データベース・システム 35－3
（1983. 5. 19）

大規模RDB向きデータベース・マシン アーキテクチャに関する一考察

佐藤清実．北村正　中村敏夫　井上潮　武田英昭
（日本電信電話公社　横須賀電気通信研究所）

1. まえがき

近年の情報化社会の発展に伴い，データベース（DB）システムは，利用形態の多様化，大規模化への対応に向けた高機能化，高性能化が切実な要求となってきている。

また，LSI技術の進歩に伴うハードウェア・コストの低下を背景に，DB処理専用のアーキテクチャを有する計算機（データベース・マシン；DB－M）の研究も活発に進められている。

DB－Mの研究に際して考慮されるデータ・モデルの多くは，DBC[1]，GDS[2]等一部のものを除き（これらは，データ・モデルを特定しない），関係モデル（RDB）である。今後予想されるDBの高水準化動向を考えると，今後とも，RDB処理の高速化が，DB－M研究の重要テーマと言える。

このRDB処理を大きく分類すると，RESTRICTION，PROJECTIONのように単一リレーションに対する演算と，JOIN，DIVISIONのように複数リレーションに対する演算に分けることができる。

これまでに提案されているDB－Mを，この観点から整理すると以下のようになろう。

(i)単一リレーション演算の高速化
- ①二次記憶装置の専用化
  - ・RAP[3]　・CASSM[4]
  - ・EDC[5]
- ②記憶装置コントローラのインテリジェント化
  - ・CAFS[6]　・DBC
  - ・サーチ・プロセッサ[7]

(ii)複数リレーション演算の高速化
- ①ハッシュ化ビットアレイによる事前ふるい落し機能
  - ・CAFS　・LEECH[8]
- ②ソータの利用
  - ・DSDBC[9]・GRACE[10]

大規模RDBの実現に当って，これらのDB－Mで使用されている手法を適用することを考えた場合，

- DB規模の増大に伴い，システム・コストの占める磁気ディスク装置等ファイル系コストの割合いが増加するため，上記の(i)①のような二次記憶装置を専用化する手法は，システム・コストを増加させるため得策でない。
- (ii)②のようなソータを利用する場合は，ソータの容量が，関係演算の単位となるリレーション・サイズに左右されるため，設定すべきソータ容量が，ハードウェア利用効率面から重要である。

このように，従来，DB－Mの研究においては，「大規模化」（DB量，トラフィック等）に対する配慮が，必ずしも主要な議論のテーマとされなかった。

本稿では，リレーション・サイズが数100MB（10^6タプル程度），DB量が数10GB程度の大規模システムを対象に，実現性が高いと考えられるDB－Mアーキテクチャを提案する。

本稿で提案するアーキテクチャは，
①既存の汎用可動ヘッド磁気ディスクを二次記憶媒体とする。
②可能な限り，演算対象となり得ないデータは，事前に削除する機構を充

実させる。

③機能プロセッサ群を，同報機能を有するスイッチで結合することにより，ビルディング・ブロックによる柔軟なシステム構成を可能とする。

④演算機能を，言語／制御系機能から独立させ，演算系の並列／パイプライン処理による高性能化を図る。

等の特徴を有する。

2. 大規模RDB実現上の課題

ここでは，従来のDB－M研究の動向を踏まえ，大規模RDBをコスト／パフォーマンス的に妥当なものとしてシステム化するために考慮すべき主要な項目を，DB量，リレーション・サイズの両面から整理する。

この他，大規模化に当っては，トラフィック面での考慮が必要となるが，これは双方の項目に含めて述べることとする。

2.1 DB量

DB量の増加に伴いシステムとして重要となるのは，DB実体を格納する二次記憶装置や，コントローラ等のファイル類の量（即ちコスト）であろう。

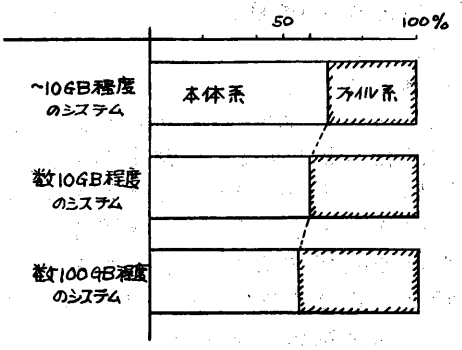


図2.1　システム・コスト構成例

DBシステムのコストを，CPU，メモリ等の本体系コストとファイル系コストに分類し，幾つかの大規模システムについて，ラフに試算すると図2.1のようになる。ファイル系コストのシステム・コストに占める割合いは，DB量の変化とともに，全体の30%～50%程度の範囲となる。

近年，個体ファイルとして，磁気バブル・メモリの開発が進められているが，数10GB以上にも及ぶDB用のバルク・メモリとして使用されることはコスト面から当面なく，磁気ディスク装置（DKU）が主流となると予想される。

DKUは，基本的にDB量に応じた台数が必要となるが，さらに，トラフィック等による性能面からの台数への影響（例えば，アクセス競合を避けるための収容率の低下）が加わる。

従って，DKU等，二次記憶装置そのものへ，RDBのための専用機構を付加することは，ことさらに，ファイル系コストを高くする。

2.2 リレーション・サイズ

リレーション・サイズは，関係演算のためのCPU，メモリ等システムとして高価な処理系リソース量に関係するた

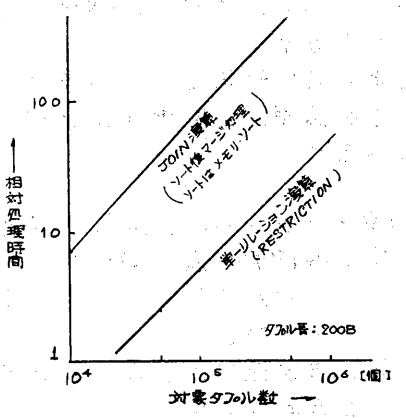


図2.2　演算の種類と処理時間の傾向
（非インデックスDB時）

め，トラフィック量と合わせて考慮すべき重要な項目である。

特に，JOIN 演算は単一リレーション演算に比べ，1桁以上もの処理時間を要し（図2.2），それだけシステムでの処理系リソースの使用率が高い。

そのため，JOIN 演算の対象タプルを可能な限り，事前に削除しておくことが，大規模化時に重要となる。

通常，JOIN演算は，再構成処理時等を除いて，RESTRICTION / PROJECTION のような単純検索と組合わされて使用されるので，これらの単純検索を先行して行わせたり，従来から種々提案されているふるい落し機能[4][9]を用いることは，有効な手法である。

リレーションサイズに関して，もう1つ重要なことは，一般に，リレーションサイズをシステムとして特定できないため，リレーションサイズの変動に対して柔軟な処理系リソースの確保が必要となる。これは，特に，処理系リソースを多く使用するJOIN 演算時に重要である。

中でも，JOIN 演算の高速化手法として提案されているハードウェア・ソータのハードウェア量は，数100MB（10^6 タプル程度）程度のリレーションを対象とした場合，10^3～10^4 Kゲート程度にも及ぶと予想される。そのため，大容量ソータを直接利用する方式を，様々なリレーションサイズが存在する現実のシステムに適用することは，リソースの有効利用面から問題となる。

また，リソースの有効利用を図るため，リレーションを分割して，比較的小容量の複数ソータで並列処理させる方式では，ソータへのデータ分配時に，偏りが生ずると，分割した意味が無くなるため，大規模化に当っては，この分配手法が重要となる。

3. アーキテクチャの概要

3.1 実現目標

本稿で提案するDB－Mの検討に当っては，次の目標を設定した。

(i) リレーションサイズとして数100MB（10^6程度のタプル数），システムとして数10 GB程度のDBをサポートする。

(ii) 単一リレーションに対する演算，又は，複数リレーションに対する演算のレスポンス・タイムが10秒程度の性能を持つ。

(iii) トラヒック的には，数100～数1000件/時 程度の能力を持つ。

3.2 構成概念

2章で述べた課題の解決に向けて，本稿で提案するDB－Mの構成概念を図3.1 に示す。

主たる特徴は，以下の点である。

(1) DB用のバルク・メモリとして汎用のDKUを使用する。

レスポンスの向上を図るため，リレーションを複数のDKUに分割格納し，並列処理する。

(2) 単一リレーション演算機構（SER）は，複数のDKUで共用することを考慮して，DKUコントローラ相当部分に付加する。

このことにより，DB量に直接関係するDKUの専用化によるコスト増加を避けることが可能となり，かつ，トラヒック見合いの演算機構の設置が可能となる。

この演算機構は，(1)の条件下で並列動作する。

(3)JOIN 演算は，3ステージに分割して行う。即ち

①結合可能性の無いタプルをふるい落すとともに，複数のソータに，リレーションを分割する処理（FL）

②分割されたリレーションをソートする処理（SOR）

③結合処理（一部のEXC）

の3ステージで行う。

タプルのふるい落しでは，両リレーションを並列処理させ，かつ，両リレーション・サイズ差によって生ずるアイドル時間を削減するために，クロス・ハッシュ方式(後述)を採用する。

ソータへのデータ分配法では，従来行われているハッシング法[10]とは異り，統計的処理法を採用する。

(4)各プロセッサは，スイッチで結合する。

従来，結合方式として増設の容易性，コスト面から光ループのような，リング・バスによる結合方式が利用されている。

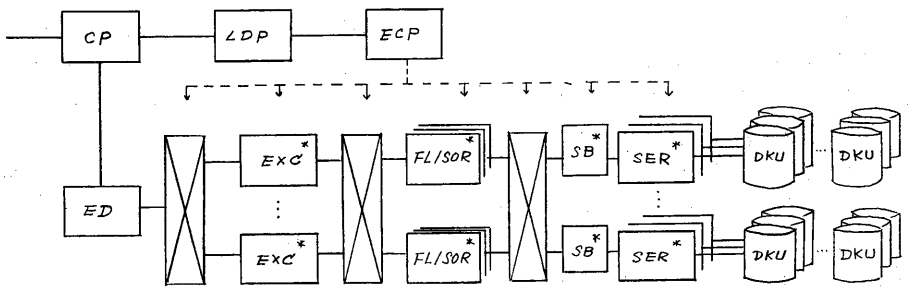
現状の光ループの転送レートは数10 MB/S 程度である。そのため，例え単一リレーション演算機構等により転送データ量が削減されるとしても，リレーション・サイズ数100 MB，数1000件/時のトラヒックを処理するシステムでは，転送路ネックが生ずる。

従って，性能的にスイッチによる結合方式が必要となるが，コストの観点からの考慮が必要である。

(5)言語処理／制御機構と演算機構を分離する。

10^6タプル程度の大量タプルを処理する場合には，ディレクトリによるマッピング処理，命令の最適化等を行う言語処理時間の全体に占める割合いは極めて小さく，数%以下と想定される。

従って，マルチ・トランザクションを処理するシステムでは，言語処理機構は，マルチ・トランザクションで共用し，演算機構は，1トランザクション対応に構成可能とすることが，システム構成上効果的である。



* いずれも，必要に応じて増設

CP : Communication Processor
LDP : Language & Directory Processor
ECP : Execution Controler
ED : Editor
EXC : Executer
FL/SOR : Filter & Sorter
SER : Sercher
SB : Selected data Buffer

図3.1 構成概念

特に，演算装置をスイッチ結合とし，各プロセッサ・リソースを演算に先立って，あらかじめスイッチ制御により割当てることは，データが，そのパスを通過するだけで良く，タプル単位の切替制御が不要となるため，通信オーバヘッドの削減とともに，パイプライ処理の容易化が期待できる。

4．構成要素の概要

ここでは，3章で述べた構成概念の要素のうち，ふるい落しと結合を実現する要素を中心に概要を述べる。

4.1 検索装置（SB & SER）

検索装置は，RESTRICTION / PROJECTIONの単一リレーションに閉じる演算を行うSER（Sercher）と，その演算結果を蓄積するSB（Selected data Buffer）から構成される。

リレーションは，そのデータ量に応じた適当な台数のDKU上に

①各DKUのデータ量が極力平均化し，

②DKU内では，極力同一シリンダ内に，

③同一シリンダ内では，極力連続トラックに

という管理アルゴリズムで格納される。

SERは，DKの回転速度と同期して各タプルについて演算を施し，その結果得られたタプルをSBに書き込む。

SBの必要量は，あらかじめ予想することができないため，SERの演算条件によって容量不足が生ずることがあるが，その場合は，SBをセグメント化しておき，ECP経由で新しいSBの割付けを受け，データ転送先を変更することとしている。

4.2 事前ふるい落し／分配ソート装置（FL／SOR）

FL／SOR（Filter & Sorter）では，JOIN演算対象タプルに対する事前ふるい落し装能，ソート装能及び，複数台のソータを動作させるためのデータ分配装能とその分配に必要なデータを収集する統計解析装能を実現する。（図4.1参照）

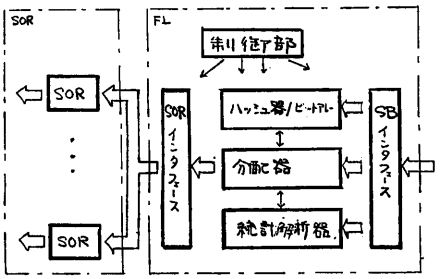


図4.1 FL/SORの装能ブロック概念

(1)事前ふるい落し装能

JOIN演算対象リレーションに前処理を施し，演算量を削減する方式としては，従来，ハッシュ化ビットアレイを用いた以下の2方式が提案されている。

(i)片ハッシュ方式

ハッシュ器，ビットアレイを各々1台用意し，一方のリレーションのみをふるい落す

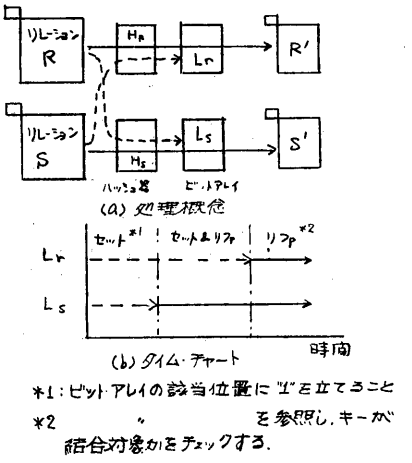
(ii)シリアル・ハッシュ方式

ハッシュ器及び2台のビットアレイを用い，2つのリレーションのふるい落しを逐次に行う。

片ハッシュ方式は，制御が単純であるが，片方のリレーションのみのふるい落しであるため，後段のソート，結合（マージ）の負荷削減の効果が少ないこと，どちらのリレーションをふるい落しの対象とすべきかの選択が困難であるという問題がある。

シリアル・ハッシュ方式は，処理が逐次のため，ふるい落し時間が長いという欠点がある。

本稿での方式は，ＳＢとＦＬを同報機能を持つスイッチで結合し，２つのリレーションのふるい落しを並列に実行するもので，クロス・ハッシュ方式[11]と呼んでいる。方式概念を図4.2に示す。



*1：ビットアレイの該当位置に"1"を立てること

*2 〃 を参照し，キーが結合対象かをチェックする。

図4.2 クロスハッシュ方式概念

アルゴリズムの概要は，以下のとおりである。（図4.2参照）

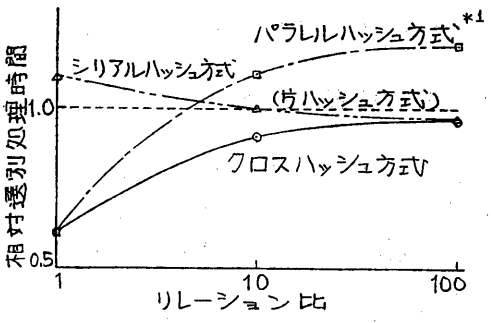
① リレーションＲ，Ｓ（|R|<|S|とする）のタプルをリレーション対応に並列に読み出し，結合属性をハッシングして，Ｒをビットアレイ Lsに，Ｓをビットアレイ Lrにセットする。

② Ｒのセットが終了した時点で，Ｓのセット中のタプルを記憶するとともにLsのモードを切替え，Ｓの結合属性値を同報機能により２つのハッシュ器に送り，Lrはセットを続け，Lsはリファを行う。

③ Ｓのセットが終了した時点でLrのモードを切替え，Ｒの全タプルを順次リファし，LsはＳの先頭からモードを切替えた時点のタプルまでをリファする。

リレーションの読出し開始から，リファ終了までの時間（選別時間）をシミュレーションにより評価した結果を図4.3に示す。

この結果から，クロス・ハッシュ方式は，従来の方式に比べ，最大40％程度の性能向上を図ることができることがわかる。



*1：パラレル・ハッシュ方式とは，セット，リファの切替えを，2つのリレーションで同時に行う。

◎ 処理時間は，プロセッサの処理時間と入出力時間の合計値

◎ リレーションは，セット時はDKU，リファ時はSBから読み込む

図4.3 クロス・ハッシュ方式の効果

(2)分配ソート技能

複数のソータを用いて並列演算を行うためには，結合属性値により，２つのリレーションのタプルを，それぞれソータに分配する必要がある。

このために，従来，高速性を生かしたハッシング関数を用いた手法が提案されている。この方法は処理が簡単なため，振分け時間が短いという利点がある反面，ハッシング関数とデータの特性の組合わせに依っては，分配結果に偏りが生ずる。

本稿での方式は，ソートの前処理として行う事前ふるい落し処理が，セットとリファの２段階で行われることを利用するもので，セット時に，結合属性値の統計データを収集しておくことにより，ＳＥＲでの演算後

のタプルの分布状況が推定できる。そのため，リファ・モードへの移行に先立って，収集した統計データを処理（単純な範囲指定程度の情報を得る処理を施す）した後，分配を行うことにより，後続のソートや結合処理の遅れの影響を極めて小さくして，かつ，偏りの問題を避けることが可能となる。

4.3 結合演算装構（ＥＸＣ）

図3.1 の構成概念中のＥＸＣは，関係演算の他，論理演算等を専用に行う演算プロセッサの総称であり，ここで述べる結合処理も，一部のＥＸＣで行われる。

タプルは，あらかじめソートされているので，単に，マージするだけで良いように思われるが，同一キー値を持つタプルが複数存在する場合には，結合処理のバック・トラックが必要となり，タプル数に比例した処理時間で終らなくなる。

特に，JOIN演算の最初のステージでふるい落し処理を行っているため，同一キー値を持つタプルが集中する可能性が高くなる。そのため，バック・トラックを避けるアルゴリズムが必要となる。

そこで，本稿では，タプルを格納するためのスタックと，スタックの深さのレベル毎にタプル連結器を備えたスタック・マージ方式[12]によりマージを行う。（図4.4，図4.5）

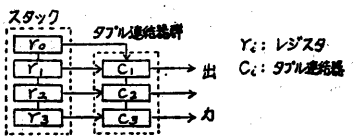


図4.4 スタックとタプル連結器の概念図

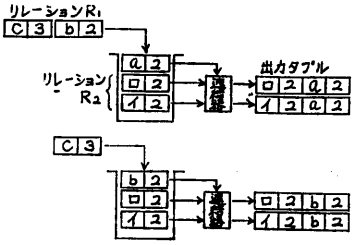


図4.5 スタック・マージ方式の概念

この方式のアルゴリズムは，以下のとおりである。

① ２つのソートされたリレーションのタプルを１つづつ取出して，キー値の大きさを比較し，キー値が等しくなければ，小さい方のタプルを破棄して次のタプルを取出して比較する。

② キー値が等しい場合は，一方のリレーションからタプルを連続して取り出し，異るキー値が現われる迄スタックに蓄積する。

③ スタックに蓄積された複数のタプルと，他方のリレーションの１タプルを，タプル連結器を用いて一斉に結合して出力する。

用意するスタック段数や，リレーションの大きさとスタック対象リレーションの関係をパラメータとして，スタック・マージ方式の性能比較を行った結果を図4.6，図4.7 に示す。

この結果から，スタック・マージ方式は，ある程度のスタック段数を準備することにより，重複したキーを持つタプルが多くなっても，比較的安定した処理量となること，リレーションの大きさが異る場合は，タプル数の小さいリレーションのタプルをスタックに蓄積した方が効率が良いことが判る。

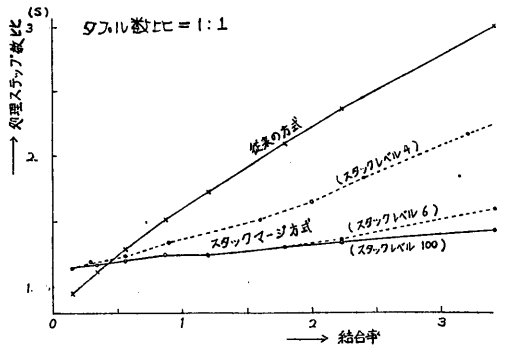
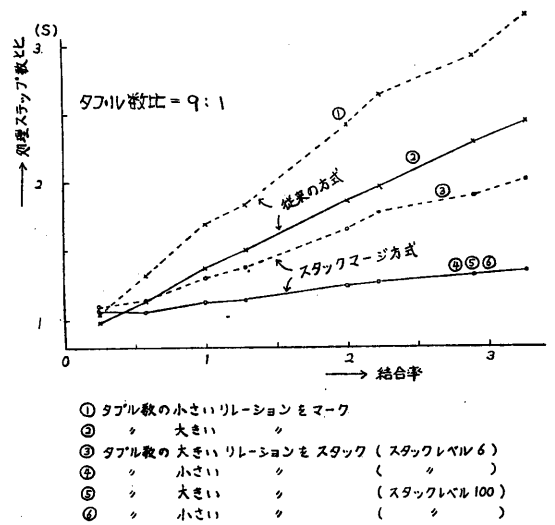


図4.6 スタック・マージ方式の性能(リレーションのタプル数が同一の場合)



① タプル数の小さいリレーションをマーク
② 〃 大きい 〃
③ タプル数の大きいリレーションをスタック (スタックレベル6)
④ 〃 小さい 〃 (〃)
⑤ 〃 大きい 〃 (スタックレベル100)
⑥ 〃 小さい 〃 (〃)

※性能の尺度として、処理ステップ比、タプルのレジスタへのロード、キー値の比較、タプルの連結等をそれぞれ1ステップとしてカウント。

※タプルのキー値の分布は一様分布とし、同一キー値の重複度合いは、データ結合率 = 結合後のタプル数/結合前のタプル数で評価。

図4.7 リレーション入れ替えの効果

5. あとがき

本稿では、スイッチ結合を用い、大規模化に向けて柔軟なシステム構成を可能とするアーキテクチャについて、その基本的概念を述べた。

さらに、スイッチにより、データの流れ制御が容易であるSTAGEによるJOIN演算方式について論じ、ふるい落し処理、結合処理のシミュレーションによる評価を中心に、各部の概要を述べた。

本稿で述べたアーキテクチャは、基本構想の段階であり、今後、制御系/更新系処理の詳細化を図り、システムトータルな評価を実施する予定である。

最後に、日頃御指導を頂いている関係各位に感謝致します。

参考文献

(1) Banerjee J. et al "DBC - A Database Computer for Very Large Databases", IEEE Trans. on Computer, vol.C-28, no6, pp414-429, June, 1979
(2) 箱崎他, "データベースマシン実験システム", 信学会計算機研究会資料, EC-79-68, 1980
(3) Schuster S.A. et al. "RAP.2-An Associative Processor for Databases and Its Applications", IEEE Trans. on Computers. vol C-28, no6, pp. 446-458, June 1979
(4) Su S.Y.W, Lipovski G.J. "CASSM: A Cellular system for very large databases" Proc. VLDB, pp456-472, 1975
(5) 植村他, "磁気バブルデータベースマシン", 「大型プロジェクトパターン情報処理システム研究成果発表会論文集」, pp 263-279, 1980.
(6) Babb E. "Implementing a relational database by means of specialized hardware", ACM TODS 4.1 pp 1-29, 1979
(7) 佐藤他, "データベース処理のための専用サーチプロセッサ", 情処論文誌 vol23, no4, July, 1982
(8) McGregor D.R. et al. "High performance hardware for database systems", Syst. for Large Data Bases, North Holland Publishing Co., 1976
(9) 田中 "データストリーム方式のデータベースコンピュータ", 情処記号処理研究会資料, 12-14, pp97-103, 1980
(10) 喜連川他 "HashとSortによる関係代数マシン", 信学技報, EC 81-35, 1981
(11) 武田他 "並列ふるい落しアルゴリズムに関する一考察", 情処第26回全国大会, 4F-13
(12) 井上他 "大規模データベース向き結合演算方式" 昭和58年度信学会総合全国大会 S1-3