関係演算高速化プロセッサ†

武田英昭†† 佐藤哲司††
中村敏夫†† 速水治夫††

リレーショナルデータベースプロセッサ RINDA のコンポーネントである関係演算プロセッサ ROP の構成とそこで用いている処理アルゴリズムについて述べる．ROP は，リレーショナルデータベースの基本的な処理であり，かつ，汎用計算機では処理負荷の重い，ソート，結合といった関係演算を高速化する専用ハードウェアである．ソートは内蔵するソートハードウェアで高速化している．結合演算は，処理をふるい落とし，ソート，マージ結合の3つのフェーズに分け，このうち，ふるい落としとソートを ROP に内蔵した専用ハードウェアで処理することにより高速化している．ふるい落としフェーズは，ハッシュ化ビットアレイを用いて実現しており，そこではデータ長の変化に強いハッシュ関数を用いている．ソートフェーズではソート件数の増加に対する拡張性に優れた多段マージソートを採用している．本論文では，ROP の構成と処理アルゴリズムについて述べた後，ベンチマークを用いた性能実測を行い，従来のソフトウェアのみのシステムに比べ，ソート，結合処理，グループ化計数が 10 倍～100 倍程度高速化されることを示す．

1. はじめに

近年，データベースマシンは研究段階から製品化の段階へと移ってきている．データベースマシンの目的はデータベース処理の高速化であり，現在製品化されているデータベースマシンは，高速化のための基本的な実現手法から次の2種類に分類できる．1つは，複数の汎用プロセッサで並列処理を行うことで高速化を図る並列処理型データベースマシンであり，代表例として DBC/1012[1] がある．汎用並列計算機にチューニングされたデータベース処理ソフトウェアを搭載したものもこの範疇に入るといえる．他の1つは，ソフトウェアでは負荷の重い処理を専用ハードウェアにより実現することで高速化を図る専用ハードウェア型データベースマシンであり，Server/8000[2] がこれにあたる．

汎用プロセッサの性能向上にはめざましいものがあり，並列処理型データベースマシンも将来的には期待できる．しかし，現状の商用プロセッサを用いて飛躍的な高速化を達成するには高い並列度が必要であり，必然的に，装置規模が大きくなる．

著者らは現状のソフトウェアによるデータベース処理において弱点とされている，インデックスの利用が困難な非定型の検索処理を専用ハードウェアを用いて高速化することにより，データベース処理の総合的な性能向上を目指した専用ハードウェア型データベースマシン RINDA (RelatIoNal DAtabase processor)[3),4)] を開発した．RINDA では，リレーショナルデータベース処理において基本的な演算であり，かつ，従来のソフトウェアでは負荷の重かった，サーチ処理，ソート処理，結合演算等を高速化している．

RINDA はハードウェアコンポーネントとして，サーチを高速化する内容検索プロセッサ (CSP : Content Search Processor) と，ソート，結合演算を高速化する関係演算プロセッサ (ROP : Relational Operation accelerating Processor) からなる．

本論文では，文献4)で示された RINDA の基本的な設計方針に基づいて，ROP の構成，そこで用いている処理アルゴリズム，および，性能評価結果を示す．

ソート処理は，ROP 内のハードウェア（ソータ）により高速化している．結合演算は，結合可能性のないタプルを除去するふるい落としフェーズ，結合対象となるキー* を昇順に並べ換えるソートフェーズ，および，ソート後のキーのマージとタプルの連結を行うマージ結合フェーズからなる3フェーズジョイン方式[4)] で実現し，前2フェーズを ROP に内蔵する専用ハードウェアで高速に処理する．

ふるい落としフェーズの特徴は，ハッシュ化ビットアレイ[6)] を設定・参照するためのハッシュ関数にある．ハッシュするデータを一定の長さに区分してから定数との乗算を施し，その結果をシフトしながら重ね合わせることで，種々の長さのデータを効率良くハッ

† An Accelerating Processor for Relational Operations by HIDEAKI TAKEDA, TETSUJI SATOH, TOSHIO NAKAMURA and HARUO HAYAMI (NTT Communications and Information Processing Laboratories).

†† NTT 情報通信処理研究所

* 処理対象となるアトリビュートの集まりを本論文ではキーと呼ぶ．

シュしている．ソートフェーズは，マルチウェイマージ処理を繰り返す多段マージソート法[8]を専用ハードウェアを用いて高速化している．ソータは，一次元アレイ構造のソートアレイ[7]を用いて並列比較を行うことで大量データの高速ソートを実現している．

さらに，ROP では，従来のデータベースマシンであまり考慮されていなかった専用ハードウェアへのデータの入出力に伴う処理対象アトリビュートの切り出し処理，および，使用頻度の高い統計処理の1つであるグループ化計数処理も専用ハードウェア化している．

2章では RINDA の概要について述べ，3章では ROP のハードウェア構成について述べる．4，5章では，各々ふるい落とし，ソートブロックの構成とそこで用いている処理アルゴリズムについて述べる．最後に，6章で ROP の性能評価結果を示す．

2. RINDA の概要

2.1 開発目的

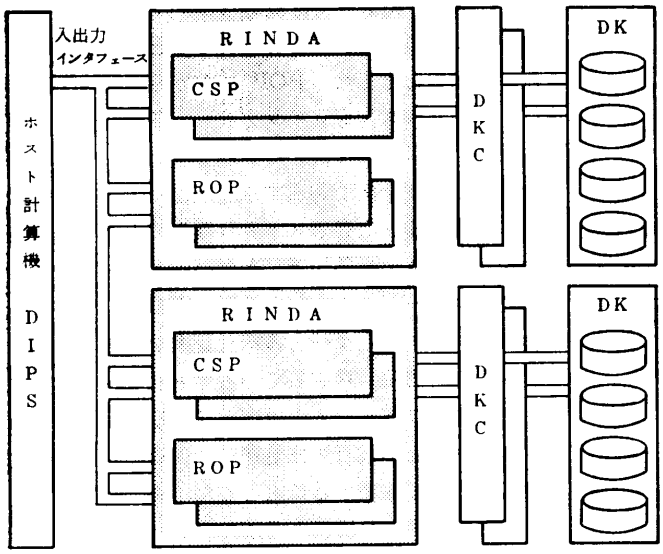
リレーショナルデータベース処理は，特定のアトリビュートをキーとして検索／更新を行う定型処理と，不特定のアトリビュートを用いて種々の検索／更新を施す非定型処理に分けられる．一般のリレーショナルデータベース処理ソフトウェアでは，インデックスを利用して定型処理を実用上十分な速度まで高速化している．

しかし，非定型処理には一般にインデックスを利用できないことから，従来のソフトウェアのみによる処理では多大の時間を要していた[9]．主な原因は2つ考えられる．1つは，主記憶に読み出してきたリレーションに対し逐次的に条件判定を行わなければならず，そのためのサーチ処理に時間がかかることである．他の1つは，非定型処理では単一のリレーションのグループ化，または複数のリレーションを対象とする結合処理，副問合せを伴うことが多く，それに伴うソート処理に時間がかかることである．

以上述べた2つの問題点を解決するために，サーチ処理とソート処理を専用ハードウェアで実行するデータベースマシン RINDA を開発した[4]．

2.2 RINDA の構成

ホスト計算機に RINDA を接続したシステム構成例[4]を**図1**に示す．内容検索プロセッサ CSP は，ディスクに格納されたリレーションをサーチし，検索条件に合致するタプルの選択とアトリビュートの抽出を行い，結果をホスト計算機の主記憶に転送する．関係演算プロセッサ ROP は，ホスト計算機の主記憶上にあるリレーションを一度すべて取り込み，ソート，結合演算の前処理等を行った後，それらの処理結果をホスト計算機に転送する．CSP と ROP を独立のハードウェアとしたことにより，システムごとに異なるサーチ処理とソート処理との負荷バランスに対応できる構成となっている．各 CSP, ROP は，それぞれ独立に入出力インタフェースでホスト計算機に接続され，ホスト計算機からのチャネルコマンドにより制御される．ホスト計算機と CSP/ROP 間のデータ転送は複数のタプルが格納されたページ単位で行われる．



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

図1 RINDA を使用したシステム構成例
Fig. 1 Typical system organization with RINDA.

3. ROP の構成

3.1 基本思想

ROP は，非定型処理で従来問題となっていた処理（2.1 節）のうちで，ソートをベースとした以下の3種類の処理を専用ハードウェア化することとした．

① ソート処理
② グループ化計数処理
③ 2リレーションの結合演算処理

ソフトウェアによるこれらの処理の CPU

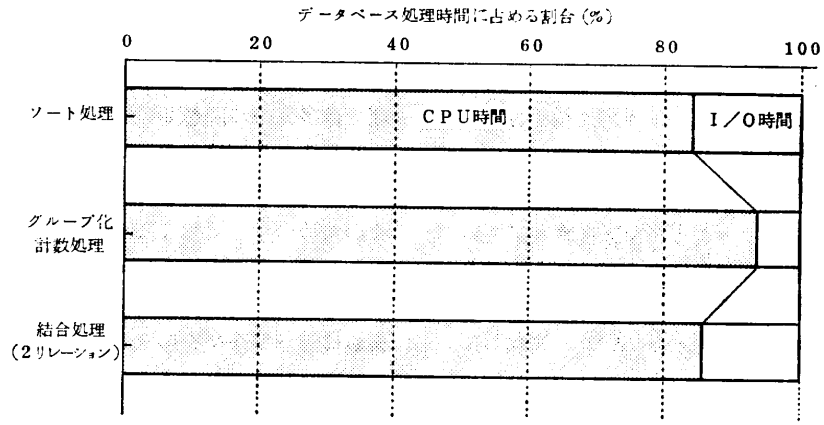


図 2 データベース処理時間の内訳
Fig. 2 The ratio database processing time.

時間と I/O 時間の割合を**図 2** に示す．図 2 で評価の対象としているリレーションは，ソート，グループ化計数が 1 万件，結合が 1,000 件と 10 万件，タプル長は各々 208 バイトであり，小型汎用機の DIPS-V 30[10] 上で測定した．いずれの処理も CPU 時間の割合が高く，この部分を専用ハードウェア処理することにより 5〜10 倍程度の高速化が達成できる．さらに，大容量バッファを設ける等により，I/O を高速化することで 10 倍以上の処理の高速化が期待できる．

ROP では，①〜③の CPU 処理から演算回数の多い処理要素を抽出し，それらを専用ハードウェア化した．専用ハードウェア化する機能の選定は，処理対象となるリレーションのタプル数 (n) を基準に，これよりも演算回数の多い処理要素を対象に行った．以下に①〜③の処理に対する対処方針を述べる．

①に対して：ソートの計算時間は，概ね，ソート対象となるデータの比較回数に比例する．ソフトウェアで一般に用いられるソートの比較回数は，ソートするデータ数 n に対して $O(n \log_2 n)$ である．この値は，n の増加に対して，n のさらに $\log_2 n$ 倍で増加し，これが CPU 時間が大きくなる原因となっている．$\log_2 n$ の底 2 は比較処理する CPU が 1 台であることに起因している．ROP では，並列比較を行う専用ハードウェアを用いて対数の底の値を大きくし，ソート処理時間を短縮することとした．さらに，ソート作業用のメモリを搭載することで，二次記憶へのアクセス回数の削減を図った．

②に対して：グループ化計数は他の問合せに比べ，CPU 時間の割合が高い．これは，データのソートに加え，ソートされたデータを再び走査して計数（加算）する必要があるためである．ROP では，ソータにハードウェアのグループ化計数回路を付加し，計数処理の高速化を図ることとした．

③に対して：ソフトウェアによる結合処理は，結合対象となるアトリビュートにインデックスが定義されていない場合，一般に，ソートマージ法[11]を用いている．ソートマージ法では，前述のソート処理と同様に，各リレーションのタプルのソート処理に CPU 時間を要する．さらに，各リレーションごとにソートされたタプルをマージ結合する時間も結合処理を遅くする要因である．RINDA でもソートマージ法を基本とするが，ROP によって，高速なソートと結合可能性のないタプルのふるい落としを行うことで，結合時間を削減することとした．ROP における結合演算処理方式（3 フェーズ・ジョイン方式）の開発背景，処理方式については次節において詳述する．

①〜③の処理全体に共通する高速化：①〜③の処理では，二次記憶およびホスト計算機上でのデータの格納単位はタプルである一方，専用ハードウェアでの処理の対象はキーであるという処理対象の‘ミスマッチ’が存在し，タプルとキーとの相互変換が必要となる．タプルとキーでは，構成しているアトリビュートの書式およびアトリビュートの順序が異なる．ROP では，タプルとキーとの相互変換処理を，独立した専用ハードウェアとして搭載することとした．また，大容量メモリを設け，キーに変換する前のタプルを格納しておくことにより，変換作業に伴う二次記憶へのアクセスをなくし，一層の高速化を図った．

3.2 3フェーズ・ジョイン方式に基づく結合演算処理手法

3.2.1 基本アルゴリズム

結合演算処理の基本アルゴリズムは，次の2つに大別できる．なお，本節における演算回数の議論では，2リレーションのタプル数を各々 m,n $(m \leqq n)$ とし，結合対象キーはユニークなものと仮定している．

（1） ネステッドループ法[11]

一方のリレーションの各タプルの結合キーに対し，他方のリレーションの全タプルの結合キーを比較することにより結合する方法であり，演算回数はオーダ $n \times m$ となる．

（2） ソートマージ法[11]

2リレーションを結合キーで各々ソートした後，各リレーションの先頭タプルから順に結合キー同士を比較することにより結合する方法であり，演算回数は，オーダ $n \log_2 n$ となる．

データベースマシンを実現するに際し，上記の基本アルゴリズムの演算回数をオーダ n に近づけ，処理を高速化する手段として以下の方法が提案されている．

(1)に対して：ハッシュ表[12]

一方のリレーションの各タプルを結合キーでハッシュし，ハッシュ表の対応するエントリにタプルを格納する（オーダ m の処理）．その後，他方のリレーションをハッシュし，ハッシュ値に対応するハッシュ表のエントリに格納されているタプルとの比較，結合を行う．後半の処理は，ハッシュ値に衝突がなければ，比較は1回で済み，前半の処理も含めてオーダ n で結合処理を行うことができる．

(2)に対して：ソータ[13]

ハードウェア・ソータによりソートの比較を並列化し，ソートのための比較回数を n もしくは，n に近いオーダに下げることで，結合処理の演算回数をオーダ n に近づける．

結合処理の演算回数は，キー値の分布特性に依存するが，ハッシュ表，あるいは，ソータを用いることによってオーダ n に近づけることができる．ROP では，ソートの高速化のためにソータを搭載することとしているため，結合演算においてもこれを積極的に用いるソートマージ法を基本アルゴリズムとした．

3.2.2 3フェーズ・ジョイン方式

巨大なリレーションの結合処理では，各リレーションのソートを高速化するだけでは，ソート後のリレーションのマージ結合に時間を要し十分ではない．例えば，100 万件のリレーション同士をマージ結合するには，小型汎用計算機クラスで数分のオーダになる．また，ソートのためにタプルを一時的に格納するメモリも大きくなり，1タプルの長さを 200 バイトとすると，リレーションあたり 200 M バイトのメモリが必要となる．このメモリ量は，現状の装置コストを考えると現実的でない．したがって，巨大リレーションを対象とする業務に装置を適用する場合，結合対象タプルをソートマージの前に削減することが重要である．これは，結合処理の基本アルゴリズムとして，ネステッドループ法を採用しても同様である．

結合可能性のないタプルを削減する手段としては，ハッシュ化ビットアレイを用いたふるい落としが従来より提案されている[6]．ROP ではこのふるい落としをソートマージ法の前処理として利用することとし，さらに，ふるい落とし処理，ソート処理，および，ソート後のタプルのマージ結合処理をパイプライン的に重畳することにより，巨大リレーションの結合処理を高速に行うことを可能とした．このように，結合するリレーションから結合可能性のないタプルを除去するふるい落としフェーズ，残ったタプルを結合キーの値でソートするソートフェーズ，ソート済みのタプルを比較し，結合キー値の一致するタプルを連結するマージ結合フェーズの3つのフェーズで，パイプライン的に結合処理を行う方式を3フェーズ・ジョイン方式と呼ぶ．

3.2.3 3フェーズ・ジョイン方式の実現法

3フェーズジョイン方式は，以下の考え方に基づいて，ふるい落とし，ソートフェーズを ROP で実現し，マージ結合フェーズはホスト計算機で行わせている．

① ふるい落としフェーズでは，結合する2リレーションの一方，または，両方から結合可能性のないタプルをふるい落とす．前者の場合だと各リレーションを一度ずつ，後者の場合だと一方のリレーションは一度，他方のリレーションは二度処理する必要がある（4章参照）．タプル単位にハッシングを行い，ビットアレイを設定／参照するための処理量はきわめて大きくなる．

② ソートフェーズは，オーダ $n \log_2 n$ と処理量が多いことから，ROP に搭載されたソータを利用する．

③ マージ結合フェーズは，結合後の処理，例えば，ユーザへの最終出力，主問合せ等のために様々な

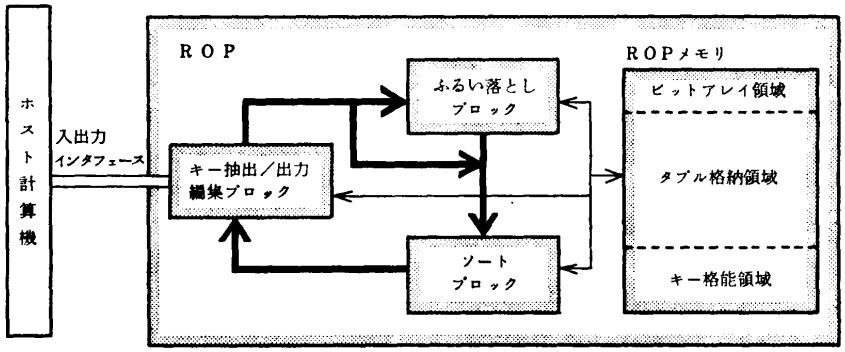


図 3 ROP のハードウェア構成
Fig. 3 ROP hardware configuration.

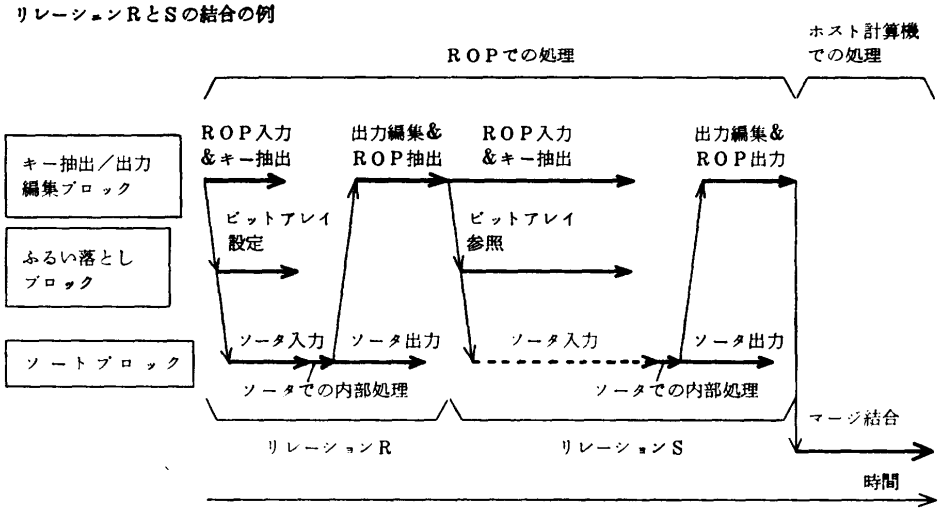


図 4 ROP における結合処理の流れ
Fig. 4 Join operation flow by ROP.

パターンのタプル編集処理が必要となる．これは専用ハードウェア，特に，LSI 化には不向きである．また，マージ結合は既にソートが行われており，オーダ n の処理であること，さらにふるい落とし処理によって，結合対象タプル数が原リレーションのタプル数よりも減少していると考えられる．

以上のように，ROP では従来から提案されているふるい落とし，ソートマージといった手法を効果的に組み合わせ，専用ハードウェア化を考慮して抽出した3つの処理要素を，ホスト計算機との役割分担の中で，各々に適した実現形態で実機上にインプリメントしたことに特徴がある．

3.3 ハードウェア構成

ROP の構成を**図 3** に示す．ROP は3つの機能ブロックと1つのメモリからなる．ふるい落としブロック，ソートブロックは各々3フェーズ・ジョイン方式での対応する機能を実現している．ソートブロックは，グループ化計数機能も実現している．

キー抽出／出力編集ブロックの機能は以下のとおりである．

（a） キー抽出機能

① キー抽出：タプルから ROP での処理対象となるキー・アトリビュートを抽出する．

② キー・アトリビュート順序の並べ換え：ソート／結合の重みの順にキー内のアトリビュート順序を並べ換える．

③ ROP 内部処理データ形式*への変換：キーのアトリビュートのデータ型を ROP 内部処理デー

* 固定長で，キーの先頭バイトから比較することによりキー値の大小が判定できるデータ形式である．

タ形式に変換する.

(b) 出力編集機能

ROP で処理した結合キーあるいはソートキーに対応するタプルをページ形式に編集する.

ふるい落とし，ソートブロックについては，各々4，5章で述べる.

ROP メモリは単一のアドレス空間であり，ホスト計算機から転送されたタプルの格納，ソート作業(キー格納領域)，および，ふるい落としで用いるビットアレイの格納のために使用する.

キー抽出→ふるい落とし→ソート，および，ソート→出力編集は各々タプル単位のパイプラインで処理する. **図4** に結合演算を例にとり，ROP の典型的な処理フローを示す.

4. ふるい落としブロック

4.1 機能と構成

ふるい落としブロックの目的は，結合可能性のないタプルの除去にある. ふるい落としブロックでは，原リレーションのタプル数の比により，一方のリレーション，あるいは，両方のリレーションをふるい落とす機能を持つ.

ふるい落としブロックの構成を**図5** に示す. ビットアレイは両方のリレーションをふるい落とす場合のために2つある. 一方のリレーションのみふるい落とす場合は，いずれかのビットアレイを使用し，図4に示した手順でふるい落としを行う. 両方のリレーションをふるい落とす手順は以下のとおりである.

① ビットアレイA，Bのすべてのビットを '0' に初期化する.

② 一方のリレーションの結合キーをハッシュし，ハッシュ値に対応するビットアレイAのビット位置に '1' を書き込む (ビットアレイの設定).

③ 他方のリレーションの結合キーをハッシュし，ハッシュ値に対応するビットアレイAのビット位置を読み出し，そこが '1' であれば，そのキーをソートブロックに転送する (ビットアレイの参照). また，このハッシュ値を用いてビットアレイBを設定する.

④ ②で処理したリレーションを再びハッシュし，ビットアレイBの参照処理を行う.

4.2 ハッシュ関数

ふるい落としの効果を左右する主要な要素としてハッシュ回路で用いるハッシュ関数がある. ふるい落としブロックのハッシュ関数に求められる条件は，任意のデータ型のキー，および，任意の長さのキーをビットアレイ上に一様に散らすことである.

ハッシュのアルゴリズムは，従来より，種々のものが提案されているが，散らし効果が良いアルゴリズムとして乗算法や除算法がある[14]. しかし，これらの方法は，数値型データ等の固定長の短いデータのハッシュを対象としており，文字列データのような長いデータや可変長データをハッシュすることはできない. 一方，可変長データを重ね合わせることで固定長化し，それをハッシュ値とするアルゴリズムとして，排他的論理和法[14]がある.

ふるい落としブロックでは，上記の乗算法と排他的論理和法を併用することで，種々のデータ型のキーを効果的にハッシュできるようにした. さらに，繰り返し文字列パターンのキーでハッシュ衝突* を避けるため，排他的論理和で重ね合わせる前に一定ビットの回転**を施した.

ふるい落としブロックのハッシュ回路の構成を**図6** に示す. 乗算は乗算結果が格納されている乗算表と読み出し回路で実現している. ハッシュ回路内は，キー分割回路で分割された部分キー単位にパイプライン処理され，ROP へのタプル入力速度に完全に追

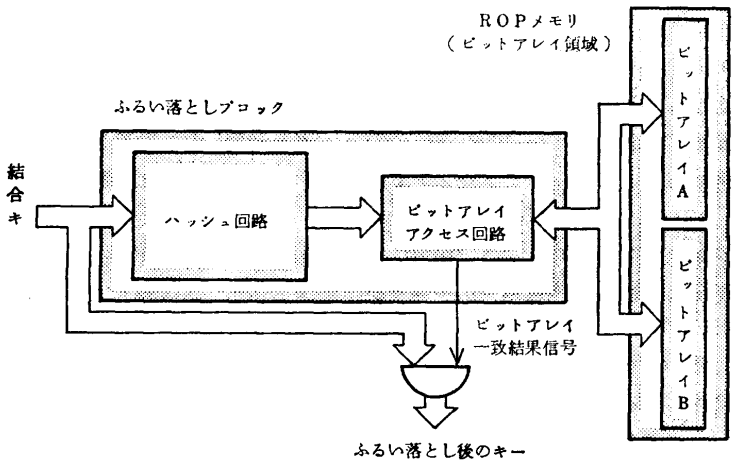


図5 ふるい落としブロックの構成
Fig. 5 Filtering block configuration.

* 例えば，日本語データ '2番地' と '22番地' を2バイトずつ重ね合わせると同一のハッシュ値になる.

** 一定ビットの回転は，ハードウェアでは配線をずらすだけで容易に実現できる.

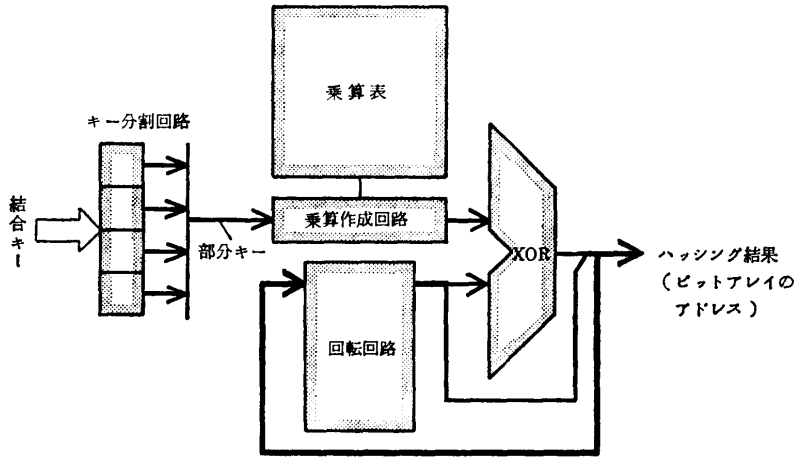


図 6 ハッシュ回路の構成
Fig. 6 Hashing circuit configuration.

従して動作する.

5. ソートブロック

5.1 従来のソータにおける問題点と解決法

ソートブロックは，キーのソートとグループ化計数，および，キーの重複除去機能を持つ．以下では，ソートブロックにおけるソート手法について議論する.

ソータは従来より種々の形式のものが提案されているが，データベース処理でよく用いられるものとしては $\log_2 n$ 型ソータ[15),16)]が知られている．しかし，$\log_2 n$ 型ソータを ROP に用いるには以下のような問題点がある.

① ROP では適用領域を広げる意味から，大量データのソートが必須である．$\log_2 n$ 型ソータで大量データのソートを行うには，比較器を実現するのに要するハードウェア量が多い．例えば，100 万件のタプルのソートを可能とするには，1 LSI/比較器として，20 個の LSI が必要である．1 枚の基板に搭載可能な LSI 数を考慮すると，数個以下に抑えることが望ましい.

② 製品化を前提に考えると，製品ランクによってソート可能件数を容易に変えられる必要がある．$\log_2 n$ 型ソータでソート可能件数を変更するには，ソート用作業メモリ量の変更に加え，比較器 LSI 数の変更も必要である．種々のランクの製品への対応を容易にするには，作業メモリ量の変更のみに抑える必要がある.

これらの問題を解決するため，著者らは単純マルチウェイマージ* を繰り返して行うことによりソートを実現する多段マージソータ[8)]を開発した．本ソータを用いることで上記の問題は以下のように解決した.

①に対して：マージの繰返しでソートを行うことにより，大量のデータのソートも一定個数の比較器 LSI で可能とした．また，2 つのデータを比較する構造が簡単な比較器を，1 LSI へ複数個実装することとした．この結果，数個の LSI で数十ウェイのマージを実現し，これらの LSI を用いて作業メモリが許す限り，いかなる大量データのソートも可能とした.

②に対して：マージを繰り返してソートすることにより，ソート可能件数の変更に対しては，マージを行う比較器 LSI 数は一定のままで，ソート作業メモリ量の変更だけで対応可能とした.

一方，繰り返してマージするため，ソータ（ソートブロック）へのキー入力完了後，キーの出力を開始するまでに‘遅れ時間’がでる．しかし，この‘遅れ時間’は以下の理由でソータへの入出力時間に比べて十分短い.

（a） ROP への入出力はタプル単位であり，一方，‘遅れ時間’中の処理の単位はキー（その長さは常にタプル長以下）である.

（b） 一般のソート用ソフトウェア（8〜16 ウェイマージソート）に比べマージウェイ数が大きいため，繰返しの回数は一般のソート用ソフトウェアより少なくて済む．例えば，マージウェイ数が 64，タプル数が 1000 万件の場合，‘遅れ時間’中の繰返し回数は，ソフトウェアでのマージ回数 4〜6 回と比較して半分以下の 2 回でよい.

（c） ROP への入出力は 1 ページごとにホスト計算機とのデータ転送制御が必要であるが，‘遅れ時間’中の処理は ROP の内部処理のみであり，連続に実行可能である．‘遅れ時間’の大きさは 6.2 節で評価する.

* 本節の‘マージ’はソートの手段として，複数のソート済みのキー列を 1 本にする操作であり，3 フェーズ・ジョイン方式における‘マージ結合’とは別の処理である.

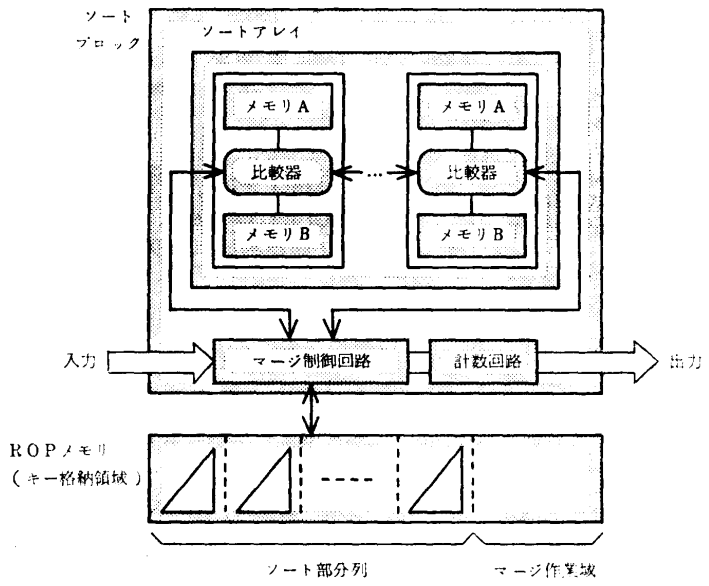


図 7 ソートブロックの構成
Fig. 7 Sorting block configuration.

5.2 ブロック構成

ROP メモリのキー格納領域を含めたソートブロックの構成を**図 7** に示す．ソートアレイ[7]はソートユニットの一次元配列であり，キーのソート，あるいは，マージを行う．各ソートユニットは2個のキーを格納するメモリとそれらの内容を比較する比較器とを持ち，各ユニットごと並列にキーの比較，転送を行う．また，マージ制御回路はソートブロックへの入出力制御，および，マージすべきソート部分列の選択制御を行う．

ソートアレイ中のソートユニットは，2つのキーのみを比較するため，構造は単純であり LSI 化に向いた構成となっている．ROP では1 LSI 中に数ユニットを実装している．

5.3 ソート・アルゴリズム

ソート処理は，マージ準備ステージ，中間マージステージ，出力マージステージの3段階で実行される[8]．

ROP へのタプルの入力とマージ準備ステージを，ROP からのタプルの出力と出力マージステージをそれぞれ重畳して実行する．したがって，ホスト計算機から見える入出力以外のソート時間，すなわち‘遅れ時間’は，中間マージステージの時間である．

本ソートアルゴリズムは，従来のソフトウェアでのソートマージ法をベースとしており，マージ準備ステージでのキーのソート処理と，中間マージ，出力マージステージでのマージ処理を同一のソートアレイで実現しているところに特徴がある．

6. 性能評価

6.1 評価条件

ROP の高速化効果を拡張ウィスコンシン・ベンチマーク[17),18)]のデータベースモデルにより評価した．評価に用いたリレーションのタプル数はリレーション TENKA, TENKB が 10,000, ONEKA が1,000, HUNKA, HUNKB が 100,000 であり，タプルの長さはシステムの制御情報を除いて 208 バイトである．評価に用いた問合せの一覧を**表 1** に示す．システム構成は，ホスト計算機として小型汎用機の DIPS-V 30[10)]，CSP：2台，ROP：1台からなる RINDA，および，容量：500 M バイトでデータ転送速度：1.8 M バイト/秒のディスクおよびディスク制御装置が各2台である．ホスト計算機と RINDA 間は 3 M バイト/秒のチャネルで接続されている．

測定はホスト計算機側で行い，CPU 時間と I/O 時間（ROP での処理も I/O に含まれる）とに区分して求めた．測定した経過時間は，特に断らない限り，問合せ文の実行開始から，処理結果をディスクに一時リレーションとして格納し終えるまでの時間である．

6.2 個別技術評価

本節では，ROP で新規に導入した技術項目について評価した．

（1） 結合キーの違いによるふるい落とし効果

ふるい落としによる結合演算処理の高速化効果については，文献5）等にもある．ここでは，ROP でのふるい落としの特徴である結合キーの長さ，データ型に依らないふるい落とし効果について評価した．評価は結合キーの長さとデータ型，および，結合元となるタプル集合の異なる，すなわち，リレーションの選択条件の異なる4種類のリレーション対について，各々の結合処理（選択のための処理を除く）に伴う I/O 時間を測定した．この I/O 時間は，ROP への転送を開始してから，ROP 内部で処理し，その結果をホスト計算機が受け取り終えるまでの時間からなる ROP 処理時間とディスク I/O 時間からなり，基本的にふるい落としの効果によってのみ変化する値である．

表1の問合せ Q1 を用いた評価結果を**表 2** に示

表 1 評価に用いた問合せ
Table 1 Executed queries in the ROP test.

問合せ		対応する SQL 文
Q1	選択結合 2リレーション	SELECT * FROM HUNKA A, HUNKB B WHERE A. xx=B. xx AND A の選択条件
Q2	選択ソート	SELECT * FROM HUNKA WHERE UNIQUE 2 〈 定数 DRDER BY UNIQUE 2
Q3	グループ化 MIN	SELECT MIN (UNIQUE 2) FROM TENKA GROUP BY HUNDRED
Q4	グループ化計数	SELECT COUNT (*) FROM TENKA GROUP BY HUNDRED
Q5	ソート	SELECT * FROM TENKA ORDER BY UNIQUE 2
Q6	結合 2リレーション	SELECT * FROM ONEKA A, HUNKB B WHERE A. UNIQUE 2=B. UNIQUE 2
Q7	選択結合 2リレーション	SELECT * FROM TENKA A, TENKB B WHERE A. UNIQUE 2=B. UNIQUE 2 AND A. UNIQUE 2 〈 1000
Q8	選択結合 3リレーション	SELECT * FROM ONEKA C, TENKA A, TENKB B WHERE C. UNIQUE 2=A. UNIQUE 2 A. UNIQUE 2=B. UNIQUE 2 AND A. UNIQUE 2 〈 1000 AND B. UNIQUE 2 〈 1000

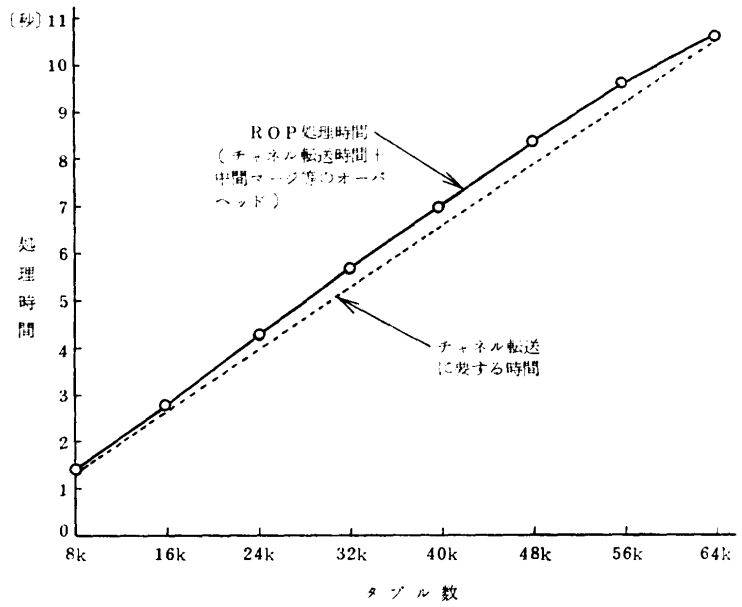


図 8 タプル数と ROP 処理時間との関係
Fig. 8 Relationship between number of tuple and ROP processing time.

表 2 結合処理時の ROP 処理時間（相対値）
Table 2 ROP processing times of join operations.

HUNKA の選択条件	2進数 (4 B)	文字列 (52B)
UNIQUE 1 〈 1000	1	1.009
UNIQUE 1 〉=1000 AND UNIQUE 1 〈 2000	0.997	1.005

す．表2は，データ型が2進数で HUNKA の選択条件が“UNIQUE 1〈1000”の場合の ROP 処理時間を基準とした相対値で示している．UNIQUE 1 はユニーク・アトリビュートのため，HUNKA から選択されるタプル数はともに 1,000 件である．表2より，データ型，選択条件の違いによる処理時間の差は1％以下であり，ディスクのシーク／サーチのタイミングの違いに起因すると考えられる．評価結果より，ROP のふるい落とし効果は，データの持つ種々の特性の変化に対して依存度が小さいといえる．

（2） ソートタプル数と ROP 処理時間の関係

ROP で採用している多段マージソータでは，出力開始までに‘遅れ時間’が生じる．ROP 処理時間とソート件数との関連，および，‘遅れ時間’を含む ROP への入出力処理以外の時間（ROP 処理でのオーバヘッド）をみるため，ROP 処理時間のソート件数による変化を評価した．評価対象とした時間は，ホスト計算機の主記憶にあるリレーションを ROP を用いて処理し，その結果を主記憶に転送し終えるまでの時間である．

表1の問合せ Q2 を用いた評価結果を**図8**に示す．図8より，ROP の処理速度は片道約 2.8 MB/秒 でほぼ一定である．チャネル転送速度は 3 MB/秒 であるから，この差が ROP での‘遅れ時間’を含むオー

パヘッドであり，この値は常に全 ROP 処理時間の1割以下と小さい．

（3） グループ化計数処理

ROP で実現しているグループ化計数処理についてソート以外のハードウェア化の効果を評価した．本来，本評価はソート以外のグループ化計数処理のみを抽出して，行うべきものであるが，ROP 内処理時間の切り分けは困難なため，ここでの評価はグループ化計数処理と，それに類似の処理でソートだけをハードウェアで実行するグループ化最小値処理と比較することとした．ソフトウェア処理では，一般にグループ化最小値処理の方がグループ化計数処理よりは処理時間

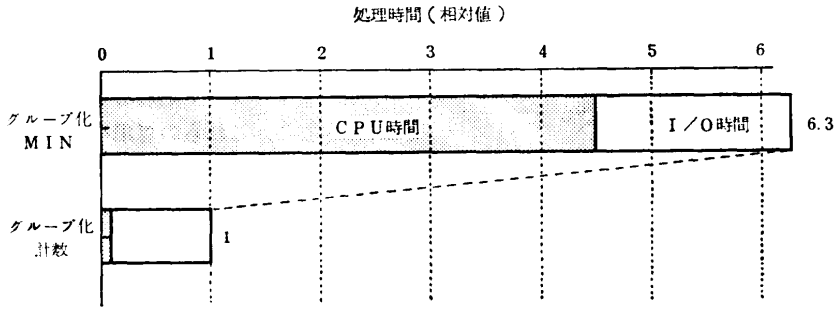


図 9 グループ化計数処理におけるハードウェア化効果
Fig. 9 Performance improvement by hardware (COUNT function with 100 partitions).

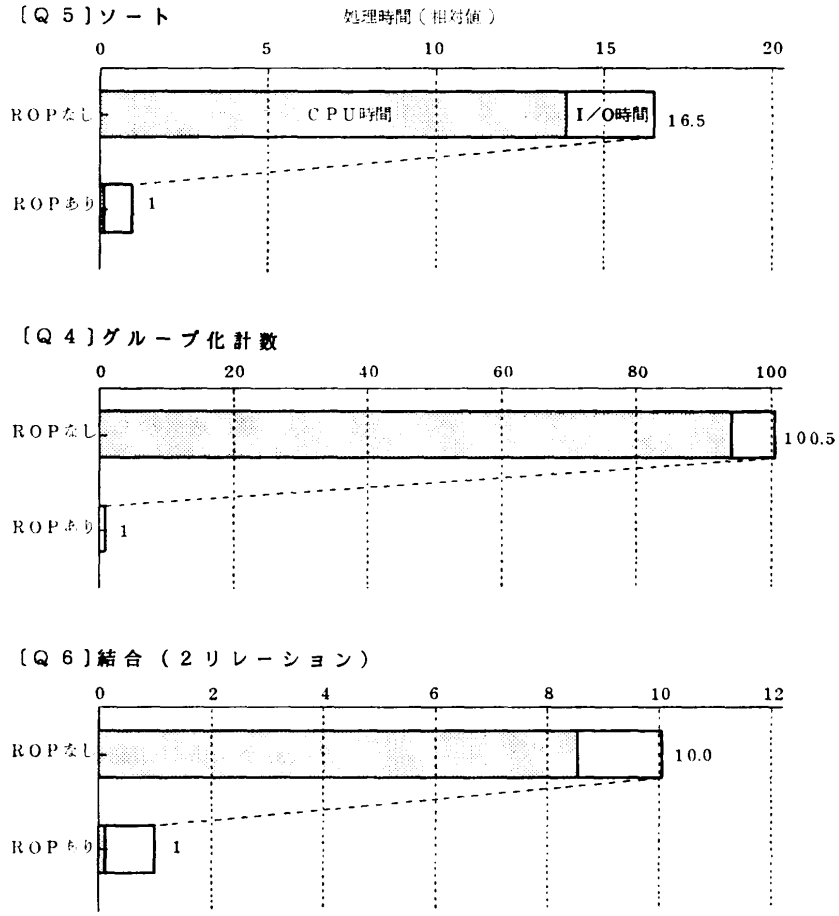


図 10 ROP による性能向上効果
Fig. 10 Performance improvement by ROP.

表 3 市販データベースマシンとの処理時間比較
Table 3 Query execution time (in seconds).
〔処理時間の単位：秒〕

問合せ	マシン（年） RINDA (1989) ROP なし	RINDA (1989) ROP あり	IDM/500 da (1985)	DBC/1012 (1987)
Q3 グループ化 MIN	26.5	6.5	36.0	8.66
Q7 選択結合（2）	52.1	17.8	84.0	35.6
Q8 選択結合（3）	33.5	18.6	36.0	27.8

は短い．

表1の問合せ Q3，Q4 を用いて評価した経過時間を**図 9** に比較して示す．図9はグループ化計数の処理時間を基準とした場合の相対値で示している．ハードウェア化により6倍以上の処理の高速化が得られることがわかる．特に，CPU 時間の削減が大きく，ハードウェア化の効果が明確に表れている．

6.3 総合評価

本節では，同一の問合せに対し，ROP を用いた場合とホスト計算機上のソフトウェアで行った場合とを比較することにより，ROP の効果を評価した．

総合評価は，表1の問合せ Q4，Q5，Q6 を用いて行った．結果を**図 10** に示す．図10は ROP を使用した場合を基準とした相対値で示してある．ROP による性能向上効果は問合せにより異なるが，10 倍～100 倍程度の向上効果が得られた．I/O 時間は，ディスクへの I/O が減少する代りに ROP への I/O が増加するため，削減効果が相対的に小さくなっている．これに対して，CPU 時間は，ROP に負荷の大半をオフロードできるため削減効果が大きい．

6.4 市販データベースマシンとの比較

市販のデータベースマシンの性能の公表値[17),19)] と RINDA との性能を比較する．評価に用いた問合せは，ウィスコンシン・ベンチマークのうちでソートを必要とする表1の Q3，Q7，Q8 である．結果を**表 3** に示す．RINDA の評価結果には CSP の効果も含まれるため，表3には ROP を使用した場合と ROP を使用しなかった場合とを併記した．表3から，RINDA，特に，ROP を備えた RINDA は他のマシンと比較しても極めて高速であることがわかる．

7. おわりに

本論文では，RINDA のキー・コンポーネントである ROP の構成と，そこで用いている処理アルゴリズムについて述べた．

RINDA はインデックス使用効果の低いデータベース問合せ処理の高速化を目的に開発したマシンであり，その中で ROP はソート処理，結合演算処理，グループ化計数処理の高速化を図っている．2リレーションの結合演算はふるい落とし，ソート，マージ結合という3つのフェーズで処理される．このうち，ふるい落としとソートフェーズを ROP で専用ハードウェアを用いて処理している．また，種々の関係演算の基本処理となるソート処理も専用ハードウェアにより高速化される．これらの専用ハードウェアを含む，広範囲な専用ハードウェア化により，従来ソフトウェアで実行した場合と比較し，10 倍～100 倍の処理の高速化を図ることができた．

RINDA は既に製品化を終え，幾つかのシステムに導入されている．

謝辞 本研究の機会を与えてくださった NTT 情報通信処理研究所の松永俊雄研究企画部長，拝原正人情報処理研究部長に感謝するとともに，有益なご指導，ご助言を与えてくださった鈴木健司，福岡秀樹，井上潮各主幹研究員に深く感謝いたします．

参考文献

1) Teradata Corp.: DBC/1012 Data Base Computer System—Introduction (1986).
2) Britton Lee, Inc.: Server/8000 for Use with ShareBase II Software (1988).
3) 速水治夫，井上 潮，福岡秀樹，鈴木健司：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，情報処理学会研究会報告，Vol. 88, No. 79, ARC-73-12, pp. 85-92 (1988).
4) 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 (1990).
5) 武田英昭，中村敏夫，北村 正：関係演算処理のための専用ハードウェアの構成とその評価，情報処理学会研究会報告，データベースシステム，57-5, pp. 35-42 (1987).
6) McGregor, D. R., Thomson, R. G. and Dawson, W. N.: *High Performance Hardware for Database Systems, Syst. for Large Data Bases*, North-Holland Publishing Company, pp. 103-116 (1976).
7) 佐藤哲司，津田伸生：一次元アレイを用いたソート処理装置の構成法，第 29 回情報処理学会全国大会論文集，4F-1 (1984).
8) 佐藤哲司，武田英昭：大容量データベース処理に適したソート手法，信学技報，DE 88-1 (1988).
9) 井上 潮，北村 正，速水治夫，中村敏夫：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，情報処理学会研究会報告，Vol. 85, No. 6, 85-DB-47-5 (1985).
10) 伊藤憲一，矢沢良一，福村好美，小浜晴雄：DIPS-V 30 の実用化，NTT 研究実用化報告，Vol. 35, No. 5, pp. 539-546 (1986).
11) Blasgen, M. W. and Eswaran K. P.: Storage and Access in Relational Data Bases, *IBM Syst. J.*, No. 4, pp. 363-377 (1977).
12) DeWitt, D. J., Gerber, R. H., Graefe, G., Heytens, M. L., Kumar, K. B. and Muralikrishna, M.: GAMMA—A High Performance Dataflow Database Machine, *Proc. 12th Int. Conf. on VLDB*, pp. 228-237 (1986).
13) Akl, S. G.: *Parallel Sorting Algorithms*, Academic Press, Inc. (1985).
14) Knott, G. D.: Hashing Functions, *Comput. J.*, Vol. 18, No. 3, pp. 265-278 (1975).
15) Tanaka, Y., Nozaka, Y. and Masuyama, A.: Pipelined Searching and Sorting Modules as Components of a Data Flow Database Computer, *Proc. IFIP '80*, pp. 427-432 (1980).
16) 喜連川優，伏見信也，桑原和宏，田中英彦，元岡 達：パイプラインマージソータの構成，信学論，Vol. J 66-D, No. 3, pp. 332-339 (1983).
17) Bitton, D., DeWitt, D. J. and Turbyfill, C.: Benchmarking Database Systems—A Systematic Approach, CSTR #526, Univ. of Wisconsin-Madison (1983).
18) DeWitt, D. J. et al.: *A Single User Evaluation of the GAMMA Database Machine, Database Machines and Knowledge Base Machines*, Kluwer Academic, pp. 370-386 (1988).
19) Simon, E.: *Update to December 1983 'DeWitt' Benchmark*, Britton Lee, Inc. (1985).

(平成元年8月1日受付)
(平成2年5月8日採録)

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 情報通信処理研究所主任研究員．IEEE 会員．

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 情報通信処理研究所主任研究員．電子情報通信学会，IEEE 各会員．

中村 敏夫（正会員）

昭和24年生．昭和47年京都工芸繊維大学工芸学部電気工学科卒業．同年，日本電信電話公社入社．現在，NTT 情報通信処理研究所情報処理研究部主任研究員．主に，言語処理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．

速水 治夫（正会員）

昭和22年生．昭和45年名古屋大学工学部応用物理学科卒業．昭和47年同大学院工学研究科応用物理学専攻修士課程修了．同年，日本電信電話公社入社．現在，NTT 情報通信処理研究所情報処理研究部主幹研究員．主に，DIPS ハードウェアシステム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．