計算機アーキテクチャ 73－12
（１９８８．１０．３１，１１．１）

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

速水治夫，　井上　潮，　福岡秀樹，　鈴木健司
ＮＴＴ情報通信処理研究所

リレーショナルデータベース（ＲＤＢ）処理の大幅性能向上を目的として、ＲＤＢ処理の中で重要性が高く、汎用プロセッサにとって負荷が重いサーチ処理、ソート処理を超高速に実行するハードウェア機構を備えた、データベースプロセッサＲＩＮＤＡを開発した。

ＲＩＮＤＡは、内容検索プロセッサ（ＣＳＰ）と関係演算プロセッサ（ＲＯＰ）から構成される。ＣＳＰとＲＯＰはそれぞれ独立のＩ／ＯインタフェースでＤＩＰＳ本体装置に接続され、ＤＩＰＳ上のデータベース管理プログラムにより制御されて動作する。

ＣＳＰは磁気ディスク装置に格納された表を指定された条件でサーチし、条件に合致した行の中から必要な列を抽出して本体装置へ転送する。ＲＯＰは本体装置から転送されてきた表を指定された条件でふるい落とし、ソートを行い、その結果を本体装置へ返送する。

インデックスを使用しない検索において、ＲＩＮＤＡを使用することにより約１０～１００倍の性能向上を達成している。

ARCHITECTURE OF RINDA - A RELATIONAL DATABASE PROCESSOR

Haruo HAYAMI, Ushio INOUE, Hideki FUKUOKA, and Kenji SUZUKI

NTT Communications and Information Processing Laboratories
1-2356 Take, Yokosuka-shi, Kanagawa, 238-03 Japan

A new relational database processor called RINDA is presented. RINDA performs key database operation, such as search and sort, at very high speed with specialized hardware.

RINDA is composed of CSPs (Content Search Processor) and optional ROPs (Relational Operation Accelerating Processor). Each CSP and ROP is attached to general purpose computers and controlled by database manegement software.

In this paper, the architecture of RINDA is descried. Performance evaluation shows RINDA accelerates non-indexed queries of relational databases by 10 to 100 times compared with conventional database management software.

1．はじめに

リレーショナルデータベース（ＲＤＢ）は、データベース（ＤＢ）の論理構造が単純な表形式であり、データアクセス時にアクセスパスを意識する必要がない。このため、ＤＢ構築およびアプリケーション（ＡＰ）の開発が容易であり、広く普及しつつある。

しかし、ＲＤＢをソフトウェアのみにより実現した場合は、インデックスを効果的に利用できる定型処理では高速な検索が可能であるが、インデックスの利用が困難な非定型処理に対しては性能低下が著しいという問題点がある。また、インデックスの付加による更新処理の性能低下も問題である。

著者らは、非定型のＲＤＢ処理の中で重要性が高く、汎用ＣＰＵにとって負担が重いサーチ処理とソート処理を超高速に実行するリレーショナルデータベースプロセッサＲＩＮＤＡを開発した[1]。

以下、第２章では設計思想、第３章ではアーキテクチャ、第４章では非定型処理に対するＲＩＮＤＡの性能向上効果を述べる。

2．設計思想

2．1　従来の問題点

ソフトウェアのみにより実現したＲＤＢでは、予めインデックスが作成されていない表の列を指定した検索処理を行おうとすると、結果が得られるまでに膨大なＣＰＵ時間と入出力時間を必要とした。この検索処理の主なボトルネックはサーチ処理とソート処理である[2,3]。以下、これらの処理における問題点を述べる。

（１）サーチ処理

最近のＶＬＳＩ技術の進歩により半導体メモリの価格は大幅に低下しつつあるが、現状では大容量ＤＢの格納媒体としては磁気ディスク装置（ＤＫ）を用いるのが一般的である。このＤＫ上に構築されているＤＢを検索するとき、インデックスを使用して検索範囲を局所化している。

しかし、インデックスが使用できない検索では、処理対象のデータを局所化することができない。従って、従来の汎用ＣＰＵの処理においては、ＤＫに格納された全データを逐次主記憶に読み込んでＣＰＵで検索条件の判定を行っていた。本処理に長時間を要するのは主に以下の要因である。

① 本体装置－ＤＫ間のデータ転送
② ＤＫの読みだし単位が１ページ（物理レコード）あるいは数ページであることによるシーク・サーチ回数の増加
③ ＤＫの読みだしと本体装置での条件判定処理がシリアル
④ 条件の種類（文字列の部分一致等）によっては、汎用ＣＰＵでの判定は負荷が重い

（２）ソート処理

非定型処理では、結合演算、データの整列出力、グループ化等でソート処理が多用される。本処理に長時間を要するのは主に以下の要因である。

① 本体装置－ワークＤＫ間のデータ転送
② 行からソートキーを抽出する処理
③ 汎用ＣＰＵでのソートの比較回数はＯ（ｎ log ｎ）

2．2　開発目的

上記の様に、ソフトウェアのみによる実現では性能に限界のあった非定型処理を高速化するために、サーチ処理とソート処理を専用ハードウェア化するＲＩＮＤＡを開発する。

すなわち、ＲＩＮＤＡは従来のデータベース管理プログラム（ＤＢＭＳ）の一部の処理を高速化する付加プロセッサとして開発する。

これによって、１つのＤＢＭＳの基で、同一のＤＢに対してソフトウェアによるインデックスを使用したアクセスとＲＩＮＤＡによるアクセスとの混在実行を可能とし、リアルタイム処理可能なＲＤＢの機能範囲を拡大する。

2．3　設計条件

システムの柔軟性、拡張性を確保するとともに、既存システムへの導入を可能とするために、以下の条件でＲＩＮＤＡを設計した。

(ｉ)ハードウェアをコンポーネント化することにより、ＤＢの規模、利用形態に応じたシステム構成を可能とする。
(ii)１つのＤＢＭＳの基で、同一のＤＢに対してＲＩＮＤＡによるアクセスと、ソフトによるインデックスを使用したアクセスとの混在実行、同時処理を可能とする。
(iii)ＤＩＰＳシリーズの本体装置各機種と接続可能とする。
(iv)ＤＢＭＳが使用可能な汎用ＤＫと接続可能とする。

3．アーキテクチャ

3．1　システム構成

設計条件を満足するため、ＲＩＮＤＡは以下のように構成される（図１、写真１）。

① サーチ処理を実行する内容検索プロセッサ（ＣＳＰ：Contento Search Processor）[4]とソート処理を実行する関係演算プロセッサ（ＲＯＰ：Relational Operation accelerating Processor）[5]により構成する。

② ＣＳＰ、ＲＯＰは各々Ｉ／Ｏインタフェースで本体装置と接続する。

ＣＳＰはＤＫに格納された表を指定された条件でサーチし、条件に合致した行の中から処理に必要な列のみを抽出して本体装置へ転送する。

ＲＯＰは本体装置から転送された表を指定された条件でソートし、その結果を本体装置へ返送する。なお、結合演算のためにソートを行う場合は結合の対象とならない行をふるい落とした後にソートを行う。

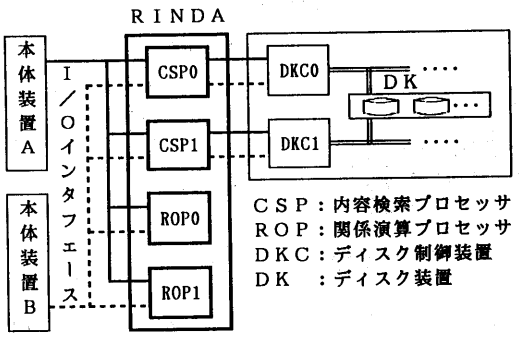

図１　システムの論理構成

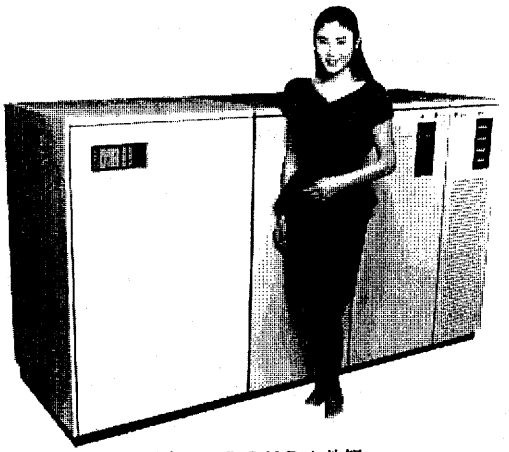

写真１　ＲＩＮＤＡ外観

3．2　システムの動作概要

ユーザ（アプリケーションプログラム（ＡＰ））がＳＱＬで記述した問合せは、本体装置上のデータベース管理プログラム（ＤＢＭＳ）により解釈され、ＤＢＭＳがＲＩＮＤＡを制御して実行される。

図２に結合演算を例として、システム全体の動作概要を示す[8]。

3．3　内容検索プロセッサ

3．3．1　機能

ＣＳＰはＤＫに格納されている単一の表に対し、表１に示す内容検索を実行し、結果を一時表としてＣＰＵに返送する。

表１　ＣＳＰの機能

機能		説明
述語判定	比較述語	〈列指定〉〈比較演算子〉〈定数〉
	IN述語	〈列指定〉[NOT] IN〈定数リスト〉
	LIKE述語	〈列指定〉[NOT] LIKE〈パタン〉
	NULL述語	〈列指定〉IS [NOT] NULL
探索条件判定		述語の任意の論理式
出力列の抽出		〈列指定〉[{,〈列指定〉}...]
集合関数演算		COUNT(*)

(注)用語、記号はＳＱＬ(ＪＩＳ)[6]に準拠

なお、複数の表に対する結合、副問合せ等は、データベース管理プログラムが単一の表に対する検索指令に分解し、それをＣＳＰに繰り返し発行して処理する。

3．3．2　論理構成

ＣＳＰの論理構成を図３に示す。

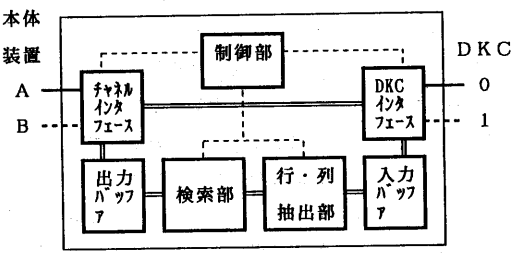

図３　内容検索プロセッサの論理構成

【検索命令】

```
ＳＥＬＥＣＴ　品名，在庫，会社．社番，社名，地区
ＦＲＯＭ　　会社，部品
ＷＨＥＲＥ　　会社．社番＝部品．社番　ＡＮＤ
　　地区＝’東京’　ＡＮＤ　在庫≦５０
```

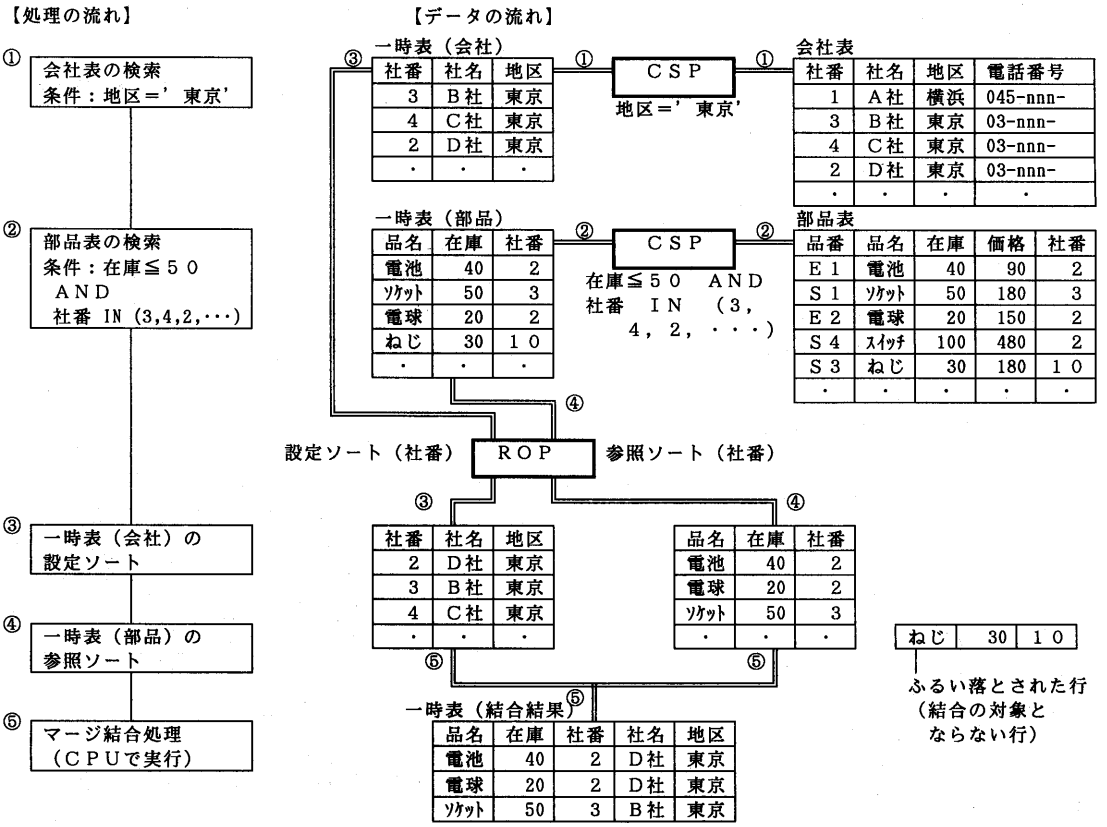

図２　結合処理を例としたシステムの動作概要

３．３．３　動作

ＤＢＭＳから起動され検索結果のデータを返却するまでのＣＳＰの動作を述べる。

①　ＤＢＭＳはＣＳＰに検索させるＤＫボリューム中の範囲、検索条件、検索結果データの転送単位容量、転送回数等を制御表（検索オーダと呼ぶ）に記述し、検索オーダをＩ／Ｏコマンドにより ＣＳＰへ転送する。

②　ＣＳＰは検索オーダを受け取ると、ＤＫに対するチャネルプログラムを作成し、検索対象の物理レコード群をマルチトラックリードにより連続的に読みだす。

③　ＣＳＰは、ＤＫから読みだした物理レコード（ページ）をＣＳＰ内でダブルバッファリングしつつ、指定された内容検索を実行し、検索結果のデータをＣＳＰ内の出力バッファに一時格納する。

④　出力バッファは検索結果データの転送単位容量に設定されており、出力バッファが満杯になると、１転送単位の検索結果を転送する。

⑤　出力バッファもダブルバッファになっており、本体装置への転送中もＤＫ読みだしと検索動作は休みなく継続する。

⑥　この様にして検索範囲の検索が終了するか、所定回数の本体装置への転送が終了すると終了情報を本体装置へ転送し動作を終了する。

この様に、ＣＳＰはＤＫからのデータ読みだし以外の処理時間をほとんど要せずに（オンザフライ処理で）内容検索を実行する。

特に、汎用ＣＰＵでは負荷の重かったLIKE述語の判定には、パタンマッチングＬＳＩを開発し、複数のLIKE述語の判定を高速処理する。

3．4　関係演算プロセッサ

3．4．1　機能

ＲＯＰは本体装置から転送された一時表に対して表２の処理を実行し、結果を一時表としてＣＰＵに返送する。

表２　ＲＯＰの機能

機　　能	説　　明
ソート	指定された列の値（ソートキーと呼ぶ）に基づき、行の順序をソートする。ソートキーとしては複数の列を任意の順序でコンカチネイトしたものが指定可能であり、各列毎に昇順／降順を指定可能である。
ふるい落とし	２つの一時表を結合する場合に、結合の可能性の無い行をソートする前に除去する。
重複除去	ソートキーの値が重複する行を除去する。
重複計数	ソートキーの値が重複する行数を重複毎に計数する。計数値は新たな列として追加される。

3．4．2　論理構成

ＲＯＰの論理構成を図４に示す。

行・キー抽出部では入力バッファに転送された一時表から、各行を取り出しＲＯＰメモリの行エリアに格納すると共に、ソートキーを抽出し行とのリンク情報を付加して、ふるい落とし部および整列部に送出する。ふるい落とし部、整列部ではソートキー（リンク情報付き）のみを扱う。ソートキーのソートが終了後、リンク情報を用いてソート結果の一時表を転送単位毎に出力バッファ上に作成する。

ふるい落とし部、整列部のアルゴリズムを以下に示す。

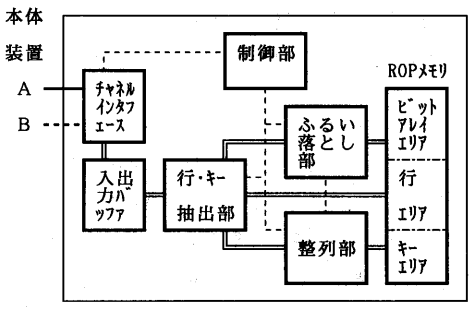

図４　関係演算プロセッサの論理構成

3．4．3　ふるい落とし部

（１）　ふるい落とし法

ふるい落しのためのソートキーの登録と参照には、ハッシュ化ビットアレイ[7]を用いている。ビットアレイはＲＯＰメモリ中のビットアレイエリアに存在する。

ふるい落としには、一方の一時表のみをふるい落とす片ハッシュ法と、両方の一時表をふるい落とす両ハッシュ法がある。

片ハッシュ法は、１つのビットアレイを用いて、以下の手順でふるい落とし処理を行う。

①　表Ｔ１の各行のソートキーをハッシングし、ハッシュ値をアドレスとしてビットアレイに'１'を設定する。

②　表Ｔ２の各行のソートキーを同一のハッシュ法でハッシングし、ハッシュ値をアドレスとしてビットアレイを参照し、そこが'１'の行のみを出力し他の行をふるい落とす。

両ハッシュ法では、２つのビットアレイを用い、上記②処理時に同時に他方のビットアレイに設定処理を行い、その後、③として表Ｔ１の参照処理を行う。

片ハッシュ法と両ハッシュ法はＤＢＭＳが２つの一時表の大きさで使い分ける[8]。

ふるい落としにより結合演算の対象行数を削減でき、ソート時間と本体装置におけるマージ結合処理時間が短縮される。

（２）　ハッシュ法

ＲＯＰではハッシュ法として乗算重ね合わせ法を採用している。

この方法では、ソートキーを一定長の切片に区切り、ソートキーの入力に同期して、先頭の切片から順に、

①乗算表により切片の値を変換（ランダマイズ）し、

②それまでの中間ハッシュ値を一定ビットシフトし、それに①の結果を重ね合わせる（ＸＯＲをとる）処理を行う。

この方法は従来から知られている乗算法とＸＯＲ法[9]を併用したものであり、ソートキー値の偏りに強いものである。

3．4．4　整列部

ＲＯＰでは多量のソートキーをソートできるようにする一方、装置を小型化するため、多段マージによるソート法[10]を採用している。

本ソート法は、ｋ個（ｋは自然数）の比較器を一次元アレイ状に配置したソートアレイとＲＯＰメモリ中のキーエリアを用い、２ｋウエイマージ処理を段階的に繰り返すことによりソートを行うものである。図５に本ソート回路の論理構成を示す。

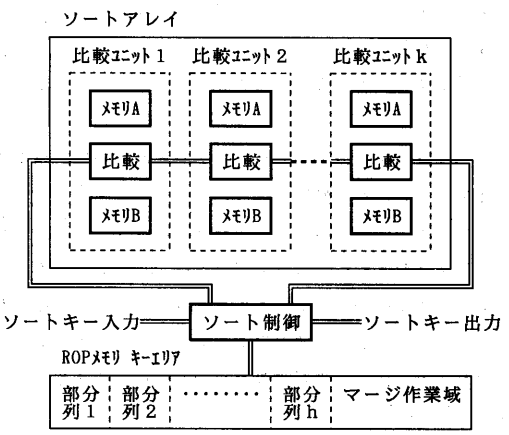

図5　ソート部の論理構成

ソート手順は、以下の２種類の段階からなり、(ｉ)を処理後、マージ結果が１本になるまで(ii)を繰り返す。

(ｉ)初期ソート段階：ソートキーを入力しながら、ソートアレを用いて、２ｋ個ずつソートし（これを部分列と呼ぶ）、キーエリアに格納する。

(ii)マージ段階　　：部分列を２ｋ本毎にマージする段階であり、以下の①を実行後、部分列が無くなるまで②を繰り返す。

①　キーエリアに格納されている各部分列の先頭ソートキー（部分列の最小ソートキー）を読みだし、ソートアレイに入力する。

②　ソートアレイから最小ソートキーを出力し、キーエリアの作業域に書き込む。次に、そのソートキーが属していた部分列の次のソートキーをキーエリアから読みだして、ソートアレイに入力する。

多段マージによるソートを行うことにより、一定のソートアレイで多量のソートキーのソートが可能となる。他方、多段マージに伴う処理時間の増加に対しては、マージウエイ数（２ｋ）を多くし、マージ段数を削減することで解決している。

３．４．５　動作

ＤＢＭＳから起動され検索結果のデータを返却するまでのＲＯＰの動作を述べる。

(１)入力系動作

①　ＤＢＭＳはＲＯＰに実行させる処理条件、および入力の一時表の転送単位容量、転送回数等を１つの制御表（関係演算オーダと呼ぶ）に記述し、関係演算オーダをＩ／Ｏコマンドによりｒｏｐへ転送する。

②　ＲＯＰは関係演算オーダを受け取ると、処理条件に基づき内部をセットアップする。

③　ＲＯＰは本体装置から転送される一時表の転送単位をダブルバッファリングしつつ、設定／参照処理と共に、あるいは単独でソートの初期ソート段階までを実行する。

④　所定回数の転送が終了すると終了情報を本体装置へ転送する一方で、ソートのマージ段階を実行する。

(２)出力系動作

①　ＤＢＭＳはＲＯＰに実行させる処理条件、および出力の一時表の転送単位容量、転送回数等を関係演算オーダに記述し、関係演算オーダをＩ／Ｏコマンドによりｒｏｐへ転送する。

②　ＲＯＰは関係演算オーダを受け取ると、処理条件に基づき内部をセットアップする。また、ソートのマージ段階の正常動作を確認する。

③　ソート結果を出力バッファに格納する。出力バッファはソート結果データの転送単位容量に設定されており、出力バッファが満杯になると、１転送単位の検索結果を転送する。

④　出力バッファもダブルバッファになっており、ソート結果を出力バッファへの格納と、本体装置への転送は並行して休みなく継続する。

⑤　この様にして所定回数の本体装置への転送が終了すると終了情報を本体装置へ転送し動作を終了する。

この様に、ＲＯＰは本体装置からのデータ転送時間以外の処理時間をほとんど要せずに、設定／参照処理とソート処理を実行する。またソートのマージ段階の処理は、本体装置側でのソート結果を受け取る準備処理と並行処理されるため、その処理時間は陽に見えることはほとんどない。

また、ＲＯＰをＣＳＰと独立の装置としたため、複数のＤＫに分散して格納された表を１台のＲＯＰでまとめてソートすることが可能である。

さらに、ＲＯＰをＩ／Ｏインタフェース接続としたことにより、ＲＯＰを複数台設置し、複数の問合せ処理に対応するソート処理を並行して実行することが可能である。

4．評価

4．1　ＲＩＮＤＡの効果

(1)ＣＳＰの効果

表全体のサーチをインデックスを使用せずに行う場合、従来の処理方式では１ないし数ページ（物理レコード）単位でＤＫから読みだし、１行単位でＣＰＵによる検索条件判定を行う。一方ＲＩＮＤＡの処理方式では、ＤＫから複数ページ（最大１シリンダ）を連続的に読みだし、検索条件判定、条件合致行の転送をＣＳＰが実行する。このため、ＤＫのシーク・サーチ回数削減による入出力時間の大幅削減とＣＰＵ負荷の削減が達成できる。

また、表を複数のＤＫに分散して格納し、複数のＣＳＰにより並行してサーチする[11]ことにより、更に入出力時間を削減できる。

(2)ＲＯＰの効果

大容量の表をソートする場合、従来の処理方式ではＣＰＵによる１行単位の比較処理による並べ替えを行うとともに、中間結果の格納のために作業ＤＫを使用する。一方ＲＩＮＤＡの処理方式では、本体装置とＲＯＰ間で往復のデータ転送を行うだけでソート処理が完結する。このため、ＣＰＵ負荷の削減と、作業ＤＫとの入出力時間の削減が達成できる

また、結合演算に対してはソートの前処理としてふるい落しを実行するため、ソート時間と本体装置でのマージ結合処理時間をさらに短縮する効果がある。

4．2　実測評価

現時点までに実施した暫定的な実測の結果を述べる。

(1)測定環境

データベースは拡張ウィスコンシン・ベンチマーク[12,13]に基づき作成した。表には１万行あり、各行は 208 バイト（制御情報を除く）である。

ハードウェア環境は、ＤＩＰＳ－Ｖ３０本体装置[14]、ＣＳＰ２台である。

測定した問合せを表３に示す。最初の３つはベンチマーク標準であり、他の２つの問合せは選択された行を転送しない条件での検索時間を評価するものと、テキストサーチを評価するものである。

表３　問合せ

問　合　せ	ＳＱＬステートメント
Non-indexed 1% Selection	SELECT * FROM tenk WHERE unique2)=5000 and unique2(5100
Non-indexed 10% Selection	SELECT * FROM tenk WHERE unique2)=5000 and unique2(6000
Non-indexed scalar MIN	SELECT MIN(unique2) FROM tenk
Non-indexed scalar COUNT	SELECT COUNT(*) FROM tenk WHERE unique2)=5000 and unique2(6000
LIKE predicate scalar COUNT	SELECT COUNT(*) FROM tenk WHERE stringu2 LIKE '%ABC%'

(2)測定結果

ＲＩＮＤＡを使用する場合と使用しない場合の測定結果を表４に示す。本比較においては、ＳＱＬ言語解析から検索結果を全てＡＰに返すまでの全処理時間を測定した。

表４に示されるように、インデックスを使用しない検索において、ＲＩＮＤＡを使用することにより約10～100倍の性能向上を達成している。

表４　問合せの処理時間（相対値）

〔暫定〕

条件／システム	Non-indexed scalar COUNT	LIKE predicate scalar COUNT
ＲＩＮＤＡ 使用	1	1
ＲＩＮＤＡ 使用せず	３７	１０４

参考のために、ＲＩＮＤＡと他のデータベースプロセッサ[13,15]とを比較した結果を表５に示す。本比較においては、検索処理の開始から検索結果がＤＢＭＳの一時表に格納されるまでの処理時間を測定した。

表５　他システムとの比較（単位：秒）

条件／システム（年）	Non-indexed		
	1% selection	10% selection	scalar MIN
ＲＩＮＤＡ (1988)〔暫定〕	1.29	1.60	7.03
IDM/500 dac (1985)	19.9	23.4	19.0
DBC/1012 (1987)	6.86	15.97	4.21

5．おわりに

リレーショナルデータベースプロセッサＲＩＮＤＡのアーキテクチャと性能評価結果を述べた。

ＲＩＮＤＡは汎用ＣＰＵにとって負担が大きいサーチ処理、ソート処理を超高速に実行することにより、インデックスを利用できない場合の検索処理時間を大幅に短縮した。

同一のＤＢに対してＲＩＮＤＡを使用するアクセスとインデックスを使用したアクセスを併用することにより、リアルタイム処理可能なＲＤＢの機能範囲が一段と拡大し、今後更に大容量ＤＢのＲＤＢが普及するものと期待できる。

最後に、日頃御指導頂いている専用システム研究部松永部長を始めＲＩＮＤＡ開発の関係者各位に深謝致します。

［参考文献］

〔1〕井上、速水、福岡、鈴木：データベースプロセッサＲＩＮＤＡのアーキテクチャ，情処37全大,5Q-4,1988

〔2〕井上、北村、速水、中村：情報提供サービスに適用可能な超大容量リレーショナルデータベースマシン，情処研報，85-DB-47-5,1985

〔3〕北村、速水、中村、井上：付加プロセッサ方式によるリレーショナルデータベースマシンアーキテクチャ，通研実報，Vol.36,No.5,1987

〔4〕速水、武田、佐藤、福岡：データベースプロセッサＲＩＮＤＡの内容検索方式，情処37全大,5Q-5,1988

〔5〕武田、佐藤、中村、福岡：データベースプロセッサＲＩＮＤＡの関係演算方式，情処37全大,5Q-6,1988

〔6〕データベース言語ＳＱＬ，ＪＩＳ　Ｘ　3005-1987

〔7〕McGregor D. R.,et al：High Performance Hardware for Database System, Syst.for Large Data Bace,1976

〔8〕中村、板倉、芳西、黒岩：データベースプロセッサＲＩＮＤＡの最適化方式，情処37全大,5Q-8,1988

〔9〕Knot G.D：Hashing function, The Computer Journal, Vol.18,No.3,1975

〔10〕佐藤、武田：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

〔11〕板倉、中村、井上：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，情処37全大,5Q-9,1988

〔12〕Bitton D.,DeWitt D.J.,et.al.：Benchmarking Database Systems: A Systematic Approach, CSTR #526, Univ. of Wisconsin-Madison, December 1983

〔13〕DeWitt D.J.,et.al.：A Single User Evaluation of the GAMMA Database Machine, Database Machines and Knowledge Base Machines, Kluwar Academic,1988

〔14〕伊藤、矢沢、福村、小浜：ＤＩＰＳ-Ｖ３０の実用化，通研実報,Vol.35,No.5,1986

〔15〕Simon E.：Update to December 1983 'DeWitt' Benchmark, Britton-Lee, Inc.,1985