# 関係データベースエンジンの
# ハードウェア構成とその制御方式

安部 公朗　伊藤 英則
（財）新世代コンピュータ技術開発機構

岩田 和秀　柴山 茂樹　酒井 浩
（株） 東芝総合研究所

## １． はじめに

ＩＣＯＴの第五世代コンピュータは、推論機構と知識ベース機構を兼ね備え、マシンと人間との間を知的インタフェースで結び、知的プログラミングができることを主な目標としている。

一般に知識情報処理を行うためには、大量のFact,Ruleを扱うことが必要であり、その処理は

(1) 大量の知識の中から、ある条件を満たすものを全て探しだす。

(2) 探しだされた知識に対し、ヒューリスティックな方法等で、組合わせの爆発を押えながら解を得る。

のタイプに分類できるが、推論マシンは(2) を担当し、知識ベースマシンが(1) を担当している。

前期３年間では、知識ベースマシンの第一ステップとして、論理型言語におけるFactが関係データベースとして取扱うことが容易であるため、大量のFactの検索・格納・更新を効率よく処理できるハードウェアとして関係データベースマシン（以下Delta と呼称する）を開発した。

関係データベース・エンジン（RDBE）は、Delta の主要な演算ユニットであり、ソート処理を主体とした関係代数演算を専用ハードウェアで、また算術系演算と全体の制御をソフトウェアで行うことにより、関係データベースに対する処理が効率よく行われるように設計されている。

本稿では、RDBEのハードウェア構成とその制御方式、大量のデータに対する処理方式、性能評価について述べる。

## ２． Delta の全体構成

### ２．１ Delta のアーキテクチャ

Delta はホストマシンである逐次推論マシンPSI の外部データベースを提供する。

Delta のアーキテクチャを図１に示す。Delta は機能分散型システムであり、次の５つのユニットからなる。

(1) IP(Interface Processor) ： LAN により接続されたホストマシンとのインタフェース処理を行う。

(2) CP(Control Processor) ： ホストマシンより送られてくる問合せコマンドを解析し、内部コマンドの生成・実行制御を行う。また、同時実行制御、データリカバリなどのデータベース制御機能を提供する。

(3) MP(Maintenance Processor) ： Delta システムの状態監視・構成管理、システムの立上げ、終了制御などを行う。

(4) RDBE(Relational Database Engine) ： 関係データベースに対する各種の演算を専用ハードウェアあるいは自身のマイクロプロセッサを用いて処理する。RDBEは４台あり、CPの制御の下に並列に動作することができる。

(5) HM(Hierachical Memory) ： 関係データの格納・変更を担当する。128Mバイトの半導体メモリと20G バイトの磁気ディスク装置とHM制御プログラムの実行を担当するHMコントローラから成る。

### ２．２ アーキテクチャの特徴

Delta は論理型言語環境下で使用されるため、そのアーキテクチャには以下の特徴がある。

(1) 内部スキーマとして、リレーションを構成するタプルを属性方向に分割し、属性毎に格納する方式を採用した。

(2) 属性情報のサーチスペースを削減するため、属性値の範囲とタプル方向の関連を示すタプル識別子（以下TIDと略称する）の値の範囲をサーチキーとする２段階クラスタリング方式を採用した。

(3) ホストマシンとの論理的コマンドインタフェースとして関係代数型コマンド(Deltaコマンド）を採用した。

(4) リレーションを属性毎に分割格納する可動ヘッドディスク（MHD ）、ディスクキャッシュ・エリアと他ユニット間転送用バッファ、作業用バッファから成るデータベースメモリユニット（DMU ）を備えた階層構造メモリを採用した。

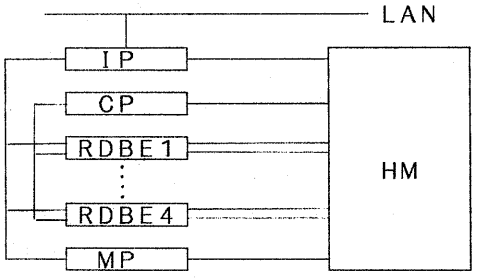



図１ Delta のアーキテクチャ

## ３． RDBEの設計思想

(1) ハードウェアによる関係代数演算の高速処理

大量のFactの中から必要なFactだけをを探索し、推論マシンに提供することが関係データベースの役割の１つである。Delta では、推論マシンとのインタフェースとして関係代数型コマンドを採用しているため、負荷の重いJoinやSelection などの演算を高速に実行することが必要である。関係代数演算を高速に実行するためには、

対象データをソートすることが必要であるのは周知である。ソータを行うソフトウェア・アルゴリズムは数多く知られているが、その処理時間は最高O(N logN) である。我々は、このソート処理をO(N)で高速に実行するためにソータという専用のハードウェアを開発し、その処理アルゴリズムとして、pipeline 2-way merge sort 法を以下の理由により採用した。

① アルゴリズムが単純で実現しやすい。
② ハードウェアでパイプライン処理が可能
③ 入出力が独立であるので、ストリーム処理が可能

また、ソータによりソートされたストリームに対し、関係代数演算をディレイ無く実行するために、ソータの後段にその専用ハードウェアであるマージャを置いた。

(2) 高速チャネルによるストリーム処理

ストリーム処理を可能にするため、HMとRDBE間に、1つのRDBE当たり入力1チャネル、出力1チャネルを設け独立にストリーム・データ転送を行えるようにした。1チャネルのデータ転送速度は、3MB/S である。

## 4. RDBEの構成と機能

### 4.1 RDBEの構成

RDBEの構成を図2に示す。RDBEは次のようなモジュールから成る。

(1) CPから送られてくるコマンドを実行するために、全てのハードウェアモジュールを制御する、または専用ハードウェアでは処理できない演算を実行するためのマイクロプロセッサ
(2) HMとRDBEとのインタフェース制御を行うHMアダプタ
(3) HMアダプタから送られてくるデータを後段のソータ、マージャ等で処理できるデータ・フォーマットに変換するINモジュール。この変換には以下のようなものがある。
① パイプラインソートが可能なように、キー・フィールドをレコードの先頭に持ってくる。
② データベースの処理では、種々のデータ形式が扱われるが、ソータでは絶対値数（符号無し2進数）のみ扱うことにしてある。そのため、整数と浮動少数点数は、絶対値数にデータ型式変換を行う。
③ Null値を判別して、Null値シグナルを発生する。
(4) 演算の対象となるレコードをソートするソータ。
(5) ソータで処理できない大量ソートと関係代数演算を実行するマージャ。

図2において、DT,PT,NL,DP はそれぞれデータ・ライン パリティ・ライン、Nullライン、重複ラインを表す。Nullラインは、データ・ライン上にNull値キーを含むタプルが存在することを表わす。重複ラインは、データ・ライン上に同じキー値を持つタプル列が存在することを表わす。

これらのモジュールは同時に動作するように制御されている。また、各モジュール間のデータ転送速度は、RDBEとHM間の転送速度と等しくなるように設計されている。通常処理対象となるデータは、IN側のHMアダプタを介して、HMから受取り、INモジュール、ソータ、マージャを通り、OUT 側のHMアダプタを介して、再びHMに送られる。

もし、RDBEがJoinのような2つのリレーションを扱う演算を実行するならば、次に示す処理を行う。

HM上に格納されている1番目のリレーションをHMアダプタを介して受取り、INモジュールでデータ・フォーマット変換し、ソータでソートし、最後にマージャのバッファに格納する。次に、2番目のリレーションも同様な処理を行った後、マージャの別のバッファに格納する。2番目のリレーションをソートしている間に、マージャは、既にバッファに格納されている1番目のリレーションの各レコードと比較を行い結果を生成する。演算結果は、OUT 側のHMアダプタを介して、HMに送られる。

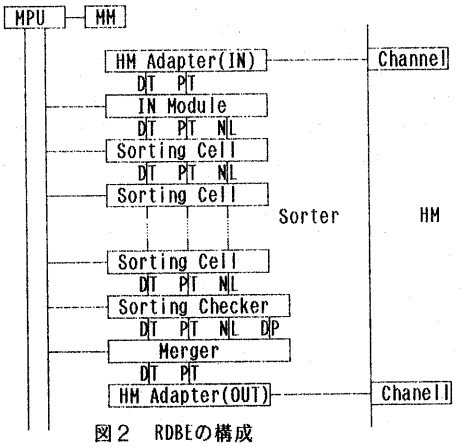


図2 RDBEの構成

### 4.2 RDBEの機能

RDBEの機能は、次の5つのパラメータで表現され、別のユニットであるCPにより指定される。

｛演算の種類、演算対象、レコードの切出し、
TID の付加、マイクロプロセッサによる処理｝

演算の種類は、文字通りRDBEで処理する演算の種類で、表1にその一覧を示す。

演算の対象は、RDBEで処理するレコード（タプル）を格納しているHMのバッファID、キー・フィールドの情報、および結果を格納するHMのバッファIDから成る。

レコードの切出しは、射影演算に相当するものである。たとえば、一般にJoinという演算を実行する場合、もとのリレーションのすべての属性が必ずしも必要というわけではなく、そのうちの一部が必要であることが多い。そこでRDBEでは、もとのリレーションの一部だけを切出すことがマージャでJoinと同時に行なえるように設計されており、

ここでは切出す範囲を指定する（図5参照）。

TID 付加は、演算の結果生成されるレコードに新TID を付加する機能を制御するものである。新TID の付加の有無とTID の初期値が指定できる。

マイクロプロセッサによる演算は、ハードウェアではサポートされていない演算を実行するための指定を行う。

表1　演算の種類

| 演算種別 | 説　　明 |
|---|---|
| Pass | MPU による演算に使用 |
| Restrict | 属性と定数との比較による選択 |
| Compare | 属性どうしの比較による選択 |
| Join | θ－Joinを行なう |
| Sort | ひとつの属性についての並べかえ |
| Unique | ひとつの属性について重複除去 |
| M-Sort | 複数の属性についての並べかえ |
| M-Unique | 複数の属性について重複除去 |
| Set | 集合演算 |
| Chech | 集合の包含関係の判定 |
| Aggregate | 集約（統計処理）演算 |
| Zone-Sort | クラスタ化に使用する |
| Delete | データの更新に使用する |

## 5．　ソータ

ソータは、与えられたレコード（タプル）の集まりを指定された順序（昇順または降順）に並べ直す処理を行う専用のハードウェアであり、pipeline 2-way merge sort アルゴリズムを採用してある。

本ソータは、次のような特徴を持つ。

(1) ソータは、12段のソーティング・セルとソート・チェッカにより構成される。ｉ段目のソーティング・セルは $2^{i-1}$ 個のソートされているレコード列を２組受取り、これらをマージして $2^{i}$ 個のソートされたレコード列を出力する。図３に示されるように、各ソーティング・セルはFIFO機能を有する２つのメモリ、比較器および制御回路から成る。

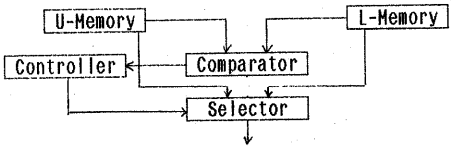



図3　ソーティング・セルの構成

(2) ソータが処理することのできる最大タプル数Kは、次式で与えられる。

$$K = \min(2^{N}, [M/L])$$

ここで、Nはソーティング・セルの個数（12）で、Mは最終段のソーティング・セルのメモリ・サイズ(64KB)、Lはタプルのサイズである。Lが16バイト以下の時にはKは4096レコードである。

(3) ソーティング・セルは、既にソートされている２つの部分レコード列をマージして、ソートされた１つの新しいレコード列を生成する機能と、入力レコードをそのまま次段のセルに転送する機能をもつ。前者の動作をソート・モード、後者の動作をパス・モードと呼ぶ。パス・モードは、次の３つの場合に必要となる動作である。

① レコード長が大き過ぎて、前段のセルを使用できない時

② レコード数が小さくて、後段のセルを使用する必要がない時。

③ ストリームを単にマージャに転送する時

(4) データベース処理においては、キー・フィールドの値が同値の時と、Null値の扱い方を考慮する必要がある。本ソータでは、キー・フィールドの値が同値の場合にはソート処理の対象となるフィールドが複数あるマルチ・ソート演算を行う時に支障をきたさない様、レコードの入力順に出力するソートを行う。Null値を含む場合には昇順、降順にかかわらず正常値の後にNull値のレコードを入力順に出力するようにしてある。

(5) ソート・チェッカは、ソータの信頼性を向上させるために、各レコードのキー・フィールドと次のレコードのそれを比較することにより、ソート結果をチェックする。また、値が同じ場合には、重複シグナルを発生させる。これは、Join演算時のデカルト積を生成する場合に有効である。

## 6．　マージャ

### 6．1　マージャの機能

マージャは、関係データベースの処理で必要となる基本演算を実行するモジュールで、RDBEの中核部である。マージャの構成を図４に示す。

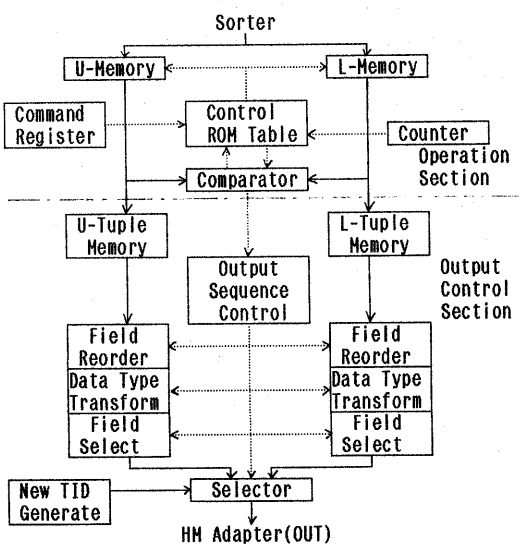



図4　マージャの構成

マージャは、演算部と出力制御部に分かれている。

演算部は、比較器、制御ROM 、FIFO機能を有する２つの64KBメモリ（U-MemoryとL-Memory）などからなる。演算部では、次のようなステップで演算を実行する。

① ２つのソートされたストリームをソータから受取り、２つのメモリのうちソフトウェアから指定された方に格納する。

② ２つのメモリのおののからそれぞれのレコードを同期して読みだし、比較器と出力演算部に転送する。

③ 各レコードのキー・フィールドの比較操作を行い、マージャ・コマンドの条件を満たすレコードを出力する。

また、演算部は、演算の種類と比較結果により、次に処理すべきレコードを決定し、このためのメモリの読みだしアドレス制御を行う。

出力制御部は、16KBのタプル・メモリ、フィールド再配置回路、フィールド選択回路、データタイプ変換回路、新TID 付加回路等からなり、ソフトウェアの制御のもとに次の機能を持つ。

(1) 出力するレコードのフィールドの再配置
(2) 出力するレコードのフィールドの選択
(3) INモジュールで行なったデータタイプ変換に対して、元のデータタイプに戻す。
(4) 出力するレコードに新TID を付加する。

これらの機能がどのようになされるのかを図５に示す。図5(a)は出力するレコードのフィールドの再配置を表わす。５つのフィールド（A,B,C,D,E;B がキー・フィールド）からなるレコードはINモジュールによって、キー・フィールドがレコードの先頭に来るように変換される。変換されたレコード②は、マージャの出力選択部により元に戻される（レコード③）。また、図5(b)は、出力するレコードのフィールドの選択を表わす。レコード④は、２つのポインタによりレコード⑤とレコード⑥に切出される。図5(c)は、出力するレコードへの新TID の付加を表わす。

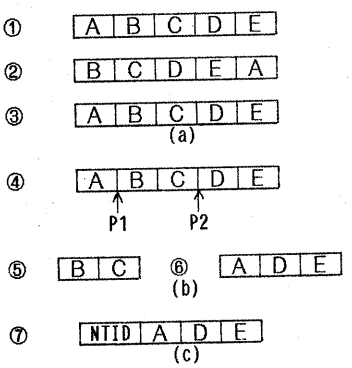



図５ 出力制御部の機能

## ６．２ マージャ・コマンド

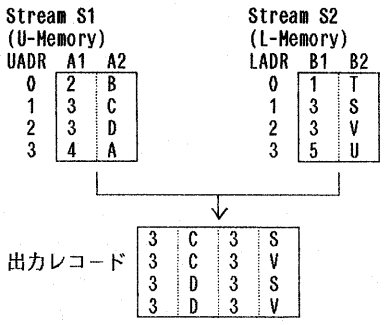
マージャで行なわれる演算は、マージャ・コマンドにもとづいて行われ、その種類を表２に示す。２つのリレーションのEqui-Join 演算を行う時に使用されるJOIN-EQ コマンドの処理例を図６に示す。JOIN-EQ コマンドの処理アルゴリズムは次の通りである。

```
if A1 > B1 then LADR := LADR + 1;
if A1 < B1 then UADR := UADR + 1;
if A1 = B1 then
  output a matched tuple pair
  if the DP of A1 and DP of B1 are on,
    then LADR := LADR + 1;
  if the DP of A1 is on, and the DP of B1 is off,
    then UADR := UADR + 1;  LADR := LADR*;
  if the DP of A1 is off, and the DP of B1 is on,
    then  LADR := LADR + 1;
  if the DP of A1 and DP of B1 are off,
    then UADR := UADR + 1;  LADR := LADR + 1;
```

ここで、LADR,UADR はそれぞれL-Memory,U-Memory のアドレスを表わす。LADR* は、同じ値を持つレコードの最初のレコードのアドレスを表わす。DPは重複シグナルを表わす。

表２ マージャ・コマンドの一覧

| コマンド名 | 機能 |
|---|---|
| PASS系コマンド | |
| LOAD | 入力レコードをUまたはLに格納 |
| PASS-1 | UまたはLのレコードをそのまま出力 |
| PASS-2 | UまたはLのレコードを重複を除いて出力 |
| SORT系コマンド | |
| SORT-EX | UとLのレコードのマージ・ソート |
| UNQ-EX | U，L内とU，Lの間の重複を除いて出力 |
| RESTRICT系コマンド | |
| REST-NULL | Lからnull値のレコードだけ出力 |
| REST-NONULL | Lからnull値でないレコードだけ出力 |
| REST-EQ | Uのレコードを条件として条件を満たすLのレコードを出力する |
| REST-NE | Uのレコードを条件として条件を満たさないLのレコードを出力する |
| REST-RANGE | Uのレコードを入力順に２個ずつ組にしてその値の範囲に入るLのレコードを出力する |
| COMPARE系コマンド | |
| COMP-ALL | UとLのレコードを入力順に連結してすべてを出力する |
| COMP-NULL | UとLのレコードを入力順に比較してどちらかがnull値の組だけを出力する |
| COMP-NONULL | UとLのレコードを入力順に比較してどちらもnull値でない組だけを出力する |
| COMP -$\theta$ | （$\theta$: EQ,NE,LT,GT,LE,GE ）UとLのレコードを入力順に比較して演算子$\theta$で指示された組だけを出力する |
| JOIN系コマンド | |
| JOIN-ALL | UとLのレコードのデカルト積を出力 |
| JOIN-NULL | UとLのレコードのデカルト積で少なくとも方方がnull値のものだけを出力 |
| JOIN-NONULL | UとLのレコードのデカルト積でどちらもnull値でないものを出力 |
| JOIN -$\theta$ | （$\theta$：EQ,NE,LT,GT,LE,GE ）UとLの$\theta$-Joinを行う |




| UADR | LADR | U タプル | L タプル | 比較結果 | 出力 |
|---|---|---|---|---|---|
| 0 | 0 | 2B | 1T | > | |
| 0 | 1 | 2B | 3S | < | |
| 1 | 1 | 3C | 3S | = | 3C3S |
| 2 | 2 | 3C | 3V | = | 3C3V |
| 2 | 1 | 3D | 3S | = | 3D3S |
| 3 | 2 | 3D | 3V | = | 3D3V |
| 3 | 3 | 4A | 5U | < | |
| END | 3 | | | | |

図６　JOIN-EQ の処理例

## ７．　ハードウェアの制御方式

この章では、データベースに対する各種処理を行うために、REのハードウェアをどのように制御しているのかについて記す。まず、HMアダプタとエンジン・コア（INモジュール、ソータ、マージャの総称）の制御方式について述べ次にエンジン・コアでは処理できない演算をどのように処理するのかを述べ、最後にエンジン・コアで１回では処理できない大量なデータに対する処理方式について述べる。

### ７．１　HMアダプタとエンジン・コアの制御方式

HMアダプタ(IN)は、HMからデータを受取り、それをエンジン・コア（INモジュール、ソータ、マージャの総称）に送る。JOINなどの演算では、演算対象となる２つのリレーション中のレコードをソータ容量(64KB)ごとにブロック・マルチプレクスして転送する。RDBE制御プログラムは、このブロックを単位としてHMアダプタを制御している。このようにした理由は、２種類の演算対象リレーションをひとつのHMアダプタで適宜切替えて入力するのに必要不可欠なためである。

一方、エンジン・コアを使用して関係代数演算を行うために、処理対象となるリレーションの量などの情報と演算の種類を前もってエンジン・コアにセットしてから起動しなければならない。このため、HMアダプタ(IN)とエンジン・コアは、常に同期して制御すればよい。

HMアダプタ(OUT) は、エンジン・コアの出力をHMに転送するのに使用される。エンジン・コアに入力されるデータと出力されるデータは量的に無関係であるので、HMアダプタ(OUT) とエンジン・コアは非同期に動作する。また、通常、出力するためのHM上のバッファは１つだけ必要だが、大量データのSort演算では2 way merge sort法を採用しているため、２つのバッファが必要となる。このため、入力が２つある場合と同じく、ブロックごとに出力先を制御している。

### ７．２　マイクロプロセッサによるデータ処理方式

データベースに対する処理には種々のものがあり、それらをすべてエンジン・コアでサポートしているのではなくて、次の演算（特殊演算と呼称する）は、RDBE内のマイクロプロセッサによりソフトウェアにより処理される。

(1) Null値同志を互いに異なるものとして扱うマルチUnique演算（Uniqueする対象のフィールドが複数あるUnique演算）および集合演算
(2) 複数の比較条件がAND/ORで結合されている、あるいは、算術演算を伴うSelection 、Join演算。
(3) あるフィールドの値を別の値（算術演算の結果など）で置換える代入演算
(4) いくつかのフィールドについて値が等しいものどうしをグループとし、各グループごとに別のフィールドの最大値、最小値、平均値などを求める集約演算。

特殊演算をエンジン・コアでなく、マイクロプロセッサでソフトウェア的に処理するようにした理由は次の通りである。

(1) エンジン・コアのハードウェア量を削減するため。
(2) RDBEの処理に柔軟性を持たせるため。
(3) これらの演算は使用頻度が低いと予想されたため。

特殊演算は、CPから送られてくるコマンドのパラメータで指定される。これに基づいて処理を行うには、各レコードごとに毎回パラメータを参照しながら処理を行うインタプリタ方式と、最初にそのパラメータに基づく処理を行うためのオブジェクト・コードを生成し、各レコードごとにそのオブジェクト・コードを実行するコンパイラ方式とが考えられる。オブジェクト・コードを生成する必要がないという点では、インタプリタ方式の方が優れているが、レコードひとつあたりの処理時間ではコンパイラ方式の方が優れている。特殊演算では、同じ形式をしたレコードがRDBEのメインメモリ上に多数連続配置されることを考慮してコンパイラ方式を採用した。そして、同じコンパイラ方式のうちでも最も高速処理ができるように、マイクロプセッサが直接実行可能なオブジェクト・コードを生成するコンパイラ方式を採用した。

特殊演算の基本的な処理方式は次の通りである。

① CPから送られてくるコマンドを解析して、ソフトウェアでの処理が必要であるか判定する。必要な場合には、コンパイラを呼出しRDBEのマイクロプロセッサが直接実行可能なオブジェクト・コードを生成する。
② HMからHMアダプタ(IN)経由で処理対象となるデータを入力する。

③ ②のデータに対して、必要があれば、エンジン・コアで関係代数演算を実行する。

④ ③の結果をHMアダプタ経由でRDBEのメインメモリに転送する。

⑤ メインメモリに格納されたデータに対して、①で生成したオブジェクト・コードを実行する。

⑥ ⑤の実行結果をメインメモリよりHMアダプタ(OUT)経由でHMに転送する。

データ量が多過ぎて一度にメインメモリ上に入りきらない場合には、④～⑥の処理を繰返す。

7．3 大量データの処理方式

ここでは、ソータ容量(64KB)を越える大量なデータの処理方式について述べる。まず、表記法について説明する。

・入力データを$A_1$～$A_m$および$B_1$～$B_n$で表わす。ただし、$A_i$ $B_i$はブロックである。

・get($A_i$) は、HMアダプタ(IN)からブロック$A_i$を受けることを意味する。

・「ソータ ＊＊＊ マージャ ×××」は、ソータとマージャの演算の種類を表わす。

・同一行に記したものは、同時に実行することを意味する。

・isempty(U)は、マージャのU-Memoryが空であれば真、そうでなければ偽となる。

その処理方式には、次に示すように演算の種類に応じて4通りある。

(1) Pass型

レコード間の比較を必要としない演算では、ソータ容量ごとにブロック化して区切ってHMから入力することにより処理する。たとえば、代入演算は次のように処理される。

```
for i=1 to m
  get(Ai)   ソータPASS  マージャPASS-1(U)
```

(2) Join型

ふたつのリレーションに関する演算で、それぞれのリレーションの各要素のすべての対について調べればよい演算は、それぞれのリレーションをソータ容量ごとにブロック化し、すべてのブロックの対について処理する。たとえば、JOIN演算は、次のように処理される。

```
for j=1 to n
  get(Bj)   ソータSORT  マージャLOAD(U)
  for i=1 to m
    get(Ai)   ソータSORT  マージャJOIN-XX(L)
```

(3) Difference型

差集合(A-B) を求めるような演算では、 Bが大量の場合、 Bをブロックに分け、$((A-B_1)-B_2)-\cdots\cdots-B_n)$を求めることにより処理する。

(4) Sort型

Sort演算は、次のように処理される。

① データ量がソータ容量以下の場合

```
get(A )   ソータSORT  マージャPASS-1(U)
```

② データ量がソータ容量の２倍以下の場合

```
get(A )   ソータSORT  マージャLOAD(U)
get(B )   ソータSORT  マージャSORT-EX(L)
```

表３ 主な演算の処理方式

| 演算の種類 | 区別 | 実現手段 | 使用するマージャコマンド | 大量データ処理方式 |
|---|---|---|---|---|
| 属性と定数とのSelection 演算 | １つの属性について、比較条件が１つだけ、あるいは複数個ある場合 | HW | RESTRICT系 | Join型 |
| | １つまたは複数の属性について、比較条件がAND で結合されている場合 | HW+SW | | |
| | 上記以外の複雑な比較条件、あるいは比較条件に算術演算を伴う場合 | SW | PASS-1 | Pass型 |
| 属性どうしのSelection 演算 | 比較条件が１つだけの場合 | HW | COMPARE 系 | |
| | 複数の比較条件がAND/ORで結合されているあるいは算術演算を伴う場合 | SW | PASS-1 | |
| Join演算 | 比較条件が１つだけの場合 | HW | JOIN系 | Join型 |
| | 複数の比較条件がAND/ORで結合されているあるいは算術演算を伴う場合 | SW | PASS-1 | Pass型 |
| Sort演算 | | HW | SORT-IN,SORT-EX | Sort型 |
| Intersection演算 | Null値どうしを同じものとみなしてよい場合 | HW | REST-EQ | Join型 |
| | Null値どうしを異なるものとして扱う場合 | HW+SW | | |
| Union 演算 | Null値どうしを同じものとみなしてよい場合 | HW | PASS-2,UNQ-EX SORT-EX | Sort型 |
| | Null値どうしを異なるものとして扱う場合 | HW+SW | SORT-IN,SORT-EX | |
| Difference演算 | Null値どうしを同じものとみなしてよい場合 | HW | REST-NE | Difference型 |
| | Null値どうしを異なるものとして扱う場合 | HW+SW | | |
| Unique演算 | Unique演算の対象となるフィールドが１つだけの場合、あるいはNull値どうしを同じものとみなしてよいマルチUnique演算 | HW | PASS-2,UNQ-EX SORT-EX | Sort型 |
| | Null値どうしを異なるものとして扱うマルチUnique演算 | HW+SW | SORT-IN,SORT-EX | |
| 代入演算 | | SW | PASS-1 | Pass型 |
| 集約演算 | | SW | PASS-1 | |

if isempty(U) then マージャPASS-1(L)
else マージャPASS-1(U)

③ データ量がソータ容量の２倍を越える場合
ソータ容量の２倍ごとの並べかえを行い、2-way merge sortを何回もくりかえすことにより処理する。
図７にソータ容量の８倍の場合の処理の例を示す。

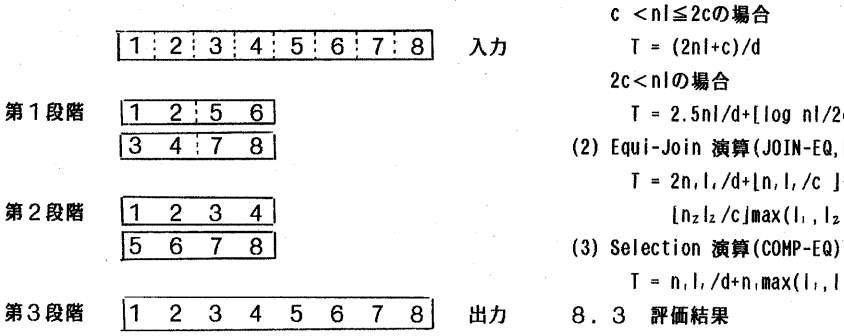



図７　大量データのSortの例

また、表３に主な演算の処理方式の概要を示す。

## ８．　RDBEの性能評価

RDBEの処理する演算のうち、代表的な演算について性能評価を行った。それについて記す。

### ８．１　性能評価方法

(1) システム構成

HMと 1台のRDBEを用いる。

(2) 演算対象データ

通常、演算対象となるリレーションは、HMの可動ヘッド・ディスク上に格納され、RDBEで処理されるためにHMの半導体メモリ上にステージングされる。しかし、ここでは演算対象となるリレーションは、RDBEで乱数発生させ、すべて半導体メモリ上に格納される。レコードの長さはすべて16バイトとする。

(3) 時間の測定方法

RDBEのコマンドの解釈からレスポンスの作成までの時間間隔をRDBEの計時モジュールを用いて測定する。

### ８．２　エンジン・コアのよる処理時間の見積り

エンジン・コアによる処理時間は、理論的に見積ることができる。以下、各演算の処理時間の理論値を記す。

なお、記号の意味は次の通りである。

n：　レコードの個数
l：　レコードの長さ
$n_i$：　i番目のリレーションのレコードの個数
$l_i$：　i番目のリレーションのレコードの長さ
g：　生成されたレコードの個数
c：　ソータ容量(64KB)
d：　データ転送速度(3MB/SEC)
T：　処理時間(SEC)
⌈a⌉：　aを切上げた結果が⌈a⌉になることを表わす。

(1) Sort演算(SORT-IN,SORT-EX)

$n l \leqq c$ の場合

$T = 2nl/d$

$c < nl \leqq 2c$の場合

$T = (2nl+c)/d$

$2c < nl$の場合

$T = 2.5nl/d+\lceil \log nl/2c \rceil 1.5nl/d$

(2) Equi-Join 演算(JOIN-EQ,REST-EQ)

$T = 2n_1l_1/d+\lceil n_1l_1/c \rceil\{n_2l_2+(c/l_1+c/l_2)\lceil n_2l_2/c \rceil \max(l_1,l_2)\}/d+g(l_1+l_2)/d$

(3) Selection 演算(COMP-EQ)

$T = n_1l_1/d+n_1\max(l_1,l_2)/d+g(l_1+l_2)/d$

### ８．３　評価結果

代表的な演算の処理時間の実測値と理論値との比較を図8～12に示す。図中、×は理論値を、 ― は実測値を表わす。

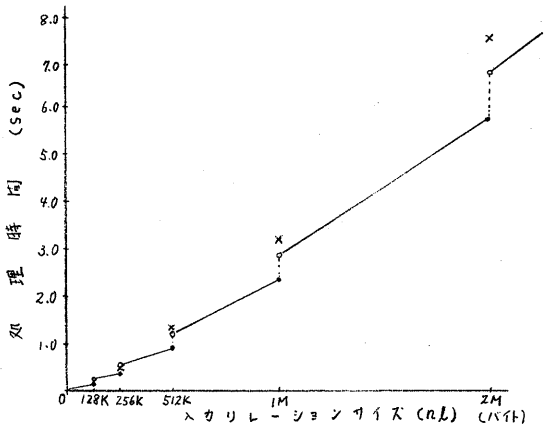



図８　Sort演算の実測時間と理論値

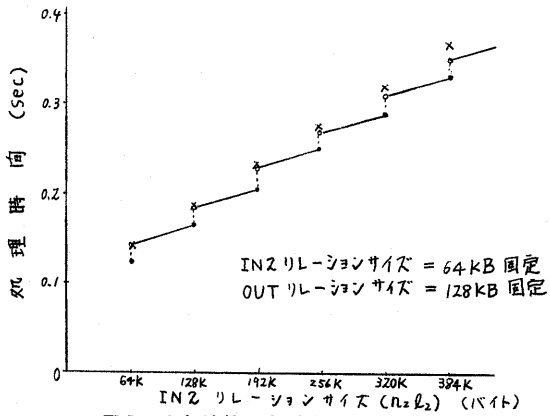



図９　Join演算の実測時間と理論値

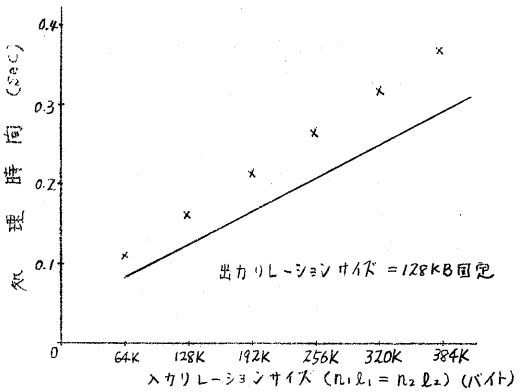



図10 Compare 演算の実測時間と理論値

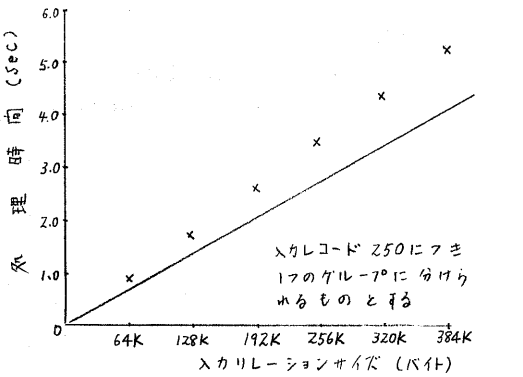



図11 集約演算の実測時間と理論値

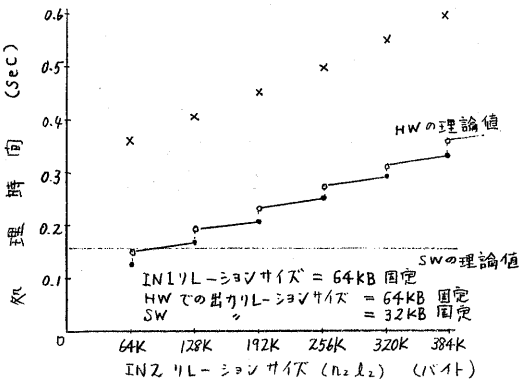



図12 Restrict演算(HW+SW) の実測時間と理論値

(注意) マイクロプロセッサによる処理については、コンパイラが生成したオブジェクト・コードとクロック・タイムより、その処理に要する時間を見積ってある。

8．4 考察

(1) 実測値が理論値よりも大きくなる理由は、RDBEのマイクロプロセッサによるコマンドの解釈やハードウェアの起動、HWでの処理の要する時間などのソフトウェアのオーバヘッドタイムを含むからである。また、オーバヘッドタイムは、処理すべきデータ量が多くなるにつれて増大する。

(2) Sort演算では、データ量の間隔が倍になるごとに階段状に上昇しているが、これは外部Sortを行うためである。

(3) Join演算では、データ量の間隔が64KBごとに階段状に上昇しているが、これはソータ容量ごとに分けて処理しているからである。

## 9． 終わりに

Delta の演算ユニットであるRDBEのハードウェア構成、制御方式、性能評価について述べた。

今後、4台のRDBEの並列処理方式の検討、およびその評価を行う予定である。

## 10. 謝辞

Delta の開発に当たって、ICOTメンバと協力し、かつインプリメントを担当された東芝、日立の研究者・技術者の各位に深謝する。

[参考文献]

(1) Knuth D.E. The Art of Computer Programming, Vol. 3,Sorting and Searching, Addition-Wesley,1973

(2) Codd, E.F. A Relational Model of Data for Large Shared Data Banks, CACM, Vol.13, No.6, June, 1970

(3) Todd, S. Algorithm and Hardware for a Merge Sort Using Multiple Processors, IBM J. RES. Develop., Vol.22, No.5, Sept., 1978

(4) Shibayama S., et al. A Relational Database Machine with Large Semiconductor Disk and Hardware Relational Algebra Processor, ICOT Technical report, TR-055, 1984

(5) Sakai H., et al. Design and Implementation of the Relational Database Engine, Proc. of Int'l Conf. on Fifth Generation Computer Systems 1984, Nov. 1984, and also ICOT TR-063

(6) Shibayama S., et al. A Relational Database Processing on an Attribute-based Schema, Proc. of IPSJ, Sep. 1984

(7) 比田井他、ソートをベースにしたデータベースマシン 情報処理学会第23回全国大会予稿 ,1981

(8) 角田、柴山、横田他、 RDBM Delta (1)-(3) 情報処理学会第26回全国大会予稿 ,1983

(9) 安部、酒井他、関係データベースエンジンの開発(1)-(6) 情報処理学会第29回全国大会予稿 ,1984

(10)酒井、安部他、関係データベースエンジンの基本演算の性能検討 情報処理学会第30回全国大会予稿 ,1985