUNKA A, HUNKB B WHERE A.UNIQUE1=B.UNIQUE1 AND A.UNIQUE1<100000
	M-A13		SELECT * FROM THUKA A, THUKB B WHERE A.UNIQUE1=B.UNIQUE1 AND A.UNIQUE1<100000
	M-A21	C sel A sel B	SELECT * FROM ONEKA C, TENKA A,TENKB B WHERE C.UNIQUE1=A.UNIQUE1 AND A.UNIQUE1=B.UNIQUE1 AND A.UNIQUE1<1000 AND B.UNIQUE1<1000
	M-A22		SELECT * FROM TENKA C, HUNKA A,HUNKB B WHERE C.UNIQUE1=A.UNIQUE1 AND A.UNIQUE1=B.UNIQUE1 AND A.UNIQUE1<10000 AND B.UNIQUE1<10000
	M-A23		SELECT * FROM HUNKA C, THUKA A,THUKB B WHERE C.UNIQUE1=A.UNIQUE1 AND A.UNIQUE1=B.UNIQUE1 AND A.UNIQUE1<100000 AND B.UNIQUE1<100000
モデルB		A sel B	SELECT * FROM HUNKA A, HUNKB B WHERE A.UNIQUE1=B.UNIQUE1 AND A.UNIQUE1< n (n: 10 ~ 100000)

3.7.2 性能評価

（1） RINDA有りとRINDAなしの性能比較

図3.10に，モデルAにおける，RINDAなしの場合とRINDA有りの場合の処理時間比を示す．2つのテーブル間の結合処理において，RINDAなしの場合，RINDA有りの場合に比べて，4.3倍から5.2倍処理時間がかかっている．また，3つのテーブル間の結合処理においても，RINDAなしの場合，RINDA有りの場合に比べて，2.6倍から3.7倍処理時間がかかっている．以上より，RINDA による性能向上効果が達成できていることがわかる．

（2）RINDAハードウェアの利用評価

図3.11に，モデルBにおける，絞込み件数ごとの処理時間比を示す．評価結果を以下に示す．

（a）最初のテーブルでの絞り込み件数が増加するに従って処理時間が増加するが，絞り込み件数300件付近で急激に処理時間が増加している．これは，絞り込み件数が少ない場合は，2番目の基本テーブルへのアクセス時，1番目の基本テーブルからの返却結果の行数が少ないため，動的最適化によって，IN述語による条件式が設定できるためである．IN述語による条件式が設定されない場合は，2番目の基本テーブルへのアクセス時の返却結果が多くなり，図3.11中の点線のように処理時間が増加すると考えられる．

（b）絞り込み件数が大きくなった70,000件付近でも，処理時間の増加率が増している．これは，ROPの利用が70,000件付近までは，片ハッシュソートマージ結合方式が選択され，それ以降，両ハッシュソートマージ結合方式に変更されたためである．片ハッシュソートマージ結合方式が利用できない場合，図3.11中の一点鎖線のように少ない絞り込み件数の場合も処理時間が増加すると考えられる．

以上の分析により，正確なデータに基づいて実行時にアクセスパス決定を行う動的最適化により，CSP，ROPの機能が有効に活用されており，効率的な処理が実現できていることがわかる．

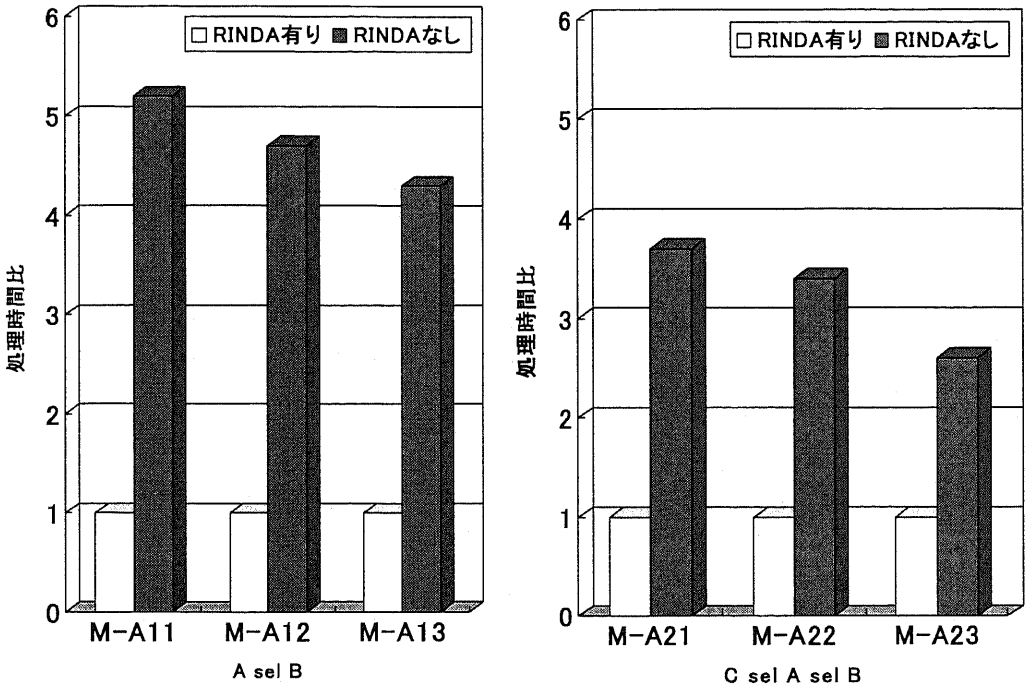


図 3.10　RINDA による処理速度向上効果

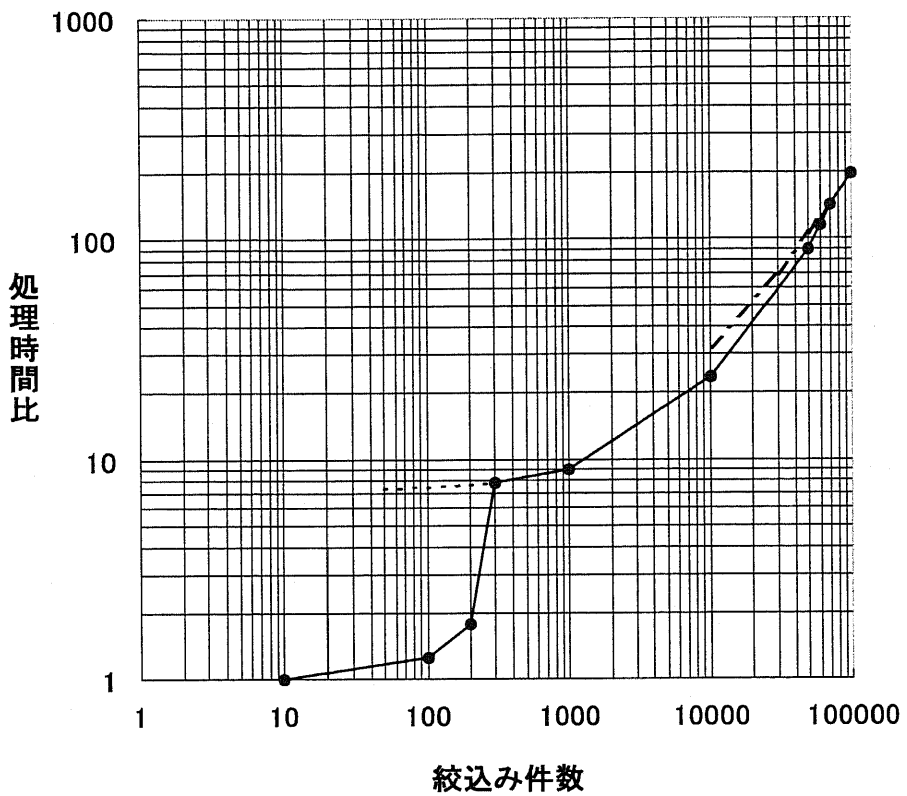


図 3.11　結合処理の処理時間比

3.8 あとがき

データベースプロセッサRINDAにおける動的最適化方式について以下の内容を示した.

（1）RINDAにおける結合処理，副問合せ処理方式として，CSP，ROPのハードウェアの機能を有効に利用する以下の処理方式を示した.

（a）CSPの条件検索機能を利用して，2番目の基本テーブルへのアクセス時に直前にアクセスした結果の行数により，CSPへのアクセス条件を生成し直し，絞り込み効果を高める実行時条件変換方式.

（b）ROPのふるい落とし機能を利用して，結合処理を高速に実行できる，片ハッシュソートマージ結合方式，両ハッシュソートマージ結合方式.

（c）副問合せの限定述語変換方式.

（2）（1）で示した方式を有効に利用できる結合処理および副問合せ動的最適化アルゴリズムを示した.

（3）RINDA有りとRINDAなしの結合処理時間の実測評価をおこない，RINDAを使用することにより，大幅な性能向上が達成できていることを示した．また，2つのテーブル間の結合処理の性能評価を行い，動的最適化方式により，RINDAのハードウェアの機能を有効に利用出来ていることを示した.

第 4 章

メモリ常駐型リレーショナル DBMS の設計と実現

4.1 はじめに

データベース管理システム（DBMS）は，統一的な形式でデータの定義・蓄積・操作を可能とし，アプリケーションプログラムの開発，およびシステム運用の効率化をもたらすことにより，システムの中核パッケージとして重要なものとなっている．DBMS の適用分野も，ビジネス分野から，ネットワークサービス分野へと広がりつつあり，これらの分野においても，利用可能な DBMS の実現が求められている．

ネットワークサービス分野の場合，たとえば，高度電話サービス等の実現においては，サービス変更，追加の容易なソフトウェア構成が必要なことから，データ管理に DBMS を適用する必要性が出てきている．この分野においては，扱うデータ量は少なく，データの操作も簡易であるが，超高トラヒックな処理が必要であり，レスポンス，スループットとも，従来のビジネス向け DBMS と比べて，一桁以上の性能向上（高性能）が必要である．また，24 時間無中断でサービスを継続する必要があるため，従来は計画停止などを必要とした DB 保守処理やバックアップ処理などについても，オンライン処理を中断することなく行うことが必要である．

一方，近年のメモリコストの減少により，全 DB をメモリ上に常駐させるメモリ常駐データベース（メモリ DB）の実現が可能となってきている．メモリ DB では，従来の磁気ディスク(DK)に DB が格納される DBMS(DK-DBMS)で必要であった I/O が不要となり，メモリ上でのみの処理でアクセスが可能となるため，高性能化を図ることができる．

以上の背景から，ネットワーク系のサービスに適用可能な DBMS として，メモリ DB 技術を主体とし，高性能かつ高可用な特徴をもつ RENA（Real-time Database Management System）を開発した．本章では，RENA の設計および実現方式を示すとともに，性能評価を行った結果，従来 DK-DBMS に比べて CPU 処理コストおよびスループットが約 10 倍以上優れていることを示す．具体的には，4.2 で，メモリ常駐型 DBMS

の関連研究を述べ，4.3 で，RENA が開発目標としたネットワークサービス分野での DBMS への要求条件を，4.4 で RENA のアーキテクチャを，4.5 でメモリ DB 記憶構造ならびにアクセス方式を，4.6 で言語処理方式を，4.7 で並行制御方式を，4.8 でリカバリ制御方式を，4.9 でサービス無中断 DB 保守方式を述べ，最後に 4.10 で，RENA の性能評価結果を示し，RENA の有効性を示す．

4.2 関連研究

文献[31]では，メモリ DB に関する研究動向がオーバビューされており，プロトタイプ，商用 DBMS が概観されている．ネットワークサービスへの適用を考えると，ショートトランザクションを主に処理する System M が適している．本システムは，リアルタイム OS である MachOS 向けに作成されたもので，2 フェーズコミットによる並行制御，ファジーチェックポイントによるリカバリ手法，固定サイズのセグメント分割，インデックスのバックアップなしによる新たな研究が行われている．しかし，ネットワークサービスでは，単一レコードアクセスの高速化が必要であり，さらに，サービス無中断での DB 保守が求められるため，ユニーク・キーアクセスの高速化と DB 保守方法の改善が必要となる．最近の商用 DBMS に Times Ten[100]がある．本システムは，本格的な商用 DBMS であり，DK-DBMS との接続，SQL，ODBC，JDBC インタフェース，ハッシュインデックス，T-tree インデックス，2 フェーズコミット，チェックポイント，レプリケーションを実現している．しかし，SQL をフル仕様で実現しており，ユニーク・キーアクセスに向けた処理速度の高速化には限界がある．また，ストアドプロシージャ等の定型処理向けの高速化手法は，DK-DBMS に依存しており，高速化の余地がある．DB 保守については，レプリケーションを利用した手法を実現しているが，サービス無中断での DB 保守は実現できていない．

RENA は，ネットワークサービス向けに，アーキテクチャ，記憶構造，アクセス手法，並行制御，ファジーチェックポイント，SQL 仕様のサブセット化，サービス無中断 DB 保守の各方式を実現した実システムであり，高速性，可用性の向上を可能としている．

4.3 ネットワークサービス分野での要求条件

ネットワークサービス分野において，フリーダイヤルサービスをはじめとする多様な高度電話サービスを，柔軟，かつ迅速に提供するための機構として AIN（Advanced Intelligent Network）が提案され研究が進められている[14][32][86][97]．AIN では，図 4.1 に示されるように，多様なサービスを柔軟に提供するため，回線接続を行う伝達層とサービス制御を行う高機能層を分けた多層構成となっている．例えば，フリーダイヤルサービスにおいては，利用者が論理電話番号（0120-XXXXXX）による発信を行うと，論理番号が伝達層から高機能層に通知される．高機能層では，物理番号（03-XXXX-YYYY）への変換が行われ，伝達層に通知される．そして，伝達層では，物理番号により接続処理が行われる．この例のように，高機能層では，サービス管理を行うために，種々のデータが管理され，様々な契機で検索／更新が行われている．また，サービス追加や仕様の変更に伴い，管理されるデータ構造の変更も発生する．このようなサービス管理においては，利用するデータ（顧客情報等）とサービスを実現するアプリケーションプログラムを分離することにより，種々のデータ管理を独立に実施可能であり，アプリケーションプログラムの追加，修正をデータと独立に実施可能な DBMS の導入が有効となる．

本分野では，従来のビジネス分野と比べ， DBMS への要求条件（表 4.1）および，データベースの利用特性は大きく異なる[38]．これらを，以下にまとめる（図 4.2）．

（1）高性能

従来のビジネス処理に使用される場合に比べ，レスポンス，スループット共に，10 倍以上の高速 DB アクセスが要求される．

（2）高可用性

24 時間無中断でサービスを提供する必要があるため，従来は，システムを停止して行っていた DB 保守処理（DB の再編成／再構成），DB バックアップ処理をサービス無中断で可能とする高可用性が要求される．

（3）データ量

対象データ量がビジネス分野に比べて比較的小規模（～数 GB）である．

（4）アクセスパターン

DB アクセスパターンが単純である．大部分の DB アクセスはユニーク・キーによる 1 レコードのみのアクセスである．また，トランザクションは，大部分がショートトランザクションである．

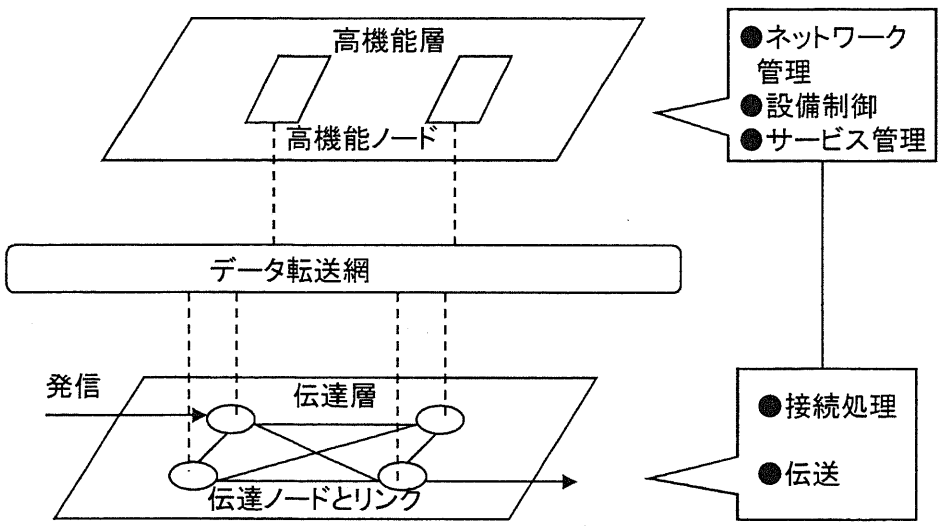


図4.1 AINサービスのアーキテクチャ

表4.1 ネットワークサービス分野のデータベース要求条件

項目		ネットワークサービス系DB	ビジネス処理系DB
処理能力	応答時間	～数十m秒	～数秒
	スループット	～数千TPS	～数百TPS
DB容量		～数GB	～数百GB
リカバリ時間（信頼性）		～数秒	～数分
DB保守時間（連続運転）		サービス無中断（～数百m秒）	計画停止（～数時間）

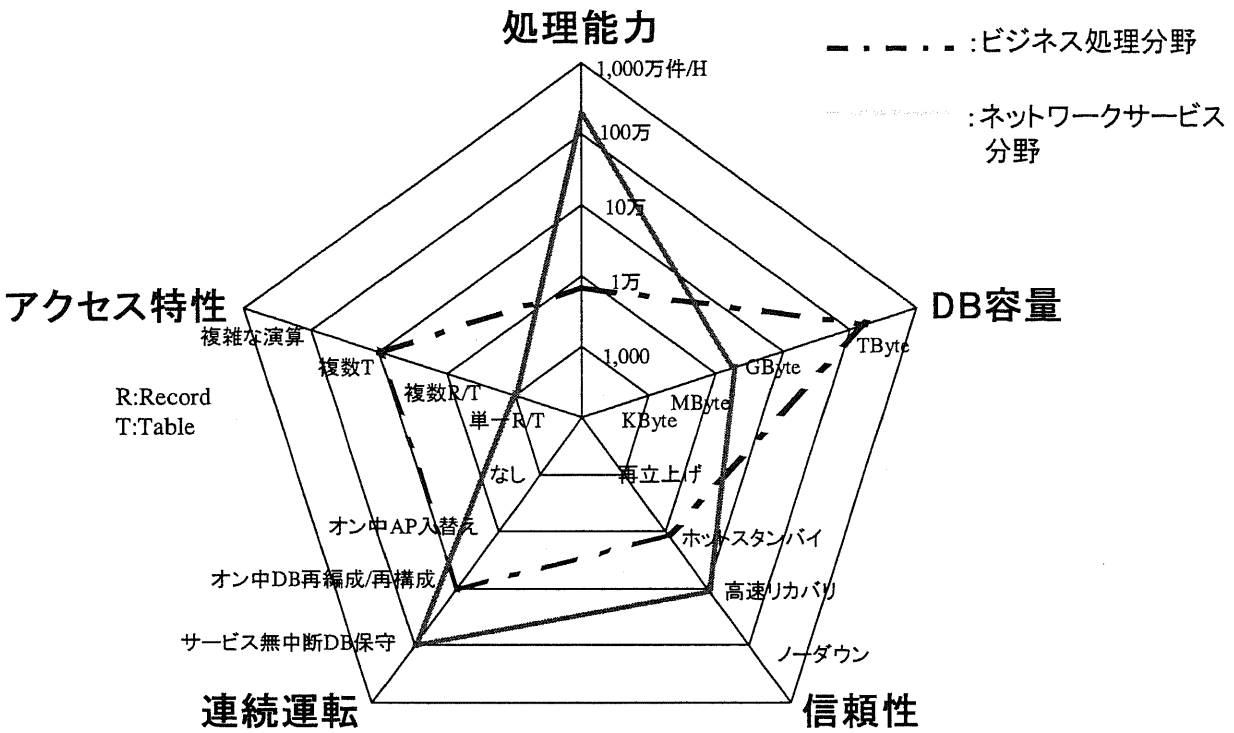


図4.2 ネットワークサービス分野のデータベース利用特性

4.4 RENAのアーキテクチャ

4.4.1 RENA設計の考え方

RENAは，ネットワークサービス分野に適用可能なDBMSとして設計したもので，4.3で述べた要求条件を満足するために，以下の考え方に基づき設計した．

（1）メモリ常駐型DBMS

高性能に対する要求条件とデータ量が少ない点から，メモリ常駐型 DBMS による方法を選択した．

（2）高性能化のアプローチ

高性能化を実現するために，ユニーク・キーアクセスに特化したアクセス手法，ショートトランザクション向けの並行制御方式，ログ取得方式，SQLのサブセット化，言語処理におけるコンパイル化を取り入れた．

（3）高可用化へのアプローチ

メモリ常駐型DBMSでは，障害対策として，DBのバックアップをDK上に取得しておく必要があり，ファジーチェックポイント方式を，また，サービス無中断DB保守のために，新旧テーブルの切替えによる DB 保守方式を取り入れた．

4.4.2 RENAアーキテクチャ

RENAのシステムアーキテクチャを図4.3に示す．データベースは，主メモリ上のデータベースとDK等の二次記憶に格納されるバックアップDBの2階層から構成される．主メモリ上のデータ構造は，メモリアクセスに適した構造を採用している．バックアップDBは主メモリ DB の完全なコピーであり，周期的にコピーが実行される．データベース更新履歴情報であるログは各トランザクション終了時に二次記憶に格納される[42][43]．

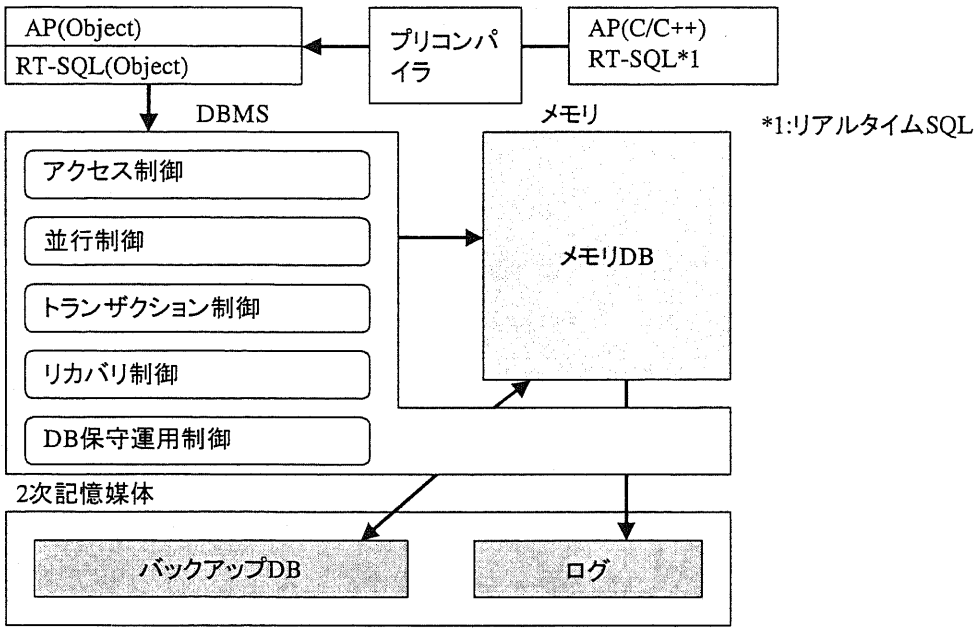


図4.3 RENAのアーキテクチャ

4.5 アクセス制御方式

ここでは，RENA のアクセス制御方式として記憶構造，インデックス構造を述べる．

4.5.1 記憶構造

RENA では，連続メモリ空間上にレコード等を連続配置したメモリセル型記憶構造（図 4.4）を採用した．本構造を用いると，ポインタ操作により高速データアクセスが可能となるとともに，DB 空間の柔軟な拡張，縮小，および可変長データの効率的な格納が可能となる．概要を以下に示す．

（1）固定長レコード用セル

固定長レコード部を格納する空間種別で，固定長レコードが配列構造で格納される．本空間は，テーブル定義時に指定された最大レコード数を格納できるサイズで，連続メモリ空間に確保される．また，固定長レコード内には可変長レコード部を指すポインタを格納する．

（2）可変長レコード用セル

可変長レコード部を格納する空間種別で，様々な長さを持つレコードが保持される．本空間は，テーブル定義時に指定されたサイズで，連続メモリ空間に確保される．

（3）インデックス用セル

インデックス部を格納する空間種別で，インデックス構成レコードが格納される．本空間はインデックス定義時に指定されたサイズで，連続空間に確保される．

本記憶構造では，定義時に連続メモリ空間を確保するが，空間が不足した場合，4.9 で示す DB 保守方式により空間サイズを拡大することができる．

4.5.2 インデックス構造

DB アクセスの高速化手法として，インデックスが一般に有効である．メモリ DB においても，インデックスは高速アクセスのための有効な手段であり，様々なインデックス構

造が提案されている[71]．RENA では，ユニーク・キーによる 1 レコードアクセスが主体である点に着目し，ユニーク・キーアクセスに適したインデックス設計を行った．メモリ DB 向けの代表的なインデックス手法に，拡張ハッシング[26]，T-tree[71]がある．しかし，ユニーク・キーによるアクセスを考慮した場合，拡張ハッシング方式が有効である．本方式は，同時実行性が劣る欠点があるが，同時実行性を改善する手法として，平野らにより，改良拡張ハッシングが提案されている[40]．RENA は，処理速度の面から改良拡張ハッシング（図 4.4）を採用した．ただし，平野らがマルチ CPU 時の同時実行性を向上させるために提案しているバケットのマルチバージョン管理は，トランザクション管理面での負荷を増加させるため，採用していない．

従来 DK-DBMS では，B-tree インデックスへのアクセスならびにレコード格納部へのアクセス時，I/O バッファ内での探索を実施し，I/O バッファに存在しない場合は，DK アクセスを実施する．その後，各ブロック内でのサーチが必要になる．RENA では，I/O に関連する処理が不要であり，また，ハッシュ索引とセル型構造によりサーチ処理コストを削減できており，従来 DK-DBMS に比べて，CPU 処理コストを削減できている．

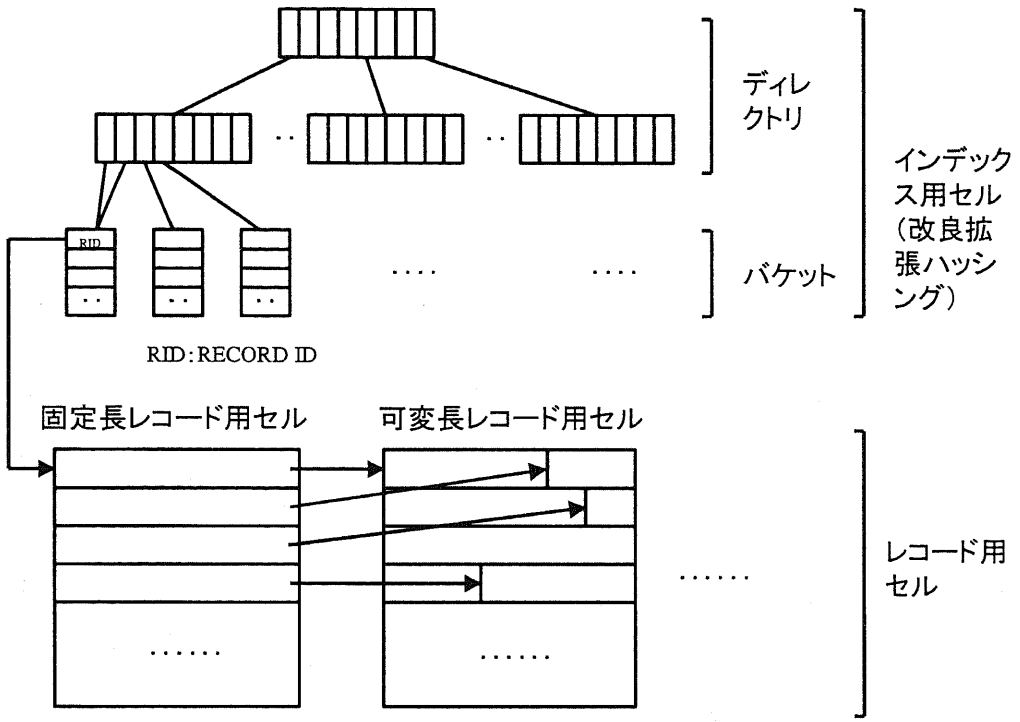

ディレクトリ
バケット
インデックス用セル（改良拡張ハッシング）
RID
RID:RECORD ID
固定長レコード用セル
可変長レコード用セル
レコード用セル

図4.4 記憶構造

4.6 言語処理方式

RENA においては，リアルタイム SQL 仕様の制定，言語処理のコンパイル化ならびに最適化により，高速な言語処理を実現している．

4.6.1 リアルタイム SQL 仕様

4.3 の特性で述べたように，ネットワークサービスにおける DB アクセスは，大部分が非常に単純である．すなわち，SQL における，結合，副問合せ，ソート等の複雑な操作が不要である．そこで，SQL の仕様を制限し，SQL のサブセットで構成されるリアルタイム SQL 仕様を制定した．リアルタイム SQL の特徴を以下に示す．

- データ定義言語は，CREATE/ALTER/DROP SCHEMA/TABLE/INDEX に限定する．
- データ操作言語は，INSERT，DELETE，UPDATE，SELECT，COMMIT，ROLLBACK に限定する．
- SELECT，UPDATE，DELETE のデータ操作言語は，ユニーク・キーに対する＝条件指定による１レコードアクセスのみが可能である．
- 上記に伴い，対象テーブルには，少なくとも一つのユニーク・キーが定義されているとともに，該当ユニーク・キーに対して，ユニーク・インデックスが定義される．
- データ型は，CHARACTER，INTEGER，SMALLINT 型に限定する．

4.6.2 言語処理方式

RENA は，埋め込み型リアルタイム SQL/C，SQL/C++を提供している．プログラマは，ホスト言語 C/C++の中に，リアルタイム SQL を記述することにより，アプリケーションプログラムを作成する．埋め込み型リアルタイム SQL の言語処理は以下の様に実行される．（図 4.5）

（１）プリコンパイル，コンパイル時

・RENA のプリコンパイラにより，ホスト言語に埋め込まれたリアルタイム SQL が展開され，ホスト言語のソースコードに変換される．この時，リアルタイム SQL の最適なアクセスパスを選択する最適化が同時に行われる．

・次に展開されたリアルタイム SQL のソースコードがコンパイラにより，リアルタイム SQL のオブジェクトコードに変換される．

・最後に，これらが，アプリケーションプログラムのオブジェクトコードとリンクされる．

（2）実行時

リアルタイム SQL を含むアプリケーションプログラムの実行時，RENA はアプリケーションプログラムのオブジェクトコードから直接コールされる．

本方式を，コンパイルド SQL 方式と呼ぶ．

従来の DK-DBMS においても，コンパイル方式を実現している．その方式では，プリコンパイル時に，SQL の解析，最適化を行い，その結果の DBMS へのアクセスコードを生成し，そのコードを二次記憶に格納し，実行時にそのコードを呼び出すことにより，処理の高速化を図っている[12][16]．この方式で，標準 SQL を実装すると，多くの処理単位（数式処理，結合処理，カーソル管理等）を管理しておき，その中から必要な処理単位をダイナミックリンクする必要がある．そのため，実行時の制御用の処理コストがかかる．一方，RENA のコンパイルド SQL 方式では，アプリケーションプログラム中に処理単位が実行順序に合わせてリンクされており，実行時の処理時間を短縮できる．

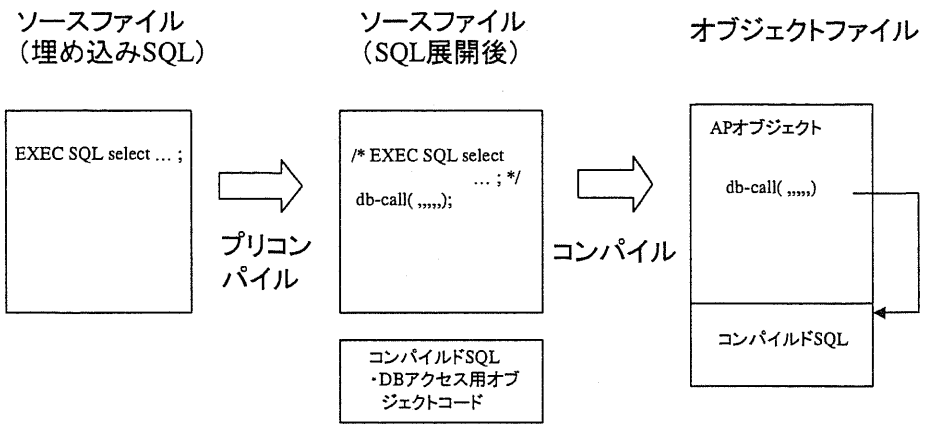


図4.5　コンパイルドSQL方式の概要

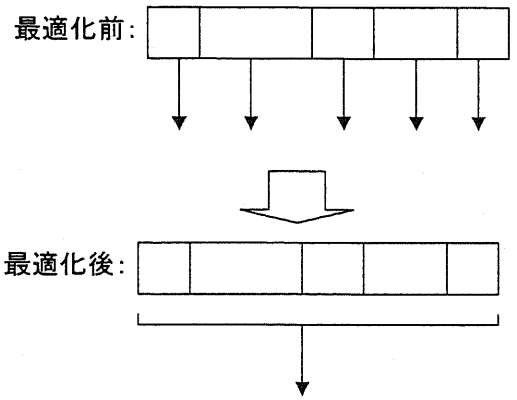
4.6.3 最適化

従来 DK-DBMS の最適化[92]は，DK アクセス回数を最小にするためのものであり，メモリ DBMS である RENA にはそのまま適用することはできない．RENA では，アクセスの単位は 1 テーブルのみであり，テーブルには，ユニーク・インデックスのみが定義されているため，条件指定されたカラムのスキーマ情報のみを使用し，利用するユニーク・インデックスを選択する方法を採っている．これ以外の最適化処理として，メモリ上のデータ配置とトランザクションの構成内容の特徴から，以下の方法を実現し，処理の高速化を図っている．(図 4.6)

（1）メモリ DB 上のデータは連続配置されていることから，DBMS と AP 間のデータ転送時における，連続配置された複数カラムを一括転送する．この方法により，領域の転送回数を削減できる．

（2）SELECT-UPDATE で構成されるトランザクションにおいては，同一レコードへのアクセスが多いことから，先行 SELECT で検索された結果を更新する UPDATE 処理での同じレコードへの検索処理を削減する．この方法により，UPDATE 処理時，従来は，レコードの探索が必要になるが，その処理を削除することができる．

1）複数カラムの一括転送
　メモリ上に連続して配置されるカラムを
APIに対し、まとめて転送する



2）先行SELECTに続くUPDATEの場合、
UPDATEでの検索処理の削減

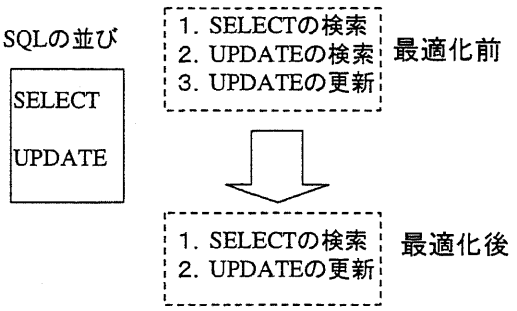


図4.6　RENAの最適化の概要

4.7 並行制御方式

一般に DBMS は，複数トランザクションの DB 並行アクセス時のデータ一貫性保証機構として，並行制御機能を有している．2 フェーズロック方式[9]はその代表的手法であり，多くの DBMS で採用されている．しかし，本方式を実現した場合，デッドロックが発生する．そのため，デッドロックを発見し，それを解消する処理が必要になる．

RENA では，DB アクセスが非常に高速であること，およびトランザクション処理が非常に短いことに着目し，スピンロック方式をベースとする並行制御方式を採用した．概要を以下に示す．

（1）ロック単位

ロックの単位は，ショートトランザクションの競合率を下げるため，レコード単位とする．

（2）制御情報

制御情報には，以下の 2 種類がある．

（a）レコードに，シリアライズ用フラグとロックモード領域を持つ．シリアライズ用フラグは，OS が提供する TEST&SET 命令によりフラグをセットする領域で，トランザクションのシリアライズ制御を実現する．ロックモードは，参照モード／排他モードを示すもので，正数が参照トランザクション数を，0 が未ロック状態を，－1 が排他ロック状態を示す．

（b）トランザクション毎に自トランザクションがロックしている個所情報を保持する．

（3）ロック制御

各トランザクションにおいて，まず，ロック該当個所に対し，シリアライズフラグのセットを行う．セットできれば，以下のロックモード処理を行う．

（a）ロックモードが排他モードかつ自トランザクションがロック済みであれば，何も処理を行わない．

（b）ロックモードが排他モードかつ自トランザクションがロック未ならば，ロック未取得とする．

（c）ロックモードが参照モードかつ自トランザクションがロック済みであれば，何も処

理を行わない.

（d）ロックモードが参照モードかつ自トランザクションがロック未であれば，ロックモードをインクリメントする.

シリアライズ用フラグのセットができない場合とロックが取得できない場合，スピンロックを行い，一定回数以上ウェイトするとエラーで処理を終了する．エラー時以外は，シリアライズ用フラグをリセットする.

ロックの解放は，トランザクション終了時に行い，ロック取得と同様，シリアライズ用フラグのセット後，ロックフラグの減算または0セットを実施し，シリアライズ用フラグをリセットする.

（4）ファジーロック処理

ロック処理の削減のために，インデックスサーチ時，バケット同定後，バケット内RIDに基づき複数のレコードを対象に検索条件の確認を行う際，未ロック状態で確認を行った後，該当するレコードのみロック処理を実施し，再度確認を行う．この方法により，チェックを行うたびにロック処理を行う必要がなくなる.

ロック単位を小さくすることによりトランザクション間の競合率を下げることが可能となる．しかし，従来DK-DBMSで採用されている，待ち制御を行う方式では，ロック単位の細分化を行うと，ロック情報の増大，および待ち制御の伴うロック･キューの増大により，競合検出のCPU処理コストが非常に大きくなる．一方RENAで用いた制御法では，待ち制御を行わないためロック･キューが発生せず，競合検出のCPU処理コストを削減できる．RENAの方式では，競合発生により，ロックエラーとなるトランザクションが増加する可能性があるが，ショートトランザクションがほとんどであるため，その確率は低い．また，ファジーロック処理により，不要なロック処理を排除しており，処理コストを削減できる.

4.8 オンライン・コンカレント・リカバリ制御方式

メモリ常駐 DB 構造では，電源ダウン等のシステム故障発生時，メモリ上の DB は保存できない．そのため，DB をシステム故障発生前に復旧するための機構が必要となる．RENA では，メモリ空間上の DB と二次記憶上のバックアップ DB の 2 階層記憶アーキテクチャを採用した．メモリ DB は定期的にバックアップ DB として二次記憶上に取得される．あわせて，トランザクション実行時に取得されるログ（DB 変更履歴）も二次記憶上に取得される．システム故障発生時は，二次記憶上のバックアップ DB とログ情報からシステム故障発生直前の DB に復元する．RENA では，従来 DK-DBMS の更新前情報(BI : Before Information)，更新後情報(AI : After Information)の両方を取得する方式と比べ，ログ取得量，I/O 回数，CPU 処理コストを低く押さえることができる方式を実現した．以下にバックアップ方式，ログ取得方式，リカバリ方式を述べる．

4.8.1 バックアップ方式

バックアップ DB を取得する有効な方式に，ファジーチェックポイント方式[36][85]がある．本方式は，バックアップ処理とトランザクション処理との排他制御が不要なため，他の処理に比べて，オンライン処理への処理時間の影響が少ないことを特徴としている．しかし，バックアップしたデータの一貫性を保証するためには，バックアップ処理中に走行するトランザクションのログを必要とする．

バックアップは，ファジーチェックポイント方式に従い，バックアップ対象データを二次記憶に出力する．このとき，出力先ファイルは，2 世代化しておき，交互に使用する．また，バックアップした DB データの一貫性を保証するために，バックアップ処理開始以降のトランザクションからのログについては，4.8.2 の手順に従いログの取得を行い，バックアップ結果とともに保存する．

本処理は，定期的に実施するが，例えば，高度電話サービスでは，数分周期で取得が行われる．

4.8.2 ログ取得方式

ファジーチェックポイント方式において，一貫性が失われる場合には，以下が考えられる．

（a）トランザクションが更新中のデータを，バックアップ処理が取得した場合．

（b）更新前のデータをバックアップ処理が取得し，その更新トランザクションが正常終了した場合．

（c）更新後のデータをバックアップ処理が取得し，その更新トランザクションが異常終了した場合．

（d）更新後のデータをバックアップ処理が取得し，その更新トランザクションが完了以前に障害が発生した場合．

（e）バックアップ処理中にI/O障害が発生した場合．

これらの問題に対し，一貫性を保証するためのログ取得方式について述べる（図4.7）．

（1）通常時のログ取得方式

一般にログはDB中のBIとAIの2種類から構成される．BIは，データ更新前に取得され，ロールバック処理に使用される．AIは，トランザクション終了時に取得され，ロールフォワード処理に使用される．RENAは，バックアップDBからロールフォワード処理により最新DBに復元する方式を採用している．通常のオンライン処理時は，トランザクション終了時，AIのみを二次記憶へ取得する．BIについては，二次記憶に出力せず，メモリ上でトランザクションをロールバックするのに用いる．この方式により，ログの取得量は1/2に削減でき，取得契機は，各トランザクションで一回のみでよい．

（2）バックアップ処理期間のログ取得方式

本節の冒頭で示した一貫性が失われる場合のうち，（a），（b），（c）について，一貫性を保証するため，更新前後でBIおよびAIをログバッファ上に取得しておき，トランザクションが正常終了した時はAIを，異常終了した時はBIを二次記憶に出力する．こうして取得したログにより，一貫性が保証される．また，取得契機は一回とすることができる．

（d），（e）については，バックアップ用ファイルを2世代化することにより，（d），（e）が発生した場合については，バックアップ処理をアボートし，前世代のバックアッ

プデータを使用することにより一貫性を保証する.

以上のログ取得方式では，バックアップ DB が 2 世代必要になり，二次記憶の容量は 2 倍となるが，トランザクション中におけるログの取得契機は，各トランザクション内で 1 回にすることができ，オンライン処理への影響が少ない DB バックアップ処理が実現できる.

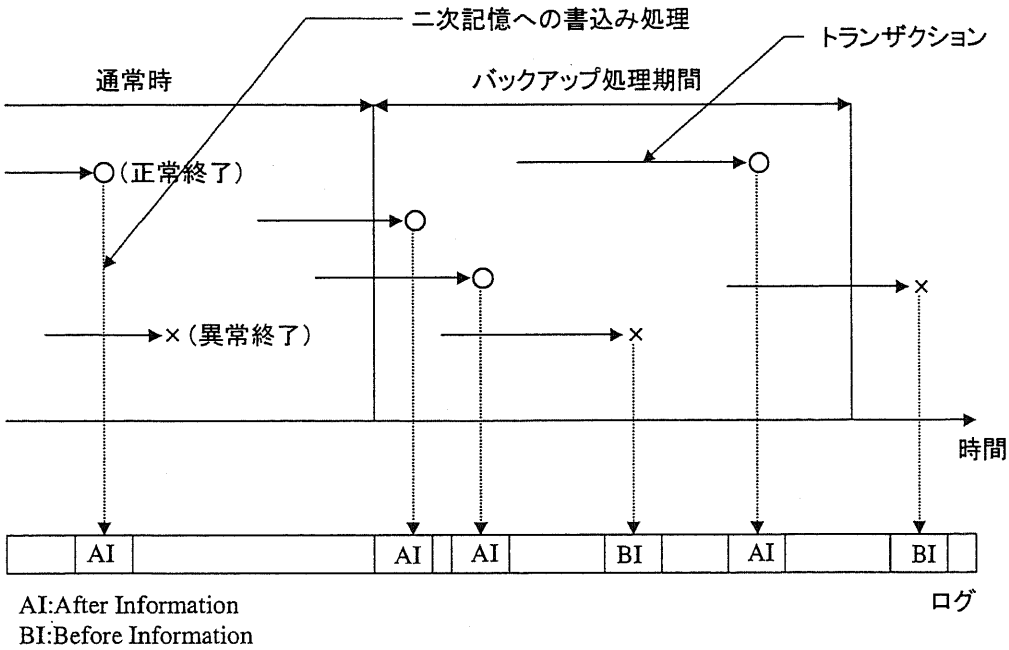

二次記憶への書込み処理
トランザクション
通常時
バックアップ処理期間
(正常終了)
×(異常終了)
×
×
時間
AI
AI
AI
BI
AI
BI
ログ
AI:After Information
BI:Before Information

図4.7 ログ取得方式

4.8.3 リカバリ方式

（1）トランザクションのロールバック処理

トランザクションがDBを更新した時に，BIをメモリ上に取得しておき，トランザクションが異常終了したとき，DBを更新前の状態に戻す．

（2）二次記憶からのリカバリ処理によるDB復元

二次記憶上の最新DBバックアップファイルをメモリ上にロードし，さらに二次記憶上のAI，BIを発生順に反映する．ただし，障害がバックアップ処理中に発生した場合は，一つ前の世代のバックアップファイルを使用する[61][62]．

4.9 サービス無中断 DB 保守方式

リレーショナル DBMS は，スキーマ変更等の DB 再構成機能およびガーベージコレクション等の DB 再編成機能を備えている．ネットワークサービスでは，これらの機能を 24 時間サービス無中断で実現する必要がある．RENA では，この要求にこたえる DB 保守方式を実現した．

DB の保守処理には，提供しているサービスの仕様変更などに伴い，DB の定義情報の変更を行う DB 再構成処理やデータベース内のガーベージコレクションを行う DB 再編成処理がある．また，RENA では，サービス仕様の変更などに伴い，全レコードのデータを一括して変更する処理についても，DB 保守処理として分類する．

高度電話サービスなどの 24 時間連続無中断運転が必要なシステムの構成要素となる DBMS では，DB 保守処理をサービスを継続しながら行う機能が要求される．従来の DBMS では，DB 保守処理は，サービスを停止したオフライン状態で行うのが一般的であり，24 時間連続無中断運転が要求されるシステムの運用に適さない．

RENA では，オンライン中での DB 保守処理を実現するために，通常のテーブルが使用する領域とは別な領域に DB 保守のテーブルを作成し，もとのテーブルで連続してサービスを行いながら，新テーブルで保守処理を併走させる方式を提案している[61][62]．本方式の詳細を以下に示す．

4.9.1 DB 保守基本方式

RENA の DB 保守基本方式を図 4.8 に示す．本方式は，以下の 4 フェーズから構成される．

（1）保守後テーブル作成フェーズ

処理対象のテーブル（以下，前テーブルと称する）と別な領域に，保守処理後のスキーマ定義情報に基づきテーブル（以下，後テーブルと称する）の領域を確保する．

（2）データ転送フェーズ

前テーブルと後テーブル間で 1 レコードづつデータ転送を行う．前テーブル内で更新ロ

ックされていたレコードおよびデータ転送後に発生した更新結果については，次の更新同期フェーズでデータ転送処理を行う．

（3）更新同期フェーズ

本フェーズでは，データ転送フェーズで転送できなかったレコードについて反映を行い，前テーブルと後テーブル間で同期処理を行う．対象となるレコードは，更新ロックされていて転送されなかったレコードと，データ転送後に更新されたレコードである．転送対象レコードを判別するために，並行制御情報として使用されている更新対象レコードの位置情報（レコードID）のみロックヒストリとしてメモリ上に保存しておき，この情報を用いて，レコード更新同期を実行する．

（4）テーブル切替えフェーズ

更新同期フェーズにおいて，更新同期の対象レコードは，全てのトランザクションの更新レコードではなく，また，レコードの転送処理は，トランザクションより短いため，ロックヒストリのエントリは減少しつづけ，最終的にはエントリはなくなる．このロックヒストリが無くなった時点で，テーブル切替え処理を行う．テーブル切替え中は，該当テーブルへのアクセスを防ぐため，DB に対して閉塞処理を行う．閉塞中の処理は，前テーブルから後テーブルへのポインタの変更のみであり，ほとんど処理時間を必要とせず，サービスが中断する時間は，数 10msec オーダであり，実質上，サービス無中断の DB 保守が実現できる．

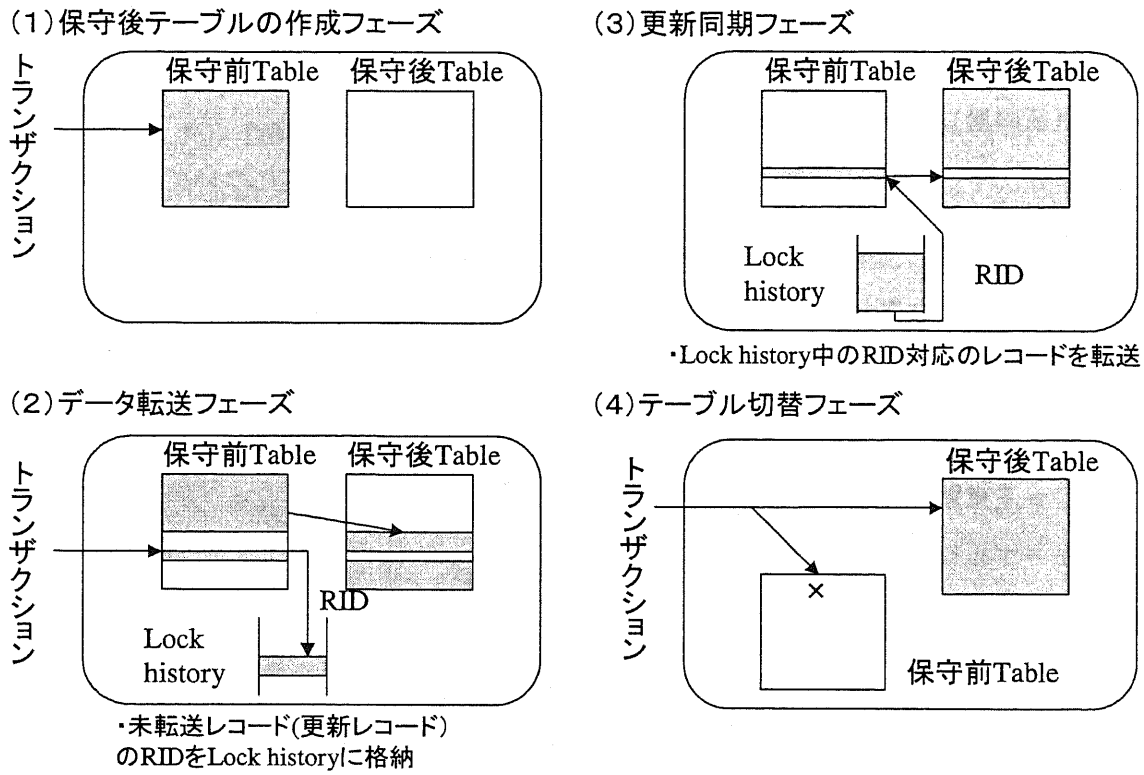


図4.8　DB保守基本方式

4.9.2 データ変換を含んだDB再構成方式

4.9.1で示した方式により，オンライン中のDB再構成処理を実現できる．しかし，一般にDB再構成処理では，DBスキーマの変更と共に，データ実体の変更を必要とする．そのため，上記方式により，オンライン中にスキーマ変更を行っても，データ変換を行う間は，通常のトランザクションは走行できないという問題が発生する．そこで，以下では，データ変換が必要なDB再構成処理についても，オンライン中に実現する方式を示す．

（1） データ変換処理

DB再構成処理に伴い必要となるデータ変換処理には，DBMSが単独で処理出来る場合とアプリケーションプログラムの介在を必要とする場合がある．前者の例としては，デフォルト値を設定する場合などがある．また，後者の例を図4.9に示す．この例のようにデータの変更内容は，DBの利用方式によって異なり，アプリケーションプログラムによる独自の変換処理が必要になる場合がある．

（2） データ変換を含んだDB再構成方式

4.9.1のDB再構成方式では，再構成前後のテーブル間でデータ転送を行う．そこで，該転送処理に着目し，データ転送処理内でデータ変換を行う方式を考えた．この方式により，スキーマ変換とデータ変換を同時に行うことができる．ただし，図4.9の例のように，データ変換処理にアプリケーションプログラムの介在を必要とする場合は，データ変換はアプリケーションプログラム内で行う．このときの処理内容を図4.10に示す．4.9.1で示した基本方式において，データ転送の契機に，以下の処理を行う．

（a）転送対象レコードをデータ変換を実行するアプリケーションプログラムに転送する（図中（A））．ただし，該当レコードの位置情報については，DBMS内に保持する（図中（B））．

（b）アプリケーションプログラムでデータ変換を実行する（図中（C））．

（c）変換後のレコードを DBMS 内に保持した位置情報に基づいて，再構成後のテーブルに反映する（図中（D））．

DBMS 内で単独にデータ変換を行う場合は，アプリケーションプログラムへ転送せず，DBMS内でデータ変換を行う．以上により，データ転送処理とデータ変換処理を同時に行

うことができる.

DB 再編成処理および一括更新処理については，上記と同様の方式で実現できる.

（ｉ）DB 再編成処理

保守後テーブル作成フェーズにおいて，前テーブルと同じスキーマ情報を持つテーブルを作成し，データ転送時にレコードの詰め直しを行うことにより実現できる[61][62].

（ii）一括更新処理

保守後テーブル作成フェーズにおいて，前テーブルと同じスキーマ情報を持つテーブルを作成し，データ転送時に，すべてのレコードに対し，同じデータ変換を行うことにより実現できる.

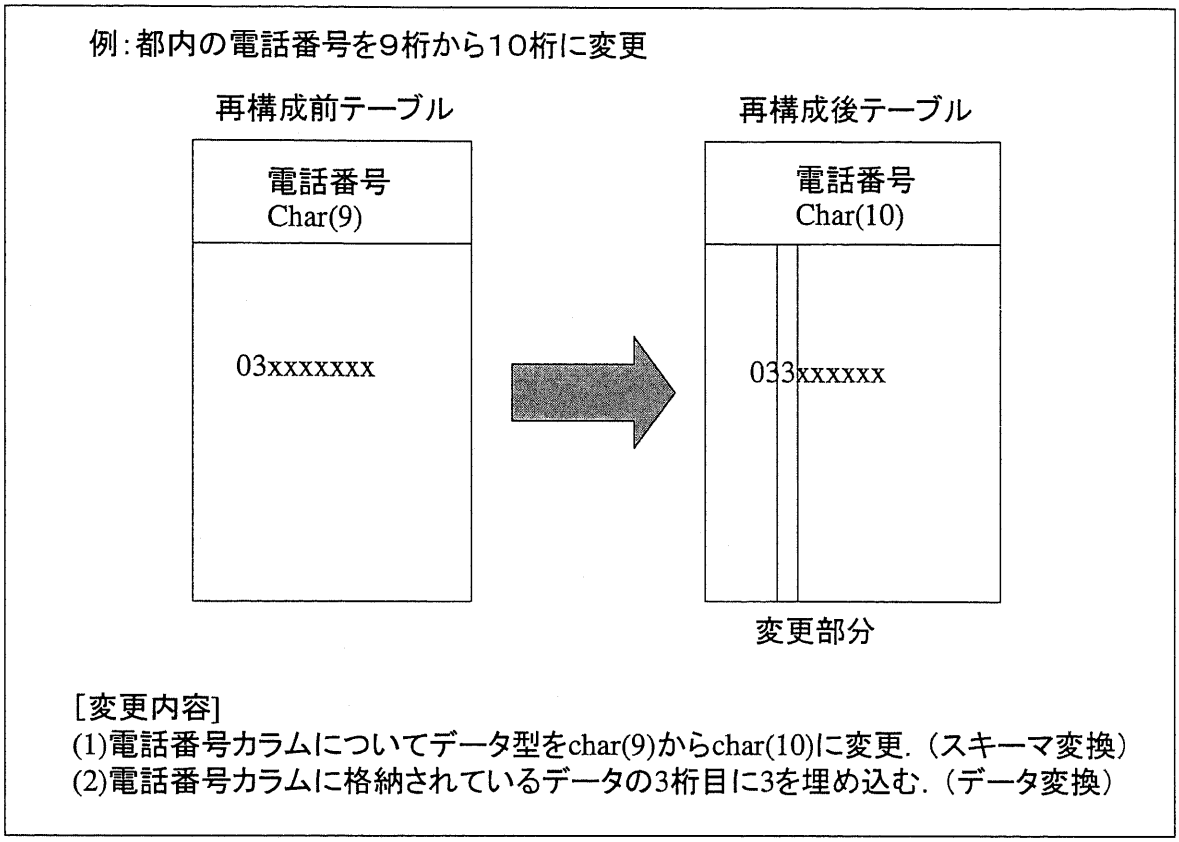
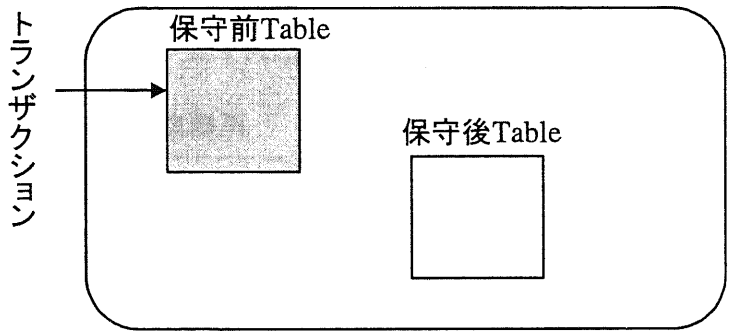
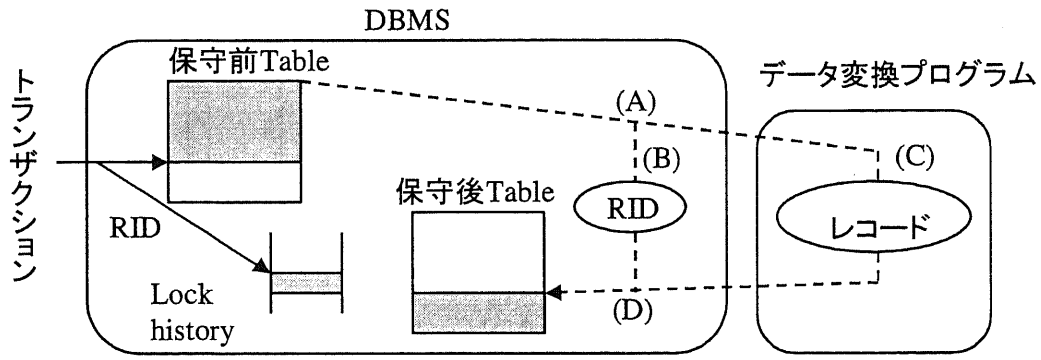

例:都内の電話番号を9桁から10桁に変更
再構成前テーブル
電話番号
Char(9)
03xxxxxxx
再構成後テーブル
電話番号
Char(10)
033xxxxxx
変更部分
[変更内容]
(1)電話番号カラムについてデータ型をchar(9)からchar(10)に変更.（スキーマ変換）
(2)電話番号カラムに格納されているデータの3桁目に3を埋め込む.（データ変換）

図4.9　データ変換の例

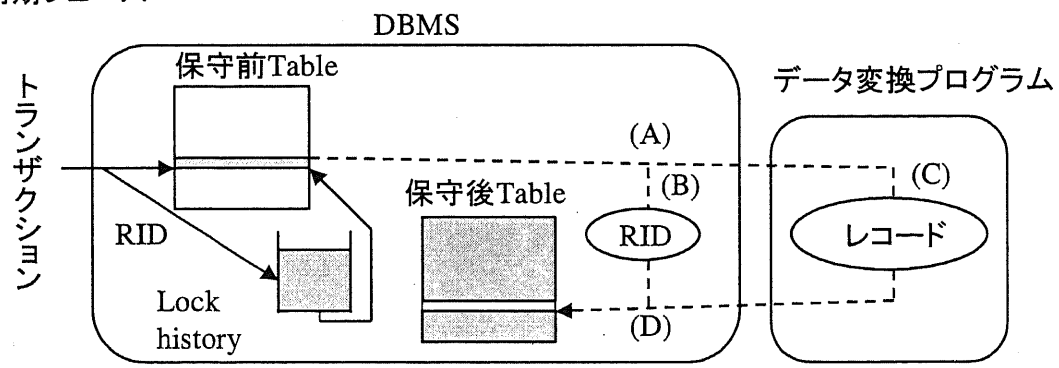
(1)保守後テーブルの作成フェーズ



(2)データ転送フェーズ



(3)更新同期フェーズ



(4)テーブル切替フェーズ

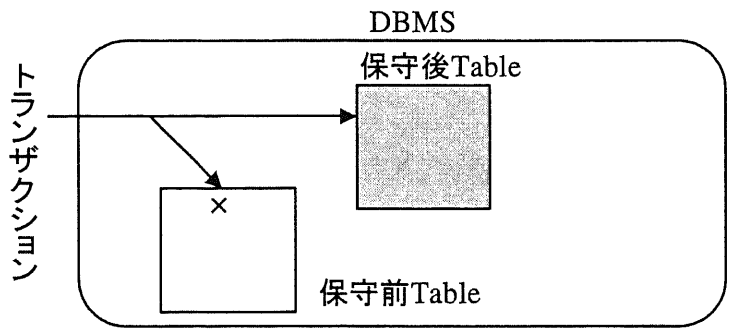


図4.10　データ変換を含んだDB再構成方式

4.10 性能評価

RENA で実現した高速性を確認するために，DB アクセス処理の CPU 処理コスト評価およびスループット評価を行った．以下に，評価結果を示す．

4.10.1 CPU 処理コスト評価

CPU 処理コストの削減効果を確認するため，高度電話サービスの 1 サービス処理を性能評価モデル（IN モデルと呼ぶ）として選び，DK アクセスベースの従来 DK-DBMS と，メモリアクセスベースの RENA の CPU 処理コストを比較した．IN モデル（表 4.2,4.3）は，7 つのテーブルからなり，アクセスは，32 個の SQL から構成される．その内訳は，SELECT が 13，UPDATE が 7，COMMIT が 12 である．

従来 DK-DBMS の CPU 処理コストを 100 とした相対評価結果を図 4.11 に示す．DBMS の処理は，大きく，アクセス制御，言語処理，並行制御，リカバリ制御に分けられる．図 4.11(a)は従来 DK-DBMS の CPU 処理コストを示す．図 4.11(b)は，従来 DK-DBMS の処理から，メモリ常駐化による I/O 処理を削減した場合の CPU 処理コストを試算したものである．図 4.11(c)は，RENA の CPU 処理コストを示したものである．以下，従来 DK-DBMS と単純に DB をメモリに常駐化した場合と，RENA との比較を示す．

（1）従来 DK-DBMS と I/O 不要化対処との比較

図 4.11(b)に示すように，メモリ常駐化のみを実施した場合，CPU 処理コストは，79.4%まで削減可能である．しかし，ネットワークサービスの要求条件を満足するまでには，削減できていない．処理部毎にみると，アクセス制御での削減のみが可能である．アクセス制御では，I/O バッファでのサーチと実 I/O 処理が不要となり，アクセス制御の 47%まで削減できるが，B-tree 構造を意識したインデックス処理とレコードサーチにおけるブロックを意識した処理は必要である．

（2）従来 DK-DBMS と RENA との比較

図 4.11(c)に示すように，RENA は従来 DK-DBMS に比べて，1/10 の CPU 処理コストとなっており，高速化が達成できていることがわかる．以下に処理部ごとの CPU 処理コ

ストの削減内容を示す.

（a）アクセス制御

アクセス制御では，従来 DK-DBMS に対し，7%まで削減できている．削減の要因には，I/O バッファでのサーチと実 I/O 処理が不要になる部分が 56%を，RENA で実現したメモリセル型記憶構造と改良拡張ハッシングによる部分が，44%を占める．

（b）言語処理

言語処理では，従来 DK-DBMS に対し，約 2%まで削減できている．削減の要因は，リアルタイム SQL 仕様により，カーソル管理処理が不要になる部分が，20%，コンパイルド SQL 方式により，実行時の処理単位の呼び出し，ダイナミックリンクが不要になる部分と最適化による部分が 80%を占める．

（c）並行制御

並行制御では，従来 DK-DBMS に対し，21%まで削減できている．削減の要因は，待ち制御が不要なスピンロック方式とファジーロックによるものである．

（d）リカバリ制御

リカバリ制御では，従来 DK-DBMS に対し，36%まで削減できている．削減の要因は，ファジーチェックポイント方式の採用によるログ取得量の削減である．

表 4.2　IN モデル（DB モデル）

テーブル	レコード長（バイト）	レコード数	
		IN モデル	ミニ IN モデル
A	56	40,000,000	40,000
B	53	3,000	3
C	10	600,000	600
D	28	40,000	40
E	1,034	8,000	8
F	9	160,000	160
G	12	8,000	8

表 4.3　IN モデル（アクセスモデル）

トランザクション	SQL(テーブル名)...
T1	SELECT(A) COMMIT
T2	SELECT(F) COMMIT
T3	SELECT(D) UPDATE(D) COMMIT
T4	SELECT(C) COMMIT
T5	SELECT(C) COMMIT
T6	SELECT(B) UPDATE(B) COMMIT
T7	SELECT(C) COMMIT
T8	SELECT(B) UPDATE(B) COMMIT
T9	SELECT(F) COMMIT
T10	SELECT(G) UPDATE(G) SELECT(E) COMMIT
T11	SELECT(G) UPDATE(G) UPDATE(E) COMMIT
T12	SELECT(D) UPDATE(D) COMMIT

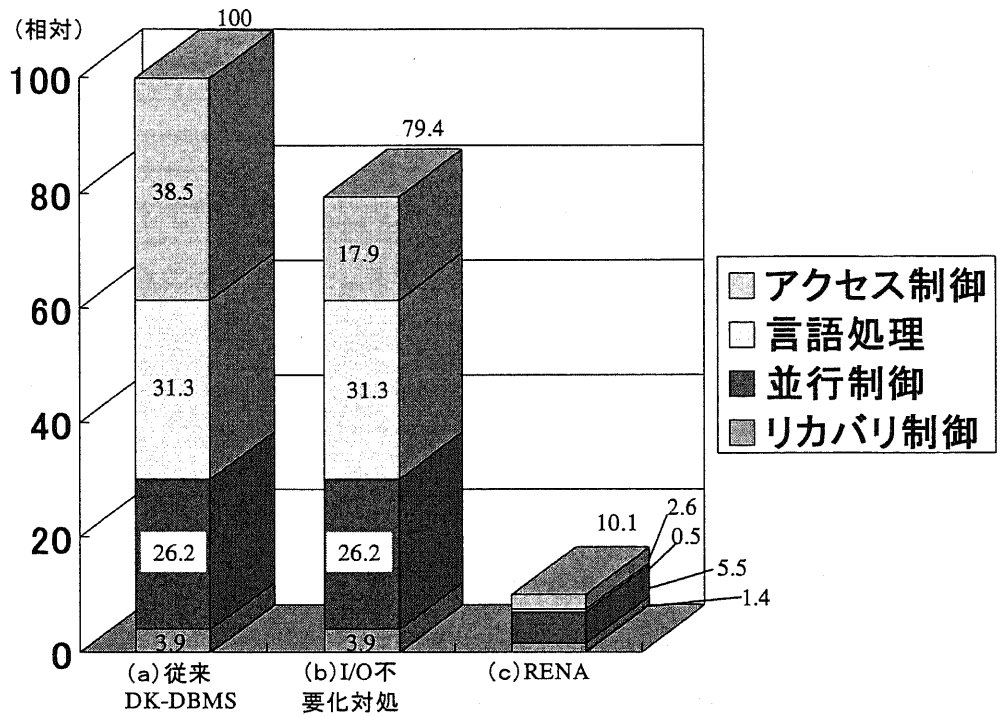

(相対)
100
80
60
40
20
0
100
79.4
38.5
17.9
31.3
31.3
26.2
26.2
3.9
3.9
10.1
2.6
0.5
5.5
1.4
アクセス制御
言語処理
並行制御
リカバリ制御
(a)従来
DK-DBMS
(b)I/O不
要化対処
(c)RENA

図 4.11　CPU 評価

4.10.2 スループット評価

市販 DBMS に対する RENA の高速性を確認するため，RENA と市販 DBMS を同一計算機の UNIX 環境上で走行させ，スループットを測定した．市販 DBMS は，ビジネス分野で代表的な 3 種類のリレーショナル DBMS を選んだ．評価モデルは，CPU コスト評価で用いた IN モデルのレコード数を 1/1000 にしたミニ IN モデル（表 4.2, 4.3）と，DBMS の性能ベンチマーク標準の 1 つである TPC-B モデル[35]を使用した．TPC-B モデルの DB サイズは，計算機のメモリ制限より，スケール 2（テーブルのレコード数 20 万件）とした．メモリ常駐型 DBMS である RENA と，DK-DBMS である市販 DBMS との測定条件を極力合わせるため，市販 DBMS に対しては，以下の性能チューニングを行った．

・全 DB のメモリ常駐化

・プライマリキーに対するユニーク・インデックスの定義

・高性能が引き出せるストアドプロシージャの利用

また，本評価は，RENA のバックアップ処理は走行させない状態で行った．

試験は，WS：Sun sparc-4/IX(IPX)，メモリ量 64MB，ディスク容量：520MB＋2GB×4，OS：Sun OS 4.1.3 上にて実施した．

各種 DBMS を単位時間当りのトランザクション数（TPS：Transaction Per Second）で比較した結果を図 4.12（ミニ IN モデル）と図 4.13（TPC-B モデル）に示す．これらの図では，RENA の TPS を 100 とした相対性能評価値を示している．図 4.12 より，RENA は市販 DBMS と比べて，7.7～37 倍の高スループットを達成していることが分かる．また，図 4.13 より，RENA は市販 DBMS と比べて，5.5～13.2 倍の高スループットを達成していることが分かる．図 4.12 に示したミニ IN モデルによる評価では，参照トランザクション同士の同時走行が可能なので，図 4.13 の TPC-B モデルに比べて，スループットの向上効果が高い．

また，各モデルでの RENA の処理時間は，ミニ IN モデルの 12 トランザクションで 10 数 msec，TPC-B モデルで数 msec であり，レスポンスでも十分な高速性を達成している．

バックアップ処理については，高度電話サービスでのバックアップ処理中の CPU 使用率の増加が，数%以内であり，通常トランザクション処理への影響は少ない．

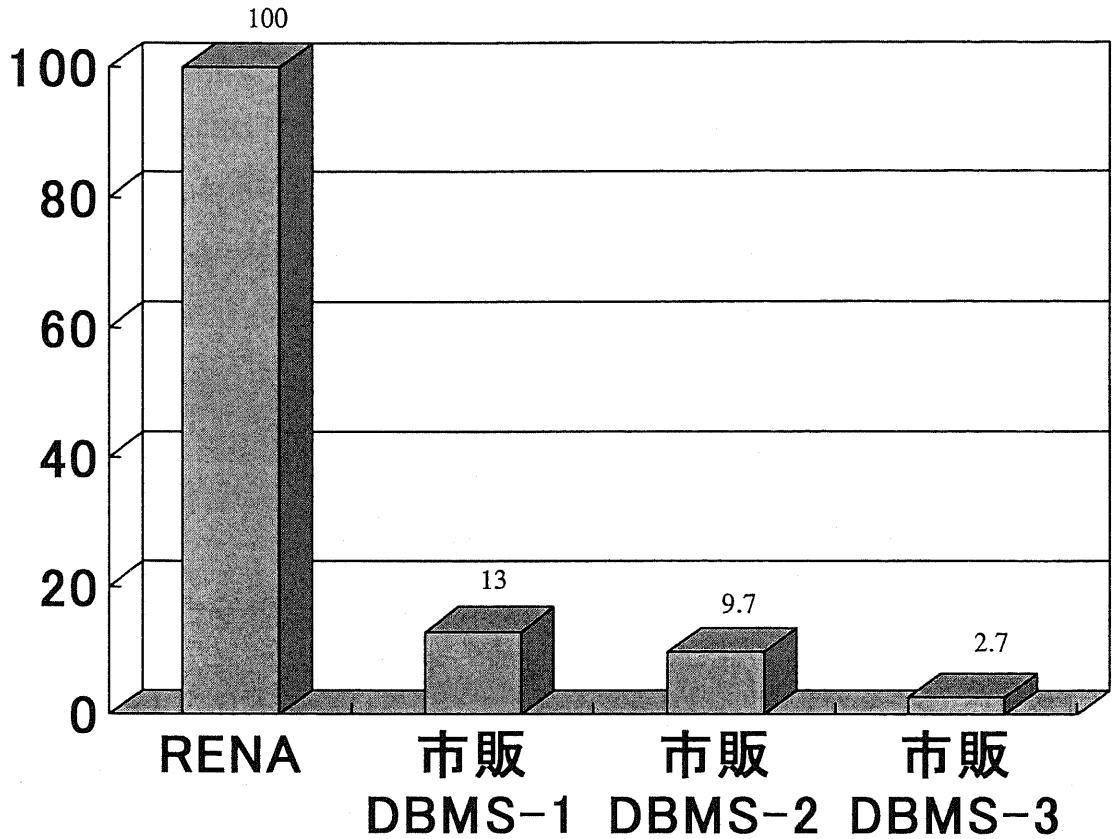


図 4.12　スループット評価（ミニ IN モデル）

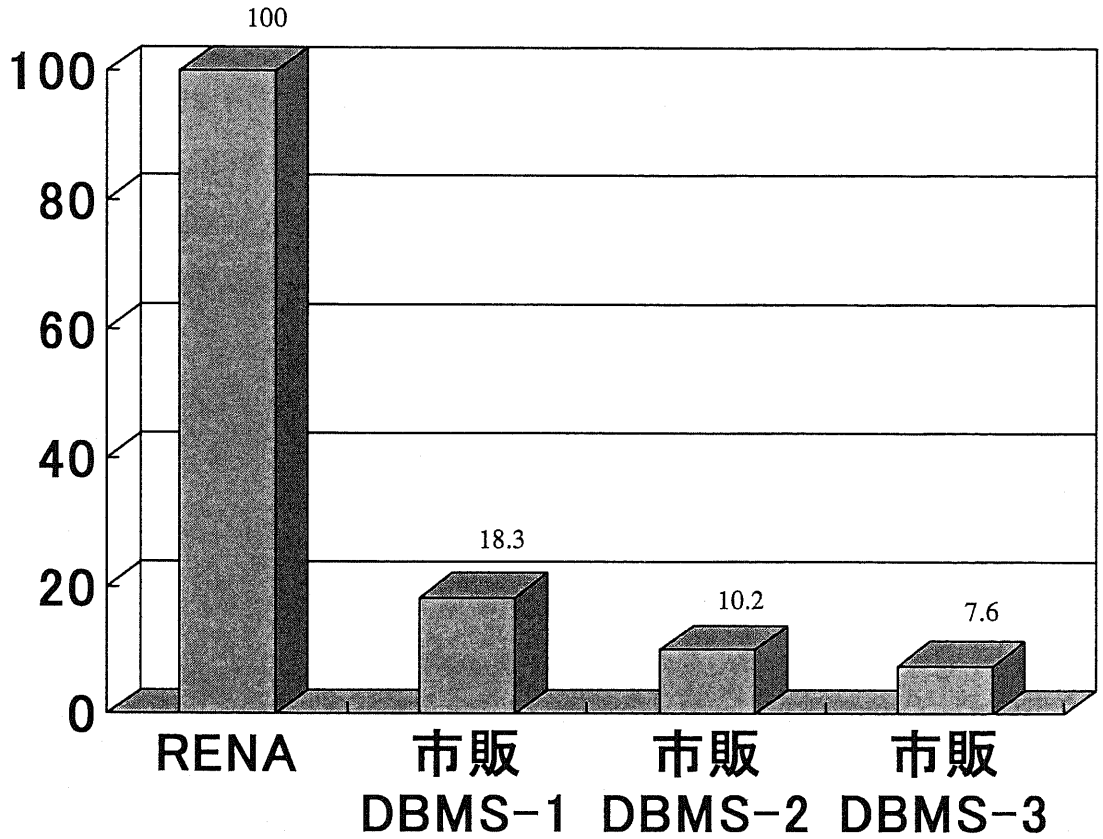


図 4.13　スループット評価（TPC-B モデル）

4.11 あとがき

ネットワークサービスに必要な高性能かつ高可用なDBMSであるRENAの設計と実現方法を示した．RENAは，本格的なメモリ常駐型リレーショナルDBMSであり，新たに，メモリセル型記憶構造，改良拡張ハッシングの実装，リアルタイムSQL仕様，コンパイルドSQL方式，最適化，ファジーロック処理ならびにサービス無中断DB保守方式を提案し，従来提案されているスピンロック方式による並行制御方式，ファジーチェックポイントによるオンライン・コンカレント・リカバリ制御方式を採り入れて実現したものである．

性能評価の結果，CPU処理コストにおいては，従来DBMSに比べて，1/10までコストを削減できており，スループットにおいては，市販DBMSに比べて，約5.5～37倍のスループットを達成できている．また，24時間無中断サービス向けに，リカバリ機能およびサービス無中断DB保守機能を実現している．現在，RENAはNTTの高度INサービスに導入され，その中で，バックアップ処理によるCPU使用率の増加が数%以内となっており，通常トランザクションへの影響がほとんどないことが実証されている．

RENAは，ネットワークサービス向けに特化して開発したDBMSであり，大容量DBやショートトランザクション以外のトランザクションが多い従来のビジネス分野への適用は困難である．しかし，RENAは，ネットワークサービスと同様のサービス特性を持つ分野での利用は可能であり，高度INサービスだけでなく，モバイル環境での位置情報の管理サーバ等への適用の可能性があり，今後，適用先の拡大が期待される．

第5章

結論

本論文では，リレーショナルデータベースの適用領域の拡大に向け，データベースプロセッサRINDA向けの最適化方式と，ネットワークサービス向けのメモリ常駐型DBMSであるRENAの実現方式について論じた．

（1）データベースプロセッサRINDAの最適化方式

データベースプロセッサRINDAの最適化方式については，従来提案されている静的最適化では精度が低いことから，新たに，実行時に最適化を行う，精度の高い動的最適化方式を提案した．RINDAを構成する内容検索プロセッサ（CSP）と関係演算プロセッサ（ROP）を有効に使用する結合処理方式，副問合せ方式を明確にし，それらを使い分ける条件を示し，最適化の中で，静的最適化と動的最適化の分担，それらのアルゴリズムを示した．実装評価の結果，最適化のアルゴリズムが有効に作用していることを示した．

具体的には，結合処理，副問合せ処理向けのCSP，ROPを利用した下記の処理方式を明らかにした．

（a）IN述語変換方式

（b）ネステッドループ方式

（c）片ハッシュソートマージ結合方式

（d）両ハッシュソートマージ結合方式

（e）限定述語の比較述語，IN述語変換方式

また，動的最適化として，上記の方式の中から，実行時に最適な方式を選択する判定アルゴリズムと条件式の変換方式を明らかにした．

最後に，結合処理方式について，性能評価を行い，RINDAなし時に比べて，2.6～5.2倍の性能向上効果があることを示した．また，結合処理時の行選択数によって，動的最適化により，有効な方式が選択されていることを明らかにした．

（2）メモリ常駐型DBMS(RENA)の実現

メモリ常駐型リレーショナルDBMSであるRENAについては，ネットワークサービス向けのDBMSへの要求条件に対する課題に対し，主に，高性能化を達成するアーキテクチャならびにアクセス方式，言語処理方式，並行制御方式を示し，さらに，高可用化を達成するオンライン・コンカレント・リカバリ制御方式，サービス無中断DB保守方式を示し，最後に実装評価を行い，有効性を明らかにした．

RENAに関しては，従来提案されている各種技術だけでなく，RENA実現のために新たな技術提案を行った，

新たに提案した技術を以下に示す，

（a）メモリセル構造によるメモリDB格納構造

（b）スピンロック方式における並行制御におけるファジーロック処理方式

（c）言語処理における，SQLのサブセットであるリアルタイムSQL仕様，DBMS実行コードをアプリケーションプログラムにリンクするコンパイルド SQL 方式，メモリ上のデータ配置特性とSQLの実行順序特性を利用した最適化方式

（d）新旧テーブル切替え方式による，サービス無中断DB保守方式

また，以下の技術については，従来技術により実現した．

（a）改良拡張ハッシングによるインデックスアクセス方式

（b）スピンロック方式による並行制御方式

（c）ファジーチェックポイント方式によるオンラインバックアップ方式，ログ取得方式ならびにリカバリ方式

上記の技術を組み合わせて，本格的なメモリ常駐型 DBMS を実装し，その性能評価を行った．まず，CPU処理コスト評価を行い，従来のDKベースのDBMSに比べて，アクセス制御，言語処理，並行制御，リカバリ制御の実現により，1/10の処理コストとなっていることを明らかにした．また，スループット評価では，市販 DBMS に比べて，5.5-37倍のスループットを達成していることを明らかにした．

現在，RENAはNTTの高度電話サービスに供されており，その中で，バックアップ処理によるCPU使用率の増加が数%以内となっており，通常トランザクションへの影響がほとんどないことが実証されている．

メモリ常駐型DBMSについては，メモリコストの低下にともない，さらなる普及の可能

性があり，msec 以下でのリアルタイムにデータを更新するサービス（たとえば，モバイル環境での位置情報管理サーバ等）への適用が可能である．

以上，本研究により，大量データの検索処理速度向上を達成するデータベースプロセッサ向けの最適化方式ならびにネットワークサービス向けのメモリ常駐型 DBMS 実現技術が明らかになった．これらの技術は，商用製品として実装され，実サービスへの適用も行われている．

謝辞

本博士論文をとりまとめるにあたり，ご指導ならびに有益なご意見をいただきました静岡大学情報学部　水野忠則教授ならびに佐藤文明助教授に深く感謝いたします．

RINDA 開発の機会を与えていただいた，NTT 情報流通総合研究所　伊土誠一所長，NTT ソフトウェア株式会社　福岡秀樹西日本支社長，鈴木健司 NTT 元主幹研究員（現東京国際大学教授），松永俊雄 NTT 元主席研究員（現東京工科大学教授）に深謝します．また，RINDA 開発にあたり議論いただいた，株式会社 NTT データ 技術開発本部　井上潮シニアスペシャリスト，黒岩淳一シニアスペシャリストならびに RINDA 開発関係者に深謝いたします．

RENA の開発にあたり，御指導いただいた，NTT ソフトウェア株式会社　田中豪部長，西日本電信電話株式会社　寺中勝美研究開発センタ所長に深謝します．また，RENA 開発にあたり，議論いただいた，NTT サイバースペース研究所　梅本佳宏主幹研究員，小林伸幸担当課長に深謝します．また，数多くの有益な助言をいただいた元 NTT 情報通信研究所，元 NTT ネットワークサービスシステム研究所の関係各位，ならびに，RENA 開発関係者に感謝いたします．

NTT サイバースペース研究所　芳西崇主幹研究員には，RINDA 開発ならびに RENA 開発において，活発な議論をいただきました．また，NTT サイバーソリューション研究所板倉一郎主任研究員には，RINDA 開発で議論いただくとともに，RENA の高度 IN サービスへの導入に尽力いただきました．多くの協力をいただいた両名に深謝いたします．

社会人博士課程への入学を許可いただいた，株式会社 NTT データ　ビジネスインキュベーションセンタ本部長　荒川弘熙常務取締役，日本電信電話株式会社　第三部門長　井上友二取締役，株式会社 NTT データ　ゲートウェイシステム本部長　澤源太郎取締役に感謝いたします．

最後に，本研究を進めるにあたり，陰ながら応援してくれた家族に感謝いたします．

参考文献

[1] Amsaleg, L., Franklin, M. J., Tomasic, A., and Urhan, T.: Scrambling Query Plans to Cope with Unexpected Delays, The 4th Int. Conf. on Parallel and Distributed Information Systems(PDIS)(Miami Beach, Florida), (1996).

[2] 安藤隆朗，小宮富士夫，喜連川優，中込宏，伏見信也：リレーショナルデータベースプロセッサ GREO の構成，信学技法，DE89-37，pp.9-15(1989).

[3] Antoshenkov, G.: Dynamic Query Optimization in Rdb/VMS, Proc. of the IEEE Conf. on Data Engineering, pp.538-547(1993).

[4] Antoshenkov, G.: Dynamic Optimization of Index Scan Restricted by Booleans, Proc. of the IEEE Conf. on Data Engineering, pp.430-440(1996).

[5] Aoe, J.: Computer Algorithms: Key Search Strategies, IEEE Computer Society Press, (1991).

[6] 青江順一：キー探索技法－Ⅰ　静的ハッシュ法とその応用，情報処理，Vol.33, No.11, pp.1359-1366(1992).

[7] 青江順一：キー探索技法－Ⅱ　動的ハッシュ法とその応用，情報処理，Vol.33, No.12, pp.1465-1472(1992).

[8] Bayer, R. and Unterauer, K.: Prefix B-trees, ACM Trans. on Database Systems, Vol.2, No.1, pp.11-26(1977).

[9] Bernstein, P., Hadzilacos, V. and Goodman, N.: Concurrency Control and Recovery

in Database Systems, Addison-Wesley, (1987).

[10] Bitton, D., Hanrahan, M. B. and Turbyfill, C.: Performance of Complex Queries in Main Memory Database Systems, Proc. Int. Conf. on Data Engineering, pp.72-81(1987).

[11] Blasgen, M. W. and Eswaran, K. P.: Storage and Access in Relational Data Bases, IBM Systems Journal, Vol.16, No.4, pp.363-377(1977).

[12] Blasgen, M.W. et al.: System R: An Architectural Overview, IBM Systems Journal, Vol.20, No.1, pp.41-62(1981).

[13] Britton Lee, Inc.: The Intelligent Database Machine – Product Description, (1984).

[14] Cancer, E., Mccann, R. and Aboudharam, M.: IN Rollout in Europe, IEEE Communications Magazine, pp.38-47(1993).

[15] Cercone, N., Boates, J. anf Krause, M.: An Interactive System for Finding Perfect Hash Functions, IEEE Software, pp.38-53(1985).

[16] Chamberlin, D. D. et al.: A History and Evaluation of System R, Communications of the ACM, Vol.24, No.10, pp.632-646(1981).

[17] Codd, E. F.: A Relational Model of Data for Large Shared Data Banks, Communications of the ACM, Vol.13, No.6, pp.377-387(1970).

[18] Comer, D.: The Ubiquitous B-Tree, ACM computing Surveys, Vol.11, No.2,

pp.121-137(1979).

[19] Derr, M. A., Morishita, S., and Phipps, G.: Adaptive Query Optimization in a Deductive Database System, Proc. of 2nd Int. Conf. on Information and Knowledge Management(Washington D. C.), (1993).

[20] DeWitt, D. J. et al.: Implementation Techniques for Main Memory Database Systems, Proc. ACM SIGMOD Conf., June, (1984).

[21] DeWitt, D. J., Ghandeharizadeh, S., Schneider, D., Jauhari, R. Muralikarishna, M. and Sharma, A.: A Single User Evaluation of the GAMMA Database Machine, Proc. of 5th Int. Workshop on Database Machines(IWDM'87), pp.43-59(Oct.1987).

[22] Eich, M. H.: A Classification and Comparison of Main Memory Database Recovery Techniques, Proc. Int. Conf. on Data Engineering, pp.332-339(1987).

[23] Eich, M. H.: MARS: The Design of a Main Memory Database Machine, Proc. Int. Workshop on Database Machines, (1987).

[24] Eich, M. H.: Foreword Main Memory Databases : Current and Future Research Issues, IEEE Trans. on Knowledge and Data Engineering, Vol.4, No.6, pp507-508(1992).

[25] Enbody, R. J. and Du, H. C.: Dynamic Hashing Schemes, ACM Computing Surveys, Vol.20, No.2, pp.85-113(1988).

[26] Fagin, R., Nievergelt, J., Pippenger, N. and Strong, H. R.: Extendible Hashing – A Fast Access Method for Dynamic Files, ACM Trans. on Database Systems, Vol.4,

No.3, pp.315-344(1979).

[27] Flajolet, P.: On Performance Evaluation of Extendible Hashing and Trie Searching, Acta Informatica, Vol.20, No.4, pp.345-369(1983).

[28] Folk, M. J. and Zoellick, B.: File Structures, Addison-Wesley, (1992), (楠木博之, 浜名祐一訳：ファイル構造, bit 別冊, pp.369-396(1997)).

[29] 福岡秀樹, 中村敏夫, 板倉一郎, 鈴木健司：DIPS データベースプロセッサ RINDA のアーキテクチャ, NTT R&D, Vol.38, No.8, pp.851-860(1989).

[30] Garcia-Molina, H. and Salem, K.: High Performance Transaction Processing with Memory Resident Data, Proc. Int. Workshop on High Performance Transaction Systems, Paris, (1987).

[31] Garcia-Molina, H. and Salem, K.: Main Memory Database Systems: An Overview, IEEE Trans. on Knowledge and Data Engineering, Vol.4, No.6, pp509-516(1992).

[32] Garrahan, J. J., Russo, P. A., Kitami, K. and Kung, R: Intelligent Network Overview, IEEE Communications Magazine, pp.30-36(1993).

[33] Gawlick, D. and Kinkade, D.: Varieties of Concurrency Control in IMS/VS Fast Path, Data Eng. Bull., Vol.8, No.2, pp.3-10(1985).

[34] Gray, J. and Reuter, A.: Transaction Processing: Concepts and Techniques, Morgan Kaufmann, (1992).

[35] Gray, J. et al.: The Benchmark Handbook for Database and Transaction

Processing Systems, Morgan Kaufmann, 2 edition, (1993).

[36] Hagmann, R.B.: A Crash Recovery for a Memory-Resident Database Systems, IEEE Trans. on Computers, Vol.C-35, no.9, (1986).

[37] 速水治夫，武田英昭，佐藤哲司，黒岩淳一：データベースプロセッサ RINDA のハードウェア機構，NTT R&D，Vol.38，No.8，pp.861-868(1989).

[38] 平野正則，鈴木孝至，塩澤恒道，芳西崇，木ノ内康夫：分散処理による高度 IN 用サービス制御ノードの構成，信学論(B-I)， Vol.J79-B-I，No.8，pp.539-550(1996).

[39] 平野泰宏，三浦史光，武田英明：同時実効性を高めた動的ハッシュ，情報処理学会第 46 回全国大会，4G-10，pp.185-186(1993).

[40] 平野泰宏,三浦史光,佐藤哲司:同時実行性を高めた改良拡張ハッシング，信学論(D-I)，Vol.J78-D-I，No.4，pp.424-433(1995).

[41] 芳西崇，板倉一郎，中村敏夫，井上潮：データベースプロセッサ RINDA における問合せ処理のアクセスパス決定方式，情報処理学会論文誌，Vol.32，No.11，pp.1412-1422(Nov.1992).

[42] Honishi, T., Kobayashi, N. and Nakamura, J.: Design and Implementation of an Enhanced Relational Database Management System for Telecommunication and Network Applications, Proc. of Pacific Telecommunications Council 18th Annual Conf., pp.698-703(1996).

[43] 芳西崇, 小林伸幸, 中村仁之輔, 田中豪: リアルタイム DBMS 構築技術, NTT R&D, Vol.45，No.1，pp.27-32(1996).

[44] ICL : CAFS-ISP, (1983).

[45] 飯塚肇, 田中英彦 : ソフトウェア指向アーキテクチャ, 第 16 章データベースマシン, オーム社, (1986).

[46] 井上潮, 速水治夫, 福岡秀樹, 鈴木健司, 松永俊雄 : データベースプロセッサ RINDA の設計と実現, 情報処理学会論文誌, Vol.31, No.3, pp.373-380(Mar.1990).

[47] 井上潮, 佐藤哲司, 速水治夫 : データベースプロセッサ RINDA, 情報処理, Vol.33, No.12, pp.1403-1408(Dec.1992).

[48] 井上潮, 芳西崇, 中村仁之輔, 中村敏夫 : データベースプロセッサ RINDA の制御方式, 電子情報通信学会データ工学研究会資料, DE-89-41, pp.41-47(Dec.1989).

[49] 井上潮, 中村仁之輔, 芳西崇, 片岡良治 : データベースプロセッサ RINDA の制御プログラム, NTT R&D, Vol.38, No.8, pp.869-876(1989).

[50] Inoue, U., Hayami, H., Fukuoka, H. and Suzuki, K.: RINDA－A Relational Database Processor for Non-Indexed Queries, Proc. of Int. Symp. on Database Systems for Advanced Applications (DASFAA'89), pp.382-386(1989).

[51] Inoue, U., Satoh, T., Hayami, H., Takeda, H., Nakamura, T. and Fukuoka, H.: A Relational Database Processor with Hardware Specialized for Searching and Sorting, IEEE Micro, pp.61-70(1991).

[52] 石川佳治 : 先進的データベースのための索引技術とその関連技術, Bit, Vol.33, No.2, pp.60-66(2001).

[53] ISO: Information Processing System-Database Language NDL, International Standard, ISO 8907, (1987).

[54] ISO: Information Processing System-Database Language SQL, International Standard, ISO 9075, (1987).

[55] 板倉一郎，芳西崇，中村仁之輔：データベースプロセッサ RINDA の副問合せ最適化方式，情報処理学会第 38 回全国大会，3Q-2，pp.952-953(1989).

[56] Ives Z.G., Florescu D., Friedman M., Levy A. and Weld D.S.: An Adaptive Query Execution System for Data Integration, Proc. of '99 SIGMOD Conf., pp299-310(1999).

[57] JIS：データベース言語 NDL，日本工業規格 JIS X 3004，(1987).

[58] JIS：データベース言語 SQL，日本工業規格 JIS X 3005，(1987).

[59] Kabra N. and DeWitt D.J.: Efficient Mid-query Re-optimization of Sub-Optimal Query Execution Plans, Proc. of '98 SIGMOD Conf., pp106-117(1998).

[60] Kambayashi, Y. and Takakura, H.: Realization of Continuously Backed-Up RAMS for High-Speed Database Recovery, Int. Symp. on Database Systems for Advanced Applications, pp.236-242(1991).

[61] 小林伸幸，芳西崇：ネットワークサービスへの適用を考慮したデータベース管理システムの実装方式について，信学技法，No.287(DE95-50～64)，pp.17-24(1995).

[62] Kobayashi, N., Honishi, T. and Umemoto, Y.: A DB Maintenance and Backup Method for Main Memory Resident Database Management Systems for Telecommunication and Network Applications, The 10th Int. Conf. on Information Networking, pp.248-253(1996).

[63] Kojima, K., Torii, S., and Yoshizumi, S.: IDP – A Main Storage Based Vector Database processor, Database Machines and Knowledge Base Machines (Kitsuregawa, M. and Tanaka, H. eds) Kluwer Academic, pp.47-60(1988).

[64] 小島啓二，鳥居俊一，吉住誠一：ベクトル型データベースプロセッサ IDP，情報処理学会論文誌，Vol.31，No.1，pp.163-173(1990).

[65] Larson, P. A.: Dynamic Hashing, BIT 18, pp.184-201(1978).

[66] Larson, P. A.: Linear Hashing with Partial Expansions, Proc. of 6th Conf. on Very Large Data Bases, CS Press, pp.224-233(1980).

[67] Larson, P. A.: Performance Analysis of Linear Hashing with Partial Expansions, ACM Trans. on Database Systems, Vol.7, No.4, pp.567-587(1982).

[68] Larson, P. A.: Linear Hashing with Overflow-Handling by Linear Probing, ACM Trans. on Database Systems, Vol.10, No.1, pp.75-89(1985).

[69] Larson, P. A.: Dynamic Hash Tables, Communications of the ACM, Vol.31, No.13, pp.446-457(1988).

[70] Larson, P. A.: Linear Hashing with Separaters – A Dynamic Hashing Scheme Achieving One-Access Retrieval, ACM Trans. on Database Systems, Vol.13, No. 13,

pp.366-388(1988).

[71] Lehman, T.J. and Carey, M.J.: A Study of Index Structures for Main Memory Database Management Systems, Proc. of 12th Conf. on Very Large Data Bases, pp.293-303(1986).

[72] Lehman, T. J. and Carey, M. J.: Query Processing in Main Memory Database Management Systems, Proc. ACM SIGMOD Conf., Washington, DC, (1986).

[73] Lehman, T. J. and Garey, M. J.: A Recovery Algorithm for a High-Performance Memory-Resident Database Systems, Proc. ACM SIGMOD Conf., San Francisco, CA, pp.104-117(1987).

[74] Lewts, T. G. and Cook, C. R.: Hashing for Dynamic and Static Internal Tables, IEEE Computer, pp.45-56(1988).

[75] Li, K. and Naughton, J. F.: Multiprocessor Main Memory Transaction Processing, Proc. Int. Symp. on Databases in Parallel and Distributed Systems, Austin, TX, pp.177-189(1988).

[76] Litwin, W.: Linear Hashing : A New Tool for File and Table Addressing, Proc. of 6th Conf. on Very Large Data Bases, CS Press, pp.212-223(1980).

[77] Litwin, W. A., Roussopoulos, N., Levy, G. and Hong, W.: Trie Hashing with Controlled Load, IEEE Trans. on Software Engineering, Vol.SE-17, No.7, pp.687-691(1991).

[78] Mendelson, H.: Analysis of Extendible Hashing, IEEE Trans. on Software

Engineering, Vol.SE-8, No.6, pp.611-619(1982).

[79] 峯村治実，浅野拓哉，佐藤誠，鹿島理華，中村俊一郎，武藤達也：並列データベースマシン HDM，信学技法，DE89-39，pp.25-32(1989).

[80] 中村仁之輔，板倉一郎，芳西崇，黒岩淳一：データベースプロセッサ RINDA の最適化方式，情報処理学会第 37 回全国大会，5Q-8，pp.385-386(1988).

[81] Nievergelt, J.: Binary Search Trees and File Organization, ACM Computing Surveys, Vol.6, No.3, pp.196-207(1974).

[82] 沖電気：データベースマシン db-1 XP，(1990).

[83] 小柳津育郎，塩川鎮雄，他：DIPS-11/5E シリーズの実用化，NTT 研究実用化報告，Vol.36，No.1，pp.49-56(1987).

[84] Pucheral, P., Thevenin, J. M. and Valduriez, P.: Efficient Main Memory Data Management Using the DBGraph Storage Model, Proc. of 16th Conf. on Very Large Data Bases, pp.683-695(1990).

[85] Rosenkrantz, D.J.: Dynamic database dumping, Proc. of ACM SIGMOD Conf. On Management of Data, pp.3-8(1978).

[86] Russo, P. A., Bechard, K., Brooks, E., Corn, R. L., Gove, W. L. and Young, J.: IN Rollout in United States, IEEE Communications Magazine, pp.56-63(1993).

[87] Salem, K. and Garcia-Molina, H.: Checkpointing Memory-Resident Databases, Proc. Int. Conf. on Data Engineering, pp.452-462(1989).

[88] Salem, K. and Garcia-Molina, H.: System M: A Transaction Processing Tested for Memory Resident Data, IEEE Trans. on Knowledge and Data Engineering, Vol.2, pp.161-172(1990).

[89] 佐藤哲司，武田英昭，津田伸生：大容量データベース処理に適したソータ構成法，情報処理学会論文誌，Vol.31，No.11，pp.1653-1660(1990).

[90] Satoh, T., Takeda, H., Inoue, U., and Fukuoka, H.: Acceleration of Join Queries by Relational Database Processor, RINDA, Proc. of 2nd Int. Symp. on Database Systems for Advanced Applications(DASFAA'91), pp.243-248(1991).

[91] 佐藤哲司，武田英昭，井上潮，福岡秀樹：データベースプロセッサ RINDA の結合演算機構の構成と評価，情報処理学会誌，Vol.32，No.8，pp.1006-1013(1991).

[92] Selinger, P. G., Astrahan, M. M., Chamberlin, D. D., Lorie, R. A. and Price, T. G.: Access Path Selection in a Relational Database Management System, Proc. of '79 SIGMOD Conf., pp.23-34(1979).

[93] Share Base Corp.: Share Base Ⅱ Technical Overview, (1988).

[94] Singhal, M.: Issues and Approaches to Design of Real-Time Database Systems, SIGMOD RECORD, Vol.17, No.1, pp.19-33(1988).

[95] Slontnik, D. L.: Logic per Track Devices, Advances in Computers, Vol.10, Frantz Alt, Ed., Academic Press, pp.291-296(1970).

[96] Stonebraker, M.: Managing Persistent Objects in a Multi-Level Store, Proc. ACM

SIGMOD Conf., Denver, CO., pp.2-11(1991).

[97] Suzuki, S.: IN Rollout in Japan, IEEE Communications Magazine, pp48-55(1993).

[98] 武田英昭，佐藤哲司，中村敏夫，速水治夫：関係演算高速化プロセッサ，情報処理学会論文誌，Vol.31，No.8，pp.1230-1241(Aug.1990).

[99] Teradata Corp.: DBC/1012 Database Computer System – Introduction, (1986).

[100] TimesTen Technical Project Manager, et al.: In-Memory Data Management Technical Paper, http://www.timesten.com/products/wp.html.

[101] 植村俊亮，前川守：データベースマシン，オーム社，(1980).

[102] Whang, K. Y. and Krishnamurthy, R.: Query Optimization in a Memory-Resident Domain Relational Calculus System, ACM Trans. on Database Systems, Vol.15, No.1, pp.67-95(1990).

[103] Wong, E., and Youssefi, K.: Decomposition – A Strategy for Query Processing, ACM Trans. on Database Systems, Vol.1, No.3, pp.223-241(1976).

[104] 矢沢良一，平野正則，他：DIPS-V30E のハードウェア構成，NTT 研究実用化報告，Vol.37，No.9，pp.523-532(1988).

筆者発表論文

A　学術雑誌等に発表した論文

（1）中村仁之輔，芳西崇，梅本佳宏，小林伸幸，佐藤文明，水野忠則：ネットワークサービス向けメモリ常駐型リレーショナル DBMS の設計と実現，情報処理学会論文誌　データベース，Vol.43，No.SIG 5(TOD 14)，pp.134-144(2002).

（2）芳西崇，小林伸幸，中村仁之輔，田中豪: リアルタイム DBMS 構築技術，NTT R&D，Vol.45，No.1，pp.27-32(1996).

（3）井上潮，中村仁之輔，芳西崇，片岡良治：データベースプロセッサ RINDA の制御プログラム，NTT　R&D，Vol.38，No.8，pp.869-876(1989).

（4）中村仁之輔，田中豪，織田敬三：リレーショナル・データベース上での統計データ管理の一方式，情報処理学会論文誌，Vol.28，No.9，pp.942-951(1987).

（5）高橋成文，若木勇，中村仁之輔：モバイル環境下で情報を停滞しないメッセージングシステム，情報処理学会論文誌，Vol.41，No.9，pp.2374-2381(2000).

（6）中村仁之輔，梅本佳宏：共通 OS におけるデータベース管理システム，NTT R&D，Vol.40，No.12，pp.1645-1652(1991).

B　国際会議に発表した論文

（1）Honishi, T., Kobayashi, N. and Nakamura, J.: Design and Implementation of an Enhanced Relational Database Management System for Telecommunication and Network Applications, Proc. of Pacific Telecommunications Council 18th Annual Conf., pp.698-703(1996).

（2）Nakamura, J., Ikeda, S.: Data Compression Technique for On-Line Database Systems Containing Large Character Set Texts, Proc. of ICTP'83, pp.373-378(1983).

C　全国大会，研究会において発表した論文

（1） 井上潮，芳西崇，中村仁之輔，中村敏夫：データベースプロセッサ RINDA の制御方式，電子情報通信学会データ工学研究会資料，DE-89-41，pp.41-47(Dec.1989).

（2） 中村仁之輔，板倉一郎，芳西崇，黒岩淳一：データベースプロセッサ RINDA の最適化方式，情報処理学会第 37 回全国大会，5Q-8，pp.385-386(1988).

（3） 芳西崇，中村仁之輔，中村敏夫，井上潮：データベースプロセッサ RINDA の問合せ処理方式，情報処理学会第 37 回全国大会，5Q-7，pp.383-384(1988).

（4） 板倉一郎，中村仁之輔，井上潮：データベースプロセッサ RINDA のデータベースアクセス方式，情報処理学会第 37 回全国大会，5Q-9，pp.387-388(1988).

（5） 板倉一郎，芳西崇，中村仁之輔：データベースプロセッサ RINDA の副問合せ最適化方式，情報処理学会第 38 回全国大会，3Q-2，pp.952-953(1989).

（6） 中村仁之輔，田中豪，織田敬三：RDB による統計データベース処理効率化のための一方式，情報処理学会第 28 回全国大会，2C-5，pp.679-680(1984).

（7） 中村仁之輔，田中豪，織田敬三：アレイ検索法におけるインデックス・サーチの一方式，情報処理学会第 29 回全国大会，5G-10，pp.887-888(1984).

（8） 中村仁之輔，田中豪，織田敬三：RDB 上でのアレイ検索法の評価，情報処理学会第 30 回全国大会，6U-2，pp.969-970(1985).

D　表彰

（1） 昭和 60 年情報処理学会第 30 回全国大会学術奨励賞：RDB 上でのアレイ検索法の評価