データベースプロセッサ RINDA の結合演算処理機構の構成と評価†

佐藤哲司†† 武田英昭††*
井上潮†† 福岡秀樹††

関係データベース処理分野では，データベースの大規模化と問い合わせの複雑化が進み，高速な処理が必要となってきている．データベースプロセッサ RINDA は，高速化に対する要求が強い複雑な条件による非定型の検索処理，統計処理等で必要なソートや結合処理を専用ハードウェアで高速化している．本論文では，RINDA で実現した結合処理の高速化手法を示す．ハッシュ化ビットアレイを用いて結合可能性のない行をふるい落とすフィルタフェーズ，残った行を並べ替えるソートフェーズとソートされた行をマージしながら連結するマージ結合フェーズからなる3フェーズジョイン法を基本とし，多くの比較演算回数を必要とするフィルタフェーズとソートフェーズをハード化した．ハッシュ化ビットアレイを設定・参照する際のハッシュ関数として，種々のキー長やキー属性に対して安定した低い衝突率が得られる乗算重ね合わせ法を考案した．RINDA は，選択処理後の一時表の行数に基づいて，ネステッドループ法，片ハッシュ・ソートマージ法，両ハッシュ・ソートマージ法から最適な結合法を動的に選択する．また，属性が異なる複数カラムからなるキーで結合できる．ベンチマークを用いて性能評価を行い，従来のソフトウェア処理と比較して，結合処理の性能を1桁向上できたことを示し，その要因を分析する．

1. はじめに

近年，関係データベースを用いたシステムでは，ギガバイトを超えるデータベースの構築や数時間もかかる統計処理が実際に行われるようになり，処理の高速化・高機能化に対する要求がますます増大している．特に，インデックスの利用が困難な非定型の検索処理は，処理自体は単純な比較に基づく条件判定やソートが主体であるが，ディスク等の2次記憶装置に格納された大量のデータを処理対象とするため，比較的少量のデータに複雑な計算を行うことを前提とした汎用計算機では効率が悪く，多くの処理時間を必要としていた．このため，データベース処理向きのアーキテクチャを持つデータベースマシン[1]が研究され，実用に供せられてきている．これらのマシンは，主に，ディスクとプロセッサ間の IO ネックの解消と，ソートやジョインなどの繰り返し演算による CPU ネックの解消を目的としている．例えば，CAFS[2] では，ディスクからのデータ読み出しと並行して選択・射影を行うプロセッサをディスク・コントローラに付加して IO ネックの解消を狙っている．ハードウェアソータを備えた GREO[3] や，汎用計算機のベクトルプロセッサをデータベース処理用に拡張した IDP[4] は，CPU ネックを解消しようとしている．また，入出力処理を行う複数の汎用マイクロプロセッサとデータベース処理向きに設計された専用プロセッサとで構成される Server/8000[5] は，IO ネックと CPU ネックの両方に対処している．

筆者らが開発した RINDA (Relational Database Processor)[6],[7] は，ディスクからのデータ読み出し速度に追従してタプルの選択と射影を行う CSP (Content Search Processor) と，ソートや結合演算を高速化する ROP (Relational Operation Accelerating Processor) の2種類のハードウェアからなり，IO ネックと CPU ネックの両方を解決している．

本論文では，RINDA システムで実現した3フェーズジョイン方式[8] に基づく結合演算の実現法を示し，そこで用いたハッシング手法について述べる．3フェーズジョイン法は，結合する2つの表から，キーに基づいて結合可能性のない行を除去するフィルタフェーズ，残った行をキーの順序に並べ替えるソートフェーズ，および，ソートされた2つの表の行を順次マージして連結するマージ結合フェーズからなる．このうち，フィルタフェーズとソートフェーズを ROP で高速に処理する．

フィルタフェーズは，結合対象の第1表でハッシュ化ビットアレイを設定し，第2表で参照することによ

† Design and Implementation of Fast Join Operations in a Relational Database Processor, RINDA by TETSUJI SATOH, HIDEAKI TAKEDA, USHIO INOUE and HIDEKI FUKUOKA (NTT Communications and Information Processing Laboratories).

†† NTT 情報通信処理研究所
* 現在 NTT 総合企画本部
NTT General Planning Headquarters

って結合可能性のない行をふるい落とす．ハッシュ化ビットアレイの設定・参照は，2つの表を結合するキーをハッシュ関数に入力し，そのハッシュ値の一致を判定することから，そこで使用するハッシュ関数は，種々の属性および長さのキーに対してハッシュ値にできるだけ衝突が起きないようにする必要がある．RINDA では，折り畳み法[9]の1つである回転重ね合わせ法と乗算法[10]を組み合わせて，種々の属性および長さのキーに対して衝突率を低くできる乗算重ね合わせ法を用いている．

第2章では，従来システムにおける結合処理の問題点を明らかにし，RINDA システムにおける解決法を示す．3章では，結合処理を高速化する観点から，ソートマージ結合法に基づく3フェーズジョイン法をハードウェア化する方針を示す．4章では，種々のキー属性や長さに対応できるハッシュ関数として乗算重ね合わせ法を提案し，可変長文字列等をキーとして評価し有効性を検証する．最後に，第5章で，結合処理を行う RINDA のアーキテクチャと，その性能について述べる．

2. 従来システムの問題点と RINDA での解決法

関係データベース処理は，あらかじめ作成されたインデックスを用いて少数の行を検索・更新する定型処理と，インデックスのない列に条件を指定した検索や，大量の検索結果をソートしたり統計処理する非定型処理に分類できる．定型処理は，問い合わせや表の構成を十分に考慮してインデックスを付与することで，ソフトウェア処理でも十分な性能が実現されている．一方，インデックスを利用できない非定型検索処理は，表の中のすべての行に対する条件判定を逐次的に繰り返す処理であり，多大の CPU 時間と IO 時間を必要とする．これらの時間は，表の大きさに比例して増大する傾向にあるが，特に，単一の表のグループ化や複数の表の結合を伴う処理では，大量の CPU 時間を必要とする．

この原因の1つとして，従来の計算機アーキテクチャが複雑な数値計算を行うことを前提としており，データベース処理，特に，非定型な検索処理で必要となる高い IO スループットと単純な演算の繰り返し処理に向いていないことが挙げられる[11]．単純な繰り返し演算の代表例に，キーの大小関係を比較して昇順あるいは降順に並べ替えるソート処理がある．

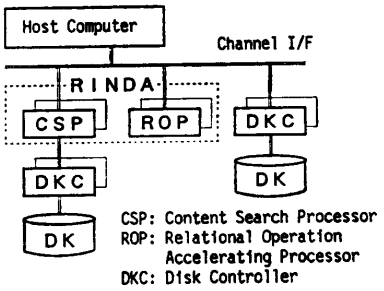


図1 RINDA システム構成例
Fig. 1 RINDA system organization.

RINDA では，上記の CPU 時間と IO 時間の短縮を目的として，**図1**に示す専用ハードウェアを開発した．2次記憶装置であるディスクに格納された表を，指定された複数の条件で検索し，条件に合致した行からなる一時表を出力する内容検索プロセッサ CSP と，CSP で得られた一時表を入力して，ソートや結合演算の前処理等を行い，結果を一時表として返却する関係演算高速化プロセッサ ROP からなる．RINDA システムでは，汎用計算機でデータベース処理を行う場合と比較して，数倍から数十倍の高速化を達成しており，専用ハードウェアを用いることによってコストパフォーマンスを1桁程度向上できた[8]．

3. 結合処理の高速化

3.1 ハードウェア化の方針

結合処理の代表的な手法として，ネステッドループ法[12]，ソートマージ法[12]，およびハッシュジョイン法[13]が知られている．第1表の各行に対して第2表を逐次的に比較して結合するネステッドループ法は，制御は簡単であるが2つの表の行数の積に比例する演算量を必要とすることから，表が小さい場合にのみ利用可能である．ハッシュジョイン法は，2つの表をハッシュ関数を用いて複数のバケットに分割し，対応するバケット間で結合を行うことから，並列処理向きといえる．ソートマージ法は，結合する表をあらかじめソートしておくことで，マージ結合の演算量を線形化できる．RINDA では，ソートマージ法の前処理としてフィルタフェーズを設け，フィルタフェーズとソートフェーズを専用ハードウェアで高速に処理し，マージ結合をホスト計算機で行う3フェーズジョイン法を実現した．

次に，専用ハードウェア化の方針を示す．フィルタフェーズは，不要な行を除去する手段としてハッシュ化ビットアレイを用いたふるい落とし手法[14]を用い

る．まず，第1表の各行から結合のためのキーを作成し，そのキーをハッシュ関数を用いてハッシュし，ビットアレイの対応する位置に設定する．次に，第2表の各行から作成したキーを，第1表で用いたと同一のハッシュ関数でハッシュし，ハッシュ値に対応するビットアレイの位置が設定してなければ，その行は結合の可能性がないとする．第1表に関する操作をビットアレイの設定，第2表に関する操作をビットアレイの参照と称し，ハッシュ値の一致・不一致によって，第2表から結合可能性のない行をふるい落とす．したがって，フィルタフェーズで用いるハッシュ関数は，種々の属性および長さからなるキーを処理できなければならない．また，ハッシュ値に衝突（Collision）が発生すると，結合可能性のない行が残る．以上示したフィルタフェーズは，結合対象となる2つの表のすべての行に対してハッシュ値を算出し，ビットアレイを設定，参照するフェーズであり，結合に用いられるキーの個数や長さ，キーの属性や分布の片寄りをあらかじめ知ることができないことから，より高度な，すなわち計算量の多いハッシュ関数を使用する必要がある．このため，フィルタフェーズは専用ハードウェア化した．

ソートフェーズは，多くのソータの研究例を挙げるまでもなく，計算量が大きい処理であるから専用ハードウェア化する．データベース処理では，アプリケーションによってソートするキーの個数や長さが大きく変動することから，これらの値に柔軟に対処でき，かつ，大容量のソータをコンパクトに実現できるマルチウェイマージソータ[15]を実現した．

マージ結合フェーズの入力は，既に，フィルタフェーズで結合可能性のない行が大部分除去され，かつ，ソートフェーズで結合対象となる2つの一時表がソートされている．このため，マージ結合の計算処理量は十分に小さくなっていると期待できる．また，マージに基づく行の連結は，ユーザによって指定された出力行の構成に大きく依存する．以上の理由から，マージ結合フェーズは，ホスト計算機上のソフトウェアで実行することにした．

3.2 ソートマージ結合法の概要

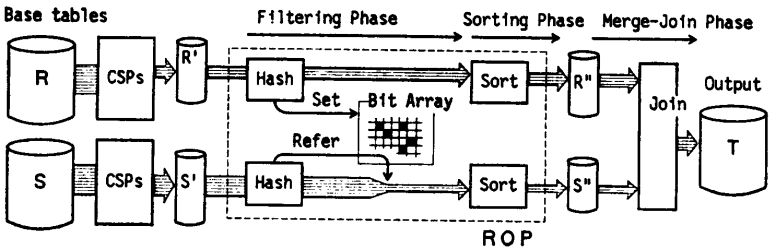
ソートマージ結合法の前処理としてふるい落とし処理を行う3フェーズジョイン法は，ふるい落としを行う表が，結合対象となる2つの表の片方か両方かによって，**図2**に示す2種類の方法が実現できる．

（a） 片ハッシュ・ソートマージ結合法

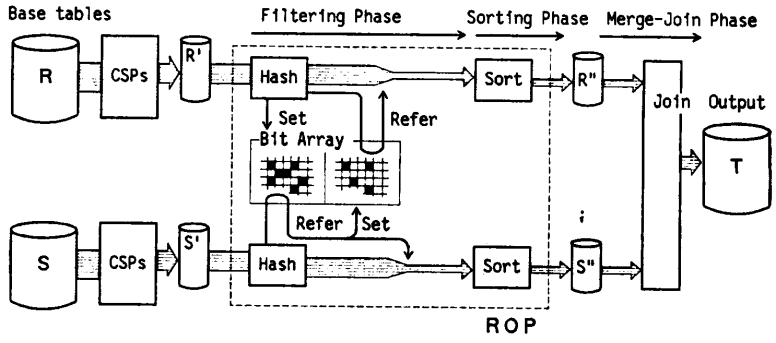
CSPで検索した第1の一時表をROPに入力し，ビットアレイの設定とソートを行い，結果をソート済み一時表として出力する．次に，第2の一時表をROPに入力し，既に設定されているビットアレイを参照してふるい落としを行った後にソートして，結果をソート済み一時表として出力する．得られた2つのソート済み一時表をホスト計算機でマージ結合する．本方法では，第2の一時表だけがふるい落とされる．

（b） 両ハッシュ・ソートマージ結合法

第1の一時表をROPに入力してビットアレイを設定する．次に，第2の一時表をROPに入力し，第1の一時表で設定されたビットアレイを参照してふるい落としを行った後に，別のビットアレイの設定とソートを行って，結果をソート済み一時表として出力する．再度，第1の一時表をROPに入力し，第2の一時表で設定されたビットアレイを用いてふるい落とし



（1） 片ハッシュ・ソートマージ結合法
（1） Single-table filtering method.



（2） 両ハッシュ・ソートマージ結合法
（2） Dual-table filtering method.

図2 RINDAを用いたソートマージ結合処理法
Fig. 2 Sort-merge join methods using RINDA.

を行った後にソートして，結果をソート済み一時表として出力する．最後に2つのソート済み一時表をホスト計算機でマージ結合する．本方法では，2つのビットアレイを用いて，第1と第2の両方の一時表をふるい落としている．

ROP で実現したソータは，登載しているメモリ容量から，キー長によってソートできる最大キー数が決まる[15]．ふるい落とし処理で残った行数がソートできる最大キー数以下であれば，オーバフローせずに一度にソートできる．このため，結合する表が極めて大規模な表であっても，ふるい落とし後の一時表の大きさ，すなわち，ソートするキー数が制限以下であればオーバフローしない．また，ふるい落としによって ROP から出力される一時表が小さくなれば，一時表の転送に要する時間とマージ結合に要する時間を短縮できる．このように，ふるい落とし処理で結合可能性のない行をできる限り除去することが，適用領域の拡大と処理の高速化を達成する上で重要となる．

4. ハッシュ関数の決定法

ハッシュ関数は，互いに異なる2個以上のキーのハッシュ値が衝突する場合がある．関係データベース処理では，キーを生成する列の属性が整数であったり文字列であったり，時には異なる属性を持つ複数の列からキーを生成する場合もある．また，キー値の分布に片寄りが生じる場合もあり，単純なハッシュ関数では衝突率が高くなると考えられる．ハッシュ値が衝突すると，十分に結合可能性のない行を除去できなくなるため，より高度なハッシュ関数が要求される．

ハッシュ関数は，キーをハッシュ表のアドレス空間に一様（ランダム）に割り当てるものが望ましい．文献 9)，10) は，短い固定長キーを前提として，過去に提案されたハッシュ法の衝突率を比較している．代表的なハッシュ関数は，除算法，乗算法，折り畳み法，平方採中法，文字解析，基数変換法などであり，キーの集合が未知な場合は，除算法が比較的衝突率が低いとしている．ふるい落とし処理では，キー長が長い場合や可変長の場合があり，そのまま除算法を適用することはできない．以下にふるい落とし処理で用いるハッシュ関数の特徴を示す．

(1) ハッシュ表は，ハッシュ値に対応するキーの有無を表示する1ビットのセルの集まりである．このため，以下に示すロードファクタを小さくできる．

$$\text{ロードファクタ}=\frac{\text{異なりキー値の総数}}{\text{ハッシュ表のセル数}}$$

(2) 衝突の軽減は重要ではあるが，衝突が起きた場合でもあふれ（オーバフロー）等の処理は不要である．

(3) 種々のデータ属性に対して，同一のハッシュ関数が適用できる必要がある．

(4) 比較的長い可変長のキーを扱える必要がある．

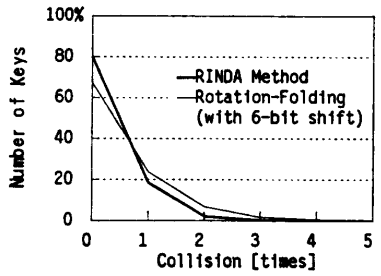
比較的長い可変長のキーを扱う方法に，キーを固定長の複数の切片に分割して，それらを排他的論理和を用いて重ね合わせる排他的論理和法[11]がある．しかし，文字列，特に日本語文字列の文字コードは，あるビットに1が出現する確率に偏りがあるため，単純に1バイトあるいは2バイトの切片を用いて重ね合わせると，結果のハッシュ値に偏りが生じる．

ハッシュ値の偏りを軽減する手法として，適当なビット数をずらして重ね合わせたり，ビット順を入れ換えて折り畳む方法等が知られている．ROP では，分割した切片に乗算を施した後に，得られた結果を一定ビット回転して重ね合わせる乗算重ね合わせ法（以下では RINDA 方式と呼ぶ）をハードウェア化した．乗算は文字コードの偏りを分散するのに，回転重ね合わせは可変長キーを同一のハードウェアで扱うのに有効な手法といえる．

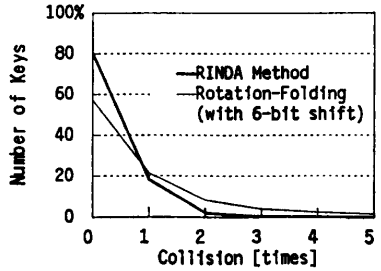
RINDA 方式の有効性を検証するために実際のデータを用いて評価した結果を**図3**に示す．図の横軸は，同一のハッシュ値に衝突した回数を，縦軸はキーの割合を示している．従来法である回転重ね合わせ法は，回転数を0から7まで変えて評価を行い，最も衝突の発生が少なかった回転数6の場合を示している．図3(a)は長さ16バイト・固定長のランダム数字列を，図3(b)は英語辞書の見出し語を用いた場合であり可変長の英単語である．いずれの場合であっても RINDA 方式が優れているといえるが，特に，キーの分布に偏りがある英単語の例では，回転重ね合わせ法の衝突なしの比率が 60% 未満に対して，RINDA 方式は 80% の単語が衝突なしである．また，キーの衝突回数で比較すると，RINDA 方式は，ランダム数字列と英単語でほぼ同一の結果が得られているのに対して，回転重ね合わせ法は，文字コードの分布やキー長に依存しており，英単語の例では，ハッシュ値が衝突するキー数が5を越える割合も 10% 程度と高い．RINDA 方式では，いずれの評価でも3個以上のキーのハッシュ値はほとんど衝突しなかった．

次に，ハッシュ表の大きさ（セル数）が衝突の度合いに与える影響を示す．ビットアレイの大きさを変えて所定の個数のキーをハッシュすることによって，前述のロードファクタを変えて評価した．ハッシュ関数に RINDA 方式を用いて，ロードファクタを1から1/8まで変えた時の衝突の度合いを評価した結果を**図4**に示す．評価に用いたキーは，図3(b)と同一である．ロードファクタが1の時，すなわち，キーの個数とセルの個数が等しい時は，約4割のキーが衝突なし，残りの約4割のキーが1回の衝突，2割のキーが2回以上の衝突を起こしている．ロードファクタが1/4以下になると，衝突なしの割合が8割を超え，2回以上衝突する確率はほぼゼロとなる．

以上の結果より，ハッシュ化ビットアレイを用いたふるい落とし処理では，ロードファクタを小さくする，すなわち，ビットアレイのサイズを大きくすることによってキーの衝突を回避できる．すなわち，ハッシュするキーの個数が増加するに伴って，ビットアレイのサイズを大きくすると，衝突率を一定にできる．



(a) ランダム数字列〈16バイト固定長〉
(a) Random numeric 〈16-byte fix〉.



(b) 英単語〈可変長〉
(b) English words 〈Variable length〉.

図3 RINDA 方式におけるハッシング効果
Fig. 3 Hashing effect on RINDA method (235 k keys/1 M-bit cells).

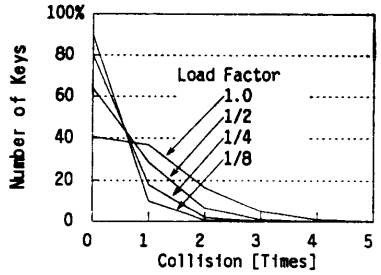


図4 ロードファクタに対するハッシング効果の違い
Fig. 4 Hashing effect for load factor.

5. 結合処理の実現と性能評価

5.1 RINDA での結合処理の実現

フィルタフェーズとソートフェーズを実現した ROP の内部構成を**図5**に示す．回路ブロックの構成については文献17)に詳しいので，ここでは結合処理を実現する上での ROP の特徴を示す．

(1) 動的最適化方式：結合対象となる表を CSP を用いてアクセスする際に，選択・射影後の一時表に含まれる行数をカウントし，この値を基に最適な結合方式を決定する動的最適化を実現している[16]．第1および第2の一時表が小さく，ディスク上にワーク域を作成せずに結合できるときは，ROP は使用せずに CSP で得られた検索結果の一時表をホスト計算機上でネステッドループ結合する．それ以外の場合は，ROP を用いた3フェーズジョインを行う．第1の一時表のサイズが第2の一時表のサイズより十分に小さい時は片ハッシュ・ソートマージ結合法を，上記以外，すなわち，第1および第2の一時表のサイズが大きくかつ均衡している場合，および，ソータ容量を超えてオーバフローが起こる場合は，両ハッシュ・ソートマージ結合法を使用する．

(2) 内部キー方式：一般に，表は複数の異なる属性を持つ列からなり，属性値のタイプには，整数，符号あり／符号なしの十進数や文字列等がある．また，属性値として値が未定のナルが許されている場合もある．これらの多種多様な属性の集まりからなる行を直接ソートする試み[18]もあるが，ハードウェアの複雑さから実現上の限界があった．

ROP では，フィルタフェーズにおけるハッシュ関

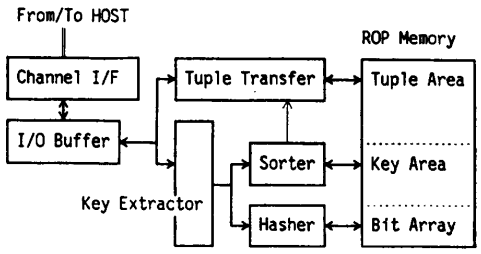


図5 ROP の構成
Fig. 5 ROP block diagram.

数の入力キー，およびソートフェーズで使用するキーを ROP の内部キーとして抽出する専用ハードウェアを設けている．この内部キーは，キーの先頭位置からの比較で大小関係が判定できる絶対値数で，可変長文字列やナル値は定義長で固定長化している．内部キーを専用ハードウェアで作成することにより，異なる属性を持つフィールドを複数個連結したキーを，高速にフィルタリングあるいはソーティングできる．

(3) 作業メモリ方式：ROP では，行データから作成した内部キーを用いてフィルタリングおよびソーティングを行う．処理結果は一時表として返却することから，内部キーとは独立に行データを格納している．内部キーの最後部に行データの格納位置を示すポインタを付与し，ソート後にポインタから行を読み出して一時表を作成する．したがって，ROP 内部には，内部キー，行データ，および，フィルタリングで使用するビットアレイを格納するための記憶部が必要となる．これら3種類のデータを独立のメモリ装置に格納したのでは，メモリの使用効率が低くなるばかりかメモリの実装密度も低下し，小型化・低価格化を阻害する要因となる．

ROP では，大容量メモリチップを高密度に実装した1つの作業メモリを，行格納域，内部キー格納域，およびビットアレイ格納域に動的に分割して使用する．ビットアレイ域の大きさは，ハッシュ値のロードファクタを一定として衝突率を低く保つために，作業メモリの一定比率の領域を割り当てる．残りの領域を行格納域と内部キー格納域に分割して使用するが，行の長さは可変長で，最悪条件では行ごとに長さが異なる場合がある．このため，行を格納するごとに作業メモリのオーバフローを検出する回路を設けて，真に作業メモリが満杯になるまで処理できるように設計してある．

5.2 性能評価

RINDA を用いることによる結合処理の高速化効果をウィスコンシンベンチマーク[19]により評価した．結合する表の大きさが1万行×1万行と 10 万行×10 万行の2つの場合について，それぞれ，結合する一方の表に選択条件を付けて行数を 10% に絞り込んでから結合する．

評価に用いたシステムは，ホスト計算機が小型汎用機 DIPS-V 30 E[20]，CSP が2台，ROP が1台と，容量が 1.3 GB でデータ転送速度が 3 MB/秒のディスクおよび制御装置各2台で構成した．汎用計算機上のデータベース管理システムは，オプション指定によって RINDA を使用する場合としない場合を選択できる．評価は，問い合わせを発行してから結果をすべて一時表に格納完了するまでの経過時間をホスト計算機上で測定して行い，**図6**では RINDA を使用した場合の経過時間を基準として，CPU 処理時間と IO 処理時間を分離して示した．RINDA を使用した場合の CSP と ROP の処理時間は，ホスト計算機からは IO 時間に含まれる．

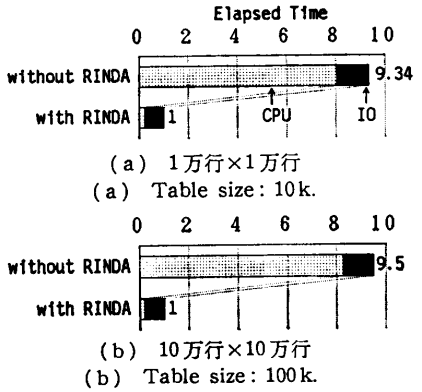


図 6 RINDA による結合処理の高速化
Fig. 6 Performance improvement of join operation (Normalized by time with RINDA).

図6の結果から，RINDA を用いることで結合処理時間を約 1/10 に短縮できることがわかる．また，結合処理対象の表が1万行から 10 万行へと大きくなった場合でも一定比率の高速化効果が得られている．特に，ホスト計算機での CPU 処理時間の短縮効果が大きく，RINDA 使用でマージ結合時間を大幅に短縮できるといえる．以下では，RINDA を用いて高速化が図れた要因について考察する．

本問い合わせを RINDA で実行した場合は，CSP で 10% の選択を行うことから，5.1 節(1)に示した動的最適化機構により片ハッシュ・ソートマージ結合法を用いて処理する．10万行の表同士の結合を例に示すと，最初に，1つの 10 万行の表から 10% 選択した結果である1万行の一時表を ROP に入力してハッシュ化ビットアレイを設定するとともに，ソートされた1万行の一時表を出力する．次に，他方の 10 万行の一時表を ROP に入力して先に設定したハッシュ化ビットアレイを用いてふるい落としを行う．ウィスコンシンベンチマークでは，結合するキーはユニークであるから，ハッシュ値に衝突がなければ，ROP から1万行だけがソートされた一時表として出力される．すなわち，ROP のふるい落とし機能によって，出力さ

れる一時表の大きさが 1/10 となり，一時表の作成時間と転送時間が削減される．ホスト計算機上でマージ結合する一時表は，結合可能性のない行が削除され，かつソートされていることから，CPU 時間も大幅に削減されている．

6. おわりに

関係データベース処理の中で，特に，高速化が要求されている結合処理の高速化手法について示した．データベースマシン RINDA は，結合処理を含む非定型の問い合わせ処理を専用ハードウェアを用いて高速化している．RINDA で実現した結合処理の特徴を以下に示す．

（1） 結合する表から結合可能性のない行をふるい落とすフィルタフェーズ，残った行を並べ替えるソートフェーズ，ソートされた行をマージしながら連結するマージ結合フェーズからなる3フェーズジョイン法を実現した．ハードウェア化の容易性とシステム構成の柔軟性とから，フィルタフェーズとソートフェーズを専用ハードウェア化した．

（2） ハッシュ化ビットアレイを用いたフィルタフェーズのハッシュ関数として，キーの長さ，属性が変わっても衝突が起こりにくい乗算重ね合わせ法を用いた．

（3） ネステッドループ法，片ハッシュ・ソートマージ法，両ハッシュ・ソートマージ法の3種類の結合法を実現し，結合する2つの表の大きさに基づいて最適な手法を実行時に選択する動的最適化を実現した．

（4） ベンチマークを用いた性能評価を行い，従来のソフトウェア処理と比較して，結合処理の性能を1桁向上できた．

RINDA は，大規模データベース処理分野で，特に，結合処理や統計処理等の非定型な検索処理を多用するシステムに導入され，十分な処理時間の短縮効果が得られている．

謝辞 本研究の機会を与えてくださったNTT情報通信網研究所松永俊雄プロジェクトリーダ，拜原正人情報処理研究部長，ならびに，性能測定にご協力いただいた中村敏夫主任研究員，黒岩淳一研究主任に感謝いたします．

参考文献

1) 喜連川優，伏見信也：データベースマシン，情報処理，Vol. 28, No. 1, pp. 56-67 (1987).
2) Babb, E.: Implementing a Relational Database by Means of Specialized Hardware, *ACM Trans. Database Systems*, Vol. 4, No. 1, pp. 1-29 (1979).
3) 安藤隆朗，小宮富士夫ほか：リレーショナルデータベースプロセッサ GREO の構成，信学技報，DE 89-37 (1989).
4) 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).
5) Britton Lee, Inc.: Server/8000 for Use with ShareBase II Software (1988).
6) 速水治夫，井上 潮ほか：リレーショナルデータベースプロセッサ RINDA のアーキテクチャ，情報処理学会計算機アーキテクチャ研究会資料，88-ARC-73-12, pp. 85-92 (1988).
7) 井上 潮，速水治夫ほか：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 (1990).
8) 井上 潮，北村 正ほか：情報提供サービスに適用可能な超大規模リレーショナルデータベースマシン，情報処理学会データベース・システム研究会資料，47-5 (1985).
9) Lum, V. Y., Yuen, P. S. T. and Dodd, M.: Key-to-Address Transform Techniques: A Fundamental Performance Study on Large Existing Formatted Files, *Comm. ACM*, Vol. 14, No. 4, pp. 228-239 (1971).
10) Knott, G. D.: Hashing Functions, *Comput. J.*, Vol. 18, No. 3, pp. 265-287 (1975).
11) Boral, H. and Redfild, S.: Database Machine Morphology, *Proc. of 11th Int. Conf. on Vary Large Databases*, pp. 59-71 (1985).
12) Blasgen, M. W. and Eswaran, K. P.: Storage and Access in Relational Data Bases, *IBM Syst. J.*, No. 4, pp. 363-377 (1977).
13) Kitsuregawa, M., Tanaka, M. and Moto-oka, T.: Application of Hash to Data Base Machine and Its Architecture, *New Generation Computing*, Vol. 1, No. 1, pp. 62-74 (1983).
14) McGregor, D. R., Thomson, R. G. and Dawson, W. N.: High Performance Hardware for Database Systems, in *Systems for Large Data Bases*, Lockemann, P. C. and Neuhold, E. J. eds., North-Holland Publishing Company, pp. 103-116 (1976).
15) 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).
16) 中村仁之輔，板倉一郎ほか：データベースプロセッサ RINDA の最適化方式，第37回情報処理学会全国大会論文集，5Q-8, pp. 385-386 (1988).
17) 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 (1990).

18) 伊藤文英，島川和典ほか：可変長レコード用関係データベース処理エンジンの試作とソート処理性能の評価，情報処理学会論文誌，Vol. 30, No. 8, pp. 1033-1045 (1989).
19) Bitton, D., DeWitt, D. J. and Turbyfill, C.: Benchmarking Database Systems—A Systematic Approach, CTSR # 526, Univ. of Wisconsin-Madison (1983).
20) 矢沢良一，平野正則，山口利和，岡田靖史：DIPS-V 30 E 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).

(平成2年9月10日受付)
(平成3年5月 7 日採録)

佐藤 哲司（正会員）

昭和32年生．昭和55年山梨大学工学部電子工学科卒業．同年，日本電信電話公社入社．主に論理回路の大規模集積技術，ハードウェアソータ，データベースマシンの研究に従事．現在，NTT 情報通信網研究所主任研究員．電子情報通信学会，IEEE 各会員．

武田 英昭（正会員）

昭和30年生．昭和54年北海道大学工学部電気工学科卒業．昭和56年同大学院工学研究科電気工学専攻修士課程修了．同年，日本電信電話公社入社．以来，データベースマシンの研究開発に従事．現在，NTT 総合企画本部担当課長．IEEE 会員．

井上 潮（正会員）

昭和28年生．昭和50年名古屋大学工学部電気学科卒業．同年，日本電信電話公社入社．現在，NTT 情報通信網研究所主幹研究員．オンライン情報検索システム，データベース管理システム，データベースマシンの研究実用化に従事．電子情報通信学会，IEEE-CS 各会員．

福岡 秀樹（正会員）

昭和25年生．昭和47年大阪府立大学工学部電子工学科卒業．同年，日本電信電話公社入社．現在，NTT 関西テクニカルサポートセンタ所長．主に DIPS 本体系装置の研究実用化を行い，現在データベースマシンの研究実用化に従事．電子情報通信学会会員．