動的マークビットによるリレーショナル演算の並列処理方式とその評価

PARALLEL PROCESSING OF RELATIONAL OPERATIONS WITH TEMPORARY MARK BITS AND ITS EVALUATION

清木康　田中浩一　上林憲行　相磯秀夫
Yasushi KIYOKI, Kouichi TANAKA, Noriyuki KAMIBAYASHI, Hideo AISO
慶應義塾大学工学部
Faculty of Engineering, Keio University

1 はじめに

リレーショナル・モデルはその強固な理論的基盤，非構造的なデータ構造，アクセスの対称性，データ独立性，非手続的な問い合せ言語の実現の容易性，集合論的な基本演算子の完備性等，従来のデータベースモデルに比べて優れている点があるにもかかわらず，均質なテーブル構造によってデータ間の関係を示す必要から同様の関係を表現するのにデータ量が他のモデルに比べて大きくなる点，さらに集合論的な演算をソフトウェアで処理するのでは非常なオーバヘッドを招来する点などが問題とされ，優れたユーザインタフェースを提供する能力があるにもかかわらず具体的にシステム化するのにそうしたデータ量の増大や処理効率の悪さからもう一つ決め手を欠いていたというのが現実であると思われる。そうした意味でリレーショナル・データベースのハードウェア化による処理効率の向上・改善はリレーショナル・データベースの命運を左右する一つの鍵である。そのためリレーショナル・データベースマシンの研究・開発はデータベースマシンの大きなハイライトである。リレーショナル・モデルを基本モデルとして採用したデータベースマシンはRAPプロジェクトのOriginal RAP[1], RAP-2[2], RAP-3[3], CAFS[4], LEECH[5], DIRECT[6], EDC[7]等が代表的である。

筆者らは現在，リレーショナルデータベースマシンSPIRIT[8][9][10]の検討を進めているが，本稿ではSPIRITで採用したリレーショナル演算を動的マークビットを効果的に活用した並列処理方式の詳細，それに基づく基本演算アーキテクチャとそのハードウェア構成，さらに他の演算方式との比較を通して本稿で提案する方式が有効であることが確認されたのでここに報告する。

本稿の構成は第2章では動的マークビット方式の基本的な特長を他の方式との比較を通して明らかにする。第3章では具体的な演算アーキテクチャの説明を通して動的マークビットのデータ構造や各基本操作の詳細を明らかにして最後にその実現のための具体的なアーキテクチャを提示している。第4章では動的マークビット方式の評価をRAPおよびDIRECTで採用されている方式を正規化して定量評価式を与えることによって行い，その結果，本稿で提案する方式の有効性を示している。さらに動的マークビット方式がコード化リレーションに対する各種の演算に対して選択的なレコード操作が可能で並列・非同期処理に適した方式であり他の方式に比べて大きな処理能力があることを明らかにしている。

2 動的マークビット方式とそれに基づくリレーショナル演算アーキテクチャの特長

本章で説明する動的マークビットを活用した演算アーキテクチャは対象となるリレーションがSequencial Codeで置換されているコード化リレーションであることを前提条件として成立しえる方式である。

以下にリレーショナル演算の具体的な各アルゴリズムの方式の比較の基準と動的マークビットを活用した方式の特長を項目別に述べる。

〔1〕マーク情報方式：リレーショナル演算の具体的な方式上の最初の選択肢は各演算ごとにリレーション

の再構成を必要とするDIRECT方式とするか，何らかの中間情報を用いて再構成を回避する方式とするかであるが，一般的に再構成コストは甚大でありRAPのような tuple に固定的に付加されているマークビットを使用して再構成を極力押える方式が有利である．

[5] 動的マークビット方式：RAPの固定マークビット方式に比べて本稿で提案する動的マークビット方式（各 tuple から分離された独自なデータ構造によって属性間および リレーション間の演算結果を保持する）の有利な点としては①データベース格納媒体上にマークビット用の格納スペースが不要②マークビットは各演算やトランザクション単位で必要に応じて動的にアロケーションされるので固定マークビットで問題であったマークビット数の制限による複雑な演算の並行能力や tuple の共有によるマルチプログラミング多重度の制限を原則的に解消している。

[3] 属性単位の演算系の確立：また演算系の基本データ単位としては属性単位（Attribute 単位）の方式を採用している．属性単位の演算方式が tuple（リレーション）単位の演算方式に比べて有利な点としては①必要最小限のデータで処理が並行可能②格納単位が属性単位であれば不要な属性が転送されずにデータ転送時間の短縮が可能 等が考えられる．

[4] 並列アルゴリズム：リレーショナル演算の projection や Join の演算は本質的には逐次的な繰り返しアルゴリズムを必要とする．このためにこのアルゴリズムを改良しないかぎり並列性を高めても（セル数を増加）セル間交信のためのオーバヘッドが無視できず効果的な並列処理を期待できないというジレンマを解決するためにコード化値と動的マークビットを体系化することによって，完全非同期に各セルが並列動作可能なアルゴリズムを考案している。

[5] 動的マークビットの体系化：動的マークビットを単なる一次元的な有効 tuple の識別情報（VTF）として利用するだけでなく，さらにある属性に対する同一種別のグループ化を示す（VTF_g），リレーション間の演算結果をリンクテーブルという形で保持する（AIR），さらにコード化された属性のユニークな種別を求めるためにそれ各セル間での交信用に使用される（UTL）等に体系化することによってセル間の交信・同期コストを 1tuple あたり 1ビットで実現可能なアルゴリズムが実現され，繰り返し演算の演算オーダを tuple 数 × 種別数（k_1）の一般的なアルゴリズムに比べて本方式によると $k_1 \times k_2$ のオーダに変換して処理を行なうことが可能となっている．一般に tuple 数に比べて種別数は少ないのでこの効果は大きい．

[6] 選択型の tuple 処理：動的マークビットの体系化によって次に処理すべき有効 tuple の情報があらかじめもって判断可能となっているので，固定マークビット方式に比べて有効 tuple のみを選択的に処理することが可能で毎回全 tuple を検索するのに比べて効率向上が著しい．

[7] 機能分散方式：本方式では動的にコード化リレーションから動的マークビットの集合（Association 情報）を生成し，それらの間の演算を通して最終結果を導びく方式をとっており，本質的に Association 間の処理とコード化リレーションから Association 情報に変換する処理を機能分散することが可能である．総合的に処理効率の向上に寄与する．

[8] リプレースメント管理の必要のないアルゴリズム：処理プロセッサのメモリ容量に限界がある場合ページ単位にリプレースメントする必要がある演算子が存在するが，本方式では各データは基本的には全部 Association 情報に変換されその意味を，保持するのでコード化リレーションの入れ替えが不要となる．これによってページ単位のリプレースメントの甚大なコストが軽減される．

3 動的マークビット方式の処理アルゴリズム

処理アルゴリズムを論理的レベル，並列処理レベル，ハードウェアレベルでその詳細を述べる．

3.1 論理アーキテクチャ

3.1.1 論理アーキテクチャの基本概念

論理アーキテクチャを図3.1に示す．Association 集合は Entity 間の関係を示した情報の集合であり，その要素は複数のマークビット列である．Entity 集合は tuple の集合であり，リレーションという部分集合に類別される．複数のマークビット列はル-

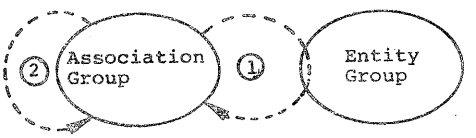

図3.1 論理アーキテクチャ

ション内のtupleの関係を示し，その1つのマークビット列はリレーション内の有効tupleを示している．矢印はこのデータ構造間の演算を示している．①は演算対象がAssociationによって選択されたEntityであり，結果がAssociationであることを示す．②はAssociationを対象とする演算で結果もAssociationであることを示す．

3.1.2 データ構造

Associationには次の3種類のデータ構造が用意されている．

(1) VTF (Valid Tuple Flags)：マークビット列によってリレーション内の有効tupleを示すものでありそのリレーション内で定義される．tupleの関連を示す基本的なデータ構造で他のデータ構造の要素ともなるが，単独でもSelection，Restriction等の結果として生成される．また次の演算の際に対象となるtupleを選択する働きを持つ．このように，有効tupleをマークビット列によって示す方法はデータが固定長であり処理が単純であるためハードウェア化が容易である．

(2) VTF_g (VTF_{group})：同一リレーションで定義されたVTFの並びで各VTFは共通の意味を持つtupleであることを示し，VTF_g全体でリレーションを分類している．各VTFをインデックスと考えればVTF_gは転置ファイルと見なせ，group by，Compound Projection等の演算時には生成される．このようにマークビットを高度に体系化することによって単なるマークビットでは不可能な演算が可能となる．

(3) AIR (Associations for Inter-Relation)：異なるリレーションで定義された2つのVTF_gの並びである．2つのVTF_gの対応するVTF同志は互いにJoinされるtupleを示している．VTF_gとVTF_gを結びつけることによってリレーション内およびリレーション間の演算も新たなリレーションを再構成することなく遂行可能である．

$VTF = \{1001\cdots\cdots01\}$：マークビット列

$VTF_g = \{VTF_1, \cdots\cdots, VTF_n\}$：各VTFは同一リレーションで定義されている．

$AIR = \{VTF_{g_1}, VTF_{g_2}\} = \{(VTF_{11}, \cdots VTF_{1n}), (VTF_{21}, \cdots, VTF_{2n})\}$：$VTF_{gi}$と$VTF_{gj}$は互いにJoinされるtuple群を示す．

3.1.3 基本操作

① EntityからAssociation情報への変換

VTFによって選択されたTupleからVTF，VTF_g，AIRを生成する．

* (リレーション，VTF) → VTF：Selection Restriction
* (リレーション，VTF) → VTF_g：Group by，Projection
* (リレーション，VTF，リレーション，VTF) → AIR：Join

② Association間の演算

* $VTF \oplus VTF \longrightarrow VTF$
* $VTF_{g_1} \oplus VTF_{g_2} \longrightarrow VTF_{g_3}$：$a_{1i} \oplus a_{2i} \rightarrow a_{3i}$ ここでa_{ij}はVTF，$\oplus$はbitでその論理演算
* $VTF_g \longrightarrow VTF$：$a_1 \oplus a_2 \oplus \cdots \oplus a_m \rightarrow VTF$
* $VTF_{g_1} \otimes VTF_{g_2} \longrightarrow VTF_{g_3}$：$VTF_{g_1} = (a_1, \cdots, a_m)$，$VTF_{g_2} = (a_1', \cdots a_n')$
  - Compound Projectionに利用する
    $VTF_{g_3} = (a_{11}'', \cdots a_{mn}'')$ $a_{ij}'' \leftarrow a_i \otimes a_j'$
  - Divisionに利用する
    $VTF_{g_3} = (a_1'', \cdots a_i'')$：$a_i$と$(a_1', \cdots a_n')$との間で演算を行ない結果によって$a_i$を$a_j''$として出力するかどうかを決定する．
* $VTF_{g_1} \otimes AIR \longrightarrow VTF_{g_2}$：$VTF_g$を定義しているリレーションを別のリレーションに変換し，リレーション間のCompound Projectionを行なうようにする．$VTF_{g_1} = (a_1, \cdots a_m)$，$AIR = \{(a_1', \cdots, a_n'), (b_1', \cdots b_m')\}$，$VTF_{g_2} = \{b_i, \cdots b_n\}$，$b_i = (a_i \cdot a_1')b_1' \oplus \cdots \oplus (a_i \cdot a_n')b_m'$
  $\oplus$：OR　・：内積　$(a_i \cdot a_j')b_j' = \begin{cases} a_i \cdot a_j' = 0 \\ \text{その他} \quad b_j' \end{cases}$
* $AIR_1 \otimes AIR_2 \longrightarrow AIR_3$：リレーションA-B間のAIR$_1$とB-C間のAIR$_2$からA-C間の関係を表わすAIR$_3$を生成する．
  $AIR = \{(a_1, \cdots, a_m), (b_1, \cdots b_m)\}$，$AIR_2 = \{(b_1', \cdots b_m'), (c_1', \cdots c_n')\}$，$AIR_3 = \{(a_1, \cdots a_m), (c_1, \cdots c_n)\}$．$(c_1, \cdots c_n) \leftarrow (b_1, \cdots b_m) \otimes AIR_2$
  a_i, b_i, c_iは異なるリレーションとする．

3.2 動的マークビットによる並列処理化

3.2.1 並列処理方式

集合演算を並列処理するためにリレーションをtuple単位に均質に分割して各セルに分担させ，各セルではその部分リレーションを対象とする処理を行ないマークビット群対象とする部分だけを生成・操作する．この結果，処理量が減るだけでなく1事とする最大マークビット量を減少する可能性がある．

例えば 1000 tuple のリレーションでは VTF_g の最大量は $1000 \times 1000 = 10^6$ bit であるが 10セルに分割すると $(100 \times 100) \times 10 = 10^5$ bit となる。

3.2.2 非同期並列処理

各セルでの処理を可能な限り非同期に実行するために UIL, Compact UIL (CUIL) という補助データ構造を導入する。これらは各セルで非同期には成立する VTF_g の要素である VTF のつながりを保つためのものである。UIL はリレーションの、ある属性中に存在するアイテム値を全種類並べたものである。各アイテム値はコード化されていることを前提としているから、VTF がマークビットによって有効 tuple 番号を示したように UIL もマークビット列によって存在するアイテム値を示すことにする。この場合各セルの UIL は共通の意味を持つのでこの UIL の各アイテム値に対応させて、この属性に対する VTF_g (転置ファイル)内の VTF を生成すれば VTF 間のつながりを保てる。CUIL は、アイテム値と関係なく単に VTF のセル間でのつながりを保存するものでセルの各 VTF が全体の VTF_g の何番目の VTF の一部分であるかを、マークビット列で表している。(図3.2)

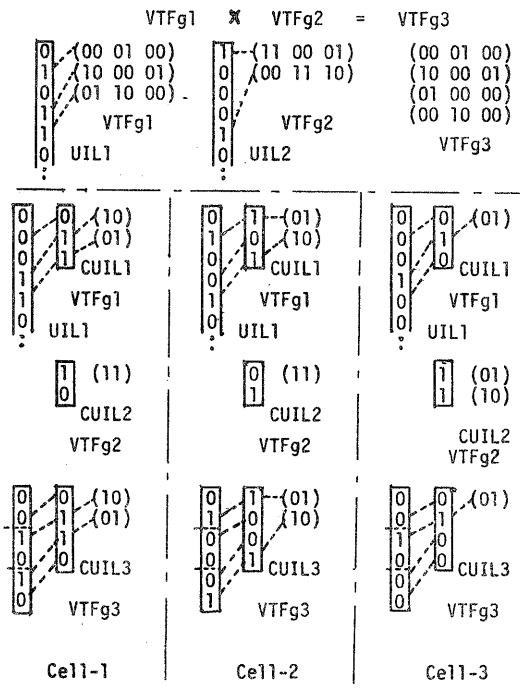

図3.2 VTF_g の並列処理

3.2.3 セル間交信

この処理方式でセル間通信とは、各セルで生成されるローカルな UIL (又は CUIL) の全ての OR をとることによって、グローバルな UIL (又は CUIL) を作成することである。これは従来の通信とは異ったものであるといえる。この交信は VTF_g, AIR を生成する演算にのみが必要となり通信量は UIL長 (bit長) であるが伝わる実質的な情報量はこの数十倍と考えられる。また、この通信と Cell内の処理は、ほとんどの場合分離できるので、機能分散処理化が可能である。

3.3 物理アーキテクチャ

3.3.1 ハードウェア構成

動的マークビット方式による並列処理演算は次のようなハードウェア構成で効果的に実現できる。

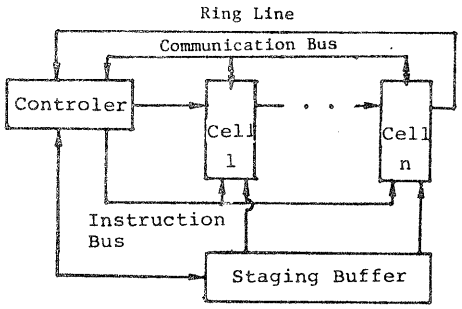

図3.3 動的マークビット方式のハードウェア構成図

・コントローラ：全セルにブロードキャストで命令を送りその完了は全セルで終了した時点で認識できる。コード値データをセルとステージングバッファの間で管理する。

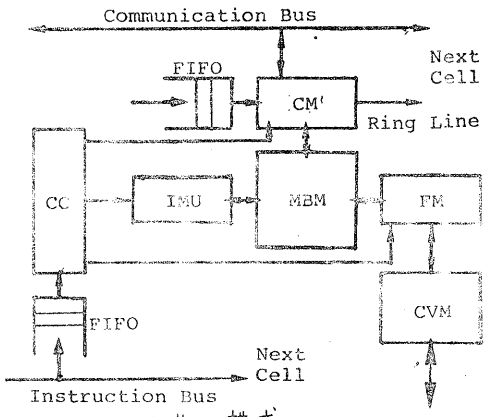

図3.4 Cell の構成

・CC：命令のシーケンスを考慮し，各ユニットが並列動作が可能なように命令を各ユニットに配る
・FM：コード値からマークビットを作成し，コード値の更新等のコード値に関する処理を全て行なう．
・IMU：マークビット間の単純な論理演算を行なう．
・CM：①セル間での通信をワイヤードORを利用して行なう。ここでは，全セルが同期しながら動作する。②命令の送り指示，関数演算の結果の累積等をセルからセルへ一方向に伝達し，最終的にコントローラに送る．
・MBM：VTF，VTFg，UIL等の全てのマークビットを格納するメモリでIMU，FM，CMの各ユニットからアクセス可能．
・CVM：リレーションの属性単位のコード値が各セルに均等に分割されて格納されているメモリ。

セル内はFM，IMU，CMに機能別のプロセッサが用意されている．それらは各々①コード値参照に対する処理②マークビット間の処理③通信に関する処理に大きく大別され，各機能を専用化して処理を遂行する．

3.3.2 ハードウェア基本命令

基本的な命令を表3.1に示す．このうちから幾つかの基本的なハードウェア命令をいくつか説明する．

・Generate UIL (ATL, VTF#1; UIL#1)

ATLとVTF#1で選択されるCVM内のアイテム値をフェッチし，そのコード化された値列によってアドレスされるUIL内のビットをマークする．VTFg，ATRを作成する際の前処理である．

・Generate VTF (ATL, VTF#1, UIL#1; VTF#2)：

ATLとVTF#1によって選択されるアイテム値自身によってアドレスされるUIL#1内のビットがマークされていれば，そのアイテム値をもつtupleを有効としてVTF#2にマークする．

・Generate VTFg (ATL, VTF#1, UIL#1; VTFg#1)

ATLとVTF#1によって選択されるアイテム値自身によってアドレスされるUIL#1内のビットがUIL#1の先頭から何番目のマークかによって，アイテム値に対応するVTFを認識し，そのVTFにそのアイテム値をもつtupleを有効としてマークする．

・Global OR UIL (UIL#1 ; UIL#2)

UIL#1のbit列をコミュニケーションバス上でワイヤードORを行ない，結果のグローバルなUILをUIL#2としておめる．

・Global OR UIL (UIL#1 ; CUIL#2)

UIL#1のbit列をコミュニケーションバス上でワイヤードORを行ない結果のグローバルなUILを得るとともに各セル内のUIL#1と比較することによってC-UILを生成する．

・Global OR UIL (UIL#1, UIL#2; CUIL#1, CUIL#2)：UIL#1のグローバルなUILとUIL#2のグローバルなUIL'とのANDによって，Join されるアイテム値を示すUIL"を作成する．

UIL#1 ← UIL#1 and UIL"
UIL#2 ← UIL#2 and UIL"

C-UIL#1とC-UIL#2はそれぞれUIL#1とUIL#2に対応するC-UILとする．

3.3.3 基本的なQueryの展開方法

① データ駆動型制御：コントローラはデータフローで表現された，セルへの命令列を持ち，下位メモリからのステージングマップによって，実行に必要なデータが揃った命令からセルへ送る．この結果処理中に最適な形で命令の実行順を決定でき，各セルにおいては，即座に実行が可能となる．

② セル内の並列処理：セル内の各ユニット(FM，IMU，CM)は並列動作が可能であるのでデータを充生成されていれば互いに並列処理を行なう．

表3.1 基本命令

	source	destination	unit
CS	ATL,VTF,OP,Value	VTF	FM
CS	ATL,ATL,VTF,OP	VTF	FM
G.UIL	ATL,VTF	UIL	FM
G.VTF	ATL,VTF,UIL	VTF	FM
G.VTFg	ATL,VTF,UIL	VTFg	FM
Gl.OR UIL	UIL	UIL	CM
Gl.OR UIL	UIL	CUIL	CM
Gl.OR UIL	UIL,UIL	(UIL,CUIL)*2	CM
Multiply	VTFg,VTFg	VTFg	IMU,CM
Multiply	VTFg,VTFg,VTFg	VTFg	IMU,CM
Select VTF	VTFg,VTFg	VTFg	IMU,CM
B.VTF	VTF,VTF	VTF	IMU
B.VTFg	VTFg	VTF	IMU
B.VTFg	VTFg,VTFg	VTFg	IMU
Total	ATL,VTF	ACC	FM
I.Total	ATL,VTFg	ACC	FM
Count Bit	VTF	ACC	IMU
Write	Value,VTF	ATL	FM

CS:Conditional Search, ATL:Attribute Location
G.:Generate, Gl:Global, B.:Boolean Operation
I.:Image, OP:Operator

```
SELECT UNIQUE S.CITY,J.CITY
FROM  S, SPJ, J
WHERE  S.S#=SPJ.S#
 AND   J.J#=SPJ.J#
```

SEQUEL

```
G.UIL(S.S# ; UIL1)
G.UIL(SPJ.S# ; UIL2)
G1.OR UIL(UIL1,UIL2 ; UIL1,CUIL1,UIL2,CUIL2)
G.VTFg(S.S#,UIL1 ; VTFg1)
G.VTFg(SPJ.S#,UIL2 ; VTFg2)

G.UIL(J.J# ; UIL3)
G.UIL(SPJ.J# ; UIL4)
G1.OR UIL(UIL3,UIL4 ; UIL3,CUIL3,UIL4,CUIL4)
G.VTFg(J.J#,UIL3 ; VTFg3)
G.VTFg(SPJ.J#,UIL4 ; VTFg4)

Multiply(VTFg2,VTFg4,VTFg3 ; VTFg5)

G.UIL(S.CITY ; UIL6)
G1.OR UIL(UIL6 ; CUIL6)
G.VTFg(S.CITY,UIL6 ; VTFg6)

Multiply(VTFg6,VTFg1,VTFg5 ; VTFg7)

G.UIL(J.CITY ; UIL8)
G1.OR UIL(UIL8 ; CUIL8)
G.VTFg(J.CITY,UIL8 ; VTFg8)

Multiply(VTFg7,VTFg8 ;VTFg9)
```

図 3.5 問い合わせの展開例

4. 動的マークビット方式の評価

前章で述べた動的マークビット方式（TMB方式）による集合演算アーキテクチャをRAP, DIRECTとの対比をまじえて評価を行なった。TMB方式の特徴は(1)選択的連想処理方式 (2)並列処理方式 (3)集合演算の結果の保持方法 (4)処理プロセッサの負荷量に対する配慮などの点である。これらは、リレーショナル演算を完全，かつ効率良く処理することを可能としている。

4.1 各演算方式の評価式と評価方法

一般にリレーションに対する集合演算は次のようなステップで実行されると仮定できる。① 演算対象tupleの認識 ② アイテム値の参照，および比較操作 ③ 演算結果の保持及び書き込み，である。①に関連してRAPではtupleに付随した固定マークビットを判断することによって行なわれる。DIRECTでは演算対象tupleと他のtupleを識別する特別なデータ構造は用意されていない。絶えず対象リレーションはすべて有効tupleであるように再構成が行なわれるので①の処理時間を必要としない。②のアイテム値の参照，及び比較操作は①の結果に従って行なわれる。③の演算結果の保持については TMB方式ではAssociation情報に変換される。RAPの場合はリレーション内の演算に対しては1ビットのマークビットを，Join等の演算では新しいリレーションを生成する。DIRECTではすべての演算の結果は新しいリレーションによって保持される。代表的なリレーショナル演算について①②③の項目別に各演算方式の評価を示す。

4.1.1 評価式

評価式内の係数 A_i, F_j, W_k は各々①.②.③に対応する係数であり評価式の一般形は(1)式のように表わされる

$$E = A_i \cdot G_1(t,n,k) + F_j \cdot G_2(t,n,k,d) + W_k \cdot G_3(t,n,k,d) \quad \cdots\cdots\cdots (1)式$$

(1)式の G_1, G_2, G_3 は t, n, k, d の関数であり各データベースマシンの特徴に応じて異なる．各係数は1データ処理に要する時間，コストあるいはある基準値に対する比，等の意味をもつ。係数の定義を次に行なう．（コストは．時間 その他の単位をとる）

- TMB方式の評価式の係数

A_{TI} ： 1マークビットを参照するのに要するコスト
A_{AU} ： G.UILにおいて1マークビットを参照するのに要するコスト
A_{AVg} ： G.VTFg 〃
F_{T2} ： 1アイテム値をフェッチ.比較操作するのに要するコスト
F_{AU} ： G.UILにおいて1アイテム値をフェッチするのに要するコスト
F_{AVg} ： G.VTFg 〃
F_{GOU} ： G1.OR UIL を実行するのに要するコスト
F_{AU} ： AND UIL 〃
F_{VTF} ： 1VTFをフェッチするのに要するコスト
W_{TI} ： 1マークビットをVTFに登録するのに要するコスト
W_{GU} ： G.UILにおいてUILに1マークビットを登録するのに要するコスト
W_{GVg} ： G.VTFg 〃 VTFg 〃
W_{WOU} ： G1.OR UILにおいて.UILを各セルへ書き込むのに要するコスト
W_{AU} ： AND UIL 〃
W_{SV} ： 1-CUILを書き込むのに要するコスト

- RAPの係数 A_{RI} ： 1マークビットを参照するのに要するコスト
F_{R2} ： 1アイテム値をフェッチ.比較するのに要するコスト，F_{R2}' ： 1アイテム値をフェッチし各セルへ登録するのに要するコスト，W_{RI} ： 1マークビットを登録するのに要するコスト，W_{R2} ： 1タップルを書き込むのに要するコスト

- DIRECTの係数 F_{D2} ： 1タップルをフェッチするのに要するコスト
F_{D2}' ： 1タップルをフェッチし.各セルへ登録するのに要するコスト
W_{D2} ： 1タップルを書き込むのに要するコスト

各係数は物理的要素で決定される値であり，変数部は各処理方式の方式上の特徴が反映される．各演算について TBM, RAP, DIRECTの各方式の評価式を示す．ただしRAPの場合，演算結果はマーク

ビットに保持され，Joinの場合はリレーションを生成する．DIRECTでは各演算後にリレーションの再構成を行なう．ゆえに常に全tupleが有効な演算対象となる．（E_T: TMB方式，E_R: RAP，E_D: DIRECT）

(1) Selection

$$E_{T\text{-}Select} = A_{T1}\cdot t/n + F_{T2}\cdot d\cdot t/n + W_{T1}\cdot d\cdot t/n \quad \cdots\cdots (1\text{-}1)式$$

$$E_{R\text{-}Select} = A_{R1}\cdot t/n + F_{R2}\cdot t/n + W_{R1}\cdot d\cdot t/n \quad \cdots\cdots (1\text{-}2)式$$

$$E_{D\text{-}Select} = F_{D2}\cdot d\cdot t/n + W_{D2}\cdot d\cdot t/(n\cdot k) \quad \cdots\cdots (1\text{-}3)式$$

(2) Projection

$$E_{T\text{-}Proj} = (A_{GU}+A_{GV})\cdot t/n + (F_{GU}+F_{GVe})\cdot d\cdot t/n + F_{GOU} + (W_{GU}+W_{GVe})\cdot d\cdot t/n + W_{GOU} \quad \cdots (2\text{-}1)式$$

$$E_{R\text{-}Proj} = 2A_{R1}\cdot k\cdot t/n + F'_{R2}\cdot k + F_{R2}\cdot(t\cdot k/n)\cdot((1+(1/k))/2) + W_{R1}\cdot(k+(d\cdot t-k)/n) \quad \cdots (2\text{-}2)式$$

$$E_{D\text{-}Proj} = F'_{D2}\cdot k + F_{D2}\cdot(d\cdot t\cdot k/n)\cdot((1+(1/k))/2) + W_{D2}\cdot k\cdot(d\cdot t/n)\cdot((1+(1/k))/2) \quad \cdots (2\text{-}3)式$$

(3) Join

$$E_{T\text{-}join} = (A_{GU}+A_{GVe})\cdot(t_S+t_P)/n + (F_{GU}+F_{GVe})\cdot(d_S\cdot t_S+d_P\cdot t_P)/n + F_{GOU} + F_{GOU} + F_{AU} + (W_{GU}+W_{GW})\cdot(d_S\cdot t_S+d_P\cdot t_P)/n + (w_{AOU}+w_{GOU}+w_{AU}) \quad \cdots (3\text{-}1)式$$

$$E_{R\text{-}join} = 4A_{R1}\cdot t_S/n + 3A_{R1}\cdot t_S\cdot t_P/n + F'_{R2}\cdot t_S + F_{R2}\cdot t_S\cdot t_P/n + 2W_{R1}\cdot d_S\cdot t_S + W_{R1}\cdot d_S\cdot t_S\cdot\alpha\cdot d_P\cdot t_P/(n\cdot k_P) + (W_{R2S}+W_{R2P})\cdot\alpha\cdot d_S\cdot t_S\cdot d_P\cdot t_P/k_P \quad \cdots (3\text{-}2)式$$

$$E_{D\text{-}join} = F'_{D2}\cdot d_S\cdot t_S + F_{D2}\cdot d_S\cdot t_S\cdot d_P\cdot t_P/n + (W_{D2S}+W_{D2P})\cdot\alpha\cdot d_S\cdot t_S\cdot d_P\cdot t_P/k_P \quad \cdots (3\text{-}3)式$$

(4) Implicit Join

$$E_{T\text{-}I\text{-}Join} = A_{GU}\cdot t_S/n + A_{GV}\cdot t_P/n + F_{GU}\cdot d_S\cdot t_S/n + F_{GOU} + F_{GV}\cdot d_P\cdot t_P/n + W_{GU}\cdot d_S\cdot t_S/n + W_{GOU} + W_{GV}\cdot\alpha\cdot d_P\cdot t_P/n \quad \cdots (4\text{-}1)式$$

$$E_{R\text{-}I\text{-}Join} = A_{R1}\cdot(t_S+t_S\cdot t_P)/n + F'_{R2}\cdot t_S + F_{R2}\cdot t_S\cdot t_P/n + W_{R1}\cdot d_S\cdot t_S + W_{R1}\cdot d_S\cdot t_S\cdot\alpha\cdot d_P\cdot t_P/(n\cdot k_P) \quad \cdots (4\text{-}2)式$$

$$E_{D\text{-}I\text{-}Join} = F'_{D2}\cdot d_S\cdot t_S + F_{D2}\cdot d_S\cdot t_S\cdot d_P\cdot t_P/n + W_{D2}\cdot d_S\cdot t_S\cdot d_P\cdot t_P\cdot k_P/n \quad \cdots (4\text{-}3)式$$

(5) Division

$$E_{T\text{-}Division} = A_{GU}\cdot(t_S+t_P)/n + A_{GVe}\cdot 2t_P/n + F_{GU}\cdot(d_S\cdot t_S+d_P\cdot t_P)/n + F_{GVe}\cdot 2d_P\cdot t_P/n + 2F_{GOU} + F_{VTF}\cdot\beta_{S(V)}\cdot k_S\cdot\beta_{P(V)}\cdot k_P + W_{GU}\cdot(d_S\cdot t_S+d_P\cdot t_P)/n + W_{GVe}\cdot 2d_P\cdot t_P/n + 2W_{GOU} + W_{SV}\cdot k_P \quad \cdots (5\text{-}1)式$$

$$E_{R\text{-}Division} = 2A_{R1}\cdot k_P\cdot t_P/n + 2A_{R1}\cdot t_S\cdot k_P + A_{R1}\cdot t_S\cdot t_P\cdot k_P/n + F'_{R2}\cdot k_P + F_{R2}\cdot(t_P\cdot k_P/n)\cdot((1+(1/k_P))/2) + F'_{R2}\cdot t_S\cdot k_P + F_{R2}\cdot t_S\cdot t_P/n + W_{R1}\cdot(k_P+(d_P\cdot t_P-k)/n) + W_{R1}\cdot(2d_S\cdot t_S\cdot k_P+d_P\cdot t_P/n) \quad \cdots (5\text{-}2)式$$

$$E_{D\text{-}Division} = F'_{D2}\cdot k_P + F_{D2}\cdot(d_P\cdot t_P\cdot k_P/n)\cdot((1+(1/k_P))/2) + F'_{D2}\cdot d_S\cdot t_S\cdot k_P + F_{D2}\cdot d_S\cdot t_S\cdot d_P\cdot t_P/n + W_{D2}\cdot k_P\cdot(d_P\cdot t_P/n)\cdot((1+(1/k_P))/2) + W_{D2}\cdot d_P\cdot d'_P\cdot t_P/n \quad \cdots (5\text{-}3)式$$

4.2 TMB方式の性能評価

(1) 選択的連想処理方式：TMB方式ではマークビットがtupleと物理的に分離されており，あらかじめ特定の有効tupleのみを選択的に操作することが可能である．この点は各式のFの項におけるdの値（$d \leqq 1$）によって表現されている．図-4.1はSelectionにおける処理時間をRAPを基準とした比によって示したものである．有効tuple数が{1/(種別数)}(=d)であると仮定するとSelectionに要する処理時間の比はdの値が小さくなる程TMB方式及びDIRECT方式はこの効果が著しい．他の演算においてもdの寄与する項によって選択的にレコード操作を行なえるので効率向上に大きく寄与する方式であるといえる．

(2) 並列処理の効果：一般にSelection（条件検索）は並列処理により1/(セル数)の処理時間となるが，Projection，Join，Implicit Join，Division等の演算ではセル間交信および多回の繰り返し比較演算が大きな負荷となる．つまり単純にセル数を増加しても交信コストを軽減する対策が必要である．TMB方式では前章で述べたように交信コストを1アイテムに対して1ビットでかつ複雑な制御を伴わない方式を考案して

いる．図-4.2，図4.3，図-4.4，図4.5はそれぞれ Selection，Projection，Join，Division の処理時間とセル数の関係を各評価式から定数の値を代入して導びいたものである．Selection では並列セル数の増加が処理効率を正比例的に向上させる．DIRECT ではリレーションの再構成コストを示す 1.3 式の第2項の効果で約100倍処理時間がかかる．Projection では RAP，DIRECT の場合（2.2）式 の F_{02}，k，（2-3）式 の F_{03} の項に漸近するので k の値が大きい場合はセル数を増しても並列性の効果が活かされなくなる．TMB 方式は $k=512$ の場合 RAP，DIRECT 方式と比べて各々約10倍～約100倍，数1000倍の効率向上がある．Join ではその結果を TMB で保持するので新しいリレーションを生成する必要がない．RAP，DIRECT では 3.2 式 第7項，3.3 式 第3項が示すリレーション再構成コスト および 3.2 式 第4項，3.3 式 第2項が示す繰り返しの比較操作の効果が図4.4に表われている．Division において RAP では（5-2）式 第4項，第6項，DIRECT では（5-3）式，第1項，第3項によってグラフは漸近する．TMB 方式では第6項がグラフを漸近させる．これは TMB 方式の VTF_g 間の演算が種別数の積の和になることからである．

(3) 集合演算結果の保持方式

集合演算の結果の保持方式は各処理方式の特徴を良く表わしている．図4.4 に示す Join の処理時間とセル数の関係は RAP，DIRECT の場合リレーションの再構成コストのためにセル数がある程度になると並列性の効果が活かされなくなる．TMB 方式ではリレーションの再構成コストの必要がないのでその点有利となる．

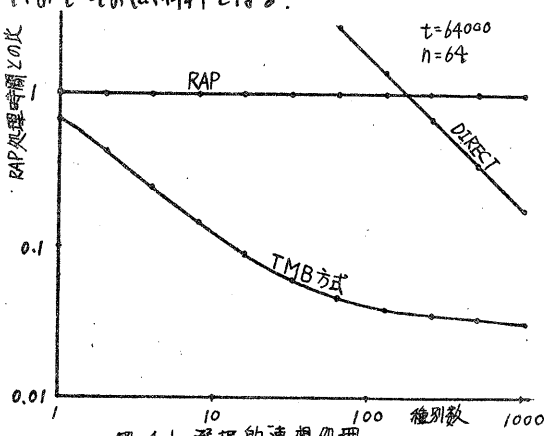

図-4.1 選択的連想処理

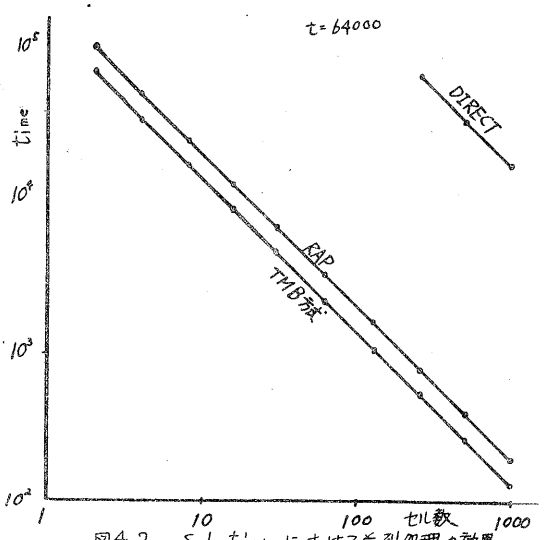

図4.2 Selection における並列処理の効果

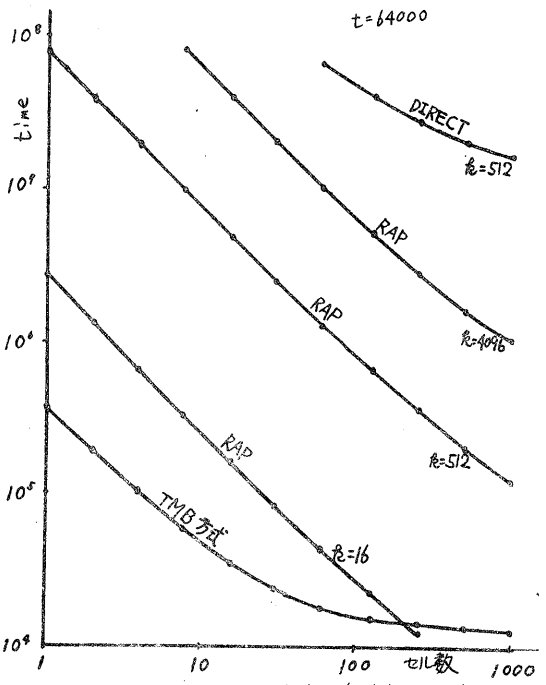

図4.3 Projection における並列処理の効果

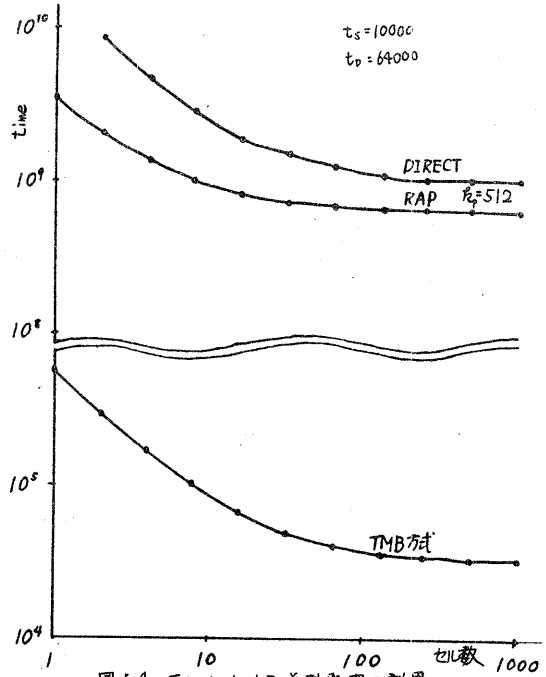

図4.4 Joinにおける並列処理の効果

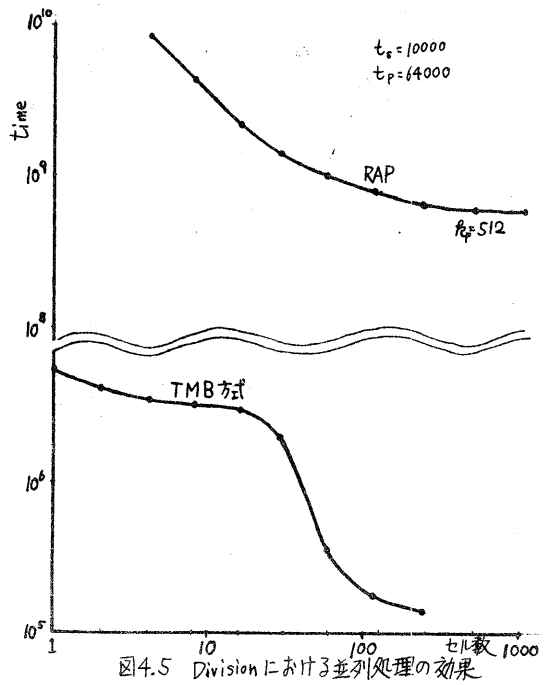

図4.5 Divisionにおける並列処理の効果

(4) 演算に必要な活性データ量

2回のSelectionとImplicit Join, Projectionから成る問い合せを例にとって各方式の活性データ量の推移について考察する.

```
SELECT UNIQUE S#
FROM   SPJ
WHERE  J#='J1'
AND    P# IN
       SELECT P#
       FROM   P
       WHERE  COLOR='RED'
```

問い合わせ例

t_p: P-リレーション タップル数
t_s: SPJ-リレーション タップル数
n: セル数 , k_p: オペランド属性 種別数
m: RAP, DIRECTにおけるタップル長 (bit)
d_p: t_p 中の有効タップルの割合
d_s: t_s 〃
$\beta_{(n)}$: k_p 中各セル内の アイテム値集合がもつ 種別数の割合の平均値 (≤ 1)

表4.1 メモリ使用量の推移

	TMB方式	RAP	DIRECT
Selection (color='RED')	$17 \times t_s$	$(m+4) \times t_s$	$m \times t_s$
Selection (J#='J1')	$17 \times t_p$	$(m+4) \times t_p$	$m \times t_p$
Implicit Join	(1) $64000n + 17 \times t_s$ (2) $64000n + 17 \times t_p$	$(m+4) \times (t_s + t_p)$	$m \times (d_s t_s + d_p t_p)$
Projection	(1) $64000n + 17 \times t_p$ (2) $64000n + 17 \times t_p + \beta_{(n)} \cdot k_p t_p$	$(m+4) \times t_p$	$m \times d_p \cdot d'_p t_p$

表4.1より TMB方式の最大メモリ使用量は

$$64000 \times n + 17 \times t_p + \beta_{(n)} \times k_p \times t_p$$

RAPでは

$$(m+4)(t_s + t_p)$$

DIRECTでは

$$m \times (d_s t_s + d_p t_p) \quad \text{である。}$$

従って RAPでは

$$m_{RAP} < \frac{64000 \times n + 17 \times t_p + \beta_{(n)} \times k_p \times t_p}{t_s + t_p} - 4$$

DIRECTでは

$$m_{DIR} < \frac{64000 \times n + 17 \times t_p + \beta_{(n)} \times k_p \times t_p}{d_s t_s + d_p t_p}$$

の場合, 各々TMB方式より, 最大メモリ使用量は小なくなる. しかしながらImplicit Join, Projection等ではRAP, DIRECT方式はTMB方式に比べて処理時間が(2)で述べたように大きくなるので, メモリ・スピード積はTMB方式の方が一般的にかなり小さくなる. またJoinが含まれる問い合せではRAP, DIRECTでは大きなメモリ領域を必要とする可能性がある. セル内の記憶領域に限界がある場合, TMB方式は他の方式に比べて本質的な違いがある. それはすべての演算がコード値集合からAssociation情報に変換する方式

を基本的に採用している。そのためにコード化リレーションに対しては繰り返してページをロードすることが原則的に必要なくなり，処理プロセッサのメモリ容量に制限がある場合（処理データがすべてメモリにロード出来ない場合）でも，多回のリプレースメントによる処理効率の劣化を未然に防いでいる。

5. おわりに

本稿では筆者らが現在検討を進めている関係リレーショナル・データベースマシン（SPIRIT）で採用されているリレーショナル演算アーキテクチャについてその特長，処理方式の詳細，具体的な実現可能なレベルのアーキテクチャ，そして他の演算方式との比較を定量的に行ない，本稿で提案している動的マークビット方式の有効性を明らかにした。

問題点としては，コード化リレーションを前提していることであるが，この点に関してはむしろデータ圧縮効果によって演算対象データ量そして転送時間を軽減することが可能であり，たとえ実システムへの変換コストを考えても十分に効率の良い方式である。

謝辞

本稿をまとめる上で貴重な御教示と御討論をいただいた本学大学院 小沢裕之君，瀬尾和男君，加藤淳君に紙面を借りて感謝致します。

参考文献

[1] E.A. Ozkarahan., et al.: "RAP - An associative processor for data base management", NCC, (1975)

[2] S.A. Schuster., et.al.: "RAP.2 - An Associative Processor for Databases and Its Applications", IEEE Trans. on computers, Vol.C-28, No.6, (June 1979)

[3] E. Babb.: "Implementing a Relational Database by Means of Specialised Hardware", ACM Trans. on Database systems, Vol.4, No.1, (Mar. 1979)

[4] D.R. McGregor., et.al.: "High performance for database systems," in Systems for Large Databases, North-Holland, Amsterdam, 1976, pp. 103-116

[5] D. J. De Witt.: "DIRECT - A Multiprocessor Organization for Supporting Relational Database Management Systems", IEEE Trans. on computers, Vol. C-28, No.6, (June 1979)

[6] 国分，他：「磁気バブルデータベース計算機EDCのアーキテクチャ」，信学技報，EC78-46，(1978)

[7] N. Kamibayashi., et.al.: "SPIRIT: A New Relational Database Computer Employing Functional-Distributed Multi-Microprocessor Configuration", The 1st International Conference on Distributed Computing Systems, (Oct. 1979)

[8] 上林，他：「リレーショナルデータベース演算のハードウェア化アルゴリズムと，それに基づくデータベースマシンアーキテクチャ」，情報処理学会 計算機アーキテクチャ委員会 35-4 (1979.9.12)