ジョイン演算のための並列実行システム

A Parallel Execution System for Join Operation

中山 敬　三上 博司　平川 正人　田中 稔　市川 忠男

Takashi Nakayama　Hiroshi Mikami　Masahito Hirakawa　Minoru Tanaka　Tadao Ichikawa

広島大学　工学部

Faculty of Engineering, Hiroshima University

1．はじめに

関係モデル[1]は，データの表現が理解しやすい，データ独立性が高い，演算が数学的に定義されているなどの優れた点を持っており，一般のデータベースへの適用の他に，より高度な機能を備えたデータベース（知識ベース）システムを構築するための基本モデルとしても期待できる．

筆者らは，すでに関係モデルに基づく高機能データベースシステムARESを開発している[2]．ARESでは，データ間の相互関連にしたがって，入力データに対してセマンティックな関連のあるデータを引きだすことができる．これにより，問合せ条件に関する従来の断定的記述に対しても，必要に応じてある程度の幅をもたせた柔軟な解釈が可能となっている．

データ間のセマンティックな関連はリレーションの形でデータベースに蓄えられる．これをセマンティック・リレーションと呼び，各定義域ごとに定義する．また，従来のリレーションについては，コンベンショナル・リレーションと呼んでこれと区別する．セマンティック・リレーションをコンベンショナル・リレーションから独立して管理することによって，セマンティックスの定義をユーザに解放することができ，かつ，これに対するシステムの追従性も保証される．

意味的な関連をもつデータの検索要求が行われた時には，まず，検索条件がかかわる属性の定義域で規定されたセマンティック・リレーションに対してセレクション操作を行い，次にその結果とコンベンショナル・リレーションとのジョイン操作が行われる．このように，検索条件にあいまい性が付与された問合せもまた，従来の関係代数演算の範囲で実行可能となる．

ところで関係モデルでは，上記のような利点がある反面，実際の処理系を通常のアーキテクチャ上に構築する場合に関係代数演算の実行効率の低さが大きな問題として現われる．データベースの高機能化に伴ってこの傾向はますます顕著となり，したがって関係代数演算の効率的な実行を支援するための研究が今後重要となる[3][4]．

本論文では，関係代数演算の中でも特に処理負荷の高いジョイン演算に注目し，これを効率よく処理するためのシステムについて述べる．以後このシステムをジョインプロセッサと呼ぶ．

ジョインプロセッサはマルチプロセッサシステムであり，ジョイン操作を並列に実行するn台のプロセッサ（スレーブユニット）とこれらのプロセッサを管理するための1台のプロセッサ（マスタユニット）から構成される．

n台のスレーブユニットで個別にジョイン操作を実行するにあたっては，ハッシュ技法を用いることによって比較演算の負荷を低減させている．すなわち，マスタユニットは，ジョイン演算を行う属性の値についてハッシングを行い，リレーションの各タップルをそのハッシュ値に対応するスレーブユニットに転送する．

これによって，ジョイン条件が"="の場合には，それぞれのスレーブユニット内ですべての操作が完了する．それ以外の場合には，さらにリレーションの一方についてスレーブユニット

間でタップルの転送を行い，そのつど各スレーブユニットでジョイン操作を行う．但し，ジョイン条件が">"，"≧"，"<"，"≦"の場合には，ハッシングによってクラスタリングされたタップルの集合の間に順序関係が得られるようなハッシュ関数を用いることにより，転送すべきタップルを制限づけることができる．また，データ転送の方向を一方向に限定することができ，したがってハードウェアとその制御が簡略化される．

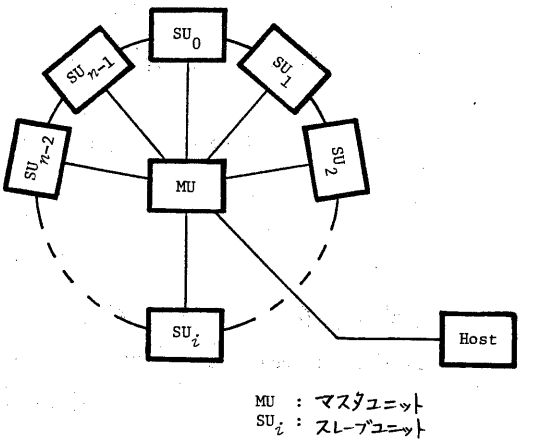
以下では，ジョインプロセッサのシステムアーキテクチャ，ならびにアルゴリズムについて詳述する[5]．さらに，マルチプロセッサ上で実際にシステムの開発を行っており，その実験結果を示す．なお，ジョイン演算の対象となる2つのリレーションを，それぞれソース・リレーションとターゲット・リレーションと呼ぶ．また，各スレーブユニットに分配されたリレーションについては，上記の区別にしたがって，それぞれソース・サブリレーションとターゲット・サブリレーションと呼ぶことにする．

2．システムアーキテクチャ

関係代数演算においては，一般にリレーションのサイズが小さくなるにしたがってその実行時間は短くなる．したがってリレーションをいくつかのサブリレーションに分割し，それらを複数個のプロセッサで並列に処理することによって全体としての実行時間が短縮できる．特にジョイン演算はリレーションの各タップルごとに独立に処理可能であり，したがって並列処理の導入によって期待される実行速度の改善度は大きい．

図1にジョインプロセッサのアーキテクチャを示す．システムは1台のマスタユニットとそれに連結したn台のスレーブユニットから構成されている．また各スレーブユニットSU_i $(0 \leqq i \leqq n-1)$は隣接するスレーブユニット$SU_{(i-1+n) \bmod n}$と$SU_{(i+1) \bmod n}$にそれぞれ接続されており，隣り合うスレーブユニットにデータの転送が行えるようになっている．

マスタユニットは，ジョイン条件が記述された属性に対してハッシュを適用し，そのハッシュ値に対応したスレーブユニットへソース・サブリレーションならびにターゲット・サブリレーションを転送する．さらに，ホストコンピュータとの入出力の管理も行う．スレーブユニットでは，対応づけられた個々のソース・サブリレーションとターゲット・サブリレーションについてジョイン操作が独立に行われる．また，必要に応じてターゲット・サブリレーションはスレーブユニット間で転送され，再びジョイン操作が行われる．



MU ：マスタユニット
SU_i ：スレーブユニット

図1　システムアーキテクチャ

3．アルゴリズム

3．1　リレーションの分配

ソース・リレーションならびにターゲット・リレーションを，それぞれいくつかのソース・サブリレーションならびにターゲット・サブリレーションに分割するためにハッシュを用いる．この場合に適用されるハッシュ関数は次の条件を満たすように決定される．

(1) 各スレーブユニットになるべく均等な数のタップルが割りあてられる．

(2) ジョイン条件が指定された属性の値i, jに対して，それらのハッシュ値をそれぞれ$h(i)$, $h(j)$とするとき，$i > j$ならば必ず$h(i) \geq h(j)$となる．

但し，(2)はジョイン条件として"<"，"≦"，">"，"≧"が指定された場合にのみ要求される．

図2にハッシュ値とその値をもつタップルが転送されるスレーブユニットとの対応を示す．ジョイン条件が"="，"≠"，">"，"≧"の場合にはハッシュ値の順序に対応して昇順に，"<"，"≦"の場合には降順になるようにスレーブユニットを割りあてる．

ハッシュ関数にこのような条件づけを与えることは次のような利点をもつ．

(1) データの流れを一方向に限定することができ，したがって制御メカニズムを簡略化することができる．

(2) ジョイン演算の実行にあたって，スレーブユニット間のデータ転送ならびに比較演算を最小限にすることができる．すなわち，

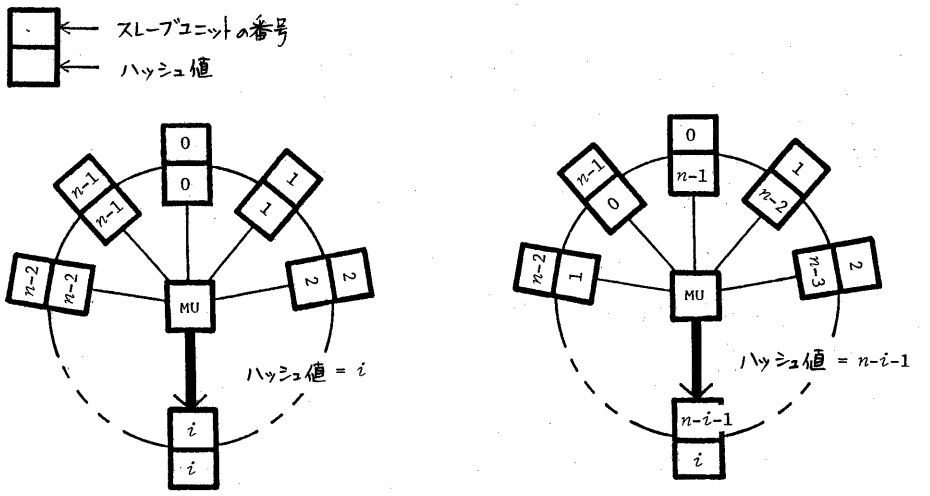
i) ジョイン条件が"="の場合には，最初にデータが割りあてられたスレーブユニット内の処理だけでジョイン演算が完了するので，スレーブユニット間でデータ転送を行う必要がない．

ii) ジョイン条件が">"，"≧"，"<"，"≦"の場合には，自分より番号の大きいスレーブユニットにだけジョイン条件を満たすデータが存在する．したがって，自分より番号の大きいスレーブユニットにだけデータを転送すればよい．

iii) ジョイン条件を必ず満たすデータが存在するスレーブユニットにだけデータを転送する．したがって，スレーブユニット間で転送されたデータに対しては比較演算を行う必要はなく，単に直積をとるだけでよい．

3.2 マスタユニットのアルゴリズム

マスタユニットは，ホストコンピュータからソース・リレーションおよびターゲット・リレーションを入力する．各タップルは，ジョイン条件が記述された属性の値から求められるハッシュ値に対応したスレーブユニットへ逐次転送される．その後，処理の終了したスレーブユニットから結果を受け取り，ホストコンピュータへ出力する．このためのアルゴリズムを以下に示す．



(a) ジョイン条件が =, ≠, >, ≧ の場合　　(b) ジョイン条件が <, ≦ の場合

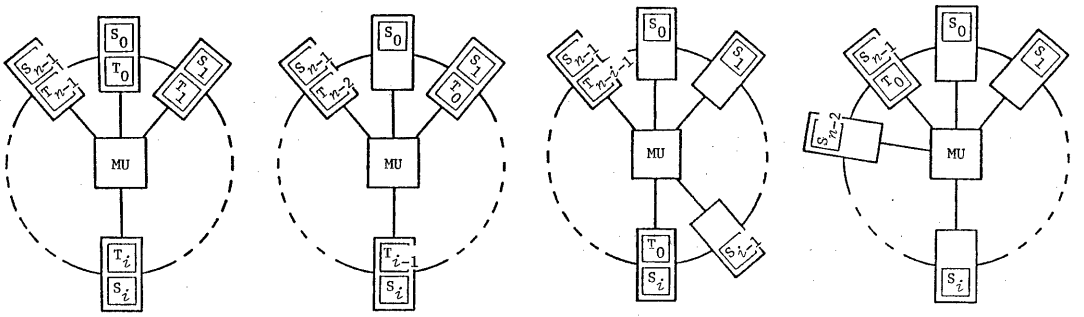
図2 ハッシュ値とスレーブユニットの対応

```
begin
  (*ソース・リレーションの転送*)
  while ソース・リレーションの終わりでない do
  begin
    ホストコンピュータからソース・リレーションを1タップル入力する;
    ジョイン条件が記述された属性の値からハッシュ値を求める;
    ハッシュ値に対応するスレーブユニットへそのタップルを転送する;
  end;
  (*ターゲット・リレーションの転送*)
  while ターゲット・リレーションの終わりでない do
  begin
    ホストコンピュータからターゲット・リレーションを1タップル入力する;
    ジョイン条件が記述された属性の値からハッシュ値を求める;
    ハッシュ値に対応するスレーブユニットへそのタップルを転送する;
  end;
  (*結果のリレーションの転送*)
  while 結果のリレーションの終わりでない do
  begin
    when スレーブユニットから結果のリレーションが1タップル転送された do
      そのタップルをホストコンピュータへ出力する;
  end;
end.
```

3.3 スレーブユニット SU_i $(0 \leq i \leq n-1)$ のアルゴリズム

スレーブユニットは，マスタユニットから転送されたソース・サブリレーションとターゲット・サブリレーションとのジョイン操作を行う．ジョイン条件が"="以外の場合には，図3，図4に示すように，その後スレーブユニット間でターゲット・サブリレーションの転送を行い，直積操作を行う．このためのアルゴリズムを以下に示す．

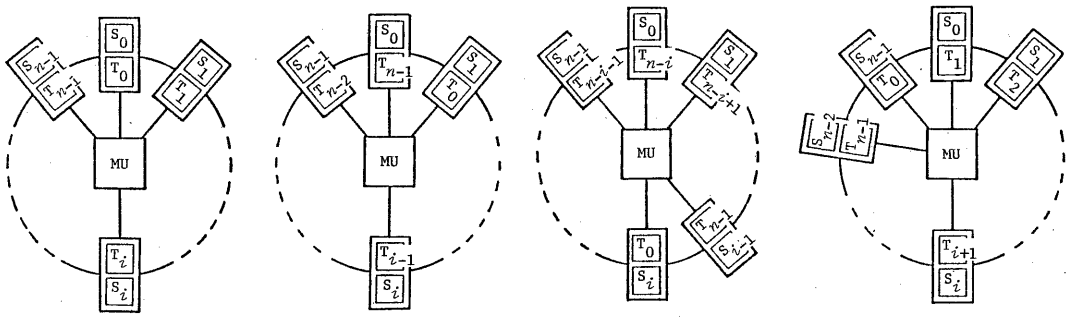
```
begin
  (*ソース・サブリレーションの入力*)
  while ソース・リレーションの終わりでない do
  begin
    when マスタユニットからソース・リレーションが1タップル転送された do
      そのタップルを受け取る;
  end;
  (*ターゲット・サブリレーションの入力およびソース・サブリレーションとターゲット・サブリレーションのジョイン*)
  while ターゲット・リレーションの終わりでない do
  begin
    when マスタユニットからターゲット・リレーションが1タップル転送された
    do そのタップルを受け取る;
      そのタップルとソース・サブリレーションとのジョインを行う.
  end;
  (*ターゲット・サブリレーションの転送およびソース・サブリレーションとターゲット・サブリレーションの直積*)
  case ジョイン条件 of
  ">", "≧", "<", "≦":
    for j:=1 to i do
    begin
      if i≠n-1 then
      ターゲット・サブリレーションをスレーブユニット SU_{i+1} へ転送する;
      when ターゲット・サブリレーションがスレーブユニット SU_{i-1} から転送された do
        ソース・サブリレーションとターゲット・サブリレーションとの直積を行う;
    end;
    if i≠n-1 then
    ターゲット・サブリレーションをスレーブユニット SU_{i+1} へ転送する;
  "=":
    for j:=1 to n-1 do
    begin
      ターゲット・サブリレーションをスレーブユニット SU_{i+1} へ転送する;
```



(a) 初期状態　(b) 1回転送後　(c) i回転送後　(d) 最終状態

S_i：ソース・サブリレーション
T_i：ターゲット・サブリレーション

図3　θ-join (>, ≧, <, ≦) の場合のデータ転送



(a) 初期状態　(b) 1回転送後　(c) i回転送後　(d) 最終状態

S_i：ソース・サブリレーション
T_i：ターゲット・サブリレーション

図4　θ-join (=) の場合のデータ転送

```
when ターゲット・サブリレーション
  がスレーブユニットSUi-1から転送さ
  れた do
  ソース・サブリレーションとターゲッ
  ト・サブリレーションとの直積を行う;
end;
end;
```

```
(*結果のサブリレーションの転送*)
while 結果のサブリレーションの終わりで
  ない do
  マスタユニットへ結果のサブリレーション
  を1タップル転送する.
end.
```

4. 実験と評価

4.1 実験システム

広島大学工学部計算機工学研究室で開発されているマルチマイクロプロセッサシステムUNIP[11]を用いて実験を行った.

UNIPの構成を図5に示す. UNIPは1台のマスタユニットとn台のスレーブユニットから構成されており, マスタユニットと各スレーブユニットは並列バスによって接続されている. また, 隣接するスレーブユニット間は互いに直列ポートによって接続されている. 各ユニットのハードウェア仕様を表1に示す.

プロセッサ間の通信はすべて同期的に行われる. すなわち, 送信側, 受信側いずれか先に通信を開始した方が待ち状態に入り, 相手が通信を開始するまで待ち続ける.

4.2 実験結果と評価

今回の実験では12台のスレーブユニットを使用した. ただし, スレーブユニットSU_{11}とSU_0は直接的には接続されていないため, この間のデータ転送はマスタユニットを介して行った. また, 各スレーブユニットに均等な大きさのリレーションが分配されるようなデータを用い, それぞれのタップルの長さは20バイトとした.

ホストコンピュータ(PC-8001)上でジョイン演算を行った場合とジョインプロセッサによってジョイン演算を行った場合について, それぞれの処理時間を実測した. ジョイン条件として"=", ">", "≠"を仮定したときのタップル数と処理時間との関係を, それぞれ図6, 図7, 図8に示す. ホストコンピュータのみで処理を行う場合と比べて, ジョインプロセッサでは明らかに実行速度の改善がなされている.

次にジョインプロセッサの場合について, タップル数を600としたときの処理時間の内わけを表2に示す. それぞれデータの転送時間と待ち時間の合計が57.4%, 71.9%, 84.7%を占めており, 転送時間の短縮と待ち時間の解消が重要である.

なお, 付録にジョインプロセッサの性能評価式を示す.

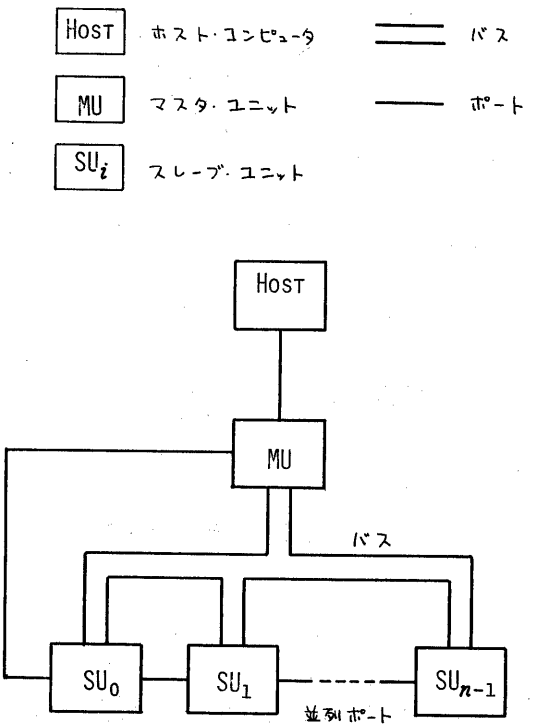


図5 UNIPの基本構成

マスタユニット

CPU	Z80A CPU (4MHz動作)
メモリ	ROM 2Kバイト, RAM 14Kバイト実装
並列バス	Z80A PIO
直列ポート	Z80A SIO
カウンタ/タイマ	Z80A CTC

スレーブユニット

CPU	Z80A CPU (4MHz動作)
メモリ	RAM 16Kバイト実装
並列バス	Z80A PIO

表1 マスタユニットおよびスレーブユニットのハードウェア仕様

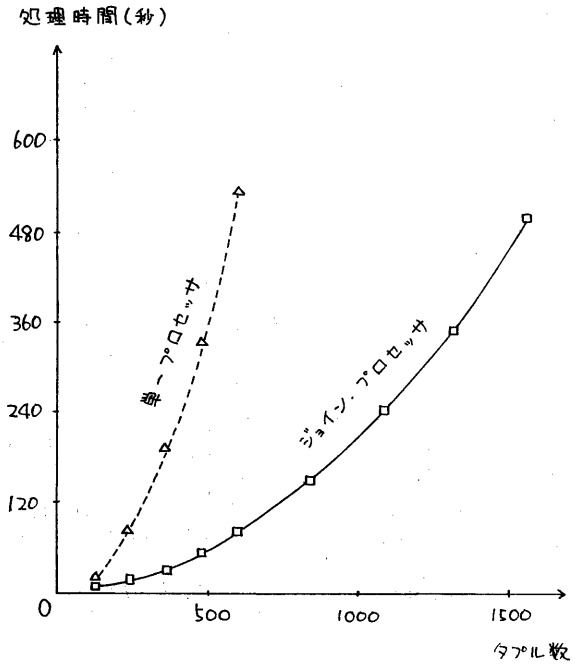


図6 単一プロセッサとの比較(ジョイン条件:=)

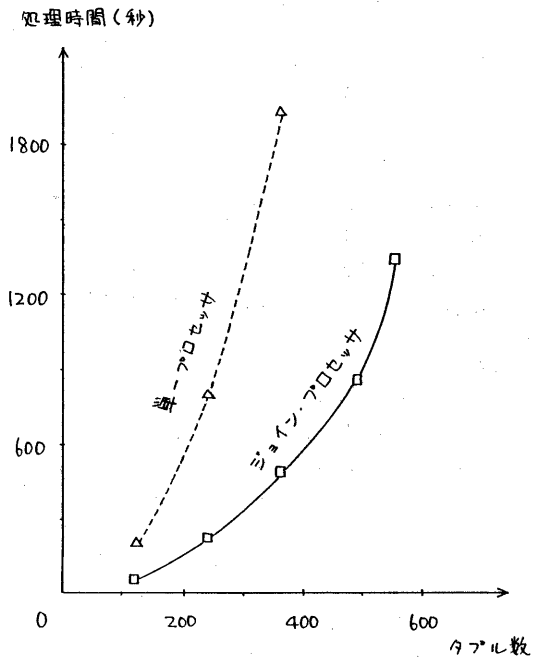


図7 単一プロセッサとの比較(ジョイン条件:>)

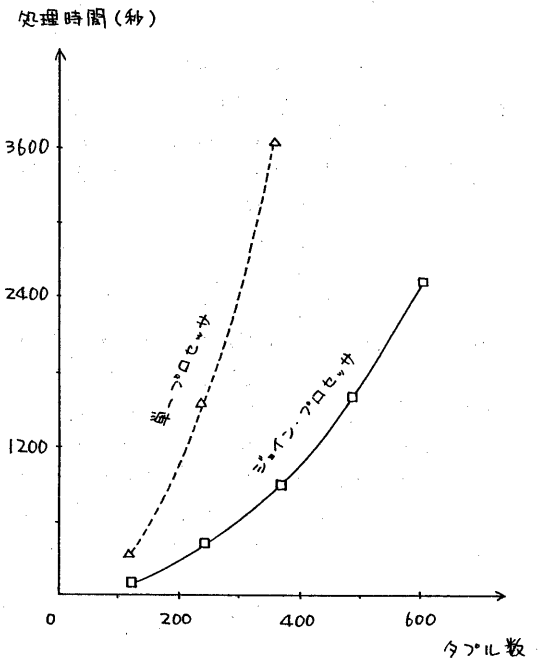


図8 単一プロセッサとの比較(ジョイン条件:≠)

比較演算子 / 内わけ	=		>		≠	
	処理時間(秒)	割合(%)	処理時間(秒)	割合(%)	処理時間(秒)	割合(%)
ソース・リレーションおよびターゲット・リレーションの転送時間	7.00	8.6	7.00	0.5	7.00	0.3
ハッシングに要する時間	10.34	12.7	10.34	0.8	10.34	0.4
比較演算に要する時間	24.26	29.9	40.48	3.0	49.98	2.0
結果のリレーションの転送時間と待ち時間	39.69	48.8	950.82	70.7	2114.11	83.8
スレーブ・ユニット間の転送時間	—	—	9.09	0.7	14.30	0.6
直積に要する時間	—	—	326.26	24.3	326.26	12.9
合計時間	81.30	100.0	1344.00	100.0	2522.00	100.0

表2 処理時間の内わけ(タップル数:600)

5. おわりに

本論文では,ジョイン演算を並列に処理するためのシステムアーキテクチャ,ならびにアルゴリズムについて述べた.システムは,1台のマスタユニットと,それと連結されているn台のスレーブユニットから構成されている.さらに,各スレーブユニットは隣接するスレーブユニットと接続されており,データの転送が行える.マスタユニットではまずリレーションのそれぞれのタップルについてハッシングを行い,次にそのハッシュ値に対応したスレーブユニットにタップルを転送する.各スレーブユニットは独立に動作し,それぞれジョイン操作を行う.

ところで,ジョイン条件が"="の場合,その条件を満足するタップルは必ず同一スレーブユニット上にある.したがって,それぞれのスレーブユニット内で処理が完了し,あらたにスレーブユニット間でデータ転送を行う必要がない.それ以外のジョイン条件をもつ場合にはスレーブユニット間でデータの転送がおこる.しかしながら,">","≧","<","≦"の条件をとるジョイン演算については,ハッシュバケット間に順序性を持つようなハッシュ関数を適用することによって,転送ならびにスレーブユニット内でのジョイン操作のコストを低減することができる.

現在,本学工学部阿江忠教授担当の計算機工学研究室で開発中のマルチマイクロプロセッサシステムUNIPを用いて,実験ならびに評価を行っており,今後データベースマシンの構成についても検討を進める予定である.

謝辞

実験設備の提供と,日頃頂いている有益なご討論に対して阿江忠教授に感謝する.また,日頃の討論と実験協力に対し,計算機工学研究室と情報システム研究室の諸氏に感謝する.

参考文献

[1] M. Brodie and J. Schmidt (Eds.), "Final Report of the ANSI/X3/SPARC DBS-SG Relational Database Task Group," ACM SIGMOD RECORD, Vol. 12, No. 4, July 1982.

[2] 平川,清水,市川,"APESにおけるデータベース操作機能の拡張" 情報処理学会論文誌, Vol.24, No.4, pp.513-520, July 1983.

[3] Special Issue on Data Base Machines, IEEE Computer, Vol. 12, No. 3, pp.7-79, March 1979.

[4] 田中,"データベースマシン," 情報処理学会誌, Vol23, No.10, pp.939-947, October 1982.

[5] T. Nakayama, M. Hirakawa and T. Ichikawa, "Architecture and Algorithm for Parallel Execution of a Join Operation," Proc., Conf. on Computer Data Engineering, 1984 (in print).
[6] T. Ae and R. Aibara, "Experimentation and Analysis of Multiprocessor Systems," Proc., Real-Time Systems Symp., pp.69-80, 1982.

付録 ジョインプロセッサの性能評価式

次のようなパラメータを用いる.

S ：ソース・リレーションのカーディナリティ
T ：ターゲット・リレーションのカーディナリティ
R ：結果のリレーションのカーディナリティ
S_i ：スレーブユニット SU_i に割り当てられたソース・サブリレーションのカーディナリティ
T_i ：スレーブユニット SU_i に割り当てられたターゲット・サブリレーションのカーディナリティ
R_i ：スレーブユニット SU_i で生成された結果のサブリレーションのカーディナリティ
α ：$S \times T$ と R の比（$R = S \times T \times \alpha$）
α_i ：$S_i \times T_i$ と S_i と T_i との間でジョイン条件を満たすタップル数の比
t_{comp} ：ソース・リレーションとターゲット・リレーションのジョイン条件が記述された属性の値を比較するのにかかる時間
t_{create} ：結果のサブリレーションを1タップル生成するのにかかる時間
t_1 ：マスタ（スレーブ）ユニットからスレーブ（マスタ）ユニットへ1タップル転送するのにかかる時間
t_2 ：スレーブユニット間で1タップル転送するのにかかる時間
n ：スレーブユニット数

(1) ジョイン条件が"="の場合

$$(S+T+R) \times t_1 + \max_{0 \le i \le n-1} (S_i \times T_i \times t_{comp} + S_i \times T_i \times \alpha_i \times t_{create})$$

(2) ジョイン条件が">", "≧", "<", "≦"の場合

$$(S+T+R) \times t_1 + \max_{0 \le i \le n-1} (S_i \times T_i \times t_{comp} + S_i \times T_i \times \alpha_i \times t_{create}) + \sum_{j=1}^{n-1} \left(\max_{j \le i \le n-1} (S_i \times T_{i-j} \times t_{create}) \right) + \sum_{j=1}^{n-1} \left(\max_{0 \le i \le n-1-j} (T_i \times t_2) \right)$$

(3) ジョイン条件が"≠"の場合

$$(S+T+R) \times t_1 + \max_{0 \le i \le n-1} (T_i \times t_2) \times (n-1) + \max_{0 \le i \le n-1} (S_i \times T_i \times t_{comp} + S_i \times T_i \times \alpha_i \times t_{create}) + \sum_{j=1}^{n-1} \left(\max_{0 \le i \le n-1} (S_i \times T_k \times t_{create}) \right)$$

$$k = \begin{cases} i-j & (i \ge j) \\ i-j+n & (i < j) \end{cases}$$

但し，上の式では処理時間と転送時間は重ならないと仮定している.