データベースマシンのアーキテクチャ水準

On Architectural Levels for Database Machine

南部　明　　相磯　秀夫
Akira Nambu　Hideo Aiso
慶應義塾大学　工学部
Faculty of Engineering KEIO Univ.

1. まえがき

近年，計算機に関する技術革新は著しく，その機能は急速に増大し，大規模で高速なシステムが利用可能となってきた．一方，社会機構も多様化・複雑化しており，情報処理に対する計算機への期待は高まるばかりである．このような背景からデータベースの概念が生まれてきたといえる．計算機の応用分野の拡大は大規模な統合システムの構築を促しており，多目的利用の統合ファイルが必要となってきた．

システムが巨大化すれば，ソフトウェアの生産負荷は大きくなる上，ハードウェアに対する柔軟性は小さくなる．また，システムに対する要求は適用範囲が広がれば広がる程，変化が大きくなり，アプリケーション毎のシステム構成をとる必要も生じてきている．したがって，幅広い要求に対応できる柔軟なシステムを設計する必要があり，データベース管理システムはこの柔軟性を与える道具といえよう．しかし，現在のソフトウェアによるデータベース管理システムでは全体のシステムの柔軟性を改善するには不十分であった．データベースマシンは一面ではこの柔軟性を増すことを狙ったものであるといえよう．

Fig．1 に従来の情報処理システムと機能分散化の例を示した．

データベースマシンは機能分散処理の一方式であり，通信制御プロセッサが前置プロセッサと呼ばれているのに従い，後置プロセッサと呼ばれている。

データベースマシンは，Fig.1. に示された構

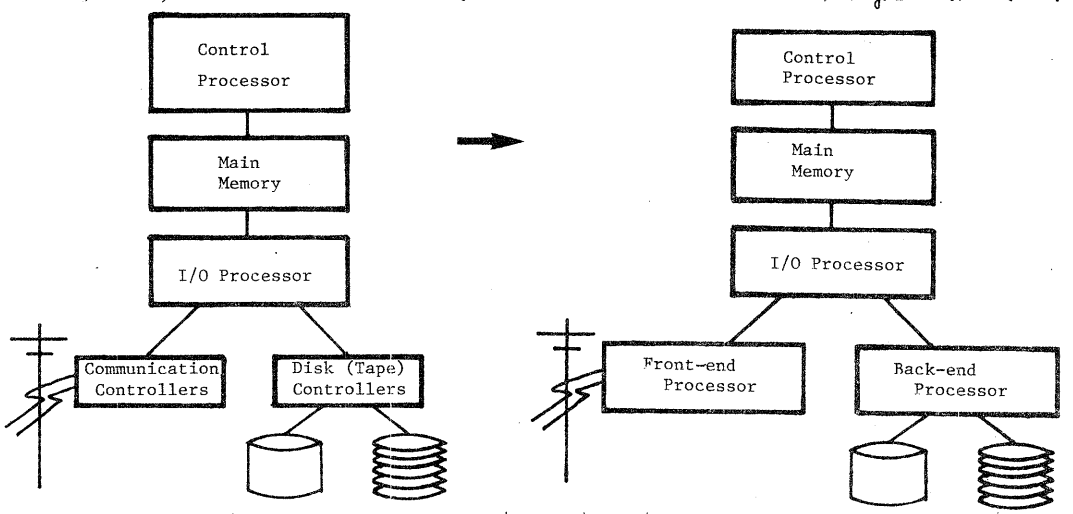

Fig. 1.　情報処理システムにおける機能分散化の例

成の他にも様々な構成がみられるが，ほとんどホスト計算機に対し，スレーブプロセッサとして動作するものである。したがって，ホストとのプロトコル水準は重要な意味を持つといえる。本論ではこのプロトコル水準に注目し，データベースマシンのアーキテクチャを考察することにする。

2. 情報処理の抽象化とデータベース管理機能の階層構造

データモデルはデータベースを設計する指針となる論理データ構造のわく組を提供するとともに，データベースを処理する上で基本となる操作群を抽象化している。

この操作群はデータベースマシンを設計する上で重要な要素となる。データベースマシンはこの操作群に基づいてデータベースを処理する。そして操作群の水準でホストとのプロトコルを設定しているものが多い。しかし，データモデルが提供する操作群とはギャップがあることは否めない。

ここでは，データモデルの操作群を大きく2つの水準に分け，いくつか例を紹介する。さらに，データベースを管理する上での機能を抽出し，操作の面から見た階層構造を考察する。

2.1 データベース操作群の水準

データモデルが提供する操作群は大きく2つに分けられる。1つはレコード操作レベルのもので，データベースの構造に依って作用する演算をみることができる。この水準の操作群を提供するデータモデルはデータベースの論理構造のわく組として，かなり物理構造を意識するものが与えられる。したがって構造を巡航する操作群が必要となってくる。この水準のデータベース操作群をL1水準演算と定義する。

他の1つは集合演算レベルの操作群であり，データベースへの操作を集合演算で抽象化する。この水準の操作群を提供するデータモデルでは，データベースの構造が多項関係あるいは2項関係で抽象化され，物理構造とは直接関連しない。この水準のデータベース操作群をL2水準演算と定義する。

[L1水準演算] L1水準演算の例として，DBTGモデル[1](ネットワーク)が提供するDML，階層構造をサポートするIMS[2]のDL/1等が挙げられる。いずれもKEY値による条件検索と構造に沿った条件検索（巡航検索）が基本演算となる。

DL/1の演算をTable 1に示す。

GU	Direct Retrieval
GN	Sequential Retrieval
GNP	Sequential Retrieval under Current Parent
GH_	As above but allow subsequent DLET/REPL
ISRT	Add New Segment
DLET	Delete Existing Segment
REPL	Replace Existing Segment

Table 1 DL/I operations (summary)

[L2水準演算] E.F. Coddの提案した関係モデル[3]におけるn項関係演算，王水林の提案した情報空間モデル[4]におけるn項関係演算と2項関係演算，およびJ. Minkerの二項関係演算[5]を取上げ，議論する。

1) 関係モデルにおける関係論理に基づく演算
（Coddの提唱したALPHAサブランゲージ[6]による）

- GET　W(β) : λ
- HOLD　W(β')
- PUT　W(β')
- UPDATE　W
- DELETE　W
- RANGE　relation-name　variable

ただし，λ：関係論理，β：ターゲットリスト
β'：単一リレーション内のターゲットリスト
W：作業領域

関係論理に基づく演算は述語表現の条件により対象リレーション群からデータを取出し，作業領域へ格納するGET操作が基本となる。他はデータベースの更新処理を抽象化している。

2) 関係モデルにおける関係代数[7]

- UNION, INTERSECTION, DIFFERENCE
- EXTENDED CARTESIAN PRODUCT
- RESTRICTION, PROJECTION, JOIN, DIVISION

古典的な集合演算とn項関係への新しく定義した演算からなる。

関係代数は全ての演算が2項演算である。そして，論理操作が明確であるため，1)に比べ処理が簡明でアルゴリズムを記述しやすい。

3) 情報空間モデルにおける情報代数 [4]

- $b[\lambda,\beta](A_1, A_2, \cdots, A_m) = \beta(\{\ell \mid \ell \in A_1 \times A_2 \times \cdots \times A_m \wedge \lambda(\ell) = \text{'true'}\})$
- $g[\rho,\gamma](A) = \gamma(A/\rho)$
- $S_1(A,B) = \{y \mid o(y) \in A \wedge y \in B\}$
  $S_2(A,B) = \{y \mid \delta(y) \in A \wedge y \in B\}$
  $S_3(A,B) = \{\delta(y) \mid o(y) \in A \wedge y \in B\}$
  $S_4(A,B) = \{o(y) \mid \delta(y) \in A \wedge y \in B\}$
  $S_5(A,B) = \{x \mid x \in A \wedge \forall y \in B(\delta(y) \neq x)\}$
  $S_6(A,B) = \{x \mid x \in A \wedge \forall y \in B(o(y) \neq x)\}$

ただし，λ：検索条件（$A_1 \times A_2 \times \cdots \times A_m$上の論理関数），
β：点生成関数，
γ：点生成関数，
ρ：点関数，
$\delta(y)$：2項関係の終点，
$o(y)$：2項関係の始点
A：ファイル，B：A上で定義された構造

情報空間モデルではデータベースの論理構造のわく組として，属性空間（データ単位間のn項関係）と事象関連（2項関係）を投影する関連空間を提供し，関係集合の分類を行っている

最初のバンドル操作 $b[\lambda,\beta](A_1, A_2, \cdots, A_m)$ は1)のGET W(ρ)：λと同様の操作である。グランピング操作 $g[\rho,\gamma](A)$ は集計操作を抽象化した演算である。1)では *image* 関数として，λやβ中に現れる。S_1からS_4までの操作は事象間関連に沿って巡航を行う演算である。

4) Minker の2項関係演算 [5]

- $R_1(a_1,a_2) \cup R_2(a_1,a_2) \longrightarrow R_3(a_1,a_2)$
- $R_1(a_1,a_2) \cap R_2(a_2,a_2) \longrightarrow R_3(a_1,a_2)$
- $R_1(a_1,a_2) / R_2(a_2,a_3) \longrightarrow R_3(a_1,a_3)$
- $R_1(a_1,a_2) \setminus R_2(a_3,a_2) \longrightarrow R_3(a_1,a_3)$
- $R_1(a_1,a_2) \wedge R_2(a_2,a_3) \longrightarrow R_3(a_2,a_3)$
- $R_1^*(a_1,a_2) \longrightarrow R_2(a_2,a_1)$

以上の6つの演算は2項関係に対しての操作である。Minkerは2項関係を単位行列を用いて表現し，6つの演算を行列演算に置き換えている。さらに推論機構をこれらの演算の上で展開している。

—— 1)～4)の比較検討 ——

特徴を列挙する。

- 2)と4)は論理操作が明確で，1)と3)に比べ，プログラム言語に近い。
- 1)と3)は述語論理表現で検索条件を示すため，自然言語とのインタフェースが良い。
- 3)はデータ構造を操作する演算を備えており，注目される。
- 4)は転置インデックスなどの補助的なデータ構造を操作する演算として期待できる。
- 1)～4)に共通する特徴として，データベース処理を集合演算で抽象化しており，非手続き的な高水準汎用データ操作言語をユーザに提供することができる。

（L2水準の演算をデータベースマシンのホスト計算機とのプロトコルとして設定する場合，大きな負荷がデータベースマシン側にかかることが予想される。したがって，現在提案されているデータベースマシンの中ではL2水準の命令を完全に用意しているものはほとんどない。）

2.2 データベース管理機能の階層構造

データベースを構築し，管理・維持してゆくために提供されるユーティリティがデータベース管理システム（DBMS）である。

DBMSの役割として

- 複雑なデータ構造の表現を可能とする。
- 多様なアクセス方式を提供する。
- データとプログラムの独立性を保証する。
- 種々の目的を持ったユーザに容易に使用できる言語を提供する。
- 機密保護・障害回復機能を有する。

が挙げられる。

DBMSの構造としてはFig.2の構造が一般的であろう。ただし，ユーザとのインタフェースとしてはL2水準で提供しているDBMSよりも，L1水準のデータ操作言語を提供するものが多い。

Fig.3は計算機システムにおいて，他のリフ

トウェア資源との関係を示した図である．

Fig.2 と Fig.3 から DBMS がオペレーティングシステム（OS）の基本システムの外にあることがわかる。つまり，記憶管理のわく外で，しかも，基本ハードウェアとは何階層も離れたところで構築されているのである。これは磁気テープ時代にその形が定まった旧式のファイル管理アーキテクチャと同等であり，技術革新の成果を取入れることが難しい原因となっている。

したがって現在のソフトウェアによるDBMSではアプリケーションが要求する性能を得られない場合が多い．Fig.5 に DBMS が適用可能なアプリケーションを図示した。座席予約システムや銀行業務システムなどのオペレーショナルシステムではリアルタイムでの高速応答や更新処理が要求される。さらに，データベースへの処理要求はパターン化しており，アプリケーション毎にシステム構成をした方がコストの点で有理なことが多い．一方，文献・特許・経済・企業等の情報を蓄積し，その情報に対する検索をサービスするデータバンク事業や企業・学校等の組織体情報を管理するインフォメーショナルシステムでは汎用の DBMS を適用しやすい．

データベースマシンの導入はデータベース処理の効率を上げ，さらに汎用DBMSの適用領域の拡大が期待できる。

3. データベースマシンのアーキテクチャ水準

データベースマシンはデータベース管理の一部を専用のプロセッサに委ねることにより処理効率および性能価格比を向上させようとするものである。2.2 で述べたように，DBMS は基本システムの外側で構築されていることから，データベースマシンを周辺装置側に置くことが一般的になってきている。これはデータベース管理機能の一部を2次記憶装置側に移行させても基本システムにはほとんど影響を与えないからである．したがってデータベースマシンはあくまでスレーブ・プロセッサであり，周辺装置の1つとして見ることができる。

データベースマシンの導入により，ホスト計算機側の負荷は減少することが期待できる。さ

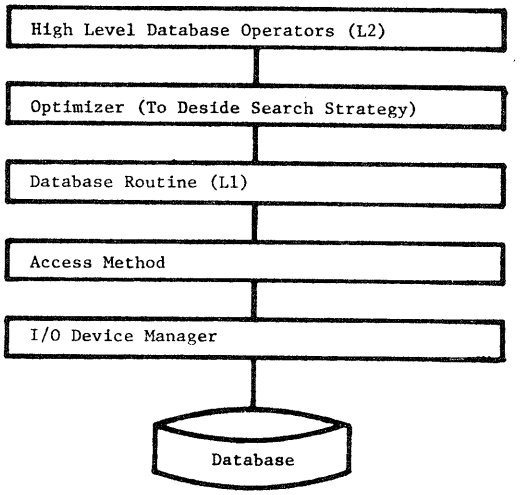

Fig. 2 DBMS の階層的モデル

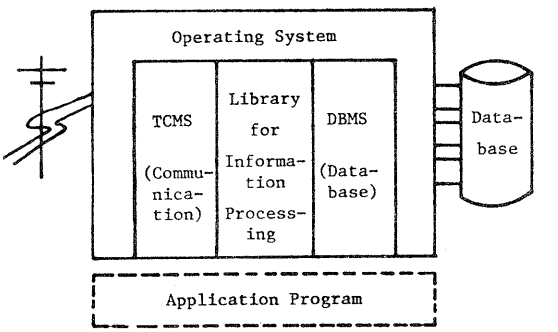

Fig. 3 ソフトウェア・リソースの構造

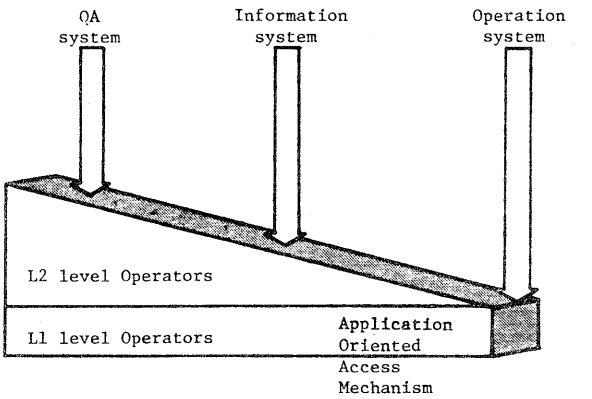

Fig. 4 DBMS の適用領域

らに，その利点として
- データベース管理のハードウェアサポートが可能となる。
- OS および アプリケーション・システムに対しデータベースの独立性が増す。

ここではデータベースマシンのホストとのプロトコル水準を比較することにより，アーキテクチャの問題点を考察する。

3.1 ホスト－データベースマシン間プロトコル水準

ホスト計算機とデータベースマシン間のプロトコル水準はデータベースマシンの処理能力のパラメータとなる。つまり，ホスト計算機での処理負担やホスト－データベースマシン間の通信量は，このプロトコル水準から予測ができる。

Table 2 では5つのデータベースマシンについて，そのホストとのプロトコル水準で分類し，ホスト側の負荷と交信回数を簡単に評価した。

L1水準のプロトコルを設定した場合，レコード単位の処理が基本となり，目的のデータ集合を得るまで，処理命令がかなり繰返されることが予想される。Table 2 を見ると，L1 水準のプロトコルが設定されている XDMS や DASD プロセッサは検索するレコードの数に比例して多くなる。また，XDMS が DBTG モデルのデータベースの管理をするのに比べ，DASD プロセッサは物理レコード（ブロック）から論理レコードを生成するだけの機能しかない。しかし，交信回数を比べると大差がない。したがって L1 水準のプロトコルを設定する場合，データベースマシンの機能を高めても交信オーバヘッドのため処理効率は上らないと考えられる。解決法としてはホスト－データベースマシン間の通信手段に高速な回線を使用することである。

L2 水準の場合，レコード集合の処理が基本であり，検索条件に合致した目的レコードの集合がデータベースマシンの応答となる。したがって複雑な検索条件で目的レコードを指定しても，ホストとの交信回数は2回ですむことが多

プロトコル水準		L1（レコード操作）		L2（レコード集合操作）		
データベースマシン名		DASDプロセッサ[10]	XDMS[11]	Hsiao's DBC [12]	CASSM[13]	RAP[14]
単純検索	ホスト側処理量	$O(\lg N)\sim O(N)$	$c\sim O(N)$	c	c	c
	ホストとの交信回数	$4\sim 2N$	$4\sim 2N$	2	2	2
巡航検索	ホスト側処理量	$O(N')$	c	c	c	c
	ホストとの交信回数	$2N'$	$2N'$	$2\sim 2N'$	2	2
プロセッサ量		d	w	$d\,dst+td+5w$	dSt	$dst+w$
プロセッサ量具体値*		8	100	2180	16000	16100
データベースの構造		Lコード	DBTGモデル	属性形データベース転置インデックス	階層構造	関係データベース
備考		・物理Lコード(ブロック)を論理Lコードに加工 ・バッファリング	・DBTGのDMLコマンドを実行する	・属性形データベースとは属性名とその特定の値を対にし，この対を複数個並べたものをLコードとする ・全属性値に対し転置インデックスを張る	・L2水準の処理要求をホスト側でCASSMの機械語に翻訳し，パケットにして転送	・Virtual RAPの場合，2次記憶とのデータ転送はRAP側のプロセッサ(μP)で管理する。
		交信オーバヘッド		データベースマシン処理オーバヘッド		

N：目的ファイルのレコード数
N'：目的構造のレコード数
d：Disk装置数
S：シリンダ数（ディスク毎）
t：トラック数（シリンダ毎）
w：高性能プロセッサ
α：インデックス量（比率）

* d=8，S=200，t=10，w=100，α=0.1

Table 2 データベースマシンのプロトコル水準による簡易的性能評価

い。

しかし，データベースマシン側に負荷の大部分がかかることを注目せねばならない。Table2を見てもわかるようにL2水準のデータベースマシンのプロセッサ量（ハードウェア規模の一応の目安）は，L1水準のデータベースマシンに比べ，大きい。

また，L2水準のデータベースマシンが提供するプロトコルで，2.で示したデータモデルが与えるL2水準の演算を包含するものはほとんどない。プロセッサの処理能力や作業領域の制限により，各データベースマシン毎にプロトコル水準は異なる。

L2水準の演算を提供するデータモデルは極て論理性の高いデータ構造をデータベースの論理構造のわく組としている。データベースマシンがL2水準の命令をハードウェアレベルで持つ場合，操作対象の物理構造がこの論理構造と同等になることが多い。例えば，関係モデルでは論理構造としてはフラット・テーブル形のリレーションしか与えていないから，関係データベースマシンではフラットテーブルへの操作を連想処理により実行せねばならない。

一般に，L2水準の演算を提供するデータベースモデルでは，論理データ構造として，ある定義に従った情報空間を与えている。したがって，ハードウェアでL2水準の演算を実行しようとすると，連想処理方式を採用する場合が多い。しかし，連想処理方式によるデータベースマシンは，性能価格比と融通性の点で問題がある。

最近では，連想処理方式の論理的実現の容易性を失わず，性能価格比を上げることを目標に転置インデックスなどの補助アクセス構造を取入れる試みが出てきた。HsiaoのDBCなどはその代表例であろう。

HsiaoのDBCではデータベースの全属性値に関し，転置インデックスを持っている。したがって，データベースマシン側で検索方法を選択（最適化問題）する必要はなく，ある一定のアルゴリズムで検索を実行すれば良い。しかし，全属性値に関する転置インデックスを持つことは，その量と管理のオーバヘッドを考えた場合，妥当ではない。（DBCではhashingの概念を利用し転置インデックスの大きさを小さくしている）　そこで，データベースマシンへの要求パターンに沿った転置インデックスを構成する方式が考えられる。

全ての属性値に関し転置インデックスをもたない場合問題となる点は以下の2点である。

- 検索方法をデータベースマシン側で決定する必要がある。
- どの属性値に関し転置インデックスを持つと効果があるか，要求パターンをモニターリングしながら，決定する必要がある。

L2水準のプロトコルが設定されているデータベースマシンでは転置インデックスの管理のため，かなりの負荷がかかることが予想される。したがって，ホストとのプロトコルに補助アクセス構造を操作する命令を加える必要があろう。

4. データベースマシン REAM HVφ のアーキテクチャと今後の展開

ここでは著者らが設計した REAM HVφ [15] のアーキテクチャの概要を述べ，今後の展開について議論する。

4.1 REAM HVφ のアーキテクチャ

全体のシステム構成をFig.5に示す。

REAM HVφ は関係代数を包含する命令をホストとのプロトコルとして設定している。このプロトコル（REAM MACRO）をTable 3にまと

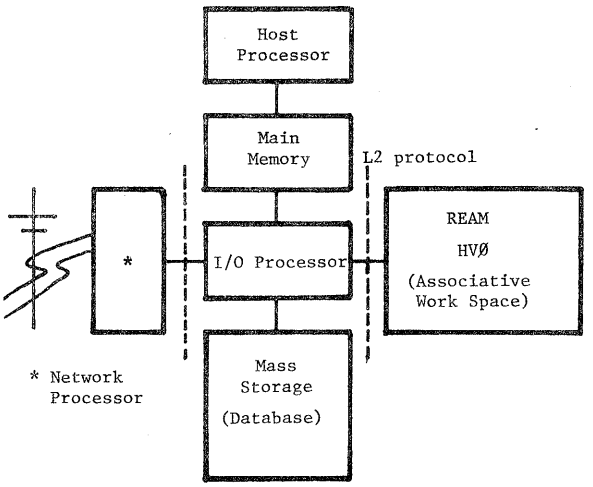

Fig. 5 REAM HVφ システムの全体構成

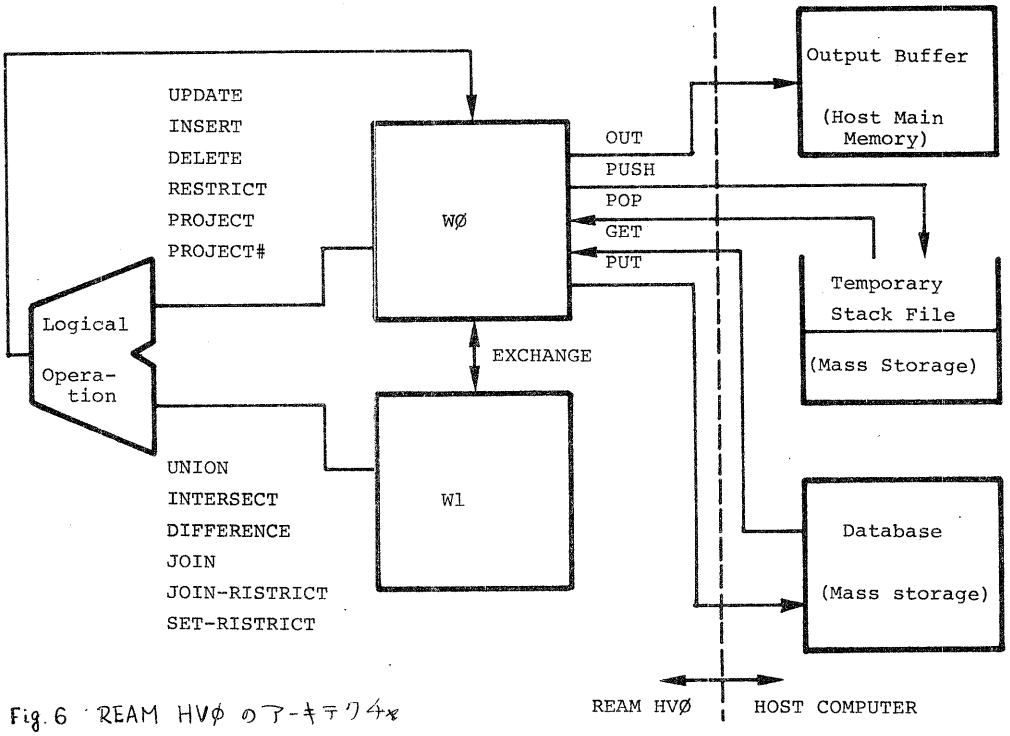

Fig.6 REAM HVØ のアーキテクチャ

TYPE	REAM MACRO INSTRUCTION	OPERATION
RELATIONAL ALGEBRA	UNION	WO ← R ∪ S
	INTERSECTION	WO ← R ∩ S
	DIFFERENCE	WO ← R - S
	RESTRICTION	WO ← R[A θ B] = {r:r∈R ∧ (r[A] θ r[B])}
	PROJECTION #	WO ← R[A] = {r[A]:r∈R} *duplicate*
	PROJECTION	WO ← R[A] = {r[A]:r∈R} *non-dupricate*
	SET RESTRICTION	WO ← R[A ϕ B]S = {r:r∈R ∧ (R[A] ϕ S[B])}
	JOIN RESTRICTION	WO ← R[A θ B]S = {r:r∈R ∧ s∈S ∧ (r[A] θ s[B])}
	JOIN	WO ← R[A θ B]S = {(r s):r∈R ∧ s∈S ∧ (r[A] θ s[B])}
STORAGE I/O	SET RELATION	WO ← T
	GET	WO ← DATABASE STORAGE
	PUT	DATABASE STORAGE ← WO
	OUT	OUTPUT FILE ← WO
*	PUSH	TEMPORARY STACK FILE ← WO, TSFP ← TSFP + 1
	POP	WO ← TEMPORARY STACK FILE, TSFP ← TSFP - 1
	EXCHANGE	WO ← W1 and W1 ← WO
**	INSERT	WO ← R ∩ T
	DELETE	WO ← R - T
	UPDATE	WO ← (R[A] ← (t[C]:t∈T))
OTHERS	MODIFY	ARITHMETIC OPERATION
	SORT	SORT OPERATION
	CONTROL	STATUS CONTROL

WO : working buffer 0
W1 : working buffer 1

R : a relation in WO
S : a relation in W1
T : a relation gived from an instruction operand

r, s, t : tupple
A, B, C : attribute

θ : ≠, =, <, ≤, >, ≥
ϕ : ⊄, ⊃, ⊂, ⊆, ⊃, ⊇

* WORKING BUFFER CONTROL
** DATABASE MANIPULATION

Table 3 REAM MACRO Instructions (summary)

めた。

REAM HUφ では L2 水準の演算を直接実行するアーキテクチャが設定されている。Fig.6 に REAM HUφ におけるデータの流れと命令の作用個所を示した。

さて，REAM HUφ のアーキテクチャ上の特徴を列挙する。

- REAM HUφ は連想処理による集合演算プロセッサであり，その記憶領域は作業領域として利用される。
- データベースの蓄積場所はホスト側の管理するマス・ストレージである。
- マス・ストレージと REAM HUφ とのデータ転送単位は，フラット・テーブル（リレーション）のコラム（ドメイン）単位である。
- ホスト側での物理データベース管理は基本ファイルシステムで行う。
- ジョブ毎に REAM MACRO 命令列をパケットにして転送するため，ホストとの交信は少ない。
- REAM HUφ は関係代数に基づく命令をハードウェアで実行する関係データベースマシンである。

REAM HUφ のアーキテクチャは情報処理システムの拡張性を損うことなく，データベースの高速処理を図ることを目的としている。

REAM HUφ のハードウェアは CCD を記憶素子とし，その特徴を生かした連想プロセッサにより設計されている。ハードウェアの詳細は本論では述べないが，処理方式の特徴を以下に述べる。

- 4つのフラグ（データ・アイテム毎）の状態を制御することにより，複雑な連想機能を実現。
- 制御用フラグをデータに付けて記憶するタグ・マシン（RAP [14] 方式）ではなく，制御用フラグは専用レジスタに置かれる。
- ロジックはマイクロプログラムで制御される。

REAM HUφ は現在ハードウェアの設計まで行ったが，実装はしていない。これは設計段階（ハードウェアの設計，同時に進めた関係データベース管理システム SVφ の開発）で生じた問題点により，アーキテクチャ上の変更が避けられなくなったからである。

なお，関係データベース管理システム SVφ は REAM HUφ のソフトウェアを開発するとと，REAM MACRO レベルでの HUφ のシミュレータとして機能することを目的とした実験システムである。現在，UNIVAC 1106 上で実動しており，TSS モードで SEQUEL の問合せができる。[16]

最後に，REAM HUφ の問題点を列挙する。

1) アーキテクチャ上の問題点

- 物理構造としてはフラット・テーブル（リレーション）のコラム（ドメイン）単位でブロックに割当てた構造しか持たない。したがって，各命令はコラム内アイテム値の全数探索を基本に実行される。
- 作業領域の大きさがハードウェア上の制限から決ってくる。したがって，処理対象のデータが作業領域を越える場合に備え，記憶管理機構を REAM HUφ 専用に用意する必要がある。
- REAM HUφ ではファイル管理をしないので，二次記憶とのアクセス時間，およびデータ転送のオーバヘッドは解決されない。
- REAM HUφ では単一ジョブしか処理できないため，応答特性が悪くなる可能性が高い。

2) ハードウェア設計上生じた問題点

- ハードウェアで処理する命令の水準が高く，ロジック量，および制御記憶が大きくなる。

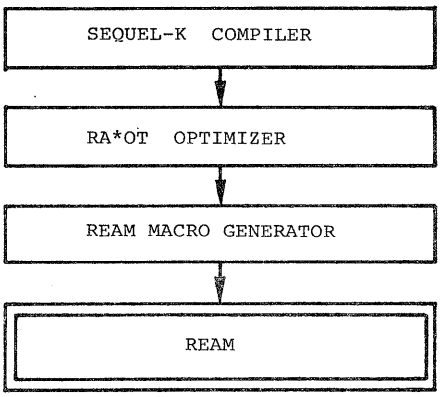

Fig.7 System Structure of Relational Database Management System 'SVφ'

・ロジックのタイミングが非常にクリティカルで，マイクロプログラムを書くのが難しい。
・ビット直列で演算処理を行うため，速度が遅い。

4.2 今後の方向

REAM HVΦで生じた問題点を考慮し，アーキテクチャの改善を提案する。

1) 分散データベース指向　分散データベースはデータ資源の共有をアクセスの局所性を生かしながら実現できる。将来，分散データベースが実現されるであろうアプリケーションは多数あり，データベースマシンも分散データベースに対処したアーキテクチャをとる必要があろう。ここではFig. 9に示される分散データベースシステムをモデルとし，ネットワーク上での標準プロトコルをL2水準で設計する。

2) Storage Processors　階層記憶システム上にセグメンテーションによる連一記憶空間を形成する。本システムの核となるマスタプロセッサである。
セグメンテーションはデータベース向きにアレンジされるが，特に次の点に注目する。
・セキュリティ機能のサポート

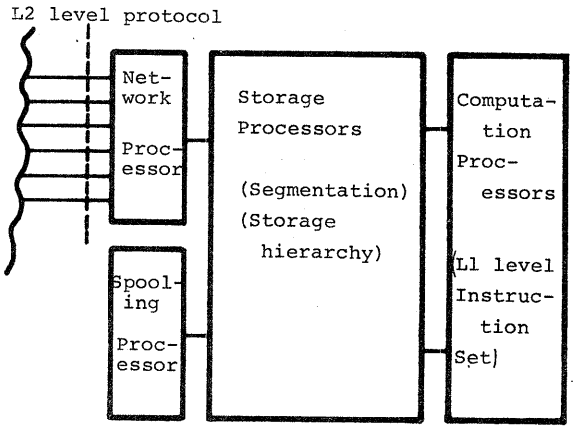

Fig.8 New Architecture of Database Machine

operator	condition	result
product	$a_1=a_3$ $a_2=a_4$	$(a_1,a_2) \longrightarrow W2$
sum	$a_1=a_3$ $a_2=a_4$	$(a_1,a_2),(a_3,a_4) \longrightarrow W2$
compose	$a_2=a_3$	$(a_2,(a_1,a_4)) \longrightarrow W2$
adiacency	$a_2=a_4$	$(a_2,(a_1,a_3)) \longrightarrow W2$
merge	$a_1=a_3$	$(a_1,(a_2,a_4)) \longrightarrow W2$
converse	$(a_1,a_2)\in W0$	$(a_2,a_1) \longrightarrow W2$

$(a_1,a_2)\in W0$, $(a_3,a_4)\in W1$

Table4 Binary operators for 2-tuples

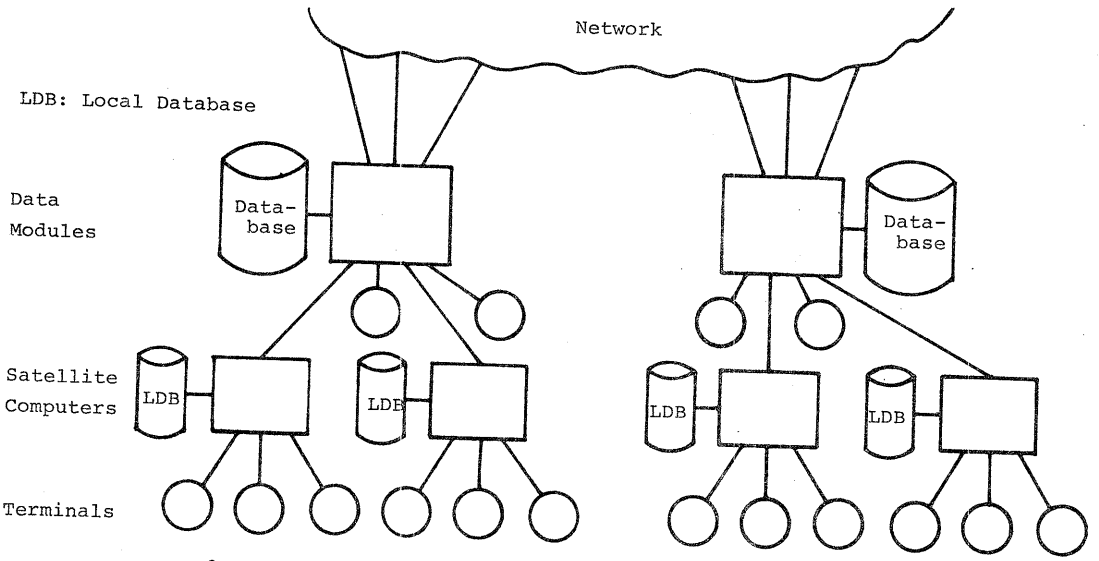

Fig.9　分散データベースシステム

・多重処理，および分散データベースの際，問題となる同時アクセス，デッドロックの回避，更新処理の制御．
・データベースへのアクセスパターンによってセグメントを最適配置する．

3) Computation Processors　2項関係の組を操作するL1水準の命令，およびデータベース処理向きにチューニングされた命令セットを有するプロセッサ群。その特徴としては次の2点が挙げられる。
・プロセッサは Storage Processor が提供する記憶空間上で演算を実行する。
・2項関係は，転置インデックスなどの補助アクセス機構を抽象化する。そこで2項関係演算をL1水準で取入れることによって複雑な検索条件の処理を効果的に実行する（Table 4に2項関係演算に基づく2項組への操作命令をまとめた．）

5. おわりに

本編では，データベースマシンのアーキテクチャをホスト計算機との間のプロトコル水準の観点から考察した。さらに，著者らが設計したデータベースマシン REAM HVφ のアーキテクチャについて述べ，その問題点を列挙した。最後に，将来のデータベースマシンの方向を探る目的で研究を進めているシステムについて，その概要を述べた。

今後，情報ユーティリティの利用形態の変化とともに，データベースマシンへの要求が高まるであろう。さらに研究を進め実用化を考える時期に来れようである。

[謝辞]　日頃から御指導，御助力頂いている産業能率短大，小林 功武教授，ならびに，本塾，所 真理雄助手に深謝いたします．
また，関係データベース管理システム SVφ を実装した以下の諸氏に深謝いたします．
上林 宏行，小花 貞夫，龍塚 博志
山形 久雄，清木 康，小沢 裕之
加藤 洋　（慶應義塾大学工学部）

参考文献

[1] Data Base Task Group of CODASYL Programming Language Committee. Report, Apr. 1971.
[2] C.J.Date, An Introduction to Database Systems second edition, p.205-p.306, Addison-Wesley, 1977.
[3] E.F.Codd, "A Relational Model of Data for Large Shared Data Banks," Comm. ACM, vol 13, No.6, Jun. 1970.
[4] I.Kobayashi, "Information and Information Processing Structure," Information Systems, vol.1, No.2, 1975.
[5] J.Minker, "Binary Relations, Matrices and Inference Developments," Information Systems, vol.3, pp. 37-47, 1978.
[6] E.F.Codd, " A Data Base Sublanguage Founded on the Relational Calculus," Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access and Control.
[7] E.F.Codd, "Relational Completeness of Data Base Sublanguages," In Data Base Systems, Courant Computer Science Symposia Series, vol.6, Englewood Cliffs, N.J. : Prentice-Hall, 1972
[8] G.D.Held, M.R.Stonebaker, and E.Wong, "INGRES—A Relational Data Base System," Proc. NCC 44, 1975.
[9] M.M.Astrahan, et al, "System R : A Relational Approach to Data Base Management," ACM Tran. on Database Systems, vol.1, No.2, Jun. 1976.
[10] T.Lang, E.Nahouraii, K.Kasuga, E.B.Fernandez "An Architectural Extension for a Large Database System Incorporating a Processor for Disk Search," 3rd Int. Conp. on VLDB, Tokyo, Oct. 1977.
[11] R.H.Canaday, et al, "A Back-end Computer for Data Base Management," Comm. ACM, vol.17, No. 10, Oct. 1974.
[12] R.I.Baum and D.K.Hsiao, "Database Computers — A Step Touards Data Vtilityes," IEEE Trans. on Computers, vol.C-25. No.12, pp. 1254-1259, Dec. 1976.
[13] G.P.Copeland, G.P.Lipovski, and S.Y.W.Su, "The Architecture of CASSM : A Cellular System for Non-Numeric Processing," Proc. 1st Annual Symposium on Computer Architecture, pp. 121-128, Dec. 1973.
[14] E.A.Ozkarahan, S.A.Schuster, and K.C.Smith, "RAP—An Associative Processor for Database Management," AFIPS Conf. Proc. , vol.44, pp. 379-388, 1975.
[15] A.Nambu, "The Architecture of REAM HVO," Internal Report, Keio Univ. , 1977.
[16] 上林他，"実験リレーショナル・データベース管理システム SVφ，"他3編，昭53通信学会全国大会，1358～1361，March，1978.