データフロー制御データストリーム処理方式データベースマシン（基本構想）

田中　譲
北海道大学・工学部

1. はじめに

1970年代には種々のアーキテクチャのデータベースマシンが盛んに研究されたが，80年代に入って，この分野は反省期を迎えているように思われる。主な反省点は，

1. 汎用データベースマシンは存在し得るか？
2. 大容量データベースをどう扱うか？
3. RASISをどう支援するのか？

の3点である。

第1点は，ニーズの分析の結果として生じた疑問というより，処理対象データのデータ長の柔軟性と，ジョイン等の複雑な演算の高速処理とを同時に満たすようなマシンを開発することの難しさに根差した疑問である。利用者のニーズとしては，IRやバンキングシステムにもDBMSを用いたいとの動きも出始めている。マシンの分化を考えることも重要であるが，一方では，

1. 可変長データが扱え，
2. タプル数nが10^6程度にもなる大規模な2つの関係のジョインをもO(n)で処理でき
3. ジョイン属性の値として，100文字以上の文字列が現われることも許す

というようなマシン・アーキテクチャの研究も必要である。本論文ではそのようなマシンを提案する。

データベースマシンの大容量化に関する反省は，最下層メモリとしてCCDや磁気バブルメモリを用いるマシン・アーキテクチャに向けられたものである。本研究でも大前提として，

"最下層メモリには移動ヘッドディスクを用いる"

という条件を課す。これに伴い，関係のセグメンテーションも必要となる。

第3点は，RASISの支援について考慮したマシン・アーキテクチャが皆無であったことについての反省である。これは，従来の研究の重点が，データベース処理全体の内の一部である関係代数演算の処理に置かれ過ぎたためである。本研究では，セグメント化された関係をオブジェクトと見做し，オブジェクト指向アーキテクチャで，他のデータベース処理モジュールを制御することを考えている。

本研究では，関係の各属性値をデータタイプ毎にコード化したコード化関係を用いて関係代数演算を行うことを提案し，この演算の高速処理のためのサブシステムと，符号化と復号化のそれぞれのためのサブシステムのアーキテクチャの概要を示す。

2. アーキテクチャの概要

2.1. 方式に関する基本的な考察

本研究の指針となった基本的な考察を以下に列挙する。

a. 普通の移動ヘッドディスクを最下層メモリとして用いるという仮定のもとでは，セレクションの実行時間を，インデクス・ファイルを持ったソフトウェア後置型データベースマシンと，これを持たないハードウェア後置型データベースマシンとで比較してみても大差ないと予想される。

b. 更新処理に於て過大な負担とならない限りは，データの冗長度を上げて

で§，処理の局所性大，処理系の単純化を図ることによって，総合的なコスト・パーフォーマンスを上げるべきである。

c. 関係モデルでは，どの属性も一様に扱われるが，現実には，検索条件式中に現われることがなく，出力の際にのみ必要となる属性がある。このようなread-onlyの属性を，関係代数演算の処理系に持ち込むことは，効率を下げるだけで，何等利点がない。

d. 性別のように，男と女の2値しか現れないような属性を持つ関係R(X, sex)は，R-male(X)とR-female(X)のように2つの関係に分けて管理処理すればよい。

e. データベースマシンとホストコンピュータの間，ホストコンピュータと入出力機器，およびネットワークとの間，入出力機器と利用者との間のそれぞれにチャネル容量の上限があり，これらを無視して，データベースマシンの高速化を図ってが無駄である。データベースマシンの入出力のチャネル容量は，せいぜい1～10 MB/secと見做してよいであろう。これは，検索結果の出力が5,000～10,000タプル/secで得られるスループットに対応する。

f. 検索結果の統計解析結果のみを出力する場合には，そのスループットの値はあてはまらない。このような統計解析用の検索処理のスループットは大きければ大きい程良い。

g. 属性値のデータ型を，数値と文字列，順序の定義の有無，出力の際にソートの指定を受けるか否か，不等号によるセレクション演算を受けるか否か，不等号によるジョインやリストリクションの対象となるか否か，四則演算の定義の有無で分類すると表2-1のようになる。id#というのは

		order	sort	θ-selection	θ-restriction	+, −, ×, ÷
Number	number	∨	∨	∨	∨	∨
	id #	∨	∨	∨		
	date	∨	∨	∨	∨	∨*
String	word	∨	∨	∨		
	text					

＊："1982-01-31" + 1 = "1982-02-1"
"1982-Jan-31" + 1 = "1982-Feb-1"

表2-1. データ型の分類

識別番号のようなデータ型で，dateは年月日を，時分秒等のデータ型を指し，text型は順序が定義されない。wordはdate型やtext型以外の文字列を指す。date型以外の文字列型は，不等号によるジョインやリストリクションの対象になることはない。date型は，number型へ変換することができる。表2-1より，number型とdate型，id#型とword型が同じ性質を持ち，これらとtext型の3つの型に分類することができることがわかる。そこで，

A型 : number, date
B型 : id#, word
C型 : text

と分類する。

h. データベースマシンのアーキテクチャを考える上で大きな問題となるのが可変長データの取り扱いである。可変長データというのは，表2-1のword型やtext型に現われる。word型やtext型を，これらに対して定義されている順序を無視して，固定長のデータへと1対1に符号化できたと

すると，ジョインやリストリクションの処理は，符号化された世界で処理できることが表2-1よりわかる．プロジェクションの処理における重複タプルの除去も，符号化された世界で処理できる．word型やtext型に対しては，四則演算も定義されていないから，このような型の属性について平均や総和の計算を要求されることはない．この型に対する統計演算は数え上げのみであり，これも符号化された世界で処理できる．復号化は，出力に対してのみ行えばよい．符号化された世界での関係代数演算処理は，固定長データの処理になり，このサブシステムのマシン・アーキテクチャを簡略化することができる．

2.2 符号化/復号化アーキテクチャ

図2-1に，2.1節のように述べた符号化関係の処理による要求処理の概念図を示す．図2-2には，処理の具体例を

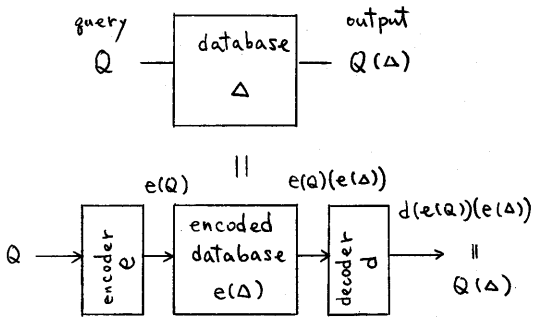


図2-1. 符号化データベースを用いた関係代数処理

示す．データベースの定義時に，各属性には固有のデータタイプが与えられるものとする．ジョインやリストリクションでは，同じデータタイプを持つ属性の間の比較のみが許される．符号化は，各データタイプ毎に行われる．

(a) type

Name = word ;
Place = word ;

attribute

Student : Name ;
Age : integer ;
Home_Town : Place ;
City : Place ;
Prefecture : Place ;

(b) Q = (R [Home_Town = City] S) [Age, Prefecture]

R

Student	Age	Home_Town
Nozaka	26	Kyoto
Habata	24	Sapporo
Masuyama	24	Shizunai

S

City	Prefecture
Kyoto	Kyoto
Sapporo	Hokkaido
Shizunai	Hokkaido
Tokyo	Tokyo

(c) encoder

Name	Use	Code
Habata	{R}	0
Masuyama	{R}	2
Nozaka	{R}	1

Place	Use	Code
Hokkaido	{S}	3
Kyoto	{R,S}	2
Sapporo	{R,S}	0
Sizunai	{R,S}	4
Tokyo	{S}	1

(d) encoded database

e(R)

Student	Age	Home_Town
1	26	2
0	24	0
2	24	4

e(S)

City	Prefecture
2	2
0	3
4	3
1	1

(e)

Age	Prefecture
26	2
24	3

(f) decoder

Code	Name	Use_Count
0	Habata	1
1	Nozaka	1
2	Masuyama	1

Code	Place	Use_Count
0	Sapporo	2
1	Tokyo	2
2	Kyoto	3
3	Hokkaido	2
4	Shizunai	2

(g)

Age	Prefecture
26	Kyoto
24	Hokkaido

図2-2. 符号化データベースの処理例

符号化の行われるのは，桁数の大きい id# 型，date 型，word 型，text 型である．date 型は，適当な関数を用いて 1 対 1 に number 型へと変換できる．桁数の大きい id# 型は，word 型として扱う．word 型と text 型は変換テーブルを用いるが，word 型では，変換テーブル中の word 型データは辞書式順序を保って管理される必要がある．したがって word 型は B-tree で，text 型は hash でテーブルを管理する．図 2.2 (c) に見るように，word 型であっても符号化データが，符号化の前のデータ間の順序を保つ必要はない．θ-ジョインや θ-リストリクションを行うことがないからである．R[Name >'Ikeda'] のような θ-セレクションは，(c) の変換表より，Masuyama, Nozaka が条件に該当することを知り (e(R)[Name = 2]) ∨ (e(R)[Name = 1]) を計算すればよい．

符号化モジュールと復号化モジュールでの処理は，基本的には，変換テーブルを関係と見做してセレクションの処理を行うことである．2.1 節の a の考察から，これら 2 つのモジュールは，ソフトウェア後置型データベースマシンを多重化することで実現するのがよいと考える．多重化は符号化テーブル（図 2.2 (c)）と復号化テーブル（図 2.2 (f)）のセグメント化を意味する．符号器はデータタイプに定義されている順序やハッシュ値の順序でセグメント化するのが望ましい．これに対し，復号器は，符号の値の順序でセグメント化されることが望ましい．図 2.1 で更新処理を考慮すると，図 2.3 のようにデータパスの追加が必要である．(5) は，検索条件を満足するタプルを削除するような場合に必要である．そのようなタプルが符号化されて (3) より得られると，各属性の符号値が復号器で調べられ，その符号値をデータベース中の何箇所で用いているかを記憶している use-count（図 2.2 (f)）が 1 だけ減じられる．その結果，どこでも用いられていないとわかった符号値は，そのデータタイプの未使用の符号値を管理している場所へと戻され，符号器からも消去される．(6) のデータパスは，新しく関係を登録するような場合に必要である．

図 2-3 の構成では，(5) と (6) には生データが転送され，しかも符号器と復号器のセグメンテーションの仕方が異なっているために (5), (6) のネットワークのスループットを上げることが難しい．そこでこれを図 2-4 のように改善する．復号器には，生データを持

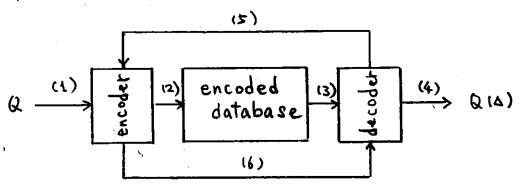
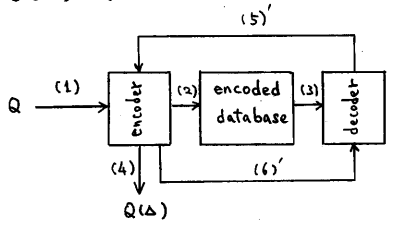


図 2-3 変換テーブルの更新



encoder　　P.A. : physical address

P.A.	Name	Use	Code
A1	Habata	{R}	0
A2	Masuyama	{R}	2
A3	Nozaka	{R}	1

P.A.	Place	Use	Code
A'1	Hokkaido	{S}	3
A'2	Kyoto	{R,S}	2
A'3	Sapporo	{R,S}	0
A'4	Sizunai	{R,S}	4
A'5	Tokyo	{S}	1

decoder

Name

Code	P.A.	U.C
0	A1	1
1	A3	1
2	A2	1

Place

Code	P.A.	U.C
0	A'3	2
1	A'5	2
2	A'2	3
3	A'1	2
4	A'4	2

図 2-4 符号化 / 復号化方式

つのではなく，生データが符号器で管理されている物理アドレス（物理ブロック番号）を持つ．これにより，(5)'(6)'のネットワークを流れるデータが固定長にでき，変換テーブルの記憶に要する記憶領域を小さくができる．この場合も，符号器と復号器でのセグメンテーション方式は異なるので，(5)'(6)'では，後述する動的クラスタリングを行う必要がある．

2.3 動的クラスタリングとソート

2.2節の符号器/復号器を例にとって説明する．両者とも物理アドレス（P.A.）と符号（C）の関係を情報として持っているが，符号器では，P.Aの順に並べられており，復合器中ではCの順に並べられている．符号器ではPAの順に処理することが望ましく，符号器より復号器へ送られるデータはP.Aの順になる．一方，復合器ではCの順に処理することが望ましい．したがって，符号器から復号器への転送では，Cについてソートする必要がある．符号器と復号器がそれぞれ単一モジュールであればこれでよいが，各々が複数モジュールより成る場合には，符号器の各モジュールからの出力を1つにまとめてソートし，これをCについて各復号器に分配する方法では多数のモジュールに対して転送路が1本となり，隘路を生じる（図2.5(a)）．逆に，Cの値で先に分配し，各復号モジュールに転送されてからソートする場合には，分配に際してデータ転送の衝突を生じる（図2.5(b)）．

これに対し，ソートと分配をネットワークの機能として統合したSDネットワークを提案する．SDはソータで，著者等の開発したソートエンジン（SOE）を基本モジュールとする．分配はD-busを用いて各符号モジュール毎に行われ，M-busによって，各復号モジュール毎

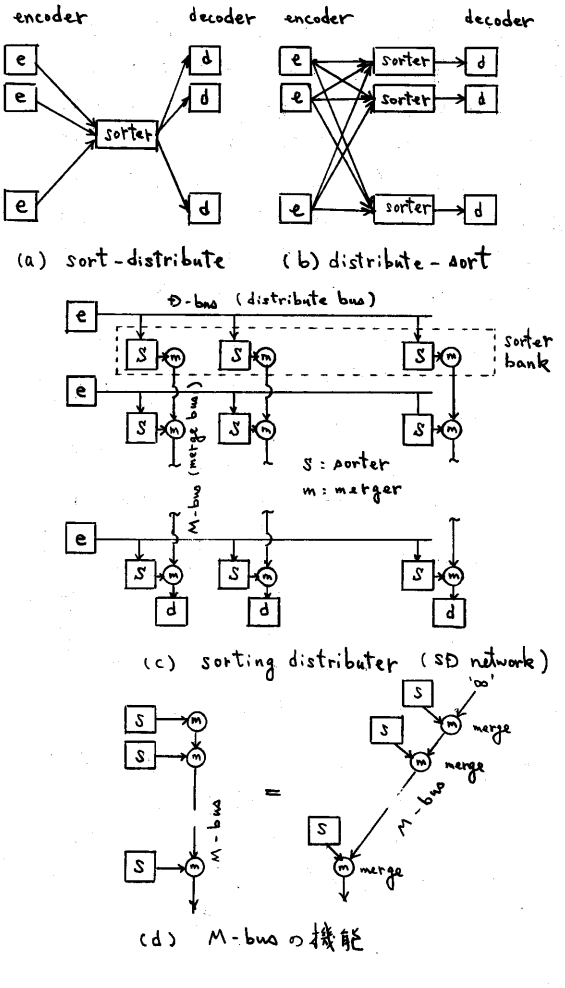


図2.5 ソートと分配の方式

に，各列のソータの出力をマージして復号モジュールへと転送する．SDネットワークはバッファでもあり，大容量を必要とするが，後述のように大容量化は通常のRAMを用いて行う．各符号モジュールより一度に送出されるデータ量は，セグメントの大きさで上限が決まる．よって，図の(c)に示すソータバンク毎に要するRAMの大きさは一定である．よって，ソータの容量を増

すために使用するRAMバンクは，ソータバンク毎に分割することができる．

データ長が固定の場合，2つの処理システム間のデータ転送において，間にソータを介在させることが望ましい場合，各システムが多重化された場合にはSDネットワークを用いることができる（図2.6(a)）．ただし，各データ

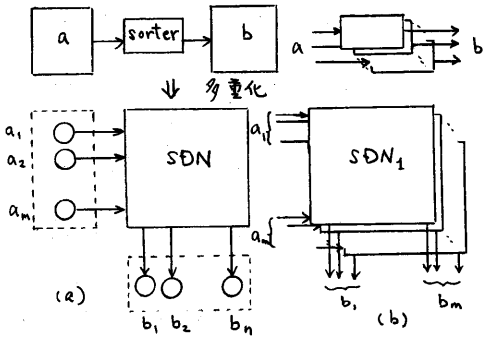


図2.6 ソータを介す転送の多重化

の転送先は，モジュラスを計算する等によって，データより計算可能とする．さらに，転送ジョブの多重化は，同図(b)のように多重度だけSDネットワークを用いることにより実現できる．

2.4 大容量ソータ

本節では，読者がソートエンジン(SOE)について既に知っておられるものと仮定する．(1)

SOEはヒープ構造を用い，図2.7(a)のような状態でキーを入力すると，同図の次経路上のデータ列にこの経路上で順序正しく新しいキーが挿入される．SOEが図(c)のような時，最下位レベルの(i+1)番目のノードと根ノードを結ぶ経路に次経路を固定し，ヒープの順序と同順序（図では昇順）にソートされたキー列を入力した後，あふれたキーをあふれた順に上から下へ(i+1)番目のノードの下に並べてみると，このアンバランスな木において根から各葉へ至るキー列はソート

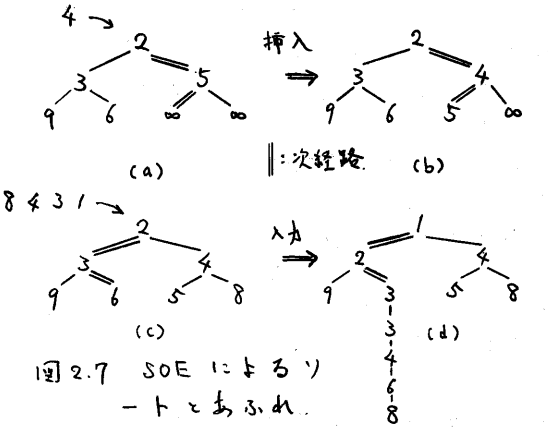
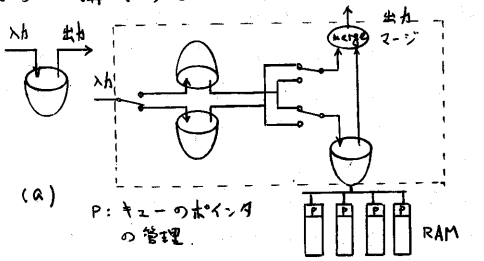
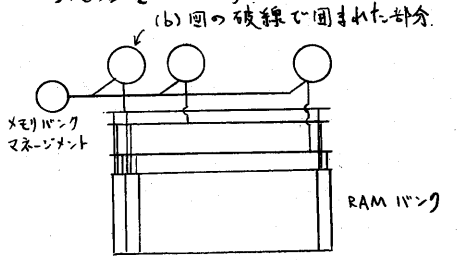


図2.7 SOEによるソートとあふれ

されている．このあふれの部分をRAMに記憶し，SOEの出力時に(i+1)番目のノードに空孔が生じたとき，あふれ部の上から順に，SOEへと送ってやると，あふれ部が一緒にソート出力することができる．昇順にソートされたブロック毎にiを変化させれば，ヒープの最下端のすべてのノードの下にあふれ部を設けることができる．今SOEを図2.8(a)のように表わすと，以上のことを用いて，大容量ソータを(b)のように構成することができる．RAMは



(b)大容量ソータ



(c) RAMバンクの共用

図2.8 大容量ソータとソータバンク

(1). Y. Tanaka et al. IFIP Congress 80 pp.427-432.

多バンク構成でもよいが，同時に複数バンクをアクセスすることはないので1バンクでもよい．多バンク構成の場合には，各バンクの割り当てを管理することにより，多バンクのRAMモジュールを複数のソータで共用し，ソータバンクを構成することができる（図2.8(c)）．

2.5. 処理方式

符号化データベースの処理は，

1. タプルセレクタ (TS)
2. タプルメーカ (TM)
3. アグリゲートオペレータ (AO)

の3つのモジュールで行う．TS, TM AOは図2.9のように結合されている．

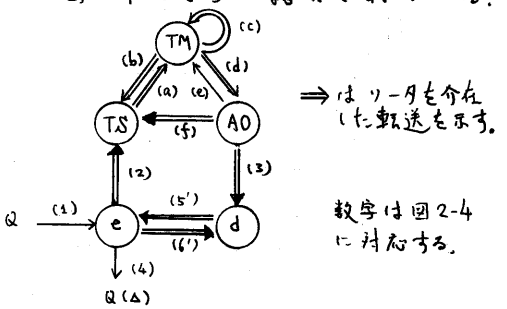
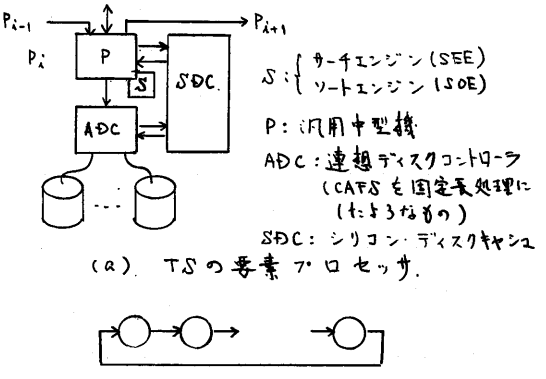


図2.9. 抽象アーキテクチャ

関係 $R(A_1, A_2, \cdots, A_n)$ の各タプルにはタプル番号を定義することができるので，これを@Rで表し，Rがあたかも@Rという属性をもつかのように見做し，$R(@R, A_1, A_2, \cdots, A_n)$ と表すことにする．Rは符号化されているとする．

TSは，各関係毎に，その関係のread-onlyでない属性毎に，$R(@R, A_1, \cdots, A_n)$ の A_i と@Rへのプロジェクション $R[A_i, @R]$ を A_i についてソートしたファイル $R[\uparrow A_i, @R]$ を持つ．TSが多重プロセッサ化される場合には，各 $R[\uparrow A_i, @R]$ 型のファイルは，A_i がnumber型やdate型の場合には，A_i の値をレンジ分けし，レンジ値のモジュラスの値でプロセッサに分配する．それ以外は，A_i の値のモジュラスでプロセッサに分配する．プロセッサ内でさらに $R[\uparrow A_i, @R]$ を分割する場合は，A_i の値の順序で分割する．各プロセッサは，図2.10に示すような構成で，プロセッサ間は同図(b)のように結合される．TSでは，セレクションを行って，該当するタプルのタプル番号をTMへ送ったり，等号によるジョインを各プロセッサ毎に独立に処理し，ジョインされるタプル対のタプル番号対の集合をTMに送る．セグメンテーションと符号化の仕方により，等号ジョインはプロセッサ毎に独立に処理することができる．θ-ジョインは，図2.10(b)の結合ネットワークを用いて処理する．TSからTMへの転送はソータを介在して行われる．多重プロセッサの場合はSDネットワークを介する．



(a). TSの要素プロセッサ.

(b). TSの要素プロセッサ間の結合.

図2.10. タプルセレクタの要素プロセッサ

$R[A_i = B_j]S[B_k = C_\ell]T$ のような演算ではTSで $((R[\uparrow A_i, @R])[A_i = B_j](S[\uparrow B_j, @S]))[@S, @R]$ と，同様な(@S, @T)の対とが求められ，両方とも@SにつきソートするようにSDネットワークに転送することにより，この3関係のジョインで残る(@S, @R, @T)の3項組を，TMの各プロセッサ中に求めることができる．

TMは，各関係Rをタプル番号のモジュラスで要素プロセッサに分割した

ファイルを，タプル番号の順序でセグメント化して持っている．各要素プロセッサは，図2.10の(a)と同様であるが図(b)の結合ネットワークは持たない．

R(↓Ai), S(↓Bi) に対して，

((R[Ai='v']) [Aj=Bk] S) [Al, Bm]

のような処理では，符号器eでvがコード化され，((R[↑Ai, @R]) [Ai='v'])[↑@R] がTMへ送られ，TMでは，この集合をXとして，各プロセッサで，Y=((R[↑@R, Aj]) [@R=@R] X)[Aj] をサーチエンジンを用いて求め，これをAjでソートしてTSに送る．TSの各プロセッサでは，Z=((R[↑Aj, @R][Aj=Aj]Y)[Aj=Bk] S[↑Bk, @S]) [@R, @S] を求め，@Rでソートして TMに送る．TMでは，各プロセッサ毎に独立に，((R[↑@R, Al]) [@R=@R]Z) [@S, Al] を求め，これをWとして，転送経路(c)を用いて，@Sでソートして TMに送る．各プロセッサは独立に ((S[↑@S, Bm]) [@S=@S] W) [Al, Bm] をサーチエンジンを用いて処理する．結果は，出力指定に従って，AlかBmのいずれか，指定がない場合は任意に決め，この属性についてソートしてAOに送る．

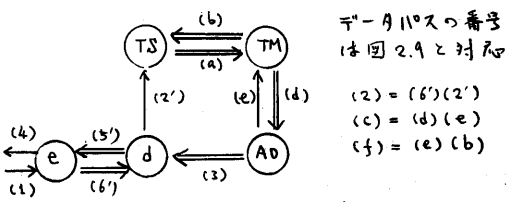
AOはグループ化しての統計演算や，重複タプルの除去などを行う．これがプロセッサ毎に独立の多重プロセッシングで処理できる．AOの要素プロセッサは，サーチエンジンとソートエンジンを持った汎用コンピュータである．

(e), (f) のデータパスは，結果として得られた関係の登録や，更新処理に用いる．

結果として得られた関係の各タプルには，AOの各プロセッサ毎に，プロセッサ番号とタプルの順番からなるタプルidがつけられ，各属性毎にバラバラにし，上の例では，Alとタプルid，Bmとタプルidの対をAl, Bmの値でソートして，復号器dに送る．復号器では，各符号をこれに対応する生データの入っている物理アドレスに変換し，符号器に転送し，符号器では，各属性毎に，生データとタプルidの対が得られる．符号器から，これらの対をタプルidの順序で1個づつ取り出し，タプルを構成することにより結果が得られる．

符号器と復号器が図2.10(a)のような要素プロセッサで構成され，この2つと，TS, TMの各プロセッサでは，検索要求にしたがって，セグメント・ファイルの先読みがなされ，シリコンメモリで構成される大容量(100～500MB)のディスク・キャッシュにステージングされるものとする．先読みと，ステージングがレディーになることをうけての処理のアクティヴェートは，データフロー制御で行う．処理の単位は，セグメントであり，これはオブジェクトと見做して管理と処理がなされる．

図2.9で2重矢印の部分にはSDネットワークが用いられるが，SDネットワークを少くするために，図2.9のかわりに図2.11でもよいと考える．



← は i番目のプロセッサをi番目に結ぶバス結合が，(4), (1)ではユニバスを表す．

図2.11. 改良した抽象アーキテクチャ

3. おわりに

本マシンは，対象データベースの規模に応じて，各モジュール毎のプロセッサの多重度，SDネットワークの多重度を自由に設定できる．制御方式の詳細は別稿に譲る．

謝辞 通産省第5世代コンピュータDDG委員の方々の議論に感謝します