CA 23 − 1
(1976. 12. 15)

データベース管理システム向き計算機アーキテクチャについて

ON THE DATA-BASE-MANAGEMENT-SYSTEM-ORIENTED COMPUTER ARCHITECTURE

京都大学工学部　渡辺勝正　萩原宏

KYOTO UNIVERSITY　Katsumasa WATANABE　Hiroshi HAGIWARA

現在広く利用されている計算機（Conventional Computer）は，必ずしも，データベース管理システム（DBMS）あるいは情報検索システムに適しているとはいえない。本論文では，たとえば，記憶装置内におけるデータ構造の実現，可変長データの操作，並列アクセス法の問題をとりあげて，ソフトウェア，ファームウェア，ハードウェアを含めた総合的な面から，大規模DBMSに適した計算機アーキテクチャを考え，DBMSの効率向上をはかるとともに，DBMSの問題の本質を追究する。

1. データベース管理システムにおける問題

データベース管理システム（DBMS）あるいは情報検索システム（以後とくに区別をせずにDBMSと総称する）の役割は，データの集りをある有機的な関係のもとで定義し，そのデータの集りからある条件に合ったデータを抽出することにあり，そのシステムは図1のように示される。

このようなシステムを実現するにあたっていくつかの問題がある。

1）DBMS向き命令体系

DDLおよびDMLにおける基本操作は，多くの場合，（数値計算に向いている）計算機の機械語命令と大きなへだたりがある。しかも，DDLおよびDMLは，より水準の高い（あるいは，非手続き的な）言語にむかう傾向にある。これらの言語翻訳の問題と関連して，DBMS向き計算機の命令体

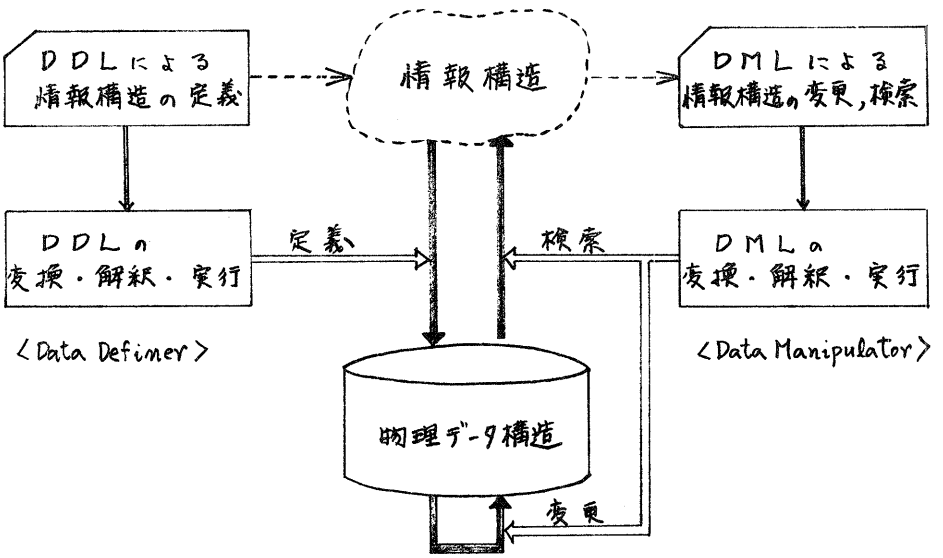


図1．データベース管理システムの構成

系はどのようなものが望ましいかを考える必要がある。

2）記憶の構造

ユーザが頭の中に描いているデータの情報構造と，記憶装置の中で実現されるデータの物理構造とは，必ずしも一致しない。ユーザはその応用に応じて様々な情報構造を定義するが，計算機の中では，1次記憶装置，2次記憶装置の構造に応じて，それを実現せざるを得ない。このとき，アクセスの効率を考慮に入れて，ブロックに分割したり，ある種の問い合わせに対してすばやく応答できるように，データ構造を再編したりすることがある。情報構造と物理構造との間のデータ構造変換の問題と関連して，どのような記憶の仕方がよいか，そのためには，どのような記憶装置の機構が必要かを求めねばならない。

3）データ転送量

取り扱われるデータベースの大きさは，1次記憶で記憶できるデータ量よりはるかに多いため，データベースを適当に分割して2次記憶に記憶しなければならない。様々なタイプの問い合わせを受け付けるDBMSにおいては，ある種の問い合わせに対して，ある特定のページだけを参照すれば良いようにページを構成することは困難であるので，（しかも，1ページ内にその問い合わせに対して必要な情報がたいていの場合わずかであるので），多くのページを1次記憶に転送しなければならない。このようなとき，大容量の1次記憶と高速転送能力をもつデータチャネルをそなえた大型計算機を望むよりも，本当に必要なページのみを指摘したり，2次記憶に検索能力をもたせたりして，転送量をへらすように工夫することの方が好ましいと考えられる。

4）データベースの変更

データベースがいくら大きくても，それが固定されたものであるならば，あらかじめ，各々の応用あるいは問い合わせの型に応じたファイルを別個に準備しておけば，応答時間を短くすることができる。この場合には，1つのデータがいくつかのファイルの中に異ったキーを持ったレコードとして記録されることがある。このようなときデータベースが変更されると，変更データに関連したファイルをみつけて，それらのいくつかのファイルを改めなければならない。一部のファイルを改めた時点で問い合わせがくると，データベース内に不一致が生じた好ましくない状態となり，問い合わせを一時待ち合わせる必要が起る。データベースの更新・拡張に耐え，しかも，柔軟なアクセス法をもった物理データ構造の実現（inverted file, chained organization とは異った自由なデータ構造の実現）が必要となる。

そのほか，データの入力方法，データの保護，データベースに関する記録等々の多くの問題がある。これらも含めて，上記の問題は，単にソフトウェア，アルゴリズムの検討といった面のみの問題ではない。以下では，これらの問題を頭に入れて，ソフトウェア，ファームウェア，ハードウェアの全体的な面から，DBMS向き計算機のアーキテクチャについて考える。

DBMSの役割りの増大する時にあたって，与えられた計算機システムの上でDBMSを実現する努力とともに，DBMSにおける基本操作および本質的な概念をとらえて，新しい計算機アーキテクチャを求めることは重要なことである。

2. ソフトウェア，ファームウェア，ハードウェアの役割り

新しい計算機システムを設計・製作するにあたって，ある機能を実現するのにソフトウェア，ファームウェア，

ハードウェアのいづれによるかを決定することは重要な問題である。それは機能の性格によって一意的に定まる場合もあるが，実現する技術的な背景とシステムの汎用性・柔軟性に対する考え方とに大きく影響される。一般に，そのような影響のもとに，1つの計算機システムにおいて，ある機能をいづれで実現するかという傾向は，歴史的につぎのように変ってきているとみられる。

1) 計算機がうまれた当初は，まさにハードウェアそのものであった。
2) ハードウェアの高速・高性能化に伴って，汎用という目標のもとに，システムプログラム，ソフトウェアパッケージとしてのソフトウェアの増大がみられる。
3) マイクロプログラミング方式の採用により，応用に応じて命令体系を変更し，可変制御構造をもった計算機として，ハードウェアの機能をファームウェアで実現した。
4) 広汎な利用と共にソフトウェアの比率が増々大きくなったが，IC による書き換え可能な制御記憶装置の実現が容易になって，処理速度の向上，ソフトウェアの簡単化（高水準言語計算機，OS のファームウェア化）をめざして，ファームウェア化する傾向がみられる。
5) LSI 技術の発展にともなって，回路の標準化，並列性の利用といった点から，再びハードウェアを中心に考えて，ソフトウェアの比率の増大をふせごうとする努力がみられる。

以上の傾向をまとめて考えると，ソフトウェアは我々が問題を考えてそれを表現しやすいように支援し，ハードウェアは，回路および設計技術からの制限を受けながら，汎用性・融通性ということにも留意して，ソフトウェアを実現するために最適な基本要素を提供し，ファームウェアは，両者の間にあって，基本要素の組み合わせによって“表現”を実現し，システムの柔軟性と設計の簡易性を保つ役割りを果しているものといえる。

したがって，ソフトウェア・ファームウェア・ハードウェア方式では，一般に，基本的なハードウェアに対して，プログラム表現に近い機械語命令をファームウェアによって解釈することで，ソフトウェアに依存した仮想機械を柔軟に構成することができる。このように，ソフトウェアによる（外部における）アルゴリズムの表現とファームウェアに対する（内部における）表現とが近接していることは，それらの間の変換が容易であり，命令数が少くなって命令取り出しの回数が減少するという利点だけでなく，我々が計算機構を頭の中に描きながらアルゴリズムを考えることができるという利点をもつ。

これに対して，ソフトウェア・ハードウェアのみの方式では，それらの間のギャップを，いづれかの歩み寄りによってうめるか，トランスレータによって，ソフトウェアによる表現をハードウェア向きに変換しなければならない。とくに，DBMS 用のソフトウェアと，汎用計算機のハードウェアについては，両者の間のギャップが（たとえば，記号列操作，データ集合の演算，可変長データの取り扱い等の面で）大きいと感じられる。

3. DBMS におけるソフトウェア・ファームウェア・ハードウェア

DBMS における問題に対する方策を，それぞれソフトウェア，ファームウェア，ハードウェアを主体とした面から，いくつかの事例[1,2,3]を参考にしてまとめる。これらの事例の中には，後置プロセサ（back end processor）方式として，データベース管理用あるいは情報検索用の専用プロセサを採用するものがあるが，それは汎用計算機

システムの機能を分離したものであって，専用プロセサにおける問題は，結局は，つぎのいづれかに関連づけられる。

3．1　ファイルディレクトリの強化

与えられた計算機システムの中で，可動ヘッドディスクのヘッドの動きを少くし，しかも，2次記憶から1次記憶に転送されるデータ量を最少におさえようとする。ファイルの中からある条件を満たすレコードを探し出すには，

i）条件を満たすレコードを含むと思われるデータブロックを抽出する，

ii）そのブロックの中で条件を満たすレコードを順に調べる。

という考えに基づく[4]。

そのため，つぎのように，与えられた"条件"から"必要なブロック"が決定できるように，ファイルディレクトリを強化するとともに，同じ条件を満たすレコードをできるだけ同じブロックに含めるようにレコードを記憶する。

レコードは簡単に（属性，値）対の集合であるとする。レコードの集りをファイルとよび，ファイルがいくつか集ってデータベースを構成するものとする。キーはレコードを特徴づける"属性－値"の対で，問い合わせは，キーの関係で表わされる。

一方，可動ヘッドディスクの1シリンダを検索単位（retrieval unit）とよび，1つの検索単位の中にいくつかのブロックがあるものとする。γ番目の検索単位の中の名前bのブロックを，

$$(\gamma, b)$$

で表わし，物理セルという。

ファイルFとキーK_iに対して，物理セルの集合

$$D(F, K_i) = \{ (\gamma, b) \mid \text{ファイルFのレコードで，キー}K_i\text{をもつものが，少くとも1つ，物理セル}(\gamma, b)\text{に含まれる} \}$$

を定義し，Fに対するキーK_iのディレクトリ・エントリーとよぶ。すなわち，キーK_iが与えられると，そのディレクトリ・エントリーを見て，キーK_iをもつレコードが少くとも1つ含まれている物理セル(γ, b)をすべて知ることができる。ファイルFのディレクトリは，F上のすべてのレコードに対するキーのすべてのエントリーから成るものである。

同じキーをもつレコードは同一のセル内にあるいは同じ検索単位の中に記憶されるようにファイルを構成する。

問い合わせQに対して，Qを満たすレコードはつぎの手順で計算される。

i）Qの中のすべてのキーK_iに対する物理セルの集合の組を抽出する。

$$\{ D(F, K_1), \cdots, D(F, K_m) \}$$

ii）Qにおけるキー間の関係Rから集合演算によってQを満たす物理セルの集合を決定する。

$$D(F, Q) = R[D(F, K_1), \cdots, D(F, K_m)]$$

iii）得られた物理セルを1次記憶にとりだして，各レコードを順に調べ，Qを満たすものを選びだす。

ファイルFにキー$(K_1, \cdots, K_j)$をもつレコードを追加する場合には，

i）$D(F, K_1), \cdots, D(F, K_j)$を求めて，

ii）これらの集合の中で最も多く共通に含まれている物理セル(γ, b)を決定し，

$$(\gamma, b) = \max_{i=1}^{j} [(\gamma, b) \in D(F, K_i)]$$

iii）その物理セルにこの新たなレコードを記憶する。

この方式により，はじめにかかげた目標は達成されるが，ファイルディレクトリに大きなスペースが必要となる。ただし，当然のことながら，ディレクトリのスペースは，ファイルのスペースに比べて少くてよい。その大きさの割合いは，1レコードの長さ，ブロックの大きさ，ファイル中の異ったキーの数，レコードの関係の複雑さによる。

3．2　中央処理装置の強化

ファームウェアあるいはハードウェアによって，記号処理向きの命令や連想処理機能を実現して，データ処理能力を向上させようとするものである。

（1）よく使われる手続きを命令として実現する

記号列の比較，転送，挿入，削除，変更，さらには，サーチ，ソート等記号処理に頻繁に使われる手続きを命令としてそなえる。それによって，命令取り出し回数が減少するとともに，機械語命令とユーザの用いる操作との関係が緊密になり，言語の翻訳が簡単になる。

（2）可変長データの取り扱い

名前（ファイル名，フィールド名），定数（10進数，記号列）およびフィールドの値は，一般に，その長さが様々であるが，これらをそのままバイト単位の可変長データとして取り扱い，データの記憶形式をユーザのみる形式に（少しでも）近づける。それとともに，命令形式も可変長のオペランドを持った可変長の形式とする。したがって，たとえば，DDLあるいはDMLで書かれたプログラム中の名前，定数は命令の中ではそのまま直接オペランドとして使われる。

命令コード｛長さ・名前または定数｝*

オペランドの個数は，個々の命令に依存する。

（3）記号処理向きアキュムレータ

記号処理（あるいは非数値処理）のための編集領域（アキュムレータ）を連想記憶または（8ビット並列の）高速シフトレジスタで実現して，前の2項をハードウェアの面からサポートする。

（4）ディレクトリの階層構成

データアクセスのための辞書を階層的に構成して，データ構造との対応を明確にしたり，データの記憶位置の決定を速くしたりする。これには2つの応用が考えられる。

その1つは，名前（ファイル名，フィールド名等）をデータ構造の階層構造にあわせて登録し，名前の階層関係を明らかにする。これは，第2項と関連して，可変長の名前をそのまま扱う場合に重要となる。

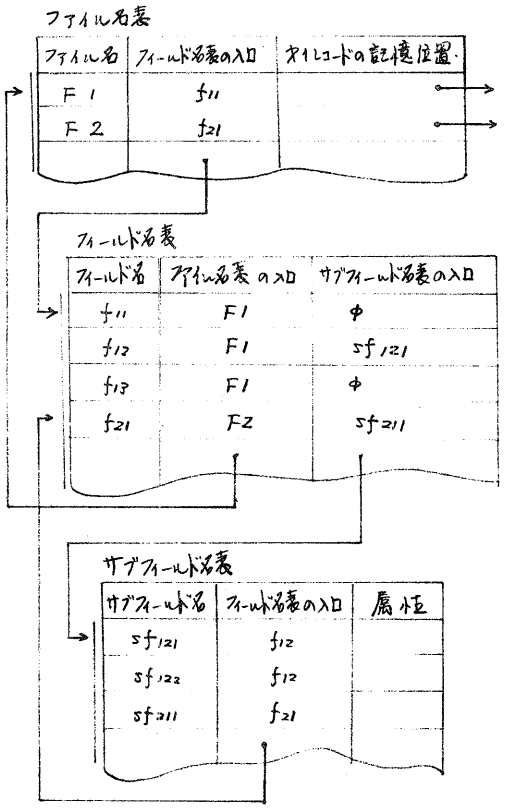


図2．名前の階層構成

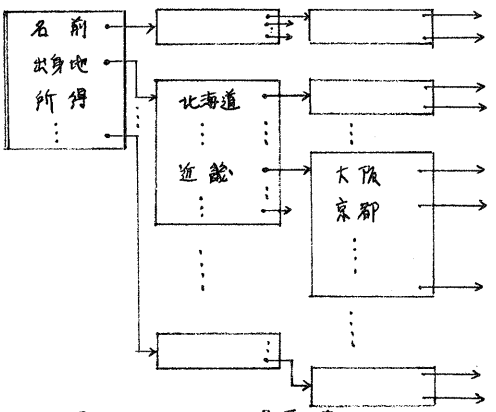


図3．キーの階層構成

もう1つは，ディスクの物理セルをつきとめるに当って，キーの階層構造に従ってキーディレクトリを構成し，所要の記憶位置を決定するものである。

オろおよびオ4項の実験から，ある程度の容量をもつ本格的な連想記憶装置の実現が待たれる。

その他，すでに実現されているものもあるが，DBMSにおいてソフトウェアだけではあまりにもオーバヘッドが大きく，ぜひファームウェア，ハードウェアによる支援が望まれるものにつぎのものが考えられる，

記憶の割付け

ガルバージ・コレクション

データの保護機構．

3．3　ファイル装置の強化

ファイル装置（ディスクあるいはドラム記憶装置）を，単にデータの記憶抽出を行うための補助記憶装置にとどめておかず，データの検索・照合機能をもたせて，2次記憶から1次記憶に送られるデータの量をへらすことを考える。検索・照合にあたっては，1語ずつを直列に行うのでなく，ディスクやドラムの構造ならびにその読み書きの速度を考慮して，トラック毎に並列に行わせる（Logic per Track Device）。

ディスクやドラムのヘッドの各々にマイクロプロセサを配置して，与えられたキーに対して，同時にサーチすると，ドラムや固定ヘッドディスクの場合には，1回転の時間で全領域をサーチすることができる。サーチの結果，必要ならば，所要のレコードを含むと判明した物理セルのみを1次記憶に送ればよい。1次記憶では，そのデータブロックに必要な処理をほどこして，場合によっては元のファイルに書きもどす。

このように，各ヘッドに特殊なマイクロプロセサをもつ装置を実現するにあたっては，各プロセサをどのように設計するか，どのような機能をもたせ，命令セットをどのように定めるかについての問題がある。

また，単純には，ディスクやドラムが1回転する間，すべてのプロセサは同じ命令に従うと考えられるが，

- 同一の命令を実行するプロセサの一部を選択することができるか，
- 各々独立に別個の命令に従わせうるか，
- 1つの命令の実行結果を（次の命令にそなえて）他の任意のプロセサに送ることができるか，

といったセルラー型プロセサ構成上の問題もある。

これらの問題が（ある程度）解決すれば，もっと速度の高い回転型メモリ（磁気バブルやCCD）あるいは半導体集積回路によるシフトレジスタに拡張して適用することができる。そのような場合には，記憶内容にデータタグをつけて，純粋のデータも，命令（プログラム）も，スタックやテーブルのような制御用データもすべて同じメモリ内に入れて，1命令実行中に次の命令を取り出す制御方式が可能となる。よって，前節オ2項で述べたような可変長の命令形式であっても同様に扱えるとともに，記号レベルでメモリやファイルが管理できる。

例．このように，1回転で全トラックの内容を検索できるように強化されたファイル装置をもつ場合の演算時間を，各プロセサの様々な特性条件のもとで考える。

演算：2つのデータ集合XとYの共通集合Z＝X∩Yを求める。

前提：1レコードは1つの値から成り，Xはm_x個のレコードを持ち，k_xトラックにまたがって記憶されている。Yはm_y個のレコードを持ち，k_yトラックにまたがって記憶されている。結果のZは，m_z個のレコードの集

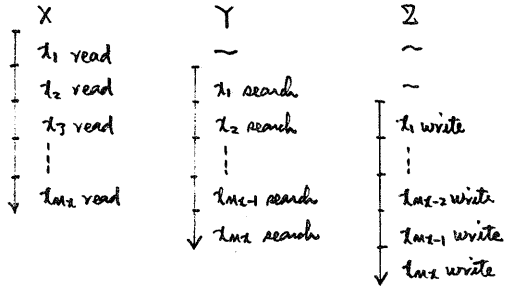
合となり，k_Zトラックに記憶できるものとする。ここに，

$0 \leqq m_Z \leqq \min(m_X, m_Y)$。

a．各ヘッドにつけられたプロセサが互いに独立に異った命令を実行し，かつ，他の任意のプロセサと通信できる場合：

- Xのレコードx_iを読み，
- x_iをYのすべてのレコードと比較する，
- $x_i = y_j$となるy_jが存在すれば，x_iをZに加える，

なる操作を並列にパイプライン的に実行すれば（m_X+2）回転で完了する。



$t_A = (m_X + 2) \times ROT$

ROTは1回転の時間．

b．他の任意のプロセサとの通信はできるが，すべて（あるいは一部）のプロセサが同じ命令しか実行できない場合：

- x_iの取り出し，
- Yの要素との比較，
- Zへの書き込み，

を順に行なわねばならないので，m_Zの値によって，$2m_X$回転から$3m_X$回転かかる。

$2m_X \times ROT \leqq t_B = (2m_X + m_Z) \times ROT \leqq 3m_X \times ROT.$

C．他のプロセサとも通信できない場合：

- Xを1次記憶に転送する ～ k_X回転
- x_iを順にファイルに送って，比較し，結果によってz_iに加える操作をパイプライン的に行う ～（m_X+2）回転
- Zを2次記憶に転送する ～ k_Z回転

$t_C \leqq (k_X + k_Z + m_X + 2) \times ROT.$

d．比較を1次記憶で行う場合：

- X，Yを1次記憶に転送する ～ $k_X + k_Y$回転
- x_iに対してYのレコードを1つずつ比較する ～ $(m_X \times FETCH) + (m_X \times m_Y \times COMPARE)$
- Zを2次記憶に転送する ～ k_Z回転

$t_D = (k_X + k_Y + k_Z) \times ROT + (m_X \times FETCH) + (m_X \times m_Y \times COMPARE)$.

FETCHは1次記憶から1レコードを取り出す時間（せいぜい1 μsec）。

COMPAREは1レコード同志の比較時間（せいぜい10 μsec）．

レコード長，トラック容量にもよるが，多くの場合，$t_A < t_C < t_B$．

ROTが10 msecのオーダであるとすれば，レコード数が500 ～ 1000位までのときは，$t_D < t_A$，それ以上になると，$t_A < t_D$になる。

1次記憶での比較照合に連想記憶を用いるとCOMPAREの回数はm_Xとなる。よって，k_X, k_Y, k_Zがm_Xに比べて充分小さければ，$t_D < t_A$となる。

回転型記憶装置上での記号列操作，あるいは，そのための命令体系は，読み出し・書き込みの機構によって大きく左右される。固定長データのみを取り扱ったり，読み出し（データ探索）のみを問題にする場合には，あまり困難はないが，とくに，可変長データを挿入・変更などの書き込みも含めて取り扱うことは，現在のディスク，ドラムの読み出し・書き込み機構だけでは無理がある。そのような操作に対しては，ランダムアクセスメモリか，シフトレジスタ型バッファを補助に用いて書き込み操作をその上で行うか，あるいは，ディスクやドラムの読み出し・書き込み機構に改良を加える必要があると考える。

4. おわりに

本論では，DBMSの問題点を整理し，それらの問題点を解決するための方策をソフトウェア・ファームウェア・ハードウェアの機能分担という面から，事例を交えて検討した。それらの方策は，局所的なもので，計算機アーキテクチャあるいは計算機システムとしてまだまとまったものではない。今後，以上の考察をもとに，さらに具体的な検討をすすめると共に，システム全体としてまとまったものを導き出したい。その際，とくに，可変長データの直接操作，検索・照合などの並列処理がポイントになると考えている。

— 参考文献 —

[1] 関野陽，植村俊亮： データベース・マシン； 情報処理，Vol. 17，NO.10（10月，1976），pp 940～946。

[2] 植村俊亮： データベースマシンのアーキテクチャ； 電気四学会連合大会，昭和51年。

[3] 関野陽： 非数値処理アーキテクチャ会議に出席して－DBMSのハードウェアサポートの研究； 情報処理学会データベース研究会資料 DB 29－1，昭和51年7月。

[4] R. I. Baum & D. K. Hsiao： A Data Secure Computer Architecture； COMPCON 76，pp 117～117。