資料 6

第５世代の電子計算機に関する調査研究報告書

―― データベースマシン・要求工学技術 ――

昭 和 55 年 3 月

JIPDEC

財団法人 日本情報処理開発協会

この報告書は，日本自転車振興会から競輪収益の一部である機械工業振興資金の補助を受けて昭和５４年度に実施した「第５世代の電子計算機に関する調査研究」の成果をとりまとめたものであります。

目　　次

第 Ⅲ 部　要求工学技術の研究

第Ⅰ部　データベース・マシーン

1. 序　論

近年データ・ユーティリティの概念が定着化しつつある（図1.1）。データ・ユーティリティとは集中，統合したデータベース・システムであり，大量のオンライン・ユーザに対して共有化したデータに対する並列的アクセスを許すものである。この原始的形式としては，銀行業務，座席予約システム等がある。また他の例としては，ＭＩＴとＩＢＭの共同開発になるNEEMIS（New England Energy Management Information System）がある。このシステムでは，各ユーザのプログラムは各仮想マシーン（ＶＭ）上にあり，別のＶＭ上のDBMSによりデータベースを共有している。これらの経験と将来の予測によれば，データベース・システムはピークロード時に1,000,000要求／秒の能力が要求される。この値は現在の高パーフォーマンス・データベース・システムの処理能力である10〜100要求／秒をはるかに越えるものである。また扱わねばならないデータの数も現在のそれ（現在でも10^{12}バイトを越す）よりはるかに大きいものになることが当然予測される。（一説では10^{18}バイトがコスト・パーフォーマンス上可能である。）このような大規模，高速のデータベース・システムをソフトウエア・データベース・システムとして実現する場合，信頼性，パーフォーマンス，セキュリティ等に大きな問題が生じる。この解決の方向として現在注目を浴びているのがデータベース・マシーンの研究である。データベース・マシーンとは，現在のＤＢＭＳの基本機能を支援する目的専用のコンピュータ・ハードウェアである。

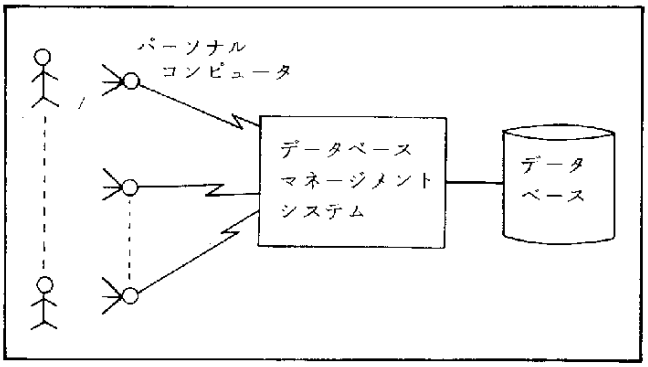


図1.1　データ・ユーティリティ

1.1 ソフトウェア・データベース・システムの問題点

(1) 複雑なデータ構造

ソフトウェア・データベース・システムの複雑さの最大の原因は，複雑なデータ構造（例えば，"B－木"等）が必要な点に起因する。DBMSでは，問合せで与えられるシンボリック名をメモリ番地へマップする機能が中心になる。つまり一般にまずシンボリック名を用いてディレクトリへアクセスし，それからディレクトリの情報により，二次メモリの探索すべきブロックを見つけ，最後にブロックを探索することにより目的のメモリ番地が得られる。シンボリック名からメモリ番地を得るためには，以上で分かる様に二次メモリとディレクトリのアクセスに対する効率の良いデータ構造が不可欠となる。update,delete等の操作に対しこれらの複雑なデータ構造を正しく保持することが，計算時間のかかる主要な原因であり，ソフトウェアが複雑になる原因である。

(2) 機能の多様性

現在のソフトウェア，DBMSは多くの機能的に異なった部分から成立している（表1.1）。これらを同一のハードウェア上で実現すると，多くの部分にボトルネックが生じ，バランスの良いシステムにすることは困難である。特に現在のvon Neumann型計算機は，数値計算や単純なデータ処理を主目的として最適化されており，例えば(1)で述べた複雑なデータ構造の処理，二次メモリとのI／O等の目的には必ずしも適したものとは言えない。

表1.1 DBMSの機能

1.	query parsing
2.	directory access
3.	directory processing
4.	data retrieval , update
5.	security

(3) セキュリティ

現在の殆どのＤＢＭＳではセキュリティのスペシフィケーションはadd on で行われているが，これは，パーフォーマンスのボトルネックになるだけではなく，統一的にセキュリティが扱いにくいため信頼性の重要な障害となる。ＱＢＥ[2] にみられるように，セキュリティを問合せ言語で指定することが最も強力な方法であるが，この場合(1)で述べた複雑なデータ構造を数多く辿る必要が生じ，パーフォーマンスに大きな影響を与える。

(4) 信頼性

ＤＢＭＳは，今後ますます大規模・複雑化する事は明らかである。また多様な応用の発生あるいは応用の変化に伴う再構成の問題を避けて通ることはできない。このような状況下では，大規模ソフトウェアの設計技術の進歩による信頼性の向上と共に，ＤＢＭＳの基本機能のハードウェア化によるソフトウェアの単純化が当然必要となるであろう。

現在のデータベース・システムではこの他にも重要な問題点が数多く存在していることは良く知られている。（例えばmulti−view support ，分散システム，non alph−numeric データベース，データベース生成システム等） しかし，現在のデータベース・マシーンの開発，研究は殆ど全て上記(1)〜(4)の問題点の解決を意図したものである。

1.2 データベース・マシーンの基本概念

前述のソフトウェア・データベース・システムの問題点を解決するために，現在の von Neumann 型計算機と本質的に異ったシステム・アーキテクチャが必要となる。データベース・マシーンの概念構成図は図1.2に示されている。この概念の基本要素を以下に述べる。

(1) フロント・バック構成

シークエンシャル・コンピュータをフロント・エンドに，データベース

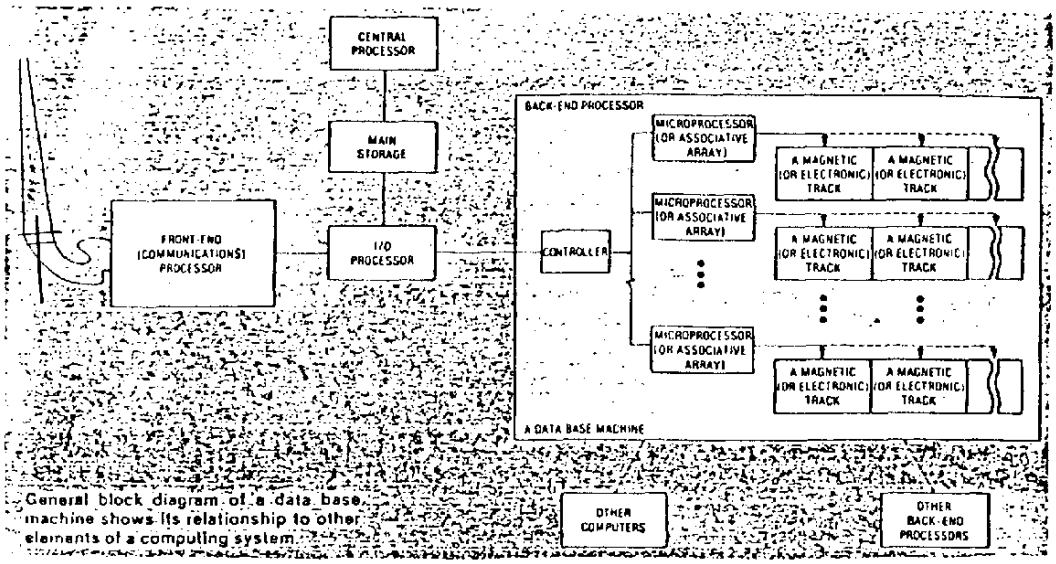


図 1. 2　データベース・マシーンの概念構成図

・マシーンをバック・エンドに置くフロント・バックの構成である。バック・エンド・データベース・マシーンに現在のＤＢＭＳの機能のどの部分までを持たせるかは，これまで提案されている各システムで各々異なる。しかし，いずれのシステムに於てもフロント・エンドの von Neumann 型計算機の仕事の中から，二次メモリに対するアクセス，1. 1 (1)で述べた複雑なデータ構造の処理等を除くことにより，ＣＰＵ，メモリサイクル，チァネル容量等に対するデータベース管理の負担が軽減される。

(2)　コンテント・アドレスと並行処理

シンボリック名から二次メモリのアドレスを決定する問題が現在のＤＢ－ＭＳの複雑さの最大の原因であるので，シンボリック名から，直接二次メモリの内容へアクセス可能なコンテント・アドレスの機構が，データベース・マシーンのアーキテクチァの基本となる。当然コンテント・アドレスの機構の結果，データベース・システムで 1. 1 (1)で述べた複雑なデータ構造が不必要なものになる。コンテント・アドレスでは，コンテントに対する探索単位に従い，二次メモリを適当に分割することにより，並行的に探索が可能であり，パーフォーマンスが飛躍的に向上する。

(3) ＤＢＭＳの高次機能のプリミティブ化

ＤＢＭＳの基本機能をデータベース・マシーンのプリミティブとすることは，パーフォーマンスの面と，信頼性の面で重要である。例えば，リレーショナル・データベースに於ける selection，projection，natural join 等，現在の von Neumann 型計算機でソフトウェアとして実現しているものを，データベース・マシーンのプリミティブとしてハードウェア機能に持たす試みは多い。

(4) 低次のマス・メモリと高次のデータベース処理間のバランス

ＤＢＭＳには多くの機能が必要であることは述べたが，各機能の専用ハードウェア化により，ボトルネックの無い，バランスのとれたシステムにすることが重要である。

2. 研究プロジェクト

2.1 データベース・マシーン研究の分類

現在までの研究プロジェクトの流れは次の様に大別できるであろう。

(1) ファームウェア・アプローチ

現在の通常の計算機の命令体系が，ＤＢＭＳに必ずしも適していないので，ファームウェアにより命令体系の強化を図ろうとするものである。この例としてはＬＩＳＴＡＲ（Linoln Information STorage and Associative Retrieval），Honeywell Ｈ６０／６４，Microdate ＲＥＡＬＩＴＹシステム等があるが，Ｉ／Ｏの含まれないファームウェアエミュレーションではＣＰＵのパーフォーマンスが高々数倍程度上げるのが精々である。

(2) セルラーロジック・アプローチ

セルラーロジックによりコンテント・アドレスを行うものである。例えば，固定ヘッドディスクの読み，書き機構に，ロジックを附随させ，セル相互間の通信形式を与えることにより，比較的容量の大きい，しかしやや速度の遅いコンテント・アドレス装置が設計される。この方式では大体容量が10^8バイト程度で，トラックの数だけ処理要素が必要となるため，コストが高くなるという傾向にある。

(3) 連想プロセサ・アプローチ

コンテント・アドレスと並行処理の観点から，連想プロセサの上でのデータベース・システムの実現は１９７０年頃から注目されていた。しかしながら，モノリシックの連想メモリは，そのコストと容量に対する限界から，この上にデータベース・システムを構成することはかなり悲観的な状態となっている。現在では，大容量のディスクに対するステージァとして連想アレイを用いる方式が有望視されている。この場合探索に比し，連想

アレイへの入出力の負荷が大きくなるため，入出力ボトルネックを解消するための研究が発展することが望まれる。

(4) 可動ヘッド・ディスク・アプローチ

可動ヘッド・ディスクはコストが安いため，大量情報の蓄積装置として，今後長期間利用され続けるであろう。そこでセルのサイズを飛躍的に増大し，セル間の通信負担を軽減するため，１シリンダの全てのトラックを並列的に読み出す構成法の研究が続けられている。

この方式では"tracks-in-parallel"読み出し中にコンテント・アドレスを行う。シリンダのサイズは非常に大きいものではあるが，データベースの構造的情報（例えばインデックス）を利用することにより，シリンダに対する遅いアクセスの数を最小化しようという試みも続けられている。この考え方は，さらにバック・エンド・コンピュータとしてのデータベース・マシーンをトータル・システムとしてみた場合，データベースの各機能を各専用ハードウェアで実現することにより，バランスのとれた，ボトルネックの少ないシステムにしようという設計方針につながってゆく。

2.2 新しい技術のサポート

前述した各アプローチ共，いずれも未だ解決されていない問題点が多いため，現在までのところ商用データベース・マシーンが実現されていない。しかしながら，近年新技術としてＣＣＤデバイス，磁気バブル・メモリ，ＥＢＡＭ（Electron-Beam-Addressed-Memories）等の技術革新が目覚ましく，大体実用段階に入ってきたと思われる。これらが段々と固定ヘッド・ディスクに置き換ってゆく事は充分予想される。これらはいろいろなアクセス・モードとアクセス・タイムを与えるので，各種のデータベース・マネージメントの仕事に有効であると思われる。実際，現在固定ヘッド・ディスクで始められたセルラー・ロジックが，上記の新しいデバイスの上で実験的には実現されつつある。

また，大容量の可動ヘッド・ディスクに対しても，シリンダ単位でのコンテント・アドレスはそれ程困難なものではないであろう。現在マイクロプロセサの技術の発達とコストの低下が著しいので，マイクロプロセサと通常のハードウェアの組合せで，シリンダ単位でのコンテスト・アドレスを実現することは技術的にも，コスト的にもかなり現実的問題であると思われる。

2.3 セルラー・ロジック装置

セルラー・ロジック装置の概念図を図2.1に示した。一般にセルラー・ロジック装置は通常の計算機によって制御される。このホスト計算機は高次のデータ操作要求をセルラー・ロジック装置のコマンドに変換する。セルラー・ロジック装置の基本構成要素は図2.1に示されているM，P，C，IOの4つである。

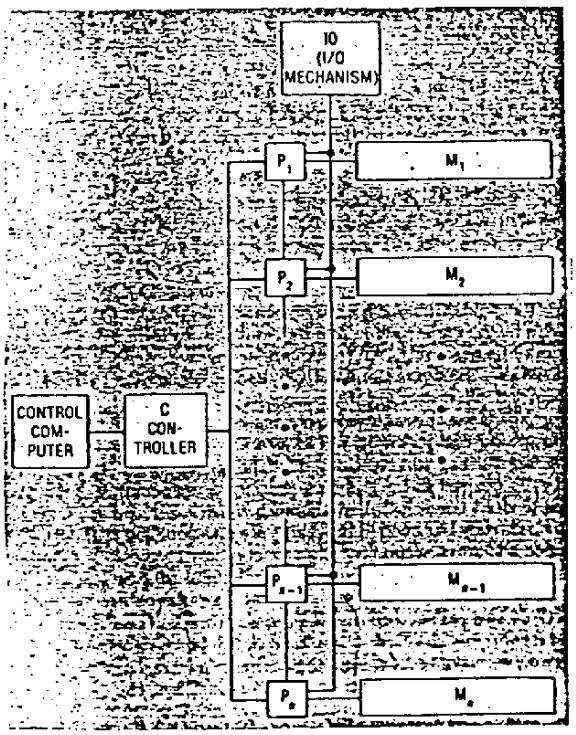


図2.1　セルラー・ロジック装置のアーキテクチャ

1）M

Mはメモリ要素M_iの集合である。M_iは次の3つ組で定義される。

M_i＝(TG, DT, SR)

TGは，データ項目に対するタグ用の領域であり，このタグによって各データ・タイプ（例えば，文字型，数値型，区切り語，ポインタ）を識別する。あるいは単にデータ

項目の区切りとして用いられる。ＤＴは実際のデータをしまう領域である。ＳＲはデータベース探索の結果ないしは中間結果をしまう領域であり，Ｉ／Ｏのためと，以後の探索のためなどに使われる。ＴＧやＤＴをストアーするメモリとしては，ディスク，ドラム，ＣＣＤシフトレジスタ，磁気バブルメモリ等回転型メモリが用いられる。このメモリは回転型であるので，走査に時間がかかる。ＳＲはＤＴと同様に回転型メモリにストアされる場合もあるが，高速化のためにランダム・アクセス・メモリ上におかれることもある。

2） P

Pは処理要素P_iの集合である。P_iはM_iに附随したロジックであり，各M_iの内容を処理する。P_i相互間の結合は各システムで異なっている。P_iの各要素を以下に列挙する。

a） レジスタ群とバッファ領域（ＣＰ）

ここにはテンプレートとなる値（即ち comparand ）を保持するレジスタや特定のフィールドを指定するためのマスクレジスタ（ＭＫ）等のコントロール用のレジスタと，Mから取り出された対象データが保持されるデータ用レジスタやバッファ領域の様なものがある。

b） 演算要素

比較演算要素：比較演算子（＝，＞，≧，＜，≦等）の計算を行う。

論理演算要素：論理演算子（AND, OR, NOT等）の計算を行う。

統計演算要素：AVERAGE, SUM, MAX, MIN等の処理を行う。

c） プリミティブ・データベース操作

データベース操作の基本である search , insert , delete , update の演算を行う。

現在のセルラー・ロジック装置の処理能力は各システムで非常に異なっている。ここでデータベースの基本操作の１つである search 操作に関して，一般的方法を簡単に述べる。

1. 全 comparand に目的データ値をセットする。
2. 対象となるフィールドに探索を限定するためのマスクをセットする。
3. M_i 中のデータが順次 CP_i 中のレジスタにロードされる。
4. 全ての P_i がマッチ操作を並行的に行う。
5. M_i に対する走査が全て終了した後，マッチ条件を満足したデータ項目あるいはレコードに対応する SR_i の領域がマークされる。
6. 単なるコンテント探索の場合，SR_i のマークされているデータ項目あるいはレコードが出力装置へ転送される。
7. コンテクスト探索の場合，SR_i のマークされているデータ項目のみが以降の探索の対象となる。即ち SR_i 領域の存在の結果，複雑な Boolean 探索が可能になる。

3) C

Cはコントローラであり，同一の命令を同時に全ての P_i で実行させる。(即ち一般的に，セルラー・ロジック装置はSIMD型－Single Instruction stream Multiple Data stream － である。) Cは命令を一つの Miから受け取る場合と，ホスト計算機から受け取る場合がある。

4) IO

入力は通常のシステムと殆ど同様に行われる。出力は通常のシステムより若干複雑で工夫が必要になる。探索結果が全ての M_i で同時に得られマークされるので，それを出力する順番を決定しなければならない。通常，P_i に優先順位を前もって与えておくか，P_i により走査された順番に出力される。

ホスト計算機のDBMSに関する仕事は以下の様なものである。

1.) ユーザとの通信を行う。

2.) 高レベル問合せ言語をセルラー・ロジック装置のプリミティブへコンパイルする。

3.) コンパイル結果をセルラー・ロジックへ渡す。その場合セルラー・ロ

ジックの書式に応じて，オペランドとしてデータが附随する。

4.) データベースの生成のためのデータを転送する。

5.) セルラー・ロジックのプログラムのエントリ・キューをスケジューリングする。

6.) データベースのセキュリティとインテグリティの管理をする。

7.) ディレクトリの管理を行なう。

＜実　例＞

(1) CASSM （Content Addressable Segment Sequential Memory）

回転型蓄積装置の読み書きヘッドにロジックを附随する試みは，1970年位から注目を浴びてきたが，CASSMはデータベースを指向した最初のものである。これは，現在のデータベースにおける主要な三つのデータモデルであるネットワークモデル，階層モデル，リレーショナルモデルのいずれをもサポート可能である。CASSMはスタンド・アローンのデータベース・コンピュータとして，固定ヘッド・フロッピーディスクで実現されている。また，CASDALという高レベルのデータ操作言語を持っている。

1) データの構成

セグメント・シークエンシャル蓄積装置は，セルの集まりから成る。各セルは全体のシークエンシャル記憶の固定長のセグメントを含んでおり，固定ヘッドディスクのディスク・トラックと処理装置から構成されている。データはディスク・トラックに沿って，ビット順次・語順次型に貯えられる。データ構成は可変長ブロック型であり，各ブロックは，40ビット語の列として扱われる。一語の構成は以下の通りである。

32ビット……デリミタ，属性・値対，文字列等を表現するのに用いる。

8 ビット……タグ及びマークビットとして用いられる。

CASSMは，各ファイルの構造を階層木として表現する。リレーショ

ナル・テーブルの場合，２レベルの木として表現される。１－レベルはリレーション全体に対応し，デリミタ語にリレーション名とレベル番号“１”を与えることによって表現される。タプルはこのデリミタの直後にしまわれる。図２.２に対応するタプルの表現例が図２.３に例示されている。各タプルの先頭にはデリミタ語があり，そこには，リレーション名とレベル番号“２”が書かれる。

SUPPLY

S#	P#	Q
S1	P1	5
S1	P2	6
S3	P1	6

SUPPLIER

S#	SN	SLOC
S1	ABC Co.	—
S2	XY Co.	NY
S3	JK Co.	NY

PART

P#	PD
P1	CAM
P2	GEAR

図２.２　部品表のリレーショナル・データベースの例

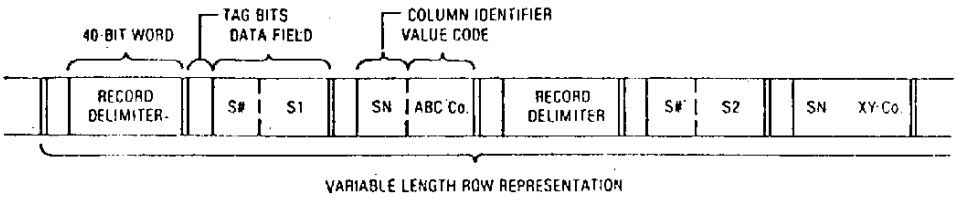


図２.３　ＣＡＳＳＭのデータ構成

このデリミタ語の残りの８ビットは，ＴＧとして利用され，問合せの対象に対応するＳビット，探索条件に対応するＱビットと，探索結果を保持するＢスタックと呼ばれる６ビットのスタックから成る。

２）処理要素間の結合方式

P_i がその前後の P_{i-1} ，P_{i+1} と直接情報を交換する方式である。CASSM では順次ファイルがセグメント化され連続的にディスク上

に貯えられるので，１タプルがメモリ要素M_iとM_{i+1}に跨って貯えられることがおこる。この様な場合，M_i中のデータについて，タプルの何番目の項目か等の情報がM_i中にはない場合が生じる。この時には，その情報をM_{i+1}，M_{i+2}……と順に調べてゆく。そのためにMi中にその情報が含まれているか，そのテーブルが続くか等の補助的レジスターを持っている。

3）ＳＲの構造

CASSMは各タプルをマークするために補助的なストーレッジとしての，ＲＡＭビットアレイを各P_i毎に持っている。CASSMが同時に数多くのタプルを選択した場合には，出力アルバイタはその中の一個のものを出力する。残りのものは各回転毎に次々と出力されるが，そのために各タプルは出力用にマークされなければならない。この目的のために，ＲＡＭビットアレイを用いる。各タプルに対し，ＲＡＭの１ビットが対応する。このＲＡＭの内容に従って，各トラックのマークのセットが行

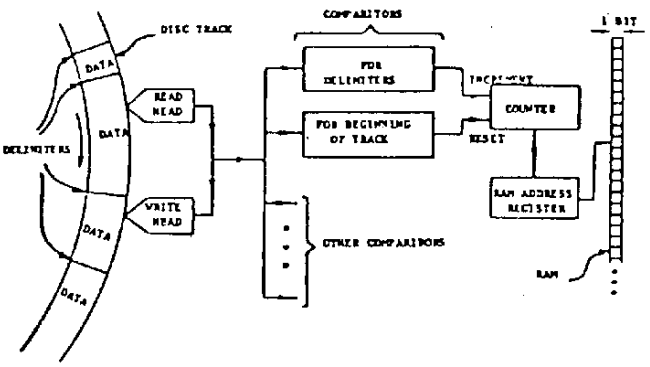


図 2.4　データ項目をマークビットに写像するためのハードウェア

なわれる。また，このＲＡＭは，コンテクスト探索における中間結果のストア・リレーショナル・アルジェブラにおけるnatural join に対するカプリング・インデックスの生成に重要な役割を担っている。RAM

アドレスの概念的構成図は図2.4に示してある。

4）データベース操作例

a） selection

「P＃＝P1かつQ＞5を満足するSUPPLY リレーションのタプルをセレクトする」

CASSMのプリミティブ演算単位で表わすと以下のようになる。

1. 1回目のディスク回転

SUPPLY リレーションに属する全てのタプルに対するデリミタのSビットをセットする。

2. 2回目のディスク回転

上記タプルに対する全てのQビットをセットする。

3. 3回目のディスク回転

Qビットの立っている各タプルに対し，＜P＃，P1＞のマッチを調べる。マッチしたタプルに対し，デリミタ中のBスタックにマークをプッシュする。

4. 4回目のディスク回転

上記探索結果に基づき，＜Q，＞5＞の探索を行ない，マッチしたタプルに対するデリミタ中のBスタックのトップと1とのORがとられ，結果が1ならコレクションビットがセットされる。

5. その後のディスク回転

コレクションビットに従い結果を出力する。（同時に多数のM_iで出力要求がある場合，各回転毎に一つの結果が出力される。）

b） natural join

「SUPPLYとPARTをP＃についてジョインする」

SUPPLY リレーションの選ばれたタプルに対し，P＃欄の各値をRAMのアドレスに変換して，そこのRAMビットをマークする。このためには，一般的には多くの回転が必要となる。例えば， SUPPLY

のＳ１タプルとＳ３タプルが別のM_iに属している時には，先ずＳ１タプルからＰ１に対するＲＡＭアドレスが決められ，Ｓ３は待たされる。続いて，ＰＡＲＴリレーションに関してＲＡＭの内容を参照しながらジョインを行なう。

5）ガーベージ・コレクション

CASSMでは，読出しヘッドから入ってきたデータは，ＲＶＬという待ち行列に入り，待ち行列から出てきたものが，書き込みヘッドで書かれるようになっている。したがって，削除されたデータはＲＶＬに入れないようにし，追加はＲＶＬに単に入れることにより，自動的にガーベージ・コレクションが行なわれる。

6）CASSMの問題点

1. データベース全体に比べ，メモリ要素の大きさが小さいためにコストが大きくなる。本質的に並行処理の程度を上げようとするとメモリ要素の大きさが小さくなるので，コストパフォーマンスのトレードオフをどこにおくかが問題となる。
2. P_i 間の相互結合形式が直接前後の処理要素間だけであるため，離れたデータ間のコミュニケーションが困難になる面がある。また，セグメント順次ファイルにより，記憶空間の利用率を向上させているので，記憶空間の利用率と並行処理の間のトレードオフが難しい。
3. ＳＲとしてＲＡＭビットアレイを利用する事により，ブール演算操作に対する能力を強めているが，各レコードあるいはフィールドの出現順とＲＡＭビットアレイの番地を対応させているために，柔軟性を失っている。例えば，この結果としてリレーショナルモデルにおける projection 操作がほとんど不可能になっている。

つまり，projection 操作をCASSMで行なおうとすると，重複したタプルが生成されてもそれを排除する手段がない。また，natural join においても，カプリング・インデックスの生成に数多くのディ

スクの回転が必要となる。この点に関し，ＣＡＦＳシステムでは，データの値に対しＲＡＭアドレスを対応させているため，join 演算，projection 演算が比較的容易になっている。

4. データベースの探索に関し，一般にディスクの回転が数多く必要なために，パフォーマンスの飛躍的向上が望みにくい。特に，探索条件を満足するデータが数多くある場合，その出力のためにパフォーマンスが極端に低下することが予想される。

(2) ＲＡＰ（Relational Associative Processor）

ＲＡＰプロジェクトはTronto 大学で１９７５年に開始され，１９７６年にＲＡＰ．１と呼ばれるプロトタイプが作製された。その後設計の変更が行われ，１９７７年にＲＡＰ．２システムが作られた。いずれもリレーショナル・データベースをサポートするスタンドアローンなデータベースマシーンであり，CASSM と同様なプログラム言語インタフェイスを持っている。

1）ＲＡＰ．１

a）M_i がディスク１トラックと１０２４ビットのＣＣＤバッファから成る。その構成図は図2.5に示した。データは，CASSM と同様にディスクのトラックにそって，ビット順次，語順次型にしまわれる。ＲＡＰは一つのリレーションのタプルに対しては固定長表現を用いている。この長さはリレーションの間では異なるが，一つのリレーションの中では全てのタプルに対しては同じである。また，一つのトラックにはただ一つのリレーションしか含まれない。一つのトラックの中では，一つのタプルは一つのブロックにしまわれる。各ブロックの最後はデリミタで示されている。ＲＡＰのトラック書式は図2.6に例示されている。一つのトラックの先頭には，特別なトラックマーカーがあり，その後に二つのヘッダーブロックが続いている。最初のブロックにはそのトラックのリレーション名が書かれ，二番目のヘッダーには属性名のリス

トがタプル表現に現われる順番でしまわれている。このヘッダの後にトラックのリレーションに含まれる各タプルが続く。全ての名前と値は，３２，１６，８ビットで表現されるが，その前にある２ビットによりその長さが指定される。

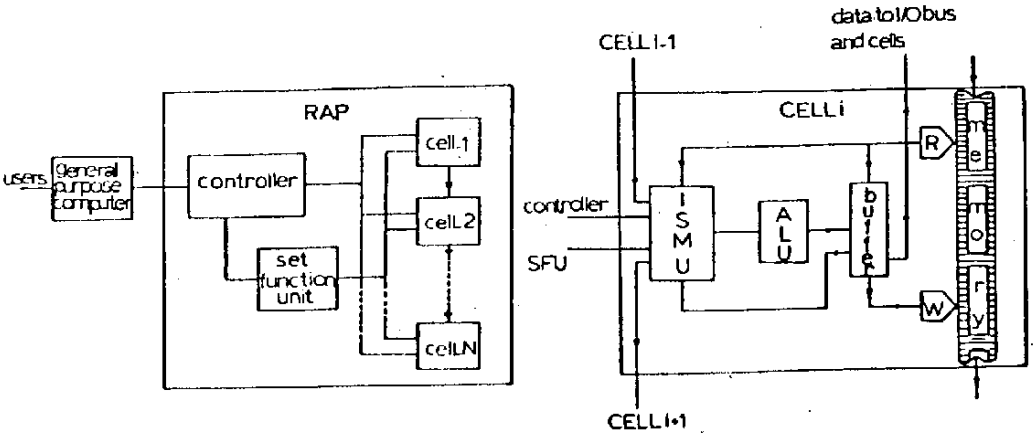
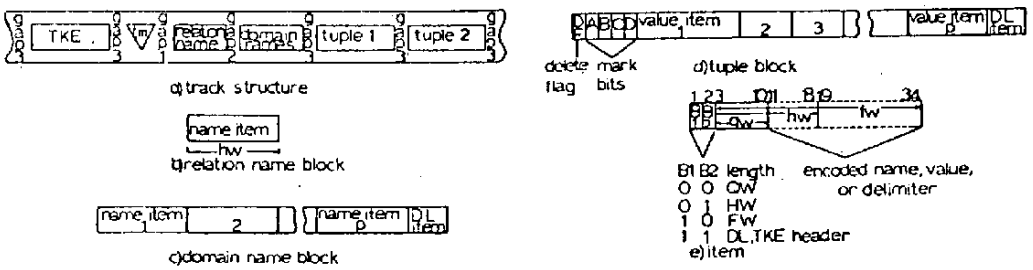


図２.５　ＲＡＰの概観



トラック書式　　タプル書式

図２.６　ＲＡＰのトラック書式とタプル書式

各タプルに対し，削除用ビットと探索中間結果の保持及び出力用の４ビットから成るマークビットが附随している。ＲＡＰはCASSMと同様，出力アルバイタを用いて探索結果の出力を行なっているがマークビットにより次に出力すべきタプルを表わしている。削除用ビットは

ハードウェアの garbage collection の対象となる。

b）処理装置間の結合方式は，一つのリレーションを表わしている多くのメモリ要素が連続的位置を取らなくともよいので P_i 間は全ての対を結合するネットワーク構造を取っている。

c）データベース操作例を以下に示す。

Selection

1. 探索条件を全てのトラックに同時に送る。例えば探索条件はマークビットの組合せで表現される。探索条件を満足するものは出力チャンネルが空き次第読み出される。
2. バッファは一つのタプル全体を保持するように設計されているが，タプルがこのバッファを通る間に附随するロジックによって，selection 条件が計算される。このロジックはK個のコンパレーターが並列的に作用するように設計されているので，K個までの Boolean selection の評価が可能である。K個の評価の結果からワイヤーロジックにより Boolean 演算により一つの結果を得る。
3. コンテクスト探索では上の selection の結果得られたタプルに対しマークが行われる。このマークは各タプルの先頭部のマークビットに書き込むことにより行われる。
4. 3.で得られたマックビットの情報を参照してそれ以降の selection が行われる。

以上の 1. 2. 3.までのステップはディスクの一回転内に行うことができる。

Join

Join 演算に対してはハードウェアでの実現方法として CROSS__MARK, CRS__COND__MARK, GET__FIRST__MARK 等が用意されている。一般に Join の演算に必要なディスクの回転数は，1＋A／Kとなる。ここにKはコンパレータの数であり，Aはもとの表のタプルの数である。

d）ＲＡＰ．１の問題点

1. CASSMの場合と同様，製作コストが大きくなる。

2. P_i 間の相互結合方式が複雑となり，一つの P_i に対してその他の全てを駆動させるためのドライバーのコストが高くなったり，通信線の長さが大きくなったり，信頼性と並列処理能力で限界が生ずる等の欠点がある。その他，基本的にはCASSMと同様な問題点がある。

1.2.の問題点とＲＡＰ．１でハードワイアで実現したコントローラに，より柔軟性を持たす目的でＲＡＰ．２の作製が行われた。

2）ＲＡＰ．２

ＲＡＰ．１とＲＡＰ．２の違い

1. ＲＡＰ．２ではコントローラがマイクロコンピュータで実現されている。

2. ＲＡＰ．１におけるディスクのトラックとＣＣＤバッファによるメモリ要素の実現方法をＣＣＤのみに変更した。

ＲＡＰ．２は１９７７年に二つのセルを持つプロトタイプが作製された。各セルは 10^6 ビットのＣＣＤトラックであり，コントローラはＰＤＰ１１／１０で実現された。更にＲＡＰ．１のｄ）の2.で述べた理由により，各処理要素間の結合を完全に無くすことにより各処理要素が独立に稼動する方法への変更を行なった。即ち各セル毎に１Ｋ語のＲＡＭバッファをＩ／Ｏ用に用意し，これがコントローラのＰＤＰ１１／１０のメモリにマップされたＩ／Ｏ領域としてみなすことにより，高速ポーリング方式で探索結果を読み込み，同期は，ＰＤＰ１１／１０でとることにしている。データ構成法，プリミティブなデータベース操作には大きな違いはない。結果的にＲＡＰ．２は現在の通常のコンピュータシステムに独立な周辺機器としてセルがついたのと類似な構造になっている。

(3) ＣＡＦＳ（Content Addressable File Store）

ＩＣＬ（International Computers Limited）社で開発用に製作されたリレーショナルデータベース用のデータベースマシーンである。特にリレーショナル・アルジェブラにおける演算 selection，projection，join等の高速化をねらったものである。これは，logic per track方式のＣＡＳＳＭ，ＲＡＰとは異なり，１２ディスクチャンネルまでのマルチプレクス出力を並列的に処理する方式を取っている。その結果 logic per track方式が並列度は高いが一走査あたりの selection能力が低かったのに対して処理要素の能力を強力にすることを可能にしている。基本的な特徴はCASSMと同様なＲＡＭビットアレイを持つことにより，処理要素の能力（特に projection と join）を高めている。図2.7に基本構造を示す。

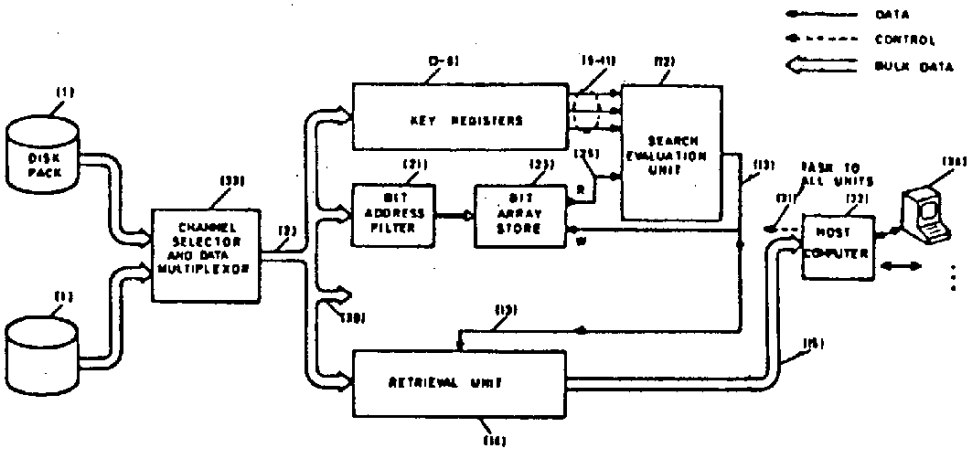


図2.7　ＣＡＦＳのビットアレーのアーキテクチャー

a）データベース操作

1.　join

図2.8のリレーションＳＰとＰＣを共通の属性であるPARTでjoinする例を考える。

問合せ：「Mullard によってＩＣＬに供給される部品のリストを求

めよ」

この場合，プリコンパイル時にカップリング・インデックス用のフィールド，Ｉパートが図2.8のように用意され，このインデックスがビットアレイの番地を示す。

SP	SUPPLIER	PART	PRICE	I PART
1	TEXAS	TRANSISTOR	9p	1
2	MULLARD	〃	〃	1
3	TEXAS	DIODE	4p	2
4	FERRANTI	〃	〃	2
5	ERIE	RESITOR	3p	3

PC	I PART	PART	CUSTOMER
1	1	TRANSISTOR	ICL
2	1	〃	CDC
3	2	DIODE	XYZ
4	2	〃	ABC
5	3	RESISTOR	ICL

図2.8　リレーションＳＰとＰＣ

a．ビットアレイＭをクリアする。

b．ＳＰの各タプル（SUPPLIER，PART，PRICE，I part）に対して

SUPPLIERが'MULLARD'と等しければ，M（I part）を1にsetする。

を実行する。

c．リレーションＰＣの各タプルに対し，もし，M（I part）が1で，かつ

CUSTOMERが'ICL'に等しければ，partをホストコンピュータに転送する。

2.　projectionの例

例　：　ＳＰをPARTとPRICEフィールドへprojectionする。

この場合も図2.9に示されるようにプリコンパイル時にインデックスI part，priceが必要である。I part，priceはjoinの例と同様

な意味を持つ操作は次の様になる。

1）ビットアレーMをクリアする。

2）全てのSP内のタプル（SUPPLIER，PART，PRICE，I part，price）に対して，

もし，M（I part，price）がセットされてなければ

ホストコンピュータにPART，PRICEを転送し，既にそのPART，PRICEの組が転送されたことを示すために，M（I part，price）をセットする。

SP

SUPPLIER	PART	PRICE	I PART,PRICE
TEXAS	TRANSISTOR	9p	1
MULLARD	〃	〃	1
TEXAS	DIODE	4p	2
FERRANTI	〃	〃	2
ERIE	RESISTOR	3p	3

図2.9　projectionの為にインディクスづけされたリレーションSP

b）問題点

1）インデックスを設けて，RAMビットアレイを指すことにより，join，projection の高速処理化を可能としている。特にこの方式ではデータの値によってRAMの番地を示しているのでprojection の結果duplicateしたタプルを除くのが簡単にできる。しかしインデックスを何らかの方法で事前に用意しなければならないのが問題である。

2）図2.7からも分かる様に，単純な比較演算の組合せでselect の条件が生成されるのでプリミティブな演算のレベルがかなり低く，joinの演算もホストでやる部分が多いので，ホストコンピュータやI／O

の負荷が大きい。

3）基本的に探索中心にできているので，適用範囲が制限されてしまう。

(4) その他

前述したセルラーロジック装置の他にも，幾つかの興味ある研究がなされている。

1）RARES

固定ヘッドディスクを想定したセルラーロジック装置であり，特に問合せの最適化の観点から，リレーショナルな問合せを並列的プロセッサに分配して実現させることが主要な点である。

2）Intelligent memory

sort，selection，updateを行うスタンドアローンマシーンでストレージとしてはＣＣＤ，バブルメモリを想定している。

3）Changマシーン

バブルメモリのmajor／minor　ループを想定したスタンドアローンデータベースコンピュータである。

上記のいずれも設計だけで終っており，インプリメンテーションは行われていない。

(5) まとめ

1. セルラーロジックでは，セルのサイズの取り方がコストパーフォーマンスの上で非常に重要な要素となる。
2. セル相互間の結合方式が同期の問題，柔軟性の問題で解決が難かしい。
3. 容量的に10^8バイト程度という限界があり速度的にも比較的遅いので，中，小規模のデータベース，あるいは大規模データベースのディレクトリファイルのハードウェア化に利用される可能性がある。
4. 今後従来の固定ヘッドディスクから，ＣＣＤ，バブルへとメモリディバイスが移行してゆくものと思われる。

2.4 連想プロセッサ

連想プロセッサのコンテント・アドレスと並行処理能力は，ＤＢＭＳのハードウェア化にとって重要な要素として早くから注目されていた。

(1) STARAN

ビット・スライス型（ビット順次・語平行の連想プロセッサで語の各ビット欄が平行的に処理される） STARANプロセッサの概念図を図2.10に示した。図2.11にアレイ，comparand レジスタ，マスクレジスタ，レスポンス・レジスタを示した。アレイはデータを含む蓄積装置であり，comparand レジスタはアーギュメントを含む。マスクレジスタは与えられた命令に対してアレイの各ビットを" enable "にするか" inhibit " にするかを決定する。レスポンス・レジスタは，探索結果を貯える。図2.11を例にして簡単な演算方式について述べる。データは name , department , 各生徒に対する degree program から成る。これらのデータは，アレイの中の前以って決められた固定長フィールドに位置している。ＩＯＲの学生名を求める問合せを考える。先づＩＯＲがcomparandにロードされ，マスクレジスタの department field に対応するビット列に"1"がセットされる。exact match 探索が行われ，matchしたレコードに対応するレスポンス・レジスタＹにビットがセットされる。次にBooleanオペレーションについて考える。問合せがＩＯＲの

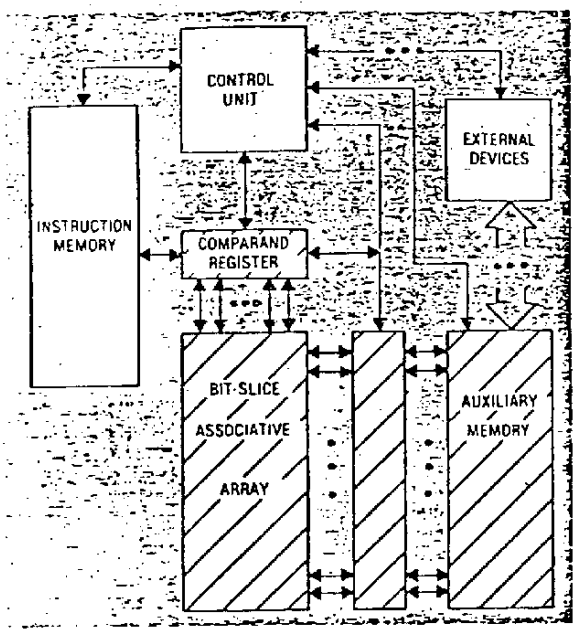


図2.10　STARANプロセッサの概念図

生徒でPh. D. program 中のものを探すことにする。前述の結果を貯えているYレジスタをXレジスタにセーブする。次に degree program フィールドの"Ph. D."値に対する exact matchを前述の方法で行う。結果がYレジスタに貯えられるので，XレジスタとＡＮＤをとることにより問合せに対する答が得られる。

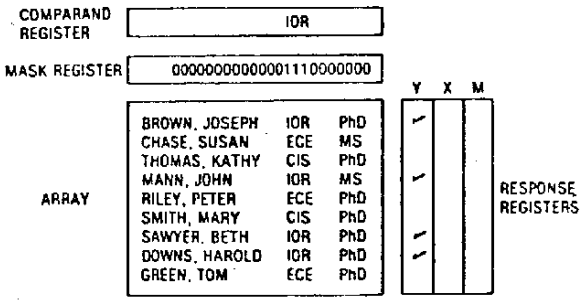


図2.11　連想メモリ

1）リサーチ・プロジェクト

a）１９７１年にDe Fiore等がＤＢＭＳを連想プロセサに適用する有利さを提唱した。ここでの彼の議論の中心は，階層データモデルを，associative normal form，すなわちＡＮＦと呼ばれるデータ構造への変換のアルゴリズムにあった。ＡＮＦは連想メモリによって効率的にマニュピュレートできるデータ構造を目指した。基本的にはこのアルゴリズムの主眼は，全ての non-simple ドメインを排除することである。例えば，次の例を考える。

Personnel (name , position , title , education history)

Education History (degree , degree date)

このアルゴリズムでは，non-simple ドメイン Education History をμで置き換え，次のＡＮＦデータ構造が得られる。

Personnel (name , position , title , μ)

Educational History (degree , degree data , μ)

De Fioreと Berra はＡＮＦデータ構造を用いたＩＦＡＭ(Information System for Associative Memories) システムを開発した。このＩＦＡＭは１９７０年にRome Air Development Centre でGoodyear

Aerospace Cooperation の連想メモリを用いて製作された。システムのブロック図を図2.12に示した。ＩＦＡＭでは，順次型インバーティド・リスト表現とＡＮＦ型の比較を行っている。ただしこの比較は，全てのデータが主メモリ中に入ることを仮定しているので一般性は少ない。その結果，順次型インバーテッド・リストでの問合せの回数と連想型メモリでの問合せの回数との比は次の様になる。

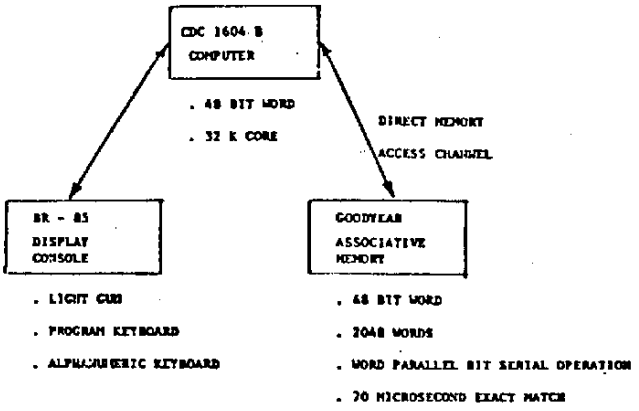


図2.12　ＲＡＤＣ連想メモリ・コンピュータ・システム

単一基準探索……探索されるリストの長さの対数に比例する。

多基準探索……約５０対１

リスト中（１６項目）の１項目の update……３０対１

全てのリストの update……３対１

メモリ容量の比では，順次リストの方がＡＮＦ型に比し２から4.5倍程度容量が必要となる。

b）Moduler等は研究目的のデータベース・システムＳＥＴＦ（Staran Evaluation and Training Facility）をＳＴＡＲＡＮ連想アレイプロセッサを用いて開発した。その構成図は図2.13に示した。この構成で特に注目されるのは，ＰＨＴＤ（Parallel Head per Track Disk）である。このディスク面は７２トラックに分かれ，そのうち６４トラックがＳＴＡＲＡＮに結ばれている。ディスク面は３８４セクターに分割され，各トラックの各セクタは２５６ビットの容量を持つ。データ

ベースを探索する時には，各トラックの１セクターが並列的にSTAR−ANにロードされ探索される。

PHTDでは１セクターのロードに１００ μ secが必要である。探索時間も大体１００μsecが必要である。

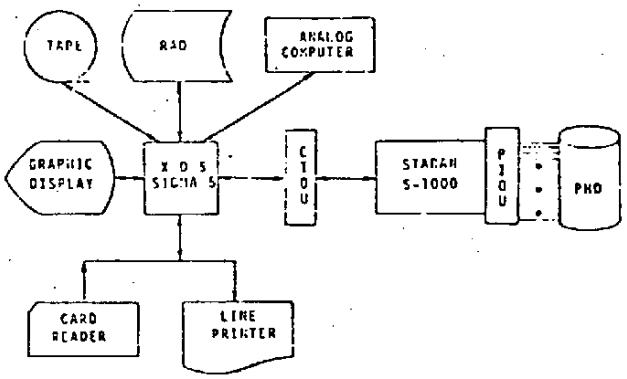


図2.13　STARAN evaluation and training facility

いま，全部のデータベースが一枚のディスク上にあるとすれば，約２回のディスクの回転（約８０$m$sec）で探索が完了することになる。

c） Linde，Gates，Peng 等は仮想的なバイト順次・語平行型の連想プロセッサ計算機システムＡＰＣＳを提案した。この構成図は図2.１４に示されている。このシステムは，二つの連想プロセッサ・ユニットと順次プロセッサ，平行Ｉ／Ｏチャネル，1.６×１０^{9} バイト／秒で連想メモリとデータ転送を行うランダム・アクセス・メモリ等から構成されている。Berraの評価によれば，ＡＰＣＳとＩＢＭ

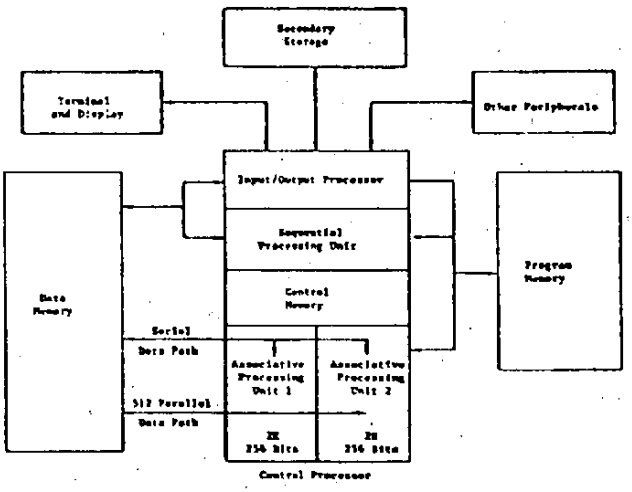


図2.14　APCS連想プロセッサ・コンピュータ・システム

370／145で同じことをした場合，ＡＰＣＳが探索速度で３２～１００倍速く，更新の速度と１５～２１０倍速くなる。

2) ビット・スライス型連想プロセッサを用いる利点と問題点

○利　点

1. 連想アレイにあるデータに対する処理の速度がコンテント・アドレスと並行処理で極めて速くなる。
2. リレーショナル・モデルの二次元表現を比較的・直接的に二次元連想アレイで表現できる。
3. 任意の属性をインデックスとしてみることができる。
4. 更新が容易である。例えば削除の場合，語に対するマスク機能によりそれ以降使われる事が禁止されるデータ項目の追加は，順番が規定されていないので，任意のリレーションの最後にしまえばよい。

○問題点

最も重要な問題点はＩ／Ｏの速度とコストである。現在までのところ，アレイにデータをロードする時間は，アレイ中のデータの探索時間より非常に大きい。例えば，二次メモリー中の8000語ブロックに対して，あるビットパターンが存在するか否かのチェックは次のようになる。

	連想プロセッサ方式	通常の計算機
ロード	41 ms	41 ms
探　索	3 ms	8 ms
	44 ms	49 ms

この問題の解決のためには，高速・大容量メモリー（データベース全体を含む）と，アレイを結ぶバンド幅の高いインタフェイスを開発するか，あるいは，通常のディスクの様な二次メモリーから高速，小容量のステージャ・スキームを開発する必要がある。例えば，高速・

大容量ストーレッジとしては，Farnsworth 等の研究がある。彼等は Rome Air Development SystemのSTARANに適用するギガバイト・ストーレッジを報告している。このブロック図は図2.15に示されている。

RADCのSTARANは4個のアレイ・モジュールを持ち，各々256×256ビットの容量がある。マス・メモリーとSTARAN間には種々のインテリジェントなデータ操作を行うことのできるインターフェイスがある。マス・メモリーとアレイ間には1024の平行ラインがあり，アレイに完全にロードするためには256ビットが順次に転送される必要がある。転送速度は300~450n sec／ビット・スライスであるので，256ビットの順次転送にかかる時間は115μsec 以下である。データは4ストーレッジ・モジュールに分けてストアされる。各モジュールは256×256ビットのランダムにセレクト可能な4096ブロックからなる。Berraの試算によるとこの様な構成の下では128Mバイトのデータベースに対し，その4%のタプルを検索するのに必要な時間は約2秒となる。

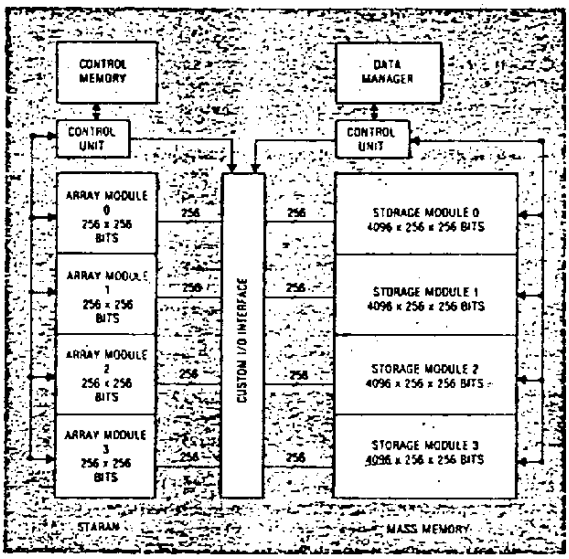


図2.15　STARAN マス・メモリ・インタフェース

(2) DIRECTプロジェクト

DIRECTはWisconsin大学で開発中のリレーショナル・データベースをサポートするスタンド・アローンのデータベース・マシーンである。その構成の概念図は図2.16に示した。

1) 構成要素

a) ホスト・プロセッサ……PDP11/45

b) バック・エンド・コントローラ……PDP11/40

c) 問合せプロセッサ……PDP11/03(28K語)

d) 連想メモリ……8個のCCD

e) 相互結合マトリックス

問合せプロセッサとCCDメモリーの間をcross pointスイッチで結合。

2) データの構成

データベースはマスメモリー中にストアされ，カレントな部分がコンテント・アドレッサブルなCCDメモリーモジュール上にある。この間の転送及びスケジューリングはコントローラーが行う。データベースは，16Kバイト単位のページ・フレームに分割され，一つのページは一つのリレーションしか含まない。ページ中のリレーションのタプルは固定長語で表現される。CCD中のデータの表現は，ビット順次，語平行型である。

3) 問合せの処理

ユーザの問合せは，ホスト計算機によって問合せのパケットに分割され，コントローラへ転送される。パケットを問合せプロセッサへ転送するスケジューリングをコントローラが行う。各問合せプロセッサ毎に異なる処理を行っており，全体としてはMIMD型のアーキテクチャである。

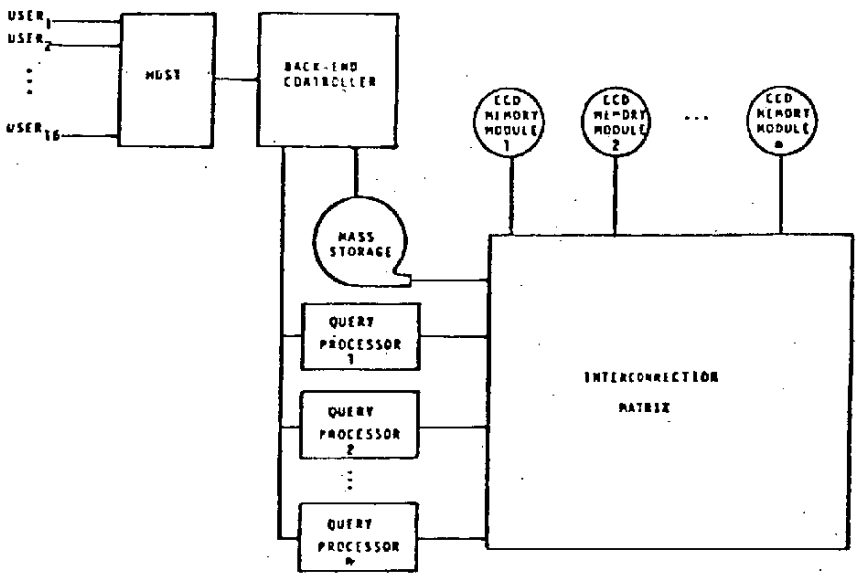


図2.16　a）DIRECTのシステム・アーキテクチャ

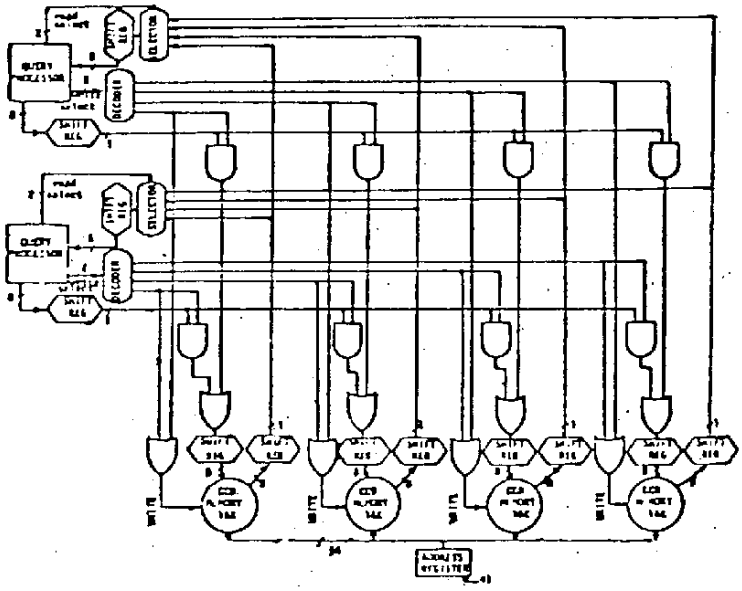


図2.16　b）2×4　DIRECTシステム構成

2.5 DBC (Data Base Computer)

(1) ＤＢＣの概要

ＤＢＣはOhio 州立大学で研究の進んでいるデータベース・マシーンである。その特徴を以下に列挙する。

(a) 構造メモリ

全体のデータベースをマスメモリーと構造メモリー（ディレクトリ・メモリー）で実現する。マスメモリーは大容量メモリーで実際のデータをストアし，構造メモリーは比較的小容量のメモリーで，ディレクトリ情報を保持する。

(b) PCAM（Partitioned Content Addressable Memory）

一般に大容量ファイルに対し，コンテント・アドレスを可能にすることは困難である。しかしながら大容量ファイルをセグメントに分割する事によって，速度は遅いが，ある程度のコンテント・アドレッサビリティが得られる。例えば，可動ヘッド・ディスクに於てシリンダー単位でコンテント・アクセスを行うように設計すると，キガバイト程度までのコンテント・アドレスが可能になる。

(c) 領域ポインタの使用

インデックスとコンテント・アドレスを併用する方式である。

図２．１７中でｆが領域ポインターである。問合せに対してディレクトリーの処理の結果，目的データーを含むマス・メモリーのセグメントを示すポインターが得られる。このセグメントの中では，コンテント・アドレスによって得られる目的データが検索される。

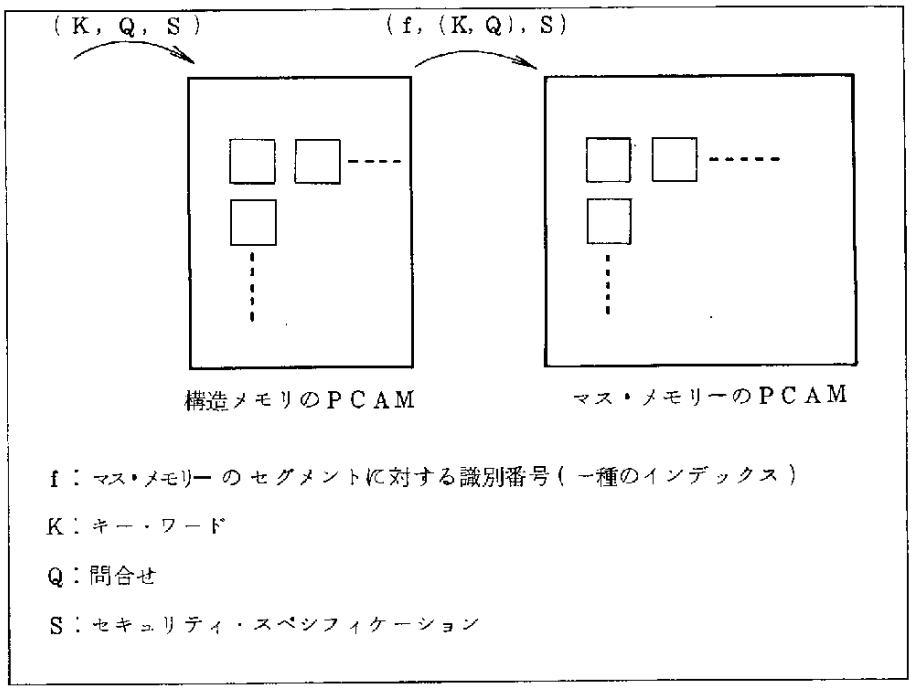


図２.１７　領域ポインターの使用例

(d)　機能単位の専用ハードウェア化

ＤＢＣのブロック図は図２.１８に示されている。

ＰＥＳ　：外界（ホスト・コンピュータ）

ＫＸＵ　：キーワードを内部表現へ変換する部分

ＳＭ　：データベースのディレクトリ情報の検索，更新，削除機能

ＳＭＩＰ　：集合演算処理装置であり，ＳＭから検索された情報の演算を行なう部分

ＩＸＵ　：ＳＭＩＰの演算結果であるロジカルなインデックスをフィジカルなアドレスに直す部分

ＤＢＣＣＰ　：全体のコントロール部分であって，ホストとのインターフェース，コマンド・スケジューリング，セキュリ

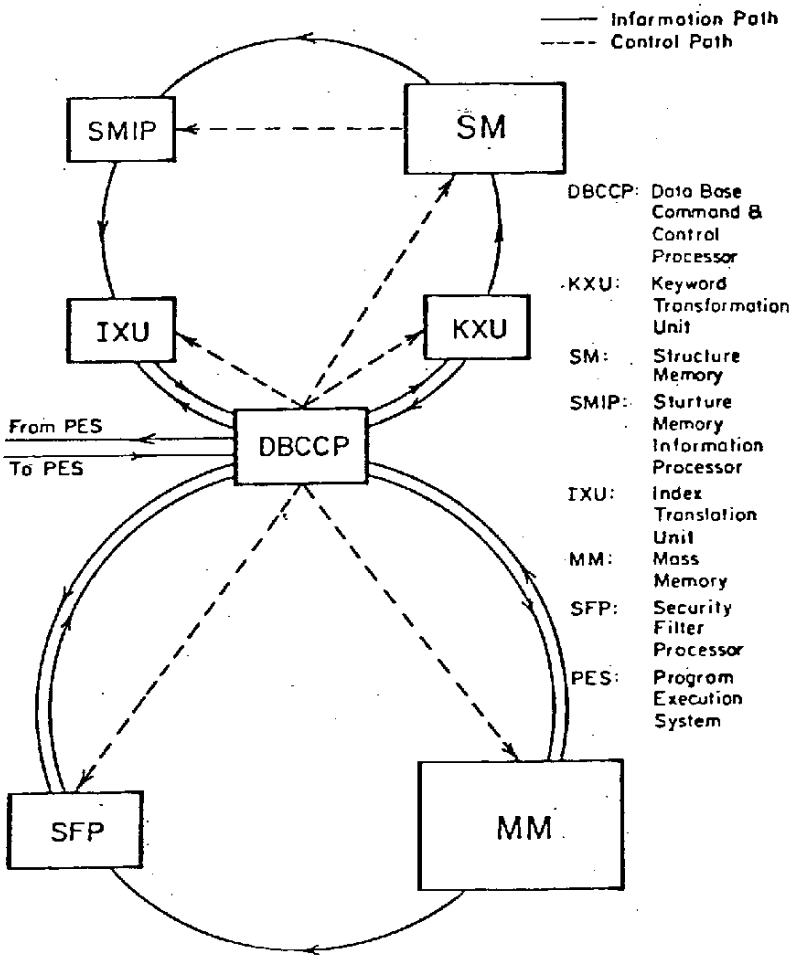


図2.18　データベース・コンピュータの概念と機能

ティ情報の構成，データーの局所化などを行なう部分

MM　：データーベースのストーレッジ

SFP　：DBCCP により作られたセキュリティ情報により，セキュリティのチェックを行なう部分

システムの各種のボトルネックを避ける為，各モジュール毎に最適なハードウェア設計を行なう。例えば一例として，

（機能ブロック）	（実　　現　　法）
DBCCP	通常の計算機の上での，専用マイクロプログラム
IXU	
KXU	
SFP	
MM	可動ヘッド・ディスクのシリンダー単位でのコンテント・アドレス
SM	固定ヘッド・ディスク（あるいはCCDメモリー，バブル・メモリー）を用いたセルラーロジック
SMIP	ＲＡＭとマイクロプロセッサ

の様な専用ハードウエア化が考えられる。

(e) データベースのクリエータがセキュリティに関する情報を指定する事により，データベースを，セキュリティ・アトムに分割する。それに基づき各ユーザの特権アクセス・リストがDBCCP 上に作られＳＦＰでセキュリティ・チェックを行なう。

○ＤＢＣの問題点

1. DBCCP とＰＥＳ（外界）との通信容量がボトルネックになる可能性がある。
2. データベース中にデータがバラバラにつまるのでＪＯＩＮ演算等では，可動ディスクのアームの動きの為，速度が遅くなる。

ＤＢＣはデータベースマシンとしてのトータル設計を目指した初めての試みであり，極めて画期的なものである。

(2) 可動ヘッドディスクにおけるコンテント・アドレス

Track-in-parallel読み出しには次の２つの型がある（図２１９）。

(a) MCSD（Multiple Content Addressability and Single data stream）

全てのトラックからのデータをマージして一つのデータストリームに

変換する。この一つのデータストリームは各々異なった比較を行なう多数のプロセッサによってコンテント・アドレスされる。

(b) SCMD (Single Content Addressability and Mutiple Data stream)

各トラックから出てくるデータストリームに対して，多数のプロセッサが同じ比較を行なう。

MCSDはディスクの一回転で多くの比較演算が可能な利点はあるが，多くのデータストリームをマージする為には高いバンドパスが要求され，1シリンダあたりのトラック数の制限が厳しい。逆にSCMDではトラックの数を多くすることは可能であるが，MCSDと同一のプロセッサを使うとすると複雑な比較演算をするにはディスクが何回転もする必要がある。Braunsch weig工科大学の探索マシーンは，MCSD方式を採用し，DBCではSCMD方式を考えている。

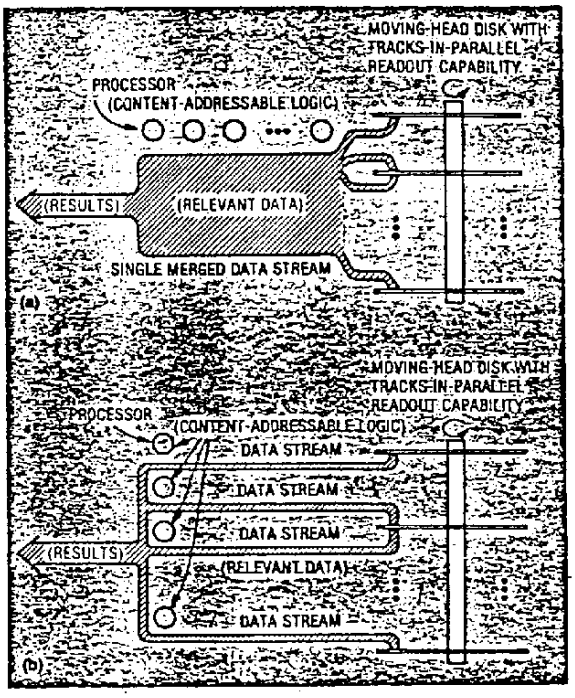


図2.19　2つの大形コンテント・アドレッサブルブロックの2つの基本アーキテクチャ

(a)多コンテント・アドレッサビリティ，単一データストリーム
(b)単一コンテント・アドレッサビリティ，多データストリーム

一般に大規模データベースは多くのシリンダを専有し，あるいは多くのディスクドライブすら専有する。ユーザーの要求に対してシリンダを

決定する方式を持たない限り，全シリンダに対するコンテント・アドレスを行なわなければならない。その意味でインデックスが必要になり，インデックスを容易に得る為のディレクトリの処理方式が問題となってくる。更にまた可動ヘッドのアームの移動を最少化するためにデータの局所化の問題が重要になる。

2.6 再構成可能アーキテクチャとデータベース・マシーン

(1) データベース管理のための再構成可能型マシンのアーキテクチャー*

テキサス大学のDaleらは，多数の（バイトスライス型の）マイクロプロセッサ，大量のメモリ，様々のプロセッサ＝プロセスおよびプロセッサ＝メモリの対応を可能にするためのスイッチング・ネットワークを仮定した，データベース管理のためのマシン・アーキテクチャーを研究している。その中で研究の重点が置かれているのは，適応型のデータ処理ということである。

システムのアーキテクチャーの概要は図2.20に描かれている。この図に示されているように，キーファイルとデータファイルの2種類のデータ構造がシステム中には存在する。キーファイルは，レコードタイプごとに分割されている。このファイルの各エントリーは，そのレコードタイプに対してキーであると指定されたデータ項目に対するデータ値の並びと，そのレコードに対する論理的識別子から成る。論理的識別子は，階層モデルおよびネットワークモデルにおいては，ルートのノードからそのレコードへの論理的パスについての情報を含む，トレースである。

データベースに対する一連の処理要求は，バッチの集合として分割される。各処理要求は，（Keyname operator value）の形の述語またはそれ

＊ 参考文献 29

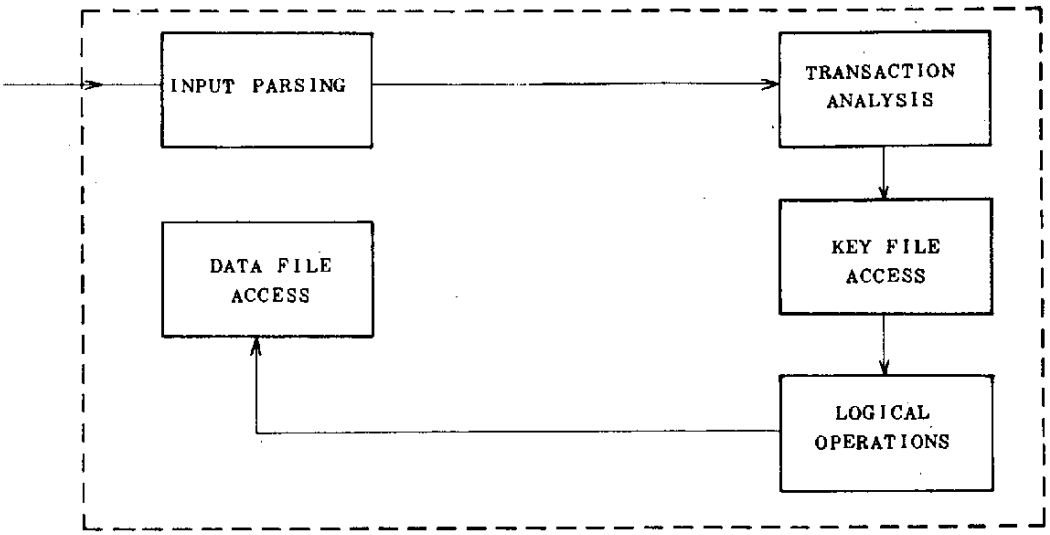


図2.20　DBMSのアーキテクチャーの概要

をブール演算子で結びつけた述語で表現され，セレクト演算をいくつも含む。そのセレクト演算の後，キーファイルから論理的識別子を読み出したり，あるいは書き込みを行なったりする。処理プロセッサのプリプロセッサは，処理要求の各バッチに対して，なされなくてはならない処理を記述したDAG（directed acyclic graph）を生成する。DAGの内部のノードはセレクションまたは集合演算であり，葉のノードはキーファイルに対応する。

処理プロセッサは，各バッチに対するDAGを受けとり，DAGの論理的順序，資源の限界，スイッチング・ネットワークのトポロジー等の制限の下に，スループットを最大にするようにマシンを適応化させた後，セレクションおよび集合演算を実行する。

セレクト演算はキーレコードに対して実行され，ある特定の述語を満たすキーレコードに対するトレースの値を生成する。キーファイルへアクセスする際のバンド幅を上げるため，次の3つのレベルの並行処理を導入している。

1）異なったタイプのキーファイルに対する並行アクセス

2）一つのタイプのキーファイルの中での異なるフィールドに対する並行アクセス

3）一つのキーフィールドからのＤＡＧ中のいくつものノードに対する並行データ転送

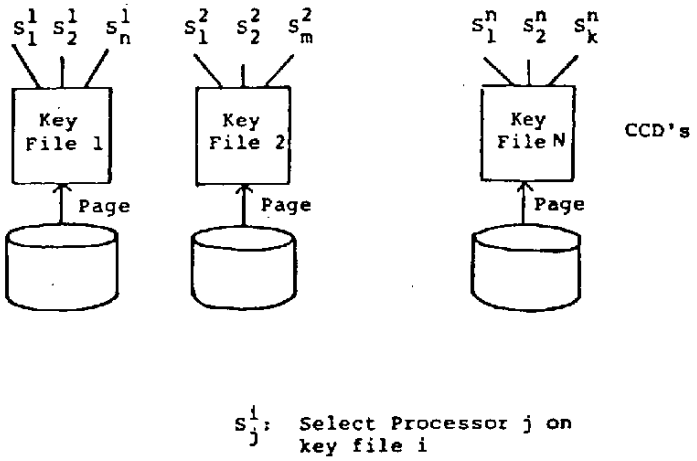


図２.２１　セレクトの処理

図２.２１は二次記憶からページ転送されたキーファイルに対する，セレクト・プロセッサの並行動作を説明している。セレクト・プロセッサの数は，負荷の状況や使える資源に依存して変化し得ることが重要である。セレクト・プロセッサはコンテント・サーチを実行し，キーファイルのサーチは１パスで終了する。

図２.２１に示したセレクト・プロセッサの構成法に関しては，種々のものが考えられる。典型的場合として，垂直型（バイト順次型）と水平型（キーレコード平行型）の２つが考えられている。

垂直型構成法は図２.２２に示すようなものであり，各セレクト・プロセッサは１バイト長であり，Ｉ／Ｏポートも１バイト長である。キーレコード中のバイトは，そのキータイプに割り当てられたすべてのセレクト・プロセッサによって読みとられる。もしそのデータが述語を満たすならば，そのトレースがＤＡＧ中の次のノードへ送られる。

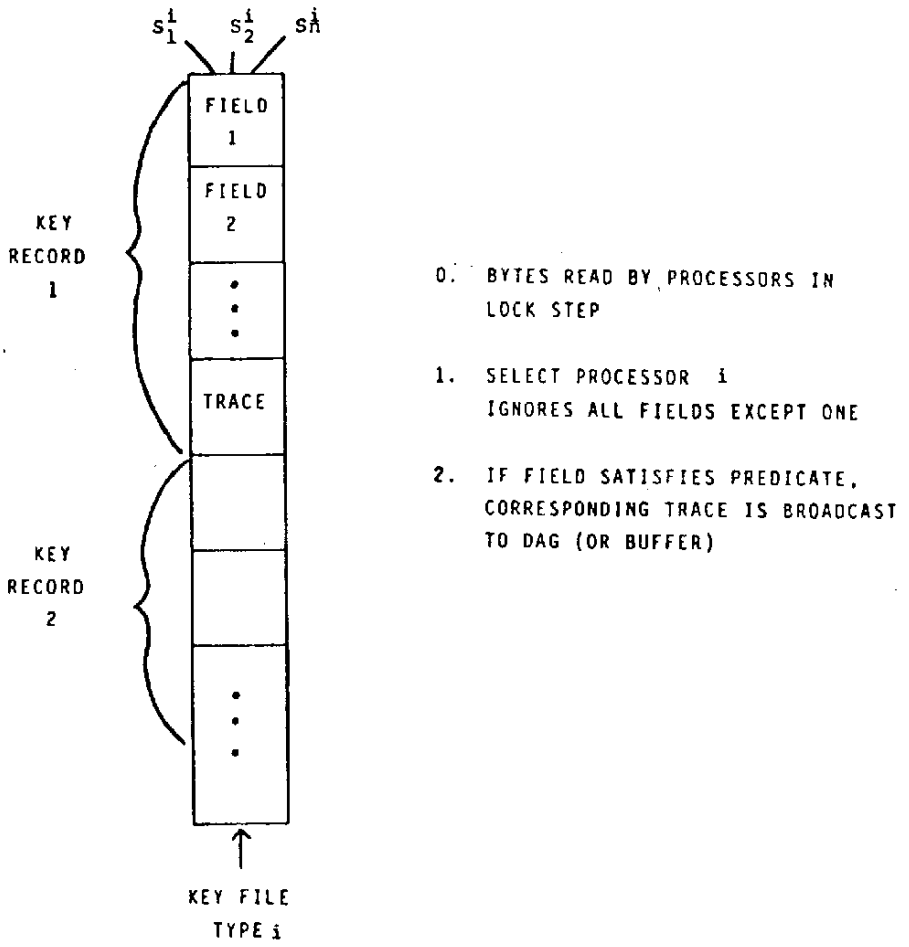


図2.2.2　バイト順次セレクション

水平型構成法においては，個々のフィールドの長さに適合するように，適当な個数のバイトスライスのプロセッサが動的に結合される。図2.2.3に示すように，これらのプロセッサによって，キーレコード全体が一度に読み込める。調べている述語を満たしているフィールドが見つかった場合には，そのセレクト・プロセッサは，そのトレースを保持しているプロセッサに対してそれをＤＡＧ中の次のノードへ送るように指示する。

集合演算を扱うために次の２つの方向を考えている。１つはパイプライ

ン処理であり，もう１つは，集合を扱うための並行アルゴリズムである。パイプライン処理に関しては，データ・フローが均一でないという問題を解決するために，パイプライン中の各ノードを動的に構成するというアプローチをとる。ノードの構成は，そのノードへの入力データがそろった時点で行なわれる。このような意味で，データ・フロー・マシンの考え方がとり入れられている。集合を扱うための並行アルゴリズムについては，まだ十分な研究が行なわれていない。

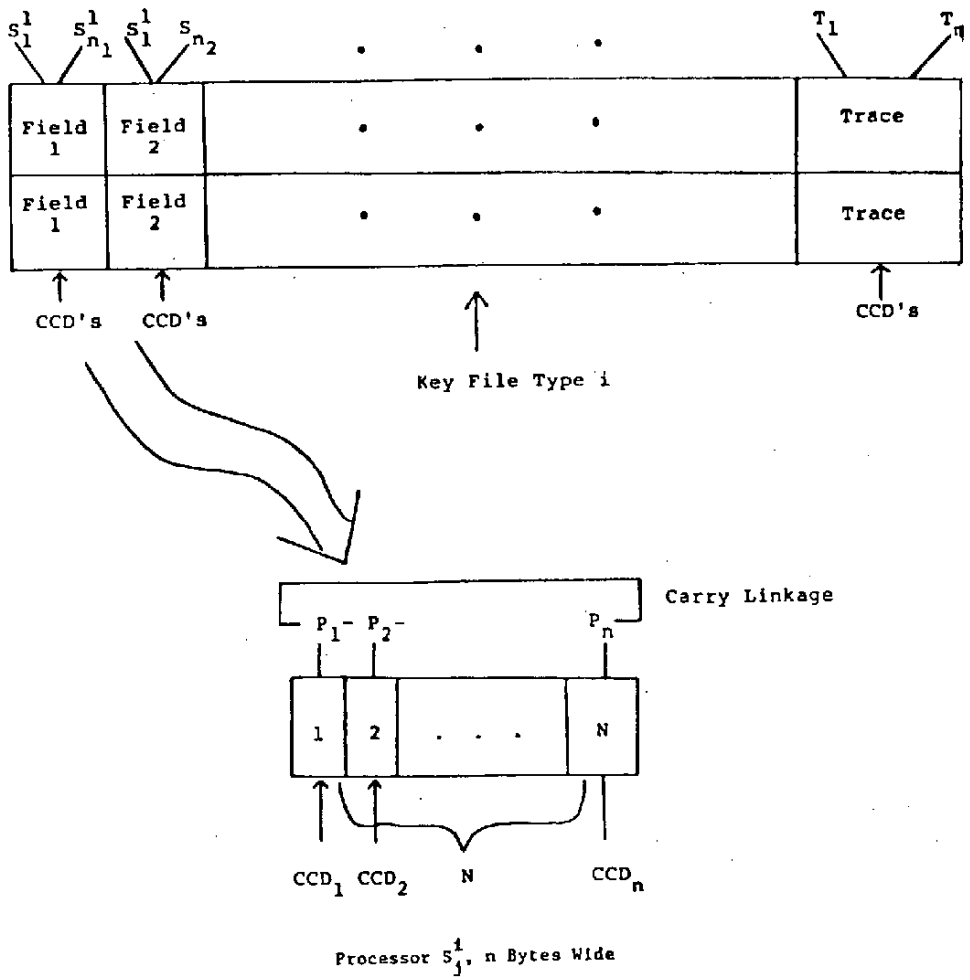


図2.23　水平型パラレルセレクション

システムの制御，オペレーティングシステムの構成，アルゴリズムの設計，スイッチング・ネットワークの構成，Ｉ／Ｏ処理等の問題については現在研究中である。待ち行列理論に基づくシステム評価についても検討中である。バイトスライス型プロセッサの動的結合によるシステムの適応化については，同じテキサス大学のＴＲＡＣ（Texas Reconfigurable Array Computer ）として実験的に実現されている。

(2) ＴＲＡＣ計算機について

1）序　論

最近，ノイマン型計算機の限界が認識されるとともに，これを越える計算機の実現をめざしてソフトウェア，ハードウェアの双方からさまざまな研究が行われている。ＴＲＡＣ計算機プロジェクトの目標は，全く新しいハードウェア構成法を提案し，その評価を通じて将来の計算機のひとつのあり方を追求することである。ＴＲＡＣ計算機研究の目的は，ブラウン（Ｊ.C. Browne ）によれば（参考文献　３０），

1. 高度な並行処理の実行が可能な計算処理システムを安価に実現するための実現法の研究
2. 再構成可能な大規模計算システムを実際的な問題に適用して得られる効果の評価
3. 大規模な計算システムの構成手段としての，資源分割という概念の評価

の三項目に要約される。

ＴＲＡＣとは，Texas Reconfigurable Array Computer の頭文字から作られた造語である。Texas は，この計算機の構想を中心となって推進しているテキサス大学オースチン校の名称に由来する。Array Computer とは，ここでは，多数の処理系を信号伝送線によって相互に，有機的に接続して構成した処理系をさすのであって，並列演算器を持つ，配列（Array ）演算志向の処理系をさすのではない。Reconfigurableと

は，再構成可能という意味である。従来の計算機は，処理系の構成が固定されていたために，計算プロセスの要求する構成が実在しない場合には，計算プロセスがハードウェア構成に妥協してプログラム化されていた。たとえば，フィルタバンクのシミュレーションを通常の計算機の上で行なうことを考えると，その計算プロセスは，おのずから，バンクを構成する各フィルタの並行動作を要求する。しかし，通常の計算機は一時点一命令実行が原則であり，並行動作は認められないので，繰り返し文と一時的変数を用いて逐次処理に改めた形でプログラム化されることが多い。ＴＲＡＣ計算機は，計算プロセスの要求する構成の計算システムを動的に実現することのできる機械を目標に構想がたてられている。これによれば，ハードウェア構成との妥協をとるための，計算プロセスにとって本質的でない制御文や変数なしにプログラムを記述することも可能となる場合がある。

ＴＲＡＣ計算機の研究成果は，計算機のソフトウェア，ハードウェア双方の構成法に大きな影響を与えると予想される。現在までに，各構成要素についてはその基本設計，システム全体についてはシミュレーションによる評価がなされている。これらの成果は，報告書として（参考文献欄参照）出版されている。

2）ハードウェア構成法

最近の電子工学のひとつの成果は，処理装置や記憶装置のようにまとまった働きを行うことのできる素子が作られるようになったことである。ＴＲＡＣ計算機は，多数の処理装置と記憶装置，ならびにそれらを相互に連絡する信号伝送系とから成る。

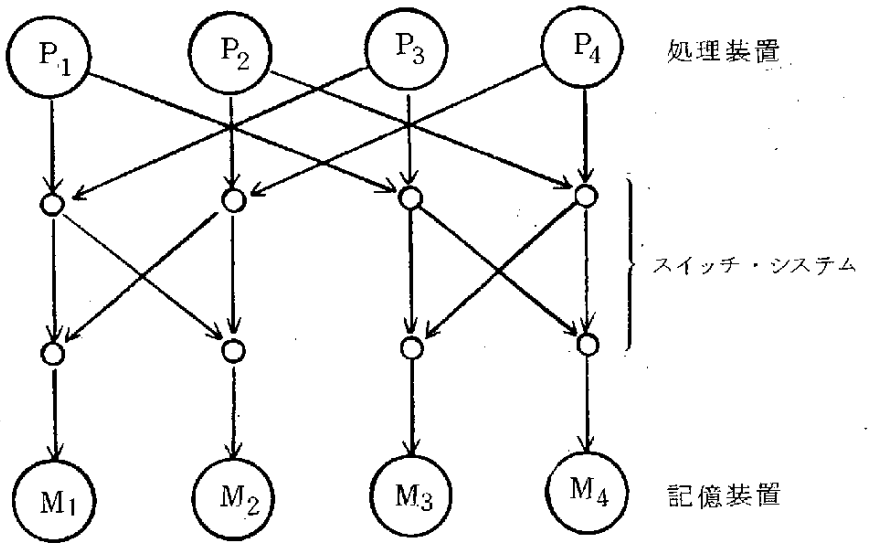
信号伝送系はスイッチの集まりからなり，任意の処理装置と任意の記憶装置を接続する信号伝送経路が構成できるようになっている。このようなスイッチ・システムは，処理装置の数をＮ，記憶装置の数をＭとするときに，ＮＭ個のスイッチを利用すれば明らかに実現可能である。し

かし，スイッチ切り替えに要する時間をへらすために，また処理装置から記憶装置までの伝送線の長さを短くするために，スイッチの総数は少ない方が望ましい。伝送線の長さは処理系の速度に関係し，これを短くとることは，処理装置の速度の向上のためには重要なことである。

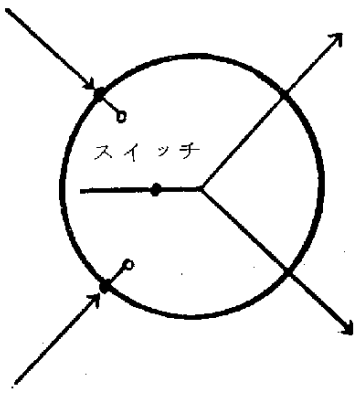
ＴＲＡＣ計算機の特徴は，スイッチ・システムの構成法にあり，ブラウン達がバンヤン樹形と呼ぶ接続法（図２．２４）を採用している点にある。この方法によれば，処理装置の数と記憶装置の数をそれぞれＮとして，総数Ｎ $\log_e$ Ｎ個のスイッチによって，条件を満たすスイッチ・システムを構成することができる。

ＴＲＡＣ計算機は，プログラムによってスイッチ・システムを動的に操作する。その結果として，ＴＲＡＣ計算機は，その時点で活用できる処理装置を動的に手続き（又は副プログラム）に割りあてる。この意味では，この機械は手続きを基本単位とするデータ・フロー機械と見なすこともできる。ＴＲＡＣ計算機は，一時点一命令一データ処理機械（ＳＩＳＤ）ばかりでなく，多命令多データ処理機械（ＭＩＭＤ），一命令多データ処理機械（ＳＩＭＤ），単純ＭＩＭＤ機械（文献参照），パイプライン機械，データ・フロー機械としての構成を実現することも可能である。

文献に示された計画によれば，ＴＲＡＣ計算機のハードウェアについては，処理装置（要素）の選定，記憶装置の構成の決定，スイッチの設計，が終了している。処理装置としては８ビット幅のマイクロプロセッサＡＭＦ７０５２を利用する。ひとつの処理装置の処理するデータは８ビット単位であるが，これを並列に動作させることによって，８×ｎ（１６，３２，６４，…）ビットのデータを単位に処理を行うことができる。記憶装置には荷電結合素子（ＣＣＤ）を採用する。記憶域はセグメントとページとから構成され，ひとつの単位は６４Ｋ byte である。記憶域の内容は四種類の情報を含んでいる。それらは，①演算のオペラ



① バニヤン樹型（N＝4）



② スイッチの概念図

図2.24　TRAC計算機のスイッチ・システムの構成法

ンド，②データ，③システムの動作に対する制御情報，特に，スイッチ・・システムに対する制御情報，記憶域のディスクリプタ等，④プログラム，の四種である。入出力はディスクを利用し，ハードウェア構成上は，

これを記憶装置のひとつとして扱う。また，信号伝送線の切り替えのためのスイッチの構成法は，G. Dujari，P. Horne による技術報告書ＴＲＡＣ－９に詳しく述べられている。これによれば，スイッチ・システムについて，その基本設計ならびにレイアウト作製が終了している。ＴＲＡＣ計算機システムの挙動について，現在シミュレーションによる解析作業が進行している。

3）ソフトウェア

ＴＲＡＣ計算機は，高度な，しかも多様な並行処理を許す構成をとっている点に特徴がある。このような計算機を利用するどのような応用分野があり，また，基本的な算法がどのような形に定式化されるかを調べることは，ソフトウェア・システムの整備を行うためには重要なことである。ＴＲＡＣ計算機のためのプログラムには，どのような並行処理の過程が含まれていても差支えない。また，ＴＲＡＣ計算機の基本命令セットには，ＳＴＡＲ計算機やＡＳＣ計算機が持っていたようなベクトル処理命令が含まれている。技術報告書ＴＲＡＣ－４，５，６はソフトウェアに関する研究成果を報告している。ＴＲＡＣ－４には，いくつかのよく知られた計算プロセスをＴＲＡＣ計算機むきのプログラムに書き下す例が示されている。プログラムの流れが，従来の良く知られたアルゴリズムと比較してかなり変わったものとなることが見られる。ＴＲＡＣ－５では，数値計算分野へのＴＲＡＣ計算機の応用を扱っている。また，ＴＲＡＣ－６では，非数値計算的な分野へのＴＲＡＣ計算機の応用が扱われている。

ＴＲＡＣ計算機のような高度な並行処理過程を許す計算機のためのソフトウェア・システムは現存するわけではない。ＴＲＡＣ計算機の提案者達は，このようなソフトウェア・システムのためのプログラムの構造として，副プログラム単位のデータ・フロー・プログラム構造を推薦している。ここで，副プログラム単位のデータ・フロー・プログラム構造とは，プログラムを構成する基本演算要素として副プログラムによって

定義される演算を考え，これから構成されるデータ・フロー・プログラムを意味している。所謂データ・フロー・プログラムは，基本演算要素として算術演算程度の低レベルな演算子を用いて構成されるが，このようなプログラム構成はＴＲＡＣ計算機にとっては有効ではない。

ソフトウェア・システムについては，基本的な調査ならびにいくつかのプログラムの作製が行われた。現在までに作製されたプログラムは，1.DEC－10のために作られたPASCALのクロスコンパイラ，2.マクロ・アセンブラ，3.ローダ，等がある。

4）結論，問題点と研究計画

ＴＲＡＣ計算機の研究は，ハードウェアの基本的な設計を終了し，試作，評価を実行する段階にある。この計算機が有効に働くためには，第1に利用者の用いる言語の設計と実現，第2にシステム構成の制御を有効に行う制御プログラムの実現，を行わなければならない。利用者にとって使いやすい，並行動作を許す言語の設計は，困難な問題である。同種の言語はＩＬＬＩＡＣ－ＩＶ 計算機の開発に際して研究されたこともあるが，使いやすさと記述力の間の両立しない傾向の調整に手間どったことが知られている。また，スイッチ・システムの制御を効率よく行うことは，ＴＲＡＣ計算機の本質と関連した重要な要求であり，制御プログラムの設計，実現には注意深さが要求される。ハードウェア構成上から見たこの計算機の性能は，1.個々の処理装置の能力，2.伝送系の情報伝送能力，のバランスに依存している。処理装置の能力が低い場合，基本演算のレベルを高くとることが難しくなる。このことは，処理装置，記憶装置間の情報転送の回数を増加させる要素として働く。また，処理装置の活動時間に占める情報伝送待ち時間が増加すると，計算機全体の処理能力は低下する。以上の点を考慮して，より能力の大きな処理装置の導入と共に，光ファイバーの利用が検討されている。

ＴＲＡＣ計算機の研究は，現在，1.利用者むき言語の設計と実現，2.

副プログラムを単位とするデータ・フローの概念に基づく並行処理プロセスの研究，3.計算システムのモデル化と評価，の側面から行われている。

3. 結　論

結論を以下にまとめる。

1. データベース・マシーンの研究は今後盛んになるであろうし，将来商業用のものも現われるであろう。
2. 現在では未だ多くのボトルネックが存在しており，また，各機能に応じたデバイスが得られているとは言えない。このようなデバイスを開発するためには，今後，データベース・システムのアブストラクション，機能分析等の研究の進展が，ハードウェア・デバイス技術の進歩と共に望まれる。
3. データベース・マシーンで用いられるデバイスの技術革新は目覚ましいものであり，現在提案されているデータベース・マシーンの各機能がさらに新しいハードウェア・デバイスに置き換っていくことが予想される。この意味で，データベース・マシーンの発展は革命的であるというよりも漸進的であろう。
4. データベース・マシーンは多くの専用プロセサの複合されたバックエンド・マシーンとして実現されるものと思われる。

参考文献

1. G. F. Coulouris, J. M. Evans and R. W. Mitchell, "Towards content-addressing in data bases," The Computer Journal, Vol. 15, No. 2, pp. 95-98, 1972.

2. L. D. Healy, G. J. Lipovski and K. L. Doty, "The Architecture of a context addressed segment-sequential storage," AFIPS Fall Joint Computer Conference, pp.691-701, 1972.

3. G. P. Copeland, Jr., G. J. Lipovski and S.Y.W. Su, "The Architecture of CASSM: A cellular system for non-numeric processing, Proc. The 1st Annual Symposium on Computer Architecture, pp.121-128, 1973.

4. C. R. DeFiore and P. B. Berra, "A data management system utilizing an associative memory," Proc. NCC, pp. 181-185, 1973.

5. R. Moulder, "An implementation of a data management system on an associative processor," Proc. NCC, pp. 171-176, 1973.

6. R. R. Linde, R. Gates and T. F. Peng, "Associative processor applications to real-time data management," Proc. NCC, pp. 187-195, 1973.

7. C. R. DeFiore and P. B. Berra, "A Quantitative Analysis of the Utilization of Associative Memories in Data Management," IEEE Trans. on Computers, Vol. C-23, No. 2, pp. 121-133, 1974.

8. P. B. Berra, "Some problemsin associative processor applications to data base management," Proc. NCC, pp. 1-5, 1974.

9. R.H. Canaday, R.D. Harrison, E.L. Ivie, J.L. Ryder and L.A. Wehr, "A Back-end Computer for Data Base Management," CACM, Vol. 17, No. 10, pp. 575-582, 1974.

10. T.M. Marill and D. Stern, "The datacomputer — A network data utility," Proc. NCC, pp. 389-395, 1975.

11. S.E. Madnik, "INFOPLEX — Hierarchical decomposition of a large information management system using a microprocessor complex," Proc. NCC, pp. 581-586, 1976.

12. E.A. Ozkarahan, S.A. Schuster and K.C. Smith, "RAP — An Associative processor for data base management," Proc. NCC, pp. 379-387, 1975.

13. C.S. Lin, D.C.P. Smith and J.M. Smith, "The Design of a Rotating Associative Memory for Relational Database Applications," ACM. Trans. on Database Systems, Vol. 1, No. 1, pp. 53-65, 1976.

14. M. Edelberg and L.R. Schissler, "Intelligent memory," Proc. NCC, pp. 393-400, 1976.

15. R.I. Baum and D.K. Hsiao, "Database Computers — A Step Towards Data Utilities," IEEE Trans. on Computers, Vol. C-25, No. 12, pp. 1254-1259, 1976.

16. S.S. Yau and H.S. Fung, "Associative Processor Architecture — A Survey," Computing Surveys, Vol. 9, No. 1, pp. 3-27, 1977.

17. D.K. Hsiao and S.E. Madnick, "Database Machine Architecture in the Context of Information Technology Evolution," Proc. 3rd Very Large Data Bases, pp. 63-84, 1977.

18. K. Kannan, "The Design of a Mass Memory for a Database Co Computer," Proc. 5th Annual Symposium on Computer Architecture, pp. 44-50, 1978.

19. D.J. DeWitt, "DIRECT – A Multiprocessor Organization for Supporting Relational Data Base Management Systems," Proc. 5th Annual Symposium on Computer Architecture, pp. 182-189, 1978.

20. H. Chang, "On Bubble Memories and Relational Data Base," Proc. 4th Very Large Data Bases, pp. 207-229, 1978.

21. J. Banerjee and D.K. Hsiao, "Performance Study of a Database Machine in Supporting Relational Databases," Proc. 4th Very Large Data Bases, pp. 319-329, 1978.

22. J. Banerjee and D.K. Hsiao and R.I. Baum, "Concepts and Capabilities of a Database Computer," ACM Trans. on Database Systems, Vol. 3, No. 4, pp. 347-384, 1978.

23. E. Babb, "Implementing a Relational Database by Means of Specialized Hardware," ACM Trans. on Database Systems, Vol. 4, No. 1, pp. 1-29, 1979.

24. S.Y.W. Su, "Cellular-Logic Devices: Concepts and Applications," IEEE Computer, Vol. 12, No. 3, pp. 11-25, 1979.

25. D.C.P. Smith and J.M. Smith, "Relational Data Base Machines," ibid, pp. 28-38.

26. P.B. Berra and E. Oliver, "The Role of Associative Array Processors in Data Base Machine," ibid , pp. 53-61.

27. D.S. Kerr, "Data Base Machines with Large Content-Addressable Blocks and Structural Information Processors," ibid , pp. 64-79.

28. G.A. Champine, "Current Trends in Data Base Systems," IEEE Computer, Vol. 12, No. 5, pp. 27-41, 1979.

29. A.G. Dale and E.I. Upchurch, "Investigation of Reconfigurable Machine Architectures for Database Management," 1979.

30. J.C. Browne, "A Proposal to the Basic Energy Research Division of the Department of Energy on Applications of a Reconfigurable Array Processor," Dept. of Compt. Sci. and Phy., The University of Texas at Austin, Apr., 1979.

31. J.R. Smullen, "Memory Management of TRAC," (TRAC-2), The Univ. of Texas at Austin, May, 1979.

32. D. DeGroot, "Technical Report TRAC-1 (Preliminary); Introduction to the Architecture of TRAC," The Univ. of Texas at Austin, Jan., 1979.

33. J. Smullen, "Technical Report TRAC-2 (Preliminary); Memory Management of TRAC," The Univ. of Texas at Austin, Jan., 1979.

34. U.V. Premkumar, "Technical Report TRAC-3; TRAC: Principles of Operation," The Univ. of Texas at Austin, Jan., 1979.

35. A.R. Tripathi, "Technical Report TRAC-4; A Preliminary Version of Bench-mark Programs for TRAC Machine," The Univ. of Texas at Austin, Jan., 1979.

36. E.T. Upchurch, "Preliminary Technical Report TRAC-5; Numerical Application of TRAC," The Univ. of Texas of Austin, Jan., 1979.

37. D.P.S. Charlu, A. Dale and E. Upchurch, "Preliminary Technical Report TRAC-6; Non-Numerical Applications of TRAC," The Univ. of Texas of Austin, Jan., 1979.

38. G.L. Lipovski, "Preliminary Technical Report TRAC-7; The Architecture of the Banyan Switch for TRAC," The Univ. of Texas of Austin, Jan., 1979.

39. D. DeGroot, et al., "Technical Report TRAC-8 (Preliminary); Report on Simulation and Scheduling of Texas Reconfigurable Array Computer," The Univ. of Texas at Austin, Jan., 1979.

40. G. Dujari, and P. Horne, "Technical Report TRAC-9; A Preliminary Report on the Node Circuit for Banyan Switch for the Texas Reconfigurable Array Computer," The Univ. of Texas at Austin, Jan., 1979.

41. R. Kapur, "Technical Report TRAC-10 (Preliminary); Implementation of the TRAC Processor," The Univ. of Texas at Austin, Jan., 1979.

42. A.R. Tripathi, and G.J. Lipovski, "Technical Report TRAC-14; Packet Switching in Banyan Networks," The Univ. of Texas at Austin, Jan., 1979.

43. J.C. Browne, and A.R. Tripathi, "Technical Report TRAC-15 (Preliminary); A Job Control Specification Language for Texas Reconfigurable Array Computer," The Univ. of Texas at Austin, Jan., 1979.

44. U.V. Premkumar, and J.P. Smullen, "TRAC: Instruction Set," May, 1979.

第II部　設計仕様処理における要求工学技術の調査

1. 序　論

システムの開発は，市場調査，要求仕様，設計，製作，検査の各段階から成るが，この様なシステムの開発を支援する工具には，紙，鉛筆，フローチャート用のテンプレート，ＨＩＰＯ等の特定のフォーマットを持つドキュメント用の用紙を別にすれば，システムの開発支援用具としては，よく知られているものとして，ミシガン大学のＩＳＤＯＳ，TRW社のＳＲＥＭ，SofTeck社のＳＡＤＴ，ＵＣＬＡのＳＡＲＡ，東京大学のＳＩＤ等がある。ＩＳＤＯＳ以外の４つは，いずれも特殊なグラフを使ってシステムを記述している。次章ではこれら各システムを簡単に述べる。ＳＲＥＭ，ＳＡＤＴ に関しては章を改めて詳述する。さらに，ＳＩＤシステムに関しては第Ⅲ部２章で詳述する。

2. 代表的システム開発工具

ISDOS の目標は，開発するシステムのドキュメントをすべてデータベースに蓄えて必要な情報が，必要な人間にいつでも手に入るようにすることである。ISDOS はシステム記述用の言語としてＰＳＬ，またその記述の解析・レポート生成用としてＰＳＡを持っている。ISDOS へのドキュメントの入力はPSL（問題記述言語）で記述することによって行われる。ＰＳＬによる目標システムの記述は，そのサブシステム（オブジェクトと呼ばれる）とそれの性質と，サブシステム間の関係を与えることによって行われる。各オブジェクト及び関係にはＰＳＬで予約された型が与えられ，ＰＳＬによる記述の解析に使われる。ＰＳＡの大きな利点の１つは，ドキュメントの変更が容易に行えることである。もう１つの利点は，ドキュメントからの情報収集分析を自動化して，システムの分析者，設計者，プログラマ，メンテナー等の生産性を向上できることである。

ＴＲＷ社のＳＲＥＭ（ソフトウェア要求工学方法論）は，要求記述言語ＲＳＬと解析支援システムＲＥＶＳから成っている。ＲＳＬではＲ－ＮＥＴと呼ばれるグラフを使ってシステムの要求を記述することになっている。　ＲＳＬではISDOSのPSL 同様，システム型及び関係型が予約されており，それらの間に許される結合が文法として規定されており，記述エラーを検出できる。

SofTech社のＳＡＤＴ（構造的解析及び設計技術）は，要求を構造的に記述分析するための方法論をＳＡＤＴ図式を使って体系化したものである。SADT図式はシステムの機能構造を表現するのに使われる。機能構造とは，システムを構成するもの（things）とそれらを関連づける関係の厳密な配置を図式的に与えることである。ＳＡＤＴの特徴はこの様な機能構造を表現するのに，階層的なインデックスをもつグラフを使った点にある。

ＵＣＬＡで研究中のＳＡＲＡは，並行システム設計用のインタラクティブ設

計工具群である。ＳＡＲＡにおいては，各システムは構造モデルと行動モデルを使って表現される。構造モデルは，モジュール，ソケット，インタラクションの３つの型を持った要素から成る階層化されたグラフ表現で，システムの構成要素及びそれらの間の関係の名前空間を記述している。一方，行動モデルはコントロール節，コントロール弧，被制御型データプロセサ，無制御型データプロセサ，データセット，データ弧の型を持つ要素から成るグラフ表現でシステムの行動を記述，分析するのに使われる。このために，各要素にはＰＬ／１に類似した言語で書かれた宣言文が与えられる。構造モデルと行動モデルは対応がとられるようになっている。

ＳＩＤは要求仕様に基づきシステムの設計過程を自動化するシステムである。ＳＩＤは，①設計者の視覚的理解のためのインタラクティブ・グラフィックス，②共有ファイル内の設計記述の検証，蓄積，更新，検索，制御のためのデータベース・システム，③共有可能な汎用設計過程をデータベース作用子からなるプログラムとしてインプリメントし，設計作用子として登録管理する設計過程ベースから成る。ＳＩＤはＲＧＦ（再帰グラフ形式論）に基づいている。RGFはシステム記述用の再帰グラフと，それを操作する再帰グラフ作用子から成る。再帰グラフによるシステム構造の記述の際には，与えられたシステムＡに関係している任意のシステムＢは，システムＢがシステムＡの，①外，②内，③境界上にあるかによってシステムＡの，①環境，②サブシステム，③インターフェイスと呼ばれ，またシステムＡとＢの間の関係は，①関連関係，②包含関係，③インターフェイシングと呼ばれる。再帰グラフはこれらの関係の差を明確に形式的にも視覚的にも区別した。頂点と弧が再帰的に構造化されたグラフである。一方，再帰グラフ作用子には，再帰グラフがテーブルで蓄えられることを仮定している都合上，結局テーブル操作子が使われる。一方，共有される設計過程としては，設計図の統合，簡略作用子，与えられた条件を満たす部分設計図を作り出す設計調査作用子，及び設計図を解析してシステムの性質を評価する設計解析・評価作用子，簡単な設計エラーを検出する検証作用子等がある。

ＳＩＤへの設計図の入力は再帰グラフに基づいて応用分野ごとに定められたグラフ的設計記述言語を使う。現在，ソフトウェアアーキテクチャ設計，並列システム設計，病院事務管理システム設計用の３つのものが定義されており，グラフィック・データベースのシステムの開発と並行して，そのインプリメントも行っている。これらの言語では，システムの型，関連の型が予約されており，それらの間に許される結合も定義されているので入力された設計図に記述ミスがないかを検査できるようになっている。

3. SREM(Software Requirements Engineering Methodology)

ＳＲＥＭはＴＲＷ社の開発したシステム開発支援方式である。ＳＲＥＭは要求を扱うのに，computer-aided systemを導入した。ＳＲＥＭは，ＲＳＬ(Requirement Statement Language…要求を表現する機械で処理できる言語)とＲＥＶＳ(Requirements Engineering and Validation System…ＲＳＬでかかれた要求の展開をsupportする一連の道具群)から成る。前者は，要求を記述するための機械処理可能な言語である。後者は，ＲＳＬのトランスレータ，システム要求を保持するデータベースである抽象システム意味論モデルＡＳＳＭ(Abstract System Semantic Model)，そのＡＳＳＭの情報を解析するためのツールの集まりなどを含むソフトウェア・パッケージである。

この技法では，システムに要求される処理が，Ｒ－ネットと呼ばれる流れ図によって表現される。Ｒ－ネットは要求を，フローストラクチャで表現して，グラフィック・ディスプレイ上でインタラクティブに操作しうるようにした，グラフによる要求の表現方法である。Ｒ－ネットはフローストラクチャを持っており，それはnode(各processing operationを表現している)とarc(nodes間をつなぎフローを表現している)とから成っている。そのうちbasic nodesはＡＬＰＨＡ(functional processing stepsを表現している)と ＳＵＢＮＥＴＳ(階層的により低レベルのプロセスフローを記述している)を含んでいる。ただし，本質的にはＳＵＢＮＥＴは，プロセスの内部的詳細を含むように拡張された一種のＡＬＰＨＡである。すべてのbasic node は出入口が１つである。従ってこれだけでは単純なシーケンスフローしか表現できないので，これに加えてfan-inとfan-outを表現するstructure nodesとして，ＡＮＤ，ＯＲ，ＦＯＲＥＡＣＨが用意されている。従ってＲ－ネットはbipartite graphである。ＡＮＤ node は同期点として使われる。つまりここにfan-inするpath のすべてが完了しないと次のＡＮＤ node以下のステップにすすめない。またＯＲ

node ではそこからでる path の内１つの条件が満たされれば他の path は無視される。また FOR EACH node は，そこに付与されている data の各々に対して，FOR EACH node に付いている１つの path を繰り返し実行することを示している。このＲ－ネットの入力は interactive graphics tool で行われるか，またはこれと同等な表現であるＲＳＬで言語として与えられる。

一方ＲＳＬで使われる各語の意味はＡＳＳＭと呼ばれるリレーショナル・データベースに蓄えられる。そのための入力情報はＲＳＬの基本的な４つのプリミティブによって，自然言語的に表現される。それらプリミティブとは，Elements （名詞に対応しており，プロセスステップを表わすALPHAや，システムに必要なデータ群を示すDATAやR-NET），Relationship （動詞に対応しており，２つのElement 間の binary relation を示す。常にその動詞の主語と述語を明確に表現する必要がある），Attributes （形容詞に対応し，Element の主要な性質を示す。名前とか番号とか text string を値として持っている）と structure （前述したＲ－ネットのフローストラクチャの一次元的表現である）の４つである。ＲＳＬは拡張可能な言語であり，必要に応じて新しい概念を，前もって用意されている概念の核集合（core set ）につけ加えることができる。ＲＳＬのトランスレータはＲＳＬ文を解析し，その内容が抽象化されて，ＡＳＳＭに入れられる。すべての入力に対し，トランスレータはすでにＡＳＳＭにある情報を参照しながら，簡単な一貫性のチェックを行なう。ＲＳＬの拡張に関する情報も，トランスレータを通してＡＳＳＭに入れられる。Ｒ－ネットによるフローグラフの表現例を図3.1に，ＲＳＬによるＲ－ネットの表現例を図3.2に，ＲＳＬによる定義例を図3.3に示した。

ＲＥＶＳのなかの自動ツールとしては，グラフィックス・パッケージ，静的な一貫性チェッカ，シミュレータ自動ジェネレータとその実行パッケージが含まれている。グラフィックス・パッケージにより，Ｒ－ネットを入力したり，ディスプレイしたり，また変更を加えたりすることができる。ＲＥＶＳにおける情報の流れを図3.4に示した。静的な一貫性チェックは，要求の一貫性や完

全性を静的にチェックする。それらには，Ｒ－ネットの構造自体をチェックするもの，データの流れを解析するもの，ＳＵＢＮＥＴを用いた階層構造について調べるものの，３つのグループがある。シミュレータ自動ジェネレータは，Ｒ－ネットの制御の流れをインプリメントするシミュレーション用のＰＡＳＣＡＬプログラムを生成する。その際，処理段階（ＡＬＰＨＡ）に対応する手続きに対するアルゴリズムやモデルは，過去の事柄を参照しながら要求工学者自身が与えなくてはならない。

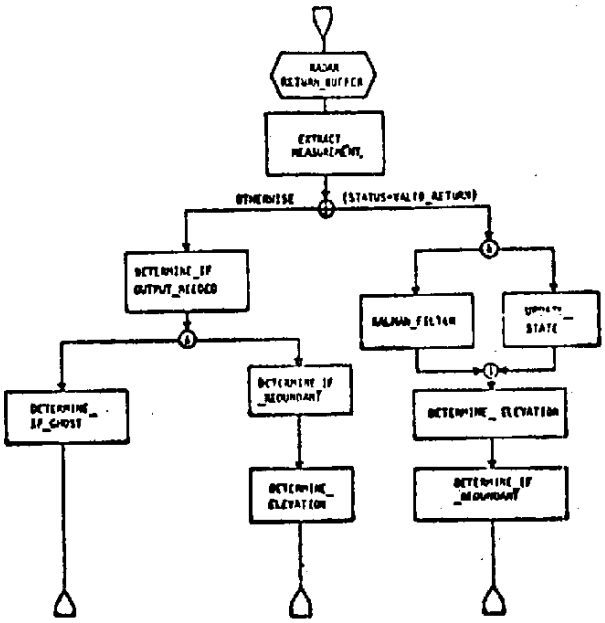


図３.１　Ｒ－ネットによる
フローグラフの表現例

```
R_NET:  PROCESS_RADAR_RETURN.
    STRUCTURE:
        INPUT INTERFACE RADAR_RETURN_BUFFER
        EXTRACT MEASUREMENT
        DO (STATUS = VALID_RETURN)
             DO  UPDATE_STATE AND KALMAN_FILTER  END
             DETERMINE_ELEVATION
             DETERMINE_IF_REDUNDANT
             TERMINATE
        OTHERWISE
             DETERMINE_IF_OUTPUT_NEEDED
             DO  DETERMINE_IF_REDUNDANT
                 DETERMINE_ELEVATION
                 TERMINATE
             AND DETERMINE_IF_GHOST
                 TERMINATE
             END
        END
    END.
```

図３.２　ＲＳＬによる
Ｒ－ネットの表現例

ＲＳＬとＲＥＶＳを用いてシステム要求を記述し検証するための一連の段階について概説する。まず，システムの機能やパーフォーマンスに対する要求が，データ処理副システム性能要求ＤＰＳＰＲ（Data Processing Subsystem Performance Requirement）とし

```
DEFINE ELEMENT_TYPE:  INPUT_INTERFACE
                      (*A port between the data
                      processing system and the
                      rest of the system which
                      accepts data from another
                      part of the system*).

DEFINE ELEMENT_TYPE:  MESSAGE
                      (*An aggregation of DATA and
                      FILES that PASS through an
                      interface as a logical unit*).

DEFINE RELATIONSHIP:  PASSES
                      (*An INPUT_INTERFACE *PASSES*
                      a logical aggregation of data
                      called a MESSAGE from the
                      outside system into the data
                      processing system*).

    COMPLEMENTARY RELATIONSHIP:  PASSED ("BY").

    SUBJECT:  INPUT_INTERFACE.

    OBJECT:  MESSAGE.
```

```
ALPHA:  EXTRACT_MEASUREMENT.
        INPUTS:  CORRELATED_RETURN.
        OUTPUTS:  VALID_RETURN, MEASUREMENT.
        DESCRIPTION:  "DOES RANGE SELECTION PER
            CISS REFERENCE 2 - 7".
        ENTERED_BY:  "M. RICHTER".

ALPHA:  DETERMINE_IF_REDUNDANT.
        INPUTS:  CORRELATED_RETURN.
        OUTPUTS:  REDUNDANT_IMAGE.
        DESCRIPTION:  "THE IMAGE OF THE RADAR RETURN
            IS ANALYZED TO DETERMINE IF IT IS
            REDUNDANT WITH ANOTHER IMAGE".
        ENTERED_BY:  "F. BURNS".

DATA:  MEASUREMENT.
       INCLUDES:  RANGE_MARK_TIME, AMPLITUDE,
            RANGE_VARIANCE, RD_VARIANCE,
            R_AND_RD_CORRELATION.
       DESCRIPTION:  "THIS IS THE ESSENCE OF THE
            INFORMATION IN THE RETURN".
       ENTERED_BY:  "F. BURNS".

ORIGINATING REQUIREMENT:
            DPSPR_3_2_2_A_FUNCTIONAL.
       DESCRIPTION:  "ACTION:  SEND RADAR ORDER
            INFORMATION:  RADAR ORDER.  IMAGE
            (REDUNDANT)".
       TRACES TO:  ALPHA COMMAND_PULSES
                   ALPHA DETERMINE_IF_REDUNDANT
                   MESSAGE RADAR_ORDER_MESSAGE
                   DATA REDUNDANT_IMAGE
                   ENTITY_CLASS IMAGE.
       ENTERED_BY:  "T. E. BELL".
```

図3.3　ＲＳＬにおける定義の例

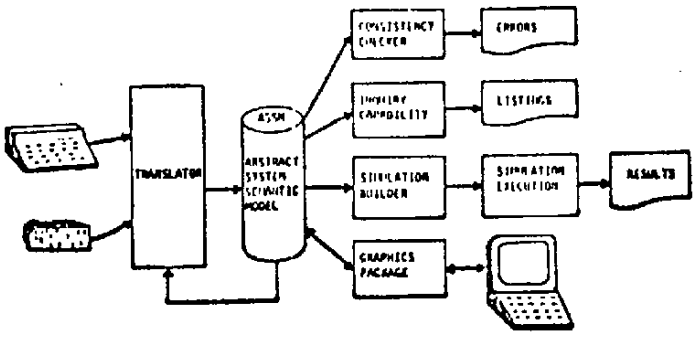


図3.4　ＲＥＶＳにおける情報の流れ

て集められる。これらは，ＲＳＬのＯＲＩＧＩＮＡＴＩＮＧ－ＲＥＱＵＩＲＥＭＥＮＴＳの形に要約され，Ｒ－ネットがＤＰＳＰＲに対する追跡可能性を持った形で生成され，一貫性や完全性のチェックがなされる。その後，Ｒ－ネットの細細化が

なされると同時に，タイミングや正確さなどについての性能要求が記述される。次に，各処理段階に機能的モデルを適合させたシミュレーションによって，システムのふるまい方についての検証が行なわれ，最後に，各処理段階に対して，実際のインプリメントのときに使えるアルゴリズムをあてはめたシミュレーションによって，要求に対する実際の解が示される。

4. SADT(Structured Analysis and Design Technique)

構造的分析法SADTはSofTech社で開発された，要求定義とシステム設計のための技法である。しかし，この技法はシステムの問題に限らずに，さまざまの分野に使われている。

この技法では，概念を記述し伝達するのに，グラフ的言語が使われる。記述さるべき主題は，SADTモデルと呼ばれる階層構造をなす一連のダイヤグラムで表現される。以下この方法の特徴を列挙する。

1) Understanding via Model Building …紙の上でモデルを作り，徹底的に理解することを目指す。

2) Support of Disciplined Teamwork

3) ALL Decisions and Comments in Written Form

4) Top-down Decomposition…複雑な問題をその構成要素に細分してゆく方法をとっているので，技法がtop-down，modular，hierarchicalそしてstructured である。SADTモデルでは，図4.1に示すように，上の水準の図式のなかのあるピースをさらにいくつかのピースに分解して記述することを繰り返すことにより，段階的にその詳細が明らかにされる。

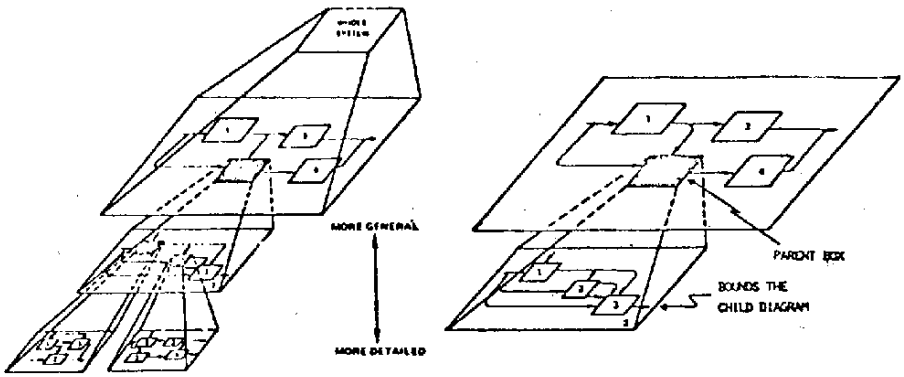


図4.1　SADTモデル

ＳＡＤＴ図式は，boxとarrowから成る。boxは分解のときのピースに対し，boxどうしを結ぶarrowはそれらの間の関係を表現している。

5） Functional Modeling Versus Implementation Modeling

「問題が何であるか」を「どのようにして問題を解き，実現させるか」からまったく切り離して考え，実際に解をみつける前に徹底的に問題を明らかにする。

6） Dual Aspects of a System

ＳＡＤＴにおいては，すべての主題はactivityとデータの二面からモデル化される。boxの四辺は，図4.2に示すように意味づけられている。activity図式では，activityがboxで示され，データがarrowで示される。したがって，各activityの詳細が段階的に記述されることになる。データ図式では，boxとarrowの役割が逆になる。どちらの図式でもboxのMECHANISMに入るarrowだけは，相互関係を意味するのではなく，そのboxをインプリメントするための手段を示す。ＳＡＤＴモデルでは，データ図式の組とactivity図式の組の両方を合わせて，初めて一体のものとなる。以上に述べたactivity図式とデータ図式のほかに，自然言語による説明文や図式の階層構造を示す頂点索引などが一緒になって一つのＳＡＤＴモデルが完成する。

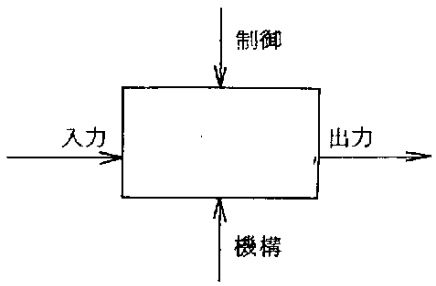


図4.2　boxの定義

7） Graphic Format of Model Representation

ＳＡＤＴではモデルはgraphical languageを使って設計されるが，その理由は

a） 徐々に詳細をみせていく。またそのときよく制御された方法をとる。

b） 正確性と簡潔性を大事にする。

c） model 間の interface を示すことに焦点をおき，これを arrowで示す。

d） 強力な分析及び設計上の vocabulary をふやす。

ことである。

図4.3に示したように，ＳＡＤＴにおける arrowは各module 間の相互関係を示すのであって，コントロールフローを示すものではない。各module にはいる arrowはその module が完了するために必要なすべての入力を示している。また各 arrowは図4.4に示すように幾つかのダイヤグラムのレベルを通して異なるレベル間の module をつなげることもできる。

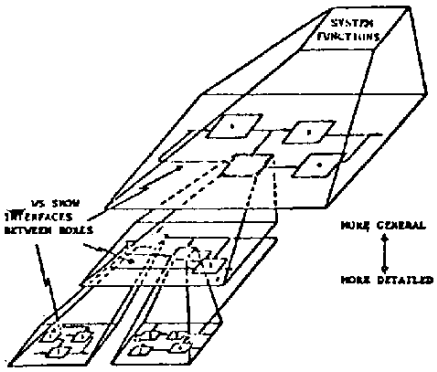


図4.3 アローインタフェイス

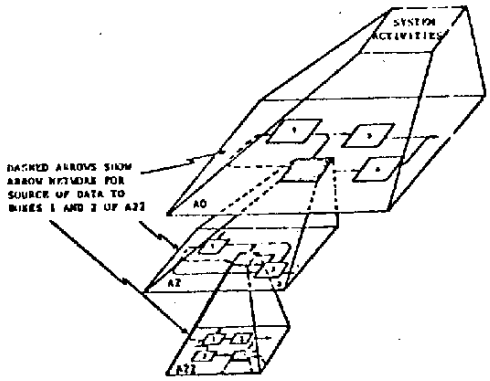


図4.4 アローネットワーク

第III部　要求工学技術の研究

1. 序　論

現在の要求工学技法の問題点を挙げると

1) 要求進化への適応性

2) 形式性の不備

3) 抽象化技法の未発達

4) ユーザーの使い易さ

等がある。

この部では，上記の問題解決のための技術を東大国井研の研究を中心として述べる。2章では問題点2), 3)を解決する一つのアプローチとして，ＲＧＦ（リカーシブ・グラフ・フォーマリズム）を基にした設計工具の開発に関して述べる。3章では，要求進化への適応性を持った進化型データベース・システム・アーキテクチャに関して述べる。このシステムでは要求仕様を記述するため，ＱＢＥに類似した表型言語を用い，問題点4)に挙げたユーザの使い易さに対する充分な考慮がはらわれている。問題点4)の解決のためには，要求を入力するためのサポートとしてのＳＢＡ（System for Business Automation）の開発が必要と思われる。ＳＢＡの概念の基本は，ＱＢＥ的な連想処理型要求仕様機構とTrigger機構，さらに，音声／画像インタフェイスから成っている。4章ではこのうち特にTrigger機構と，画像インタフェイスに関して述べる。画像インタフェイスとしては，非文字，数値型データとしての画像データの入力（汎用・高速画像処理），出力（会話型グラフィック）の両面がある。

2. 再帰グラフによる設計システム

2.1 はじめに

今日のシステムは非常に大規模であり，社会における基本的役割をたえず遂行する為に高い信頼性を持たねばならない。この様なシステムとは，例えば大型計算機のオペレーティングシステム，大規模データベース，化学プラント，医療情報システム，事務処理自動化システム等である。これらのシステムは非常に多くの副システムとそれらの間の多種にわたる相互関係から成っている。この様な複雑なシステムを高い信頼度で設計するには，従来通りの人手に頼っていてはコストがかかりすぎるので，計算機の支援に基づいた体系的アプローチが要求される。

一般的に言って，多くのシステムは，共通モジュールを含んでいる。又，この様なシステムの設計過程は基本的な共通プロセスを含んでいる。この様な共通モジュールの設計図や，共通プロセスを共有化する事ができれば多大の省労力になる。

設計図の共有化の主たる利点は次の様である。

1. 同じ設計図を繰り返し使う事からくる開発費の低減。
2. 統一した設計言語で設計図を記述する事による設計図の標準化。
3. 設計図を形式的に記述する事によって設計の正しさや設計されたシステムの性能の評価が自動化できる。

一方，共通設計プロセスの共有化の主たる利点は次の様である。

1. 共通プロセス自体，形式的に定義されるので，その正しさが保証される。
2. 確立された共通プロセスを通して設計するので，有効でかつ信頼性の高い設計結果が得られる。
3. 共通プロセスを繰り返し使う事による開発費の低減。

これらの設計図及び設計プロセスを計算機で共有管理する為には，まずそれ

らが十分形式的に定義される必要がある。これらは，設計図式及び設計作用子として形式化される。再帰グラフ理論ＲＧＴは，これら両者の定義・定理等を含む理論である。

2.2 再帰グラフ

設計図式を表現するには，従来，グラフが用いられてきた。しかし通常のグラフでは，多種にわたるシステム間の関係を弧のみで表現した為，理解のしやすさの点から難点があった。これを克服する為，まずシステム間の関係を次の3種に分けた。

1. 関連関係：システム間の対等な関係，例えばデータフロー等。
2. 包含関係：システムとそのサブシステムの間の関係。
3. モジュル・ポート関係：ポートとはシステムをモジュール（その内部詳細を知らずに使えるシステム）として設計する際に，モジュールの内部と外部の連絡を取る為の内・外両方に知られたシステム。

これらのシステム間の関係をうまく構造的に表現する数学的道具として，再帰グラフを定義した。すなわちＲ＝（NA, A, af, sn, pt, sa, ns, as）：

1. システムを表現する頂点集合Ｎ。
2. システム間の関連を表現する弧集合Ａ。
3. 各関連を，それによって結合されるシステムに対応づける弧関数 af ：$A \rightarrow 2^N \times 2^N$。
4. システム間の包含関係を表現する副頂点関数 sn ：$N \rightarrow 2^N$。
5. システム間のモジュル・ポート関係を表現するポート関数 pt：$N \rightarrow 2^N$。
6. 関連間の包含関係を表現する副関連関数 sa ：$A \rightarrow 2^A$。
7. 8. システム及び関連にそれらの名前等の意味情報ＳＲ及びＡＲを与える頂点意味関数 ns ：$N \rightarrow SR$ 及び弧意味関数 as ：$A \rightarrow AR$。

図表現としては，図 2.1 に示すようにシステムは各種の箱型で表現され，関

Notations and Meaning / Graph Component	Graphical Notation	Formula Notation	Meaning
Nodes and node semantic function	R1: 1:buffer	$N_{R1}=\{1\}$ $ns_{R1}(1)=$buffer	There is a *buffer* system.
Subnode function	R2: 1:buffer, 2:sender, 3:receiver	$N_{R2}=\{1,2,3\}$ $sn_{R2}(1)=\{2,3\}$ $sn_{R2}(2)=sn_{R2}(3)=\phi$	*Buffer* has *sender* and *receiver* as its subsystems.
Port function	R3: 1:buffer, 2:sender, 4:syn, 3:receiver, 5:syn	$N_{R3}=\{1,2,3,4,5\}$ $pt_{R3}(2)=\{4\}$ $pt_{R3}(3)=\{5\}$ $pt_{R3}(1)=pt_{R3}(4)=pt_{R3}(5)=\phi$	*Sender* and *receiver* have *syn* as their port.
Arcs, arc semantic function and arc function	R4: 1:buffer, 2:sender, 4:syn, 1:synchronize, 3:receiver, 5:syn	$A_{R4}=\{1\}$ $as_{R4}(1)=$synchronize $af_{R4}(1)=\{4;5\}$	There is a *synchronize* association directed from *syn* of *receiver* to *syn* of *sender*.
Subarc function	R5: 1:buffer, 2:sender, 4:syn, 6:in, 7:out, 1:synchronize, 2, 3, 3:receiver, 5:syn, 8:in, 9:out	$A_{R5}=\{1,2,3\}$ $sa_{R5}(1)=\{2,3\}$ $sa_{R5}(2)=sa_{R5}(3)=\phi$ $af_{R5}(2)=\{9;6\}$ $af_{R5}(3)=\{7;8\}$	*Synchronization* has two *control flows* of opposite directions as its subassociations.

図 2.1　再帰グラフ成分

連関係はそれらを結ぶ各種の折れ線で表現される。包含関係は箱型及び折れ線間の入れ子表現で表現される。またポートを表現する箱型は，それが属するモジュールを表現する箱型の境界上に描かれる。この様に再帰グラフの特徴はシステム間の各種の関係を個別にうまく表現する設計図式の構造化表現を与えたことである。

2.3　再帰グラフ作用子

設計図式は再帰グラフで表現されたが，一方，設計作用子は再帰グラフ作用子として形式的に定義される。再帰グラフ作用子は大きく分けて，原始作用子とマクロ作用子に分かれる。前者を狭義に，再帰グラフ作用子，後者を設計作

用子と呼ぶこともある。前者は再帰グラフをグラフ要素ごとに（頂点や弧を単位として）扱うが，後者はグラフ自体を非作用子として持つ。

原始作用子には，頂点や弧のグラフへの挿入作用子，及びグラフからの削除作用子，ある頂点及び弧と関連関数・副頂点関数・副弧関数・ポート関数等で関係づけられた頂点及び弧を取り出す作用子，及び頂点や弧が表現するシステムや関連の意味情報を取り出したり，書き換えたりする作用子がある。

一方，マクロ作用子（又は，設計作用子）は，設計プロセスの基本的共通プロセスを表現している。マクロ作用子には，主として次のようなものがある。

1. 統合・簡略作用子：２つの設計図式を特定の目的で１つに統合したり，設計図式の特定の副図式を作り出す。
2. 調査作用子：設計図式から，与えられた条件を満足する設計図式を生成する。
3. 解析作用子：設計図式を解析して，システム及び関連等の性質情報を取り出す。
4. 検証作用子：設計図式が特定の性質をみたす事を検証する。

この様に定義された各マクロ作用子は原始作用子及びマクロ作用子の特定の組合せとして表現できる，言い換えれば，原始作用子によって機能的に抽象化された設計図式機械上のプログラムとして実現できる。

2.4 ＲＧＴ：再帰グラフ理論*

再帰グラフ理論ＲＧＴは１９７８年，ＭＥＤＩＳ '７８で初めて公表された。ここでは，再帰グラフは病院内の情報フローの設計を記述するのに使われ，その際設計エラーをどの様にして検出するかが議論されている。典型的な設計エラーは次の様な理由でおこる。

1° 各モジュールの設計記述が

a 矛盾している。

---

*参考文献 1

b　あいまいである。　及び／又は

c　論理的にまちがっている。

2°　モジュールの統合過程において，

a　設計されたモジュールが，それに対する要求をみたさない。

b　モジュールのインターフェース間に不整合がおこる。

これらの原因によっておこる設計エラーは，非常に重大である。なぜなら，それらは検出したり修正したりするのが困難なだけでなく，設計過程に後続するシステムの開発過程において他のエラーを誘発する恐れがある。

この様な設計エラーを検出するために，病院内の情報フローを表現する再帰グラフに対して，次の様な整合（ consistency ）条件を果す。

a。境界条件（ boundary condition）……どんな弧も，ポートを通らずに，頂点の境界の内と外にある頂点を結合することはできない。これはシステムのインターフェースの欠落を検出するのに使われる。

b．方向条件（ direction condition ）……頂点の境界の一方側において，すべての弧は，ポートに対して入射するか，出射するかである。これは，どんなポートも情報のソースにもシンクにもならないことを強制している。

これらの条件は，データベースにたくわえられる再帰グラフに対して常にチェックされる。

この論文では，設計図を統合・簡略化するための設計作用子として，{JOIN，200M－IN，200M－OUT，EXTRACT，SELECT｝が定義されている。JOINは，与えられた２つの再帰グラフを，指定された共通部分を融合することによって結合する。SELECTは，与えられた再帰グラフから指定された頂点と弧からなる部分再帰グラフを生成する。200M－INは，与えられた２つの再帰グラフを，一方を他方の指定された頂点の中に，指定されたポートペアーを融合しながら，埋め込む。200M－OUTは，与えられた再帰グラフから，その指定された頂点の中に描かれた再帰グラフを取り除く。EXTRACTは，与えられた再帰グラフから，その指定された頂点の中に描かれた再帰グラフを

取り出す。これらの作用子は階層設計及びモジュラー設計の立場から図2.2の様に分類される。

方法論 / 手続	階層設計	モジュラー設計
統　　合	200M-IN	JOIN
簡　　略	200M-OUT EXTRACT	SELECT

図2.2　統合・簡略作用子

2.5　設計過程の形式論*

COMSAC79で発表された，「A Design Process Formalization」では我々は，設計図と設計過程の両方の計算機処理化及び共有化のための形式論を提案した。そこでは，再帰グラフ，再帰グラフ作用子及び統合・簡略用の設計作用子（DELETE作用子が簡略用の作用子として追加されている。）

再帰グラフ作用子（原始作用子）は，挿入，消去，検索，展開，走査用に分類されている。挿入及び消去用としては，{CRN, CRA, INI, INO, IAI, DLN, DLA}の作用子が定義されている。これらの作用子群に対しては，次の最小完全性定理が証明されている。

最小完全性定理「{CRN, CRA, INI, INO, IAI, DLN, DLA}は最小完全である。すなわち，任意の再帰グラフがこれらによって生成可能であり，そのどんな真部分集合もこの性質をもたない。」

統合・簡略作用子における，頂点の融合に対する意味が説明されている。す

＊参考文献2

なわち，JOINにおける頂点xとx′の融合は，x及びx′によって表わされるシステムが同じであるため，統合された再帰グラフでは1つに融合されるべきであるという理由で行なわれる。一方200M−1Nにおけるポートxとx′の融合は，モジュールのインターフェースを表わす頂点xとそのモジュールの埋めこまれる環境のインターフェースを表わす頂点x′が，プラグがソケットにさしこまれる様にして結合されることを意味している。この様な融合の意味をふまえて，統合過程において次の同型条件が要求される。

同型条件「JOIN，ZIN，ZIA の2つのオペランドグラフの，融合される様要求された頂点及び弧からなる部分グラフは同型でなければならない。すなわち，融合される様指定された頂点及び弧は，2つのグラフにおいて，af，sn，sa，teによって同じ様に構造化されなければならない。又，ns及びasによって与えられるシステムレコード及び関連レコードは意味的に同じカテゴリーに属さなければならない。」

以上の様な形式性をもつ再帰グラフ理論に基づいた計算機支援の設計過程は，図2.3の様に行なわれる。

設計者は，設計図の初期入力を，一次元言語で宣言的に指定するか（string form），又はインタラクティブにグラフィクスを使って逐一入力していくか（graphic form）で行なう。この様な入力形式は，トランスレータ（translator）によって原始命令の列に変換され，これを解釈実行する設計図式機械（Design Schema Machine）によって実際にデータベース内に構築される。設計作用子は，設計者の判断に従って適時，インタラクティブにコマンド形式で入力され，そのコマンドがInterpreterによって解析され，入力された作用子を記述しているプログラム（これは原始命令からなっている）が設計図式機械によって実行され，データベース中の設計図式が適当に，更新，検索，解析，検証される。この様な，初期入力及び設計行為の結果は，定められたフォーマットのレポートとして，あるいは行為の結果を表わすグラフとして設計者に表示される。

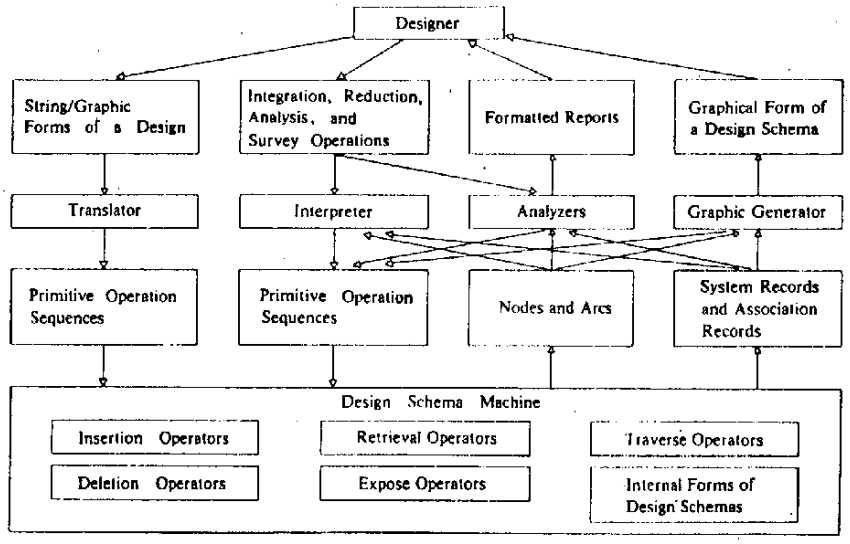


図 2.3. 計算機支援の設計過程

2.6 進化型製図形式論 (Evolutionary Drawing Formalizatinn)*

コンピューター・グラフィックスをエンジニアリングのＣＡＤに応用する場合の主要な問題の一つは，段階的に細かく図面を作成していける様な充分一般的な図面の表現法が無い事である。現在のＣＡＤは或る一つの詳細さのレベルでは充分助けになるが，もっと詳細が見たいとか，全体が見たいとかの様に色々な詳しさのレベルで図面を作成する場合，ほとんど助けにならない。リカーシブ・グラフが主にソフトウェア・デザインの為に提案されているが，階層的グラフィック・システムを達成するのに有効である。そこでリカーシブ・グラフを図面作成へ応用することを考え，その為のリカーシブ・グラフの表現法を提案する。ここではデーターベースのデーターとリカーシブ・グラフを整合させることに特に重きを置くことにする。例として，リカーシブ・グラフを化学

＊参考文献 3

プラント・デザインでの階層的図面作成に応用したものを用いる。この例ではブロック・ダイアグラムからフローシート，そして配管ダイアグラムと図面が次第に詳しく作成されていく（図 2.4.参照）。更に図面の論理的表現と物理的表現に関する問題を示す。又リカーシブ・グラフを２項関係に表現する表現の仕方を概念スキーマでの表現法として提案する。これにより既存のどのデータベースにもリカーシブ・グラフは乗せることができる。

リカーシブ・グラフは次の８つ組より成る。

$$R = (N,\ sn,\ ns,\ A,\ sa,\ af,\ aa,\ tl)$$

ここでNは頂点の集合。snは部分頂点写像で各頂点に対しその子頂点の集合を対応させる。nsは頂点意味写像であり各頂点に対し，その意味を対応させる。Aは有向辺の集合。saは部分有向辺写像で各頂点に対し子有向辺を対応させる。afは結合写像で各有向辺に対し始頂点と終頂点の順序対を対応させる。

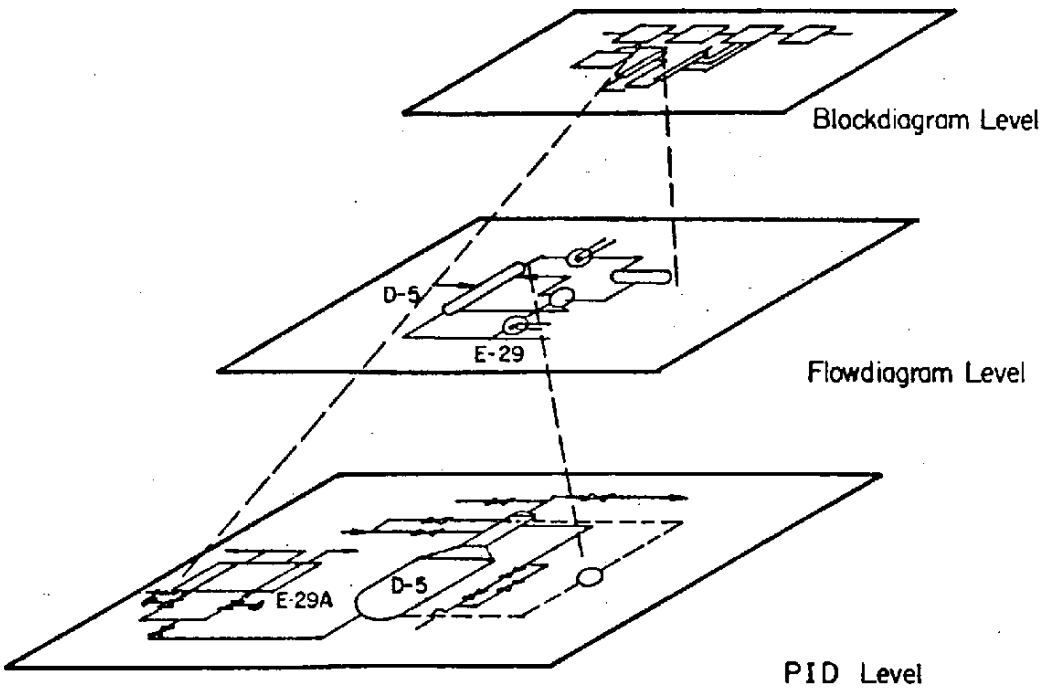


図 2.4.　設計図の進化的展開

as は有向辺意味写像であり，各有向辺に対し意味を対応させる。 tl は端点の集合である。端点は次数１の頂点で他のグラフへの結合要素として使用される。

グラフ理論的言語の多くで出来る操作に加えて８つのマクロ操作が定義されている。その内３つは統合操作で，４つは単純化操作であり，残りの１つが削除操作である。統合操作はＪＯＩＮ，ＺＩＮ，ＺＩＡ，単純化操作はＺＯＮ，ＺＯＡ，ＥＸＮ，ＥＸＡ，削除操作にはＤＥＬがある。

エンジニアリング・データベースでは元の定式化には，いくつか問題点がある。まずグラフ中の頂点や辺に対し子頂点や子有向辺の集合を決定するのに時間がかかる。又操作が行なわれるたびに頂点や辺の番号のつけ直しが行なわれる為に，部分頂点写像や部分有向辺写像を修正しなければならない。子頂点や子有向辺の情報はＥＸＮ，ＥＸＡ操作のみで使われ，その場合部分グラフがモジュールとして存在していなければならない。従ってモジュールへのポインターで部分グラフを表現しておけばＥＸＮやＥＸＡが出来る。今の場合統合や削除に操作を制限し，ポインターを使わないことも可能であるが，ポインターはそれ程の手間ではなく，有用な情報を表わすので一般性を考えここでは含めておくことにする。

次の問題は関係型式の様に構造の無いデータベース上にリカーシブ・グラフを表わす場合に生じる。各部分頂点写像や部分有向辺写像は大きさの一定しない集合で表わされる。関係型式で元の定式化のリカーシブ・グラフを表わそうとするとノーマリゼイションが問題となってくる。ここで考えているノーマリゼイションは第一段階のもので，関係型式の値域に単一の値だけが表われる様にする事である。

集合の要素が再び集合である様な重集合は避けることが望ましい。そうしないとノーマル型式で表現できないからである。

元の定式化で sn，sa，af が重集合の例である。リカーシブ・グラフの定式化を変更し重集合を避けると次の利点がある。

1） 構造情報はリカーシブ・グラフの部分に限られる。

2) 構造情報とそれ以外の情報との関係の自由度が増す。

3) リカーシブ・グラフ自体が関係型式で書ける。

リカーシブ・グラフを関係型式で頂点の関係Nと辺の関係Aの組（N, A）で表わす。Nの中の頂点n_iは（nid, ns, type, npointer）で定義される。ここでnidは頂点の数字による識別名でキーとなる。nsは元の定義と同じ。typeは頂点の種類を表わし通常頂点，端点，未定頂点からなる。npointerは更に詳細に書いたリカーシブ・グラフへのポインターである。同様に辺は（aid, as, bn, en, apointer）で表わされる。ここでbn, enは始頂点と終頂点である。typeが未定というのは，これから詳細を書くという様な時有効で，多くは表示されず単なる制限として働く。例えば化学プラントの設計でフローシートの利用フローは必要だが，処理フローに比べれば興味は薄い。これが配管ダイアグラムのレベルでは処理フローと同じものになる。

以上の情報は前の8つの操作をするのに充分である。

2項関係は概念スキーマを表わすのに便利な道具である。2項関係は関係名，主語，述語の3つ組で表わされる。NやAの様に単純なキーを持つ関係は2項関係で表現でき，そこから他のデータモデルで表わすことができる。Nを2項関係で書けば（nodesemantics, nid, ns），（nodetype, nid, type），（nptr, nid, npointer）又Aを2項関係で書けば（arcsemantics, aid, as），（arcbeqin, aid, bn），（arcend, aid, en），（aptr, aid, apointer）となる。

ポインターをグラフの辞書として使う為には更にnpointer, apointerをキーとした関係が必要になる。

化学プラントの図面展開には何レベルもあり，階層依存性がある。つまり配管ダイアグラムは論理的設計であるのに対しその詳細化の実体図面では空間的情報があるので，この2つの部分グラフを混ぜることは許されず，階層にはいくつかの階層があることが分かる。論理的情報がリカーシブ・グラフの操作の為に必要である。それは1つの階層から次の階層に受け渡される。物理的像は表

現に関係したものである。そこで図面情報を3つの部分に分けてしまうことが考えられる。

1) 論理的グラフの情報

2) 物理的表現の情報

3) その間の関係

ここでは2次元ダイヤグラム図面の表現を中心にする。問題なのは図面情報の標準の無いことであるが，このことも論理的情報と物理的情報を分けた方が良いことを示唆する。物理的情報は次の階層レベルには持って行かない。つまりフローの物性情報はフローシートには表われるが，配管ダイアグラムには表われない。

2次元図面の表現には3つの選択が可能である。1) 頂点も辺も人が指定する。2) 頂点は人が指定し辺は自動的に出す。3) 頂点も辺も自動的に出す。第一の方法は今までの製図と同じで，メモリーを多く使う。辺を記憶する時の大きさが変わるので面倒である。第2の方法は辺のメモリーが節約されるがそのかわり時間のかかるルーティング・アルゴリズムを使うことになる。第3の方法は頂点や辺の置き方の決定アルゴリズムと更に時間のかかるルーティング・アルゴリズムが必要になる。この場合ちょっとした操作によって見かけの全く異なった図面を生じる問題がある。

現在の所，2)の方式が最良である。図面生成の人間の思考過程と良く適合している。辺が次に置く頂点上にある時は，その頂点を置いてからディスプレイ・ファイルを作り直せばよい。この方式では論理情報と物理情報の関係が単純な1対1関係となる。

頂点の物理的像の最少限の情報は座標，識別図形，結合点識別記号とラベルである。頂点の形はエンジニアリングの図面で中心的な物であり，結合点識別記号は多くのシミュレーション・パッケージにおける整合性検査に使用される。表示順を同じにする為に表示順をデータとして持っても良い。辺の場合は識別記号やその他の情報を書く為の座標と水平に置くか垂直に置くかの配置情報

が必要である。

2.7 インタラクティブ設計システム*

NCC'80で発表する「SID:A System for Interactive Design」では，COMPSAC' 79で発表した，再帰グラフ理論に基づく，計算機支援設計過程を実現するためのシステムSIDのアーキテクチュアが提案されている。

SIDは，図2.5に示す様に，①設計者の視覚的診断のためのインタラクティブグラフィクス，②共有ファイル内の設計図の検証，蓄積，更新，検索，制御等を行う設計図式ベース，③再帰グラフ作用子を実行する再帰グラフプロセッサー，④設計作用子を記述したプログラムを蓄積，更新，検索，制御するデザインプロセスベース，⑤システム特有のグラフィクシンボルを登録・管理するグラフィクデータベース，及び⑥利用者とのコミュニケーションを管理するコマンド解析器から成る。

設計図式ベース内では，図2.6に示す様に設計図を表現する再帰グラフは，構造ベース，意味ベース，構造－意味写像ベースの3種の平坦なテーブルとして蓄積される。構造ベースでは，頂点識別子NID及び弧識別子AIDが，af, sn, pt, 及びsa の各関数を表わすテーブルの主検索キーとして使われる。意味ベースでは，タップル識別子TIDが，システムレコードSR及び関連レコードARを表わすテーブルの主検索キーとして使われる。構造－意味写像ベースでは，頂点識別子NIDとシステムレコードSRのタップル識別子の対がポインターアレイを形成してnsi 関数を表わし，同様に，弧識別子AIDと関連レコードARのタップル識別子の対がas 関数を表わしている。この様に，設計図式ベースでは，構造を表わす情報と意味を表わす情報を別々のテーブルにしまい，それらの対応をポインターアレイを使って行うので，構造及び意味の情報が独立に検索・更新が可能になる。

この論文では，前半においてSIDのアーキテクチュアの説明を行い，後半

＊参考文献4

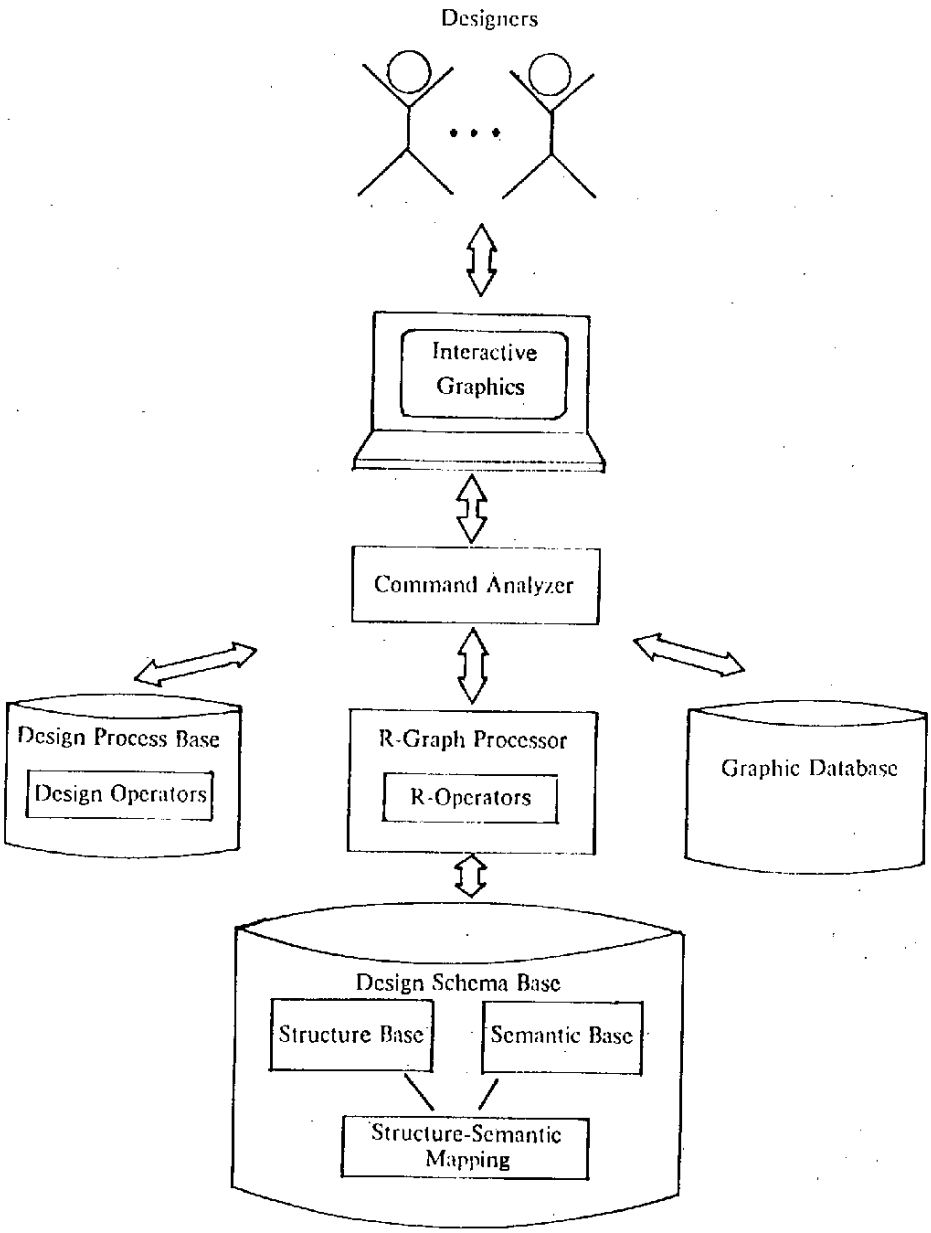


図 2.5　ＳＩＤアーキテクチュア

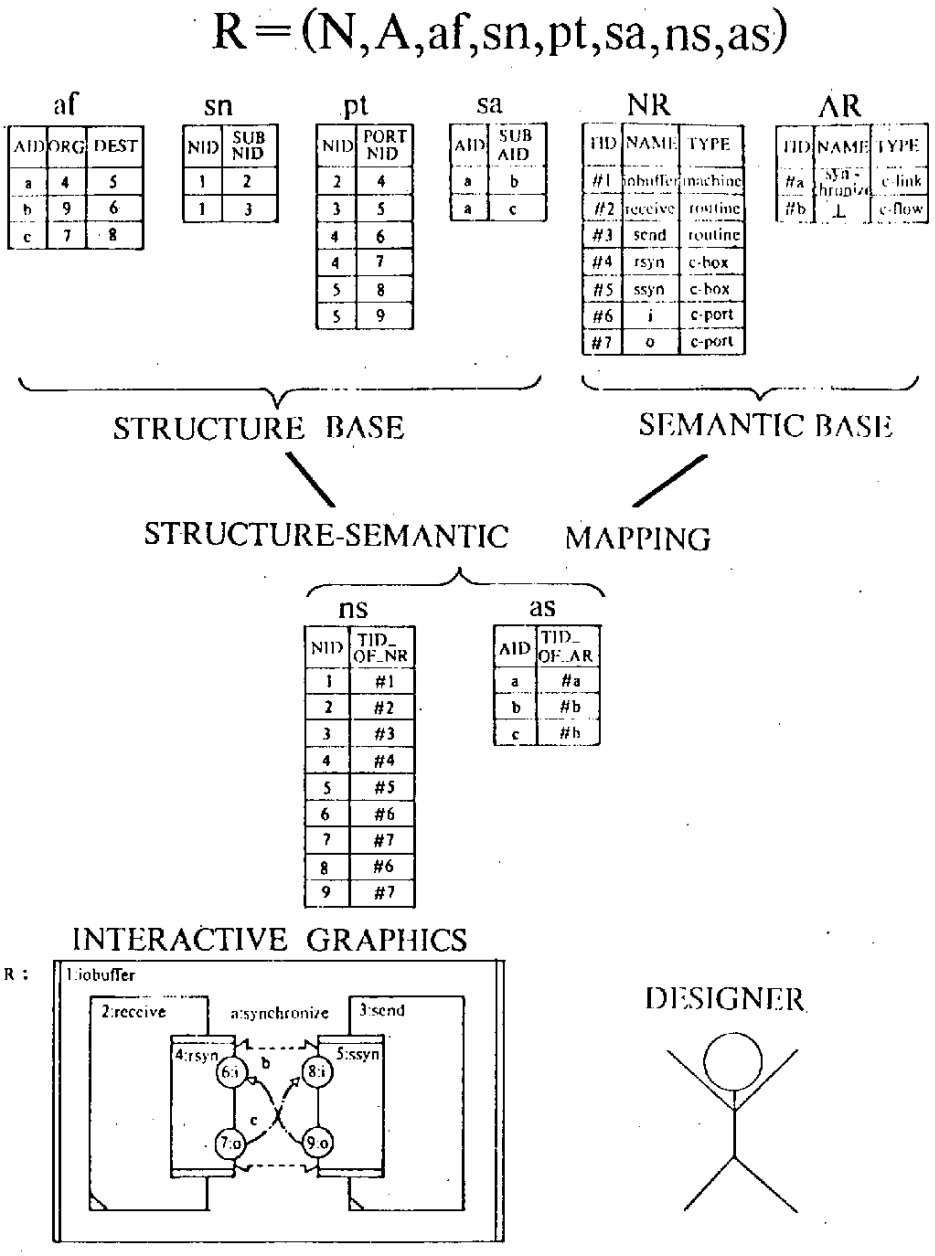


図 2.6　再帰グラフコンポーネント

において，設計作用子の説明を，並行動作システムの典型例であるパイプラインシステムを例にとって説明している。

この論文で，とりあげられている設計作用子としては，すでに公表した，[1,2,3,] 統合・簡略作用子の他に，調査作用子SURVと解析作用子MODが含まれる。

調査作用子は，与えられた設計図式を指定された視点から調査し，条件にあう部分設計図式を生成する。汎用調査作用子(SURVLR, NC, AC, SC)では，調査の視点は，頂点検索条件NC，弧検索条件AC，及び構造検索条件SCからなる。論文中では，SURVのアルゴリズムが，再帰グラフ作用子を使って記述されている。

一方，解析作用子の一例として，並行動作システムの与えられた設計図に表われる，マシーン及びルーチン間のモジュラリティーを計算する作用子が，再帰グラフの形式性を使って定式化されている。

これらの設計作用子は，任意の再帰グラフに適用可能なので，共有化することに意味がある。

2.8 グラフ設計記述言語*

MEDINFO '80 に投稿した「A Graphical Design Specification Language for a Hospital Administration System GDSL/HA」では，今までとりあつかわなかった，再帰グラフに基づくグラフ設計記述言語及び，設計図の検証を詳しくかつ精密に議論した。グラフ設計記述言語GDSLは，システム及びそれらの間の関係を言語要素とすれば，①言語要素の表現規則，②言語要素の許される結合を規定する整合規則，③言語要素の意味を記述する解釈規則からなる。

この論文では，GDSLの具体例として，病院の事務管理システムの設計用の言語GDSL/HAが定義されている。GDSL/HAでは，あつかわれるシステム

*参考文献 5

は，その性質に従っていくつかのタイプに分類され，図2.7に示す様に，視覚的理解に役立つ様なグラフィックシンボルが与えられる。関連も同様に分類され，図2.8に示す様なグラフィックシンボルを使って表現される。これらは言語要素の表現規則及び意味解釈規則を与えている。

			Type Name	Graphical Symbol	Meaning
Macro Active Systems			Department		A structured assembly of primitive systems which performs a hospital administration.
			Exceptional Procesing		A special kind of system which performs an exceptional processing.
Primitive Active System	Operation		Prepare	Ⓟ	An operation to prepare for filling in.
			Fill	Ⓕ	An operation to fill in a slip.
			Eliminate	Ⓔ	An indication that some activities are eliminated.
			Gather	⊗	An operation to gather something.
			Classify	⊗	An operation to classify something.
			Deliver	⊗	An operation to classify and deliver something to different departments.
			Count	ⓣ	An operation to calculate something.
			Communicate	◎	An operation to communicate.
			T-pick up	▽	An operation to pick up something from temporary storage.
			E-pick up	▽	An operation to pick up something from eternal storage.
			Operate	○	An operation which can not be classified into any of the above classes.
			Carrying	○	An operation to carry something.
	Storing		Receive	△	An operation to receive something.
			T-store	▽	An operation to store something temporarily.
			E-store	▽	An operation to store something eternally.
	Examination		Check	◇	An operation to examine the contents.
			Count	□	An operation to count the number of slips.
			Sign	Ⓢ	An operation to sign a slip.
Passive System			Slip		A blank slip which appears for the first time.
			Communication Port		An interface of a system for association.
			Article		An article or an actual thing other than a slip.

図2.7　GDSL/HAにおけるシステムの
表現及び解釈規則

	Type Name	Graphical Symbol	Meaning
Associations	Slip-flow	——→	A flow of slips.
	Article-flow	⟹	A flow of articles.
	Exceptional-flow	- - - - - - →	A flow of slips or articles which appears in an exceptional case.
	Association-link	—— - ——	An association other than flows (e.g., communication and comparison).

図 2.8　GDSL／HA における関連の
表現及び解釈規則

一方，整合規則は，正しい言語表現として許される，言語要素の組み合わせ様式を定める規則であり，例えば，GDSL／HA に対しては次の様なものがある。

1) 結合条件……各タイプの関連は定められたタイプのシステムのみを結合する。この許容結合の対は，結合行列で表現され，例えば，帳表の流れに対する結合行列では，i 行のタイプのシステムから j 列のタイプのシステムへ帳表の流れは，結合行列の ij 要素で指定される。

2) 目的地条件……Gather，Classify，Deliver，T－pickup，E－pickup タイプのシステムから出射する帳表流は，帳表（slip）タイプのシステムに入射する。

3) 湧き口／吸こみ口条件……任意の帳表流は，Slip，Prepare，Eliminate，T－pickup，E－pickupのどれかのタイプのシステムから始まり，Slip，Receive，Eliminate，T－store 又はE－store のどれかのタイプのシステムで終わる

4) 流量保存条件……Fill，Gather，Classify，Deliver，Count，Communicate，Operate，Carrying，Check，Sign タイプのシステムの入射帳表と出射帳表は集合として等しい。

5) 帳表流条件……任意の帳表流は，帳表を生成したり吸収したりすることはない。

これらの整合条件は，任意に与えられた設計図に対して，逐一適用され，みたされているかどうか調べられる。この様な検証作用子は言語ごとに定められて，単純な設計のミスを検出するのに役立つ。実際，この論文では，ある機関が提案した標準医事業務仕様を再帰グラフで書きなおした結果，各条件に対して違反する設計ミスがいくつかみつかった事が報告されている。

参考文献

1. Harada, M., Kunii, T.L., and Saito, M. (1978), "RGT: The Recursive Graph Theory as a Theoretical Basis of a System Design Tool DESIGN - TOOL — with an Application to Medical Information System Design," Proc. of MEDIS '78, OSAKA, Oct., pp. 503-507, 1978.

2. Harada, M., and Kunii, T.L. (1979a), "A Design Process Formalization," Proc. of COMPSAC '78, Chicago, Nov., pp. 367-373, 1979.

3. Buchmann, A.P., and Kunii, T.L. (1979), "Evolutionary Drawing Formalization in an Engineering Database Environment," Proc. of COMSAC '79, Chicago, Nov., pp. 732-737, 1979.

4. Kunii, T.L., and Harada, M., "SID: A System for Interactive Design," Proc. of National Computer Conference, NCC '80 (in press), Anaheim, May, 1980.

5. Harada, M., Kunii, T.L., Kitagawa, H., Kaihara, S., and Ohbo, N., "A Graphical Design Specification Language for a Hospital Administration System GDSL/HA — with the Application to Design Consistency Verification —," Submitted to the 3rd World Conference on Medical Informatics, MEDINFO '80, Tokyo, September, 1980.

3. 応用仕様記述とデータベース・システム
—進化型データベース・システム—

3.1 はじめに

コンピュータシステムが普及し，また社会的にも重要性が高まるにつれ，大量データ共用のためのデータベースに蓄えられるデータが増大すると共にさまざまな応用が考えられるようになってきた。また，データベースの導入時には考慮されなかった応用や新装置に対しても対応する必要が出てきている。

この分野の研究は大学・計算機メーカ等で活発に行なわれている。他研究に大きな影響を与えているものとして，理解しやすく応用の変化にも対応できる関係モデル（モデルとはデータの表現方法をいう）や，データベースシステムの柔軟な構造の標準を示したＡＮＳＩ／Ｘ３／ＳＰＡＲＣ（アメリカ規格協会計算機情報処理部門）中間報告書等がある。

進化型データベースシステムは「進化型（Evolutionary）」という言葉が示すように応用やハードウェアの変化に対して柔軟に対応し，またシステムの開発，拡張の各々の時期で最適に動作するシステムである。

東京大学国井研究室では進化型データベースシステム設計の研究を行うとともにその一形態としてＡＰＡＤ（Application Adaptable Database System：応用に適応できるデータベースシステム）と呼ばれる実験システムを提案した。ＡＰＡＤは事務自動化の応用を対象にしており，従来のシステムよりも柔軟で理解しやすいデータの見方をユーザに提供し，しかも複雑なデータベース内部のアクセスパス（データを検索する経路）を自動的に生成するという特徴を持っている。

3.2 進化型データベース・システムの概念

A. 進化型データベース・システムの目標

(1) データベース・システムのライフサイクル

図 3.1 に示すようにデータベース・システムにも人間の一生と同様にライフサイクルがあると考えられている。図 3.1 は，重なり合った 3 世代のデータベース・システムのライフサイクルを表わしている。各々の期間には次の作業を行なう。

1) 導入検討期

応用業務を分析し，必要なデータを検討する。機能・処理能力・費用等から使用するデータベース・システムを決定する。初期の応用プログラムの作成も行なう。

2) 作成期

応用業務の分析結果よりデータの論理関係，物理的な格納法を決定し，データを投入する。

3) 運用・拡張・保守期

(a) 運用

データベース中のデータを処理（検索・更新・追加・削除等）する。応用プログラムの実行や端末からの問い合わせに対する応答がこの段階に相当する。

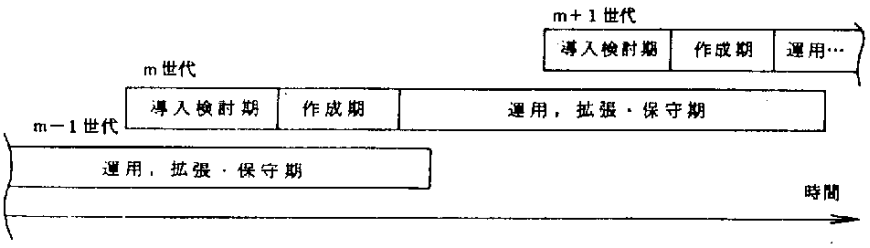


図 3.1 データベース・システムのライフサイクル

(b) 拡張・保守

作成時には考慮されなかった新しい応用のためにデータベースを適応させたりデータ削除によって生じた空き領域をつめるためにデータベースを再構成したりする。データベース適応の具体例としてはデータ項目の追加等が考えられる。このためにデータベースの再構成が必要であったり，さらに既存プログラムの修正が必要になったりする可能性がある。

(2) ライフサイクル費用の低減

ライフサイクル費用はデータベース・システムのライフサイクル中に発生する費用の累計である。発展的データベース・システムは従来のデータベース・システムよりもライフサイクル費用を低減することを目標にしている。

データベース・システムの導入，作成をより容易にすべきであるという意見があり，この考え方に沿えば運用と拡張・保守に要する費用がライフサイクル費用の大部分を占めることになる。

1) 運用時の費用対象となる処理

(a) データ処理

(b) データベース調整のための監視

2) 拡張・保守の費用対象となる処理

(a) 新しい応用の分析

(b) 新しい応用プログラム開発

(c) 新しい応用のためのデータベース適応（データベースの部分的／全体的な再構成）

(d) データベース適応に伴う既存応用プログラムの変更

(e) データベースの調整

運用費と拡張・保守費には図 3.2 に示すように密接な関係がある。すなわち拡張費用を低減するために応用に対する適応性をシステムに持たせようとすれば応用プログラムと物理媒体上のデータ間で変換を行なわなければならない。この変換はデータ処理の費用を増加させる。また，運用時のデータ追

加，削除に対し，部分的な再構成をすることにより全体的な再構成を不要にすることができるが，これもデータ処理費用を増加させる。

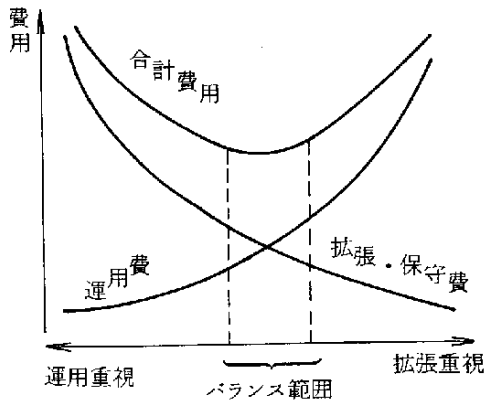


図 3.2　運用費と拡張・保守費の関係

既存システムでは拡張・保守費よりも運用費を重視している。たとえば作成時のデータの見方と異なる応用では性能が低下したり，レコード中の項目の追加に対してはデータベース全体の再構成が必要になったりする。さらには，既存プログラムの変更が必要になる場合がある。

ライフサイクル費用を低減するためには，運用費と拡張・保守費のバランスをとることが重要である。

B．発展的データベース・システムの機能

ライフサイクル費用を低減するために発展的データベース・システムでは次のような機能の強化，または追加が必要であると考えられる。

1)　運用費を低減するために

(a)　きめ細かなデータベース調整

(b)　容易な物理装置の変更

2)　拡張・保守費を低減するために

(a)　簡単で強力な応用ごとのデータ定義

（異なるレコード上の項目組み合わせ等を含む）

(b)　動的な論理関係の作成・削除

(c)　データ処理時の部分的再構成

(d)　データベース調整のための情報取得

(e)　データベース調整の自動化

上記の他，プログラマでないユーザのためのエンドユーザ言語も必要である。また，新しい応用の開発時においてはデータの見方の柔軟な変更を許し，応用が定型化したなら柔軟性よりも性能を重視して固定してしまう（プログラムと物理媒体上のデータとの変換回数を減らす）ような静と動との境界をゆるやかにする（Static／Dynamic relaxation）という概念が提案されている。

C．進化型データベース・システム設計

東京大学国井研究室では，上記要求を満たすためのデータベース・システムの基本的アーキテクチャについて検討した。先ず，ソフトウェアのライフサイクルで必要とされるコストという観点より，従来のデータベース・システムの問題点を考える。次に，データベースのメインテナンスのための，従来からの技法を概括する。その中には，IMBのDBDA，IMS／VS Monitor のようなソフトウェアツールによるアプローチ，CODASYL DBTGおよびANSI／X3／SPARCによるスキーマを通したアプローチ，ADABAS におけるようなインデックス・ジェネレータ・アプローチが含まれている。

ここで提案する〝進化型データベース・システム（Evolutionary Database System)〟の構成は図3.3のような3レベルの階層構造をなす。応用レベルは応用仕様言語（ASL）で記述され，論理構造レベルは拡張されたDDL／DMLによって記述される。応用レベルと論理構造レベルの間には応用写像関数が，論理構造レベルと物理構造レベルの間にはアクセス関数が存在する。

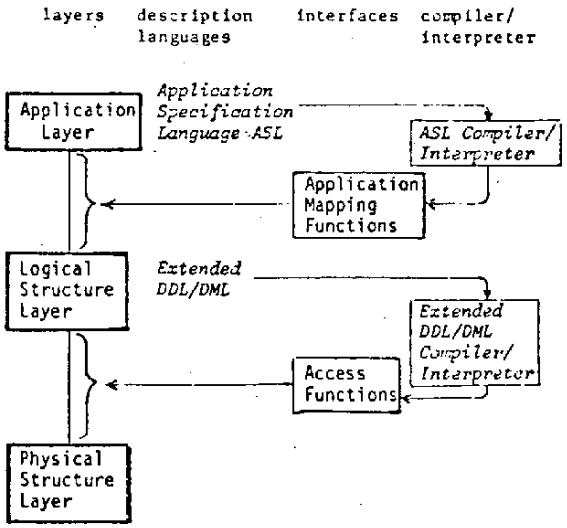


図3.3　進化するデータベース・システムの階層構造

これらは，それぞれ応用仕様言語および拡張されたDDL／DML を入力とするトランスレータによって，生成される。我々の進化するデータベース・システムにおいては，応用仕様言語における記述の中で，ある応用のどの部分が動的に変化が生じ，どの部分が静的に安定して存在するかを区別して，仕様化することができる。このような，システムの動的・静的特性の記述は，トランスレータが各レベル間の写像関数を生成する際に影響をもつ。すなわち，静的であると指定されている部分については，コンパイル方式の変換を行なうことによって，応用の実行時の効率を上げるような措置を行なう。一方，動的であると指定されている部分については，インタプリティブ方式の変換を行なうことによって，応用の動的変化に対する柔軟性をもたせる。このような，ハイブリッド構造のトランスレーション方式を用いることによって，応用の変化に対する柔軟性と，実行時の効率の向上という相反する二つの要求を同時に満たすことが可能である。このような，システムの動的・静的特性を可能にするために，拡張されたDDL／DMLでは，DMLによってDDLによってなされたデータベースの論理的構成の記述を変更することができる。従来のデータベース・システムにおけるDDL／DMLでは，データの検索，削除，挿入，変更等の操作だけがDMLでは許され，データベースの論理的構成を変更することは許されていなかった。このような拡張は，今後の応用適応型のデータベース・システムを考えてゆく上で，極めて重要となってくるであろう。

3.3 実験データベース・システムAPAD

東京大学国井研究室ではAPAD（Application Adaptable Database System）と呼ばれるデータベース・システムを提案している。このシステムは進化型データベース・システムの一形態として，特に事務自動化（表作成）を対象にした研究のためのシステムである。

(1) 用　語

APADで使われている用語を説明する。

① 表(table)

一般的な表と同様，二次元の枠組みとその中に埋め込まれた文字列で，APADではユーザのデータはすべて表の形で表わされているものとしている。APADにおいては表の名前と見出し表の形で表わされが，他のシステムでデータの見方を決めることに相当する。

表には2種類ある。

(a) 平坦な表(flat table)

その属性が横一列に並んでいる表をいう。データモデルのうち，関係モデルを表わしている(図 3.4 参照)。

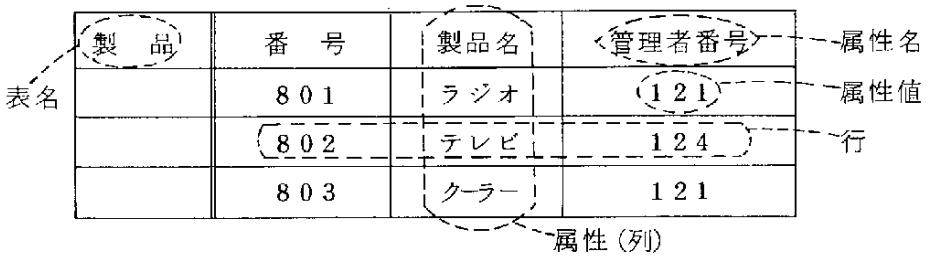


図 3.4　平坦な表の例

(b) 構造をもった表(structured table)

属性の中に細分化された属性を含む表をいう。これは階層モデルを表わしている (図 3.5 参照)。

製品部品	製品名	部　品		
		番号	部品名	個数
	ラジオ	4001	低　抗	40
		4002	コンデンサ	10
	テレビ	4001	低　抗	80

属性名(集合属性名)

属性名

(単純属性名)

図 3.5　構造をもった表の例

② つなぎ（ link ）

つなぎとは２つの表の中の行と行の対応を表わすものである。ユーザは，つなぎを使用してある表の行から別の表の対応する行をたどることができる。つなぎは網モデルを表現する際に使用する。

③ アクセスパス（ access path ）

実行時のデータアクセスを高速にするためのアクセスの経路である。アクセスファシリティ（ access facility ）とも呼び，３種類ある。

(a) SELECTOR

属性値によって行を選択する条件である。行が選択された新しい表であると考えてよい。一般形は式（ 3.1 ）のようである。

SELECTOR （T, SC）……（3.1）

ただし　T：表名

SC：条件並び

条件並びの中では属性名と定数による条件，属性名と属性名による条件，およびそれらの組み合わせが許される。

たとえば図 3.4 で製品名がラジオでかつ管理者番号が１２１の行を選択するSELECTORは式（ 3.2 ）のようになる。

SELECTOR （製品；製品名＝ラジオ　∧

管理者番号＝１２１）……（3.2）

ただし∧は〝かつ〟を表わす。

(b) LINK

②のつなぎを表現するためのアクセスパスであり，２つの表を結合する条件である。SELECTORで行が選択された表を結合してもよい。一般形は式（ 3.3 ）のようである。

LINK （T_1' , T_2' ；LC）……（3.3）

ただし　T_1' ：表名１またはSELECTOR1

T_2' ：表名２またはSELECTOR2

LC：条件並び

条件並びの中では属性名と属性名による条件，およびその組み合わせが許される。

たとえば式（3.4）のようである。

LINK（製品，製品部品；管理者番号＝番号）……（3.4）

管理者番号は製品表中の属性名であり，番号は製品部品中の属性名である。

(c) IMAGE

行をある属性値に従って順序付けする条件である。SELECTORで行が選択されていてもよい。一般形は式（3.5）のようである。

IMAGE（T′, A）……（3.5）

ただしT′：表名またはSELECTOR

A：属性名と値の上昇順，下降順指定の並び

ただし属性値の上昇順，下降順指定は，本研究では省略している。これについてはデータ検索言語QBE（Query By Example）で用いられているように，AO（Acsending Order：上昇順）またはDO（Descending Order：下降順）の接頭辞で指定する方法が考えられる。

IMAGEの例を式（3.6）に示す。

IMAGE（製品；番号，管理者番号）……（3.6）

式（3.6）は製品表中の行を番号属性の値で順序付けし，同じ番号中では管理者番号の値で順序付けすることを意味している。

④ 平坦化（flattening）

構造をもった表を平坦な表に変換する操作をいう。平坦化した結果の表には属性値によって行をまとめるためのIMAGEが生成される場合がある（図3.6参照）。

製品部品	製品名	部　品	
		番号	個数
	ラジオ	4001	40
		4002	10
	テレビ	4001	80

製品部品	製品名	番号	個数
	ラジオ	4001	40
	ラジオ	4002	10
	テレビ	4001	80

IMAGE（製品部品；製品名）

図 3.6　平坦化の例

図 3.6 中の IMAGE は元の構造をもった表が，製品名でまとめられているために生成されたものである。すなわち平坦な表の行をある属性値によってまとめておき，冗長な値（図 3.6 中では，“ラジオ”）を 1 つにすれば，構造をもった表を再現することができる。

(2)　システムの概要

APAD は，事務自動化（表作成）を対象とするシステムである。ユーザの応用を重視し，ユーザのデータの見方をデータベース中のアクセスパスに自動的に反映するという特徴を持っている。

①　簡単で強力なデータ定義

APAD では環境の変化に対して既存プログラムを安定にする（変更をなくす）ため ANSI／X3／SPARC の提案する 3 層のシステム構造 を採用している。すなわち，ユーザに近い外部レベル，中間の概念レベル，そして物理媒体に近い内部レベルからなる構造である。

内部レベルでの表現については検討していないが，概念レベルのデータモデルは理解のしやすさ，共用のしやすさのために関係モデルを採用している。外部レベルでは階層，網，関係のすべてを使用可能であり，その記述のために簡単で強力なデータ記述言語 DSE（Database Specification by Example）をサポートしている。

DSEは，既念レベルの平坦な表から外部レベルの平坦な表,構造をもった表，およびそれらの間のつなぎを記述する表形式の言語である。また，ユーザの応用の性質をよく記述するためにモード（データ処理時の処理速度指定）ライフタイム（寿命）を，定義する表ごとに指定することができる。

なお，DSEは一般ユーザでなくデータベース管理者(ANSI/X3/SPARC提案の用語でいえば応用管理者）が使用することを想定している。

② アクセスパスの自動生成

APADでは既念レベルにおいて関係モデルを採用しているためアクセスパスが存在しない。しかし，データ処理時にデータ値を参照して表の結合等を行なうのでは処理速度が低下してしまう。このためAPADではDSE記述時にあらかじめアクセスパスを作成しておく方法を提案している。DSE 記述からアクセスパスを自動的に生成するために，DDA（DSE Description Analyzer）と呼ぶプログラムを用意している。

DDAはDSE記述を入力して概念レベルの関係モデル上でのアクセスパス（アクセスファシリティとも呼んでいる）を自動的に生成する。アクセスパスは概念レベルから内部レベルに反映され，その結果実行時には高速な検索ができるようになる。すなわちDDAは，応用をよく反映したデータベース調整を人手を介さずに行なうためのプログラムである。

図3.7にAPADの構造を示す。

(3) データ記述言語

1) 概　要

データ記述言語DSEはAPADの外部モデルを記述するために提案された表を用いた例題型（example oriented）の言語である。

DSEの起動や，例題を表へ埋め込む方法等の実際の使用法については詳細な検討をしていないが，同じ表形式で例題型のデータ検索言語であるQBEの方法に従う方針である。すなわちDSE処理を起動することにより端末の画面上には表の枠組みだけが表われる。データベース管理者はこの枠組みに

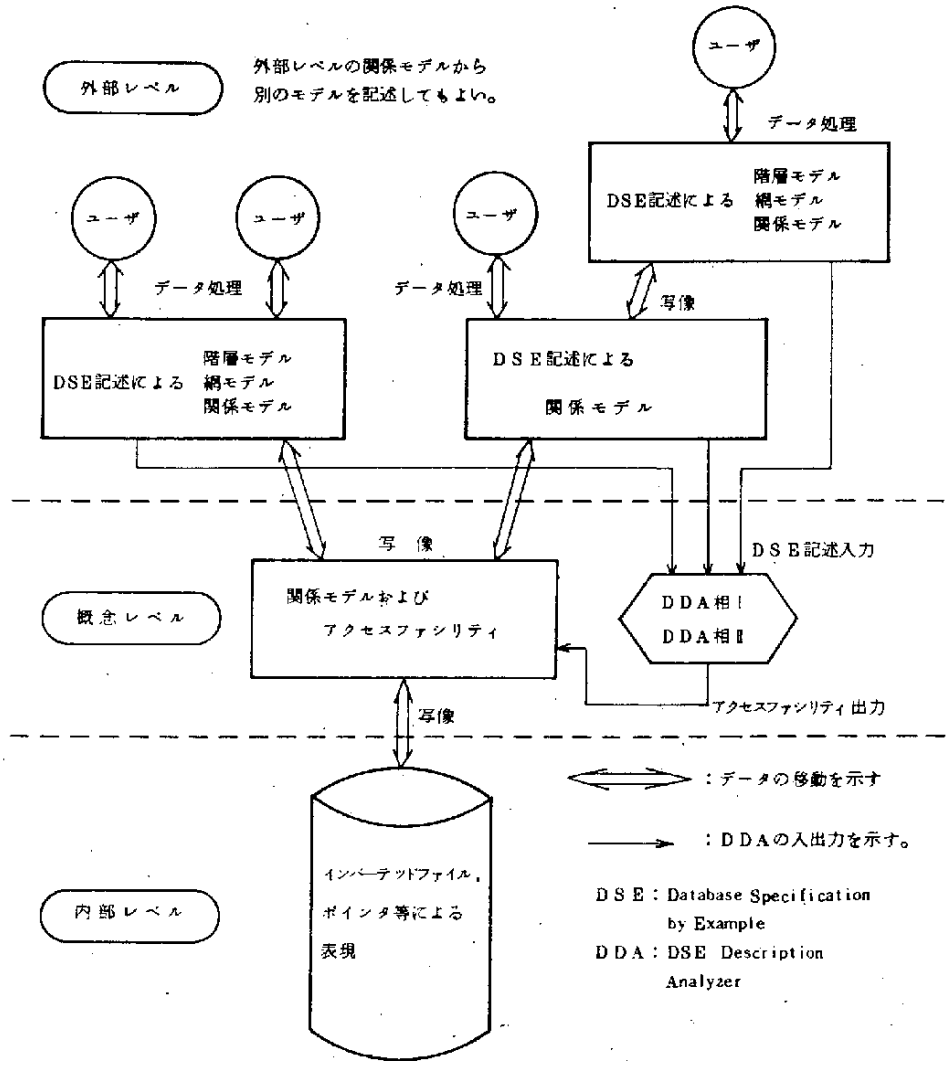


図 3.7　APADの構造

対してキーボード操作により表名，属性名，例題を埋め込み，システムに連絡する。

DSEによって概念レベルの関係モデルから外部レベルの任意の関係モデル，階層モデル，および網モデルを表現することができる。それらは概

念レベルの平坦な表から，それぞれ外部モデルの平坦な表，構造をもった表，および表間のつなぎを作成することによって行なわれる。特に表を作成する操作はコッドの提案した関係演算と同等以上の能力を持っている。

2) DSE記述の構成

新しい表またはつなぎは任意の時点で，任意の順に作成してよいが，手続きの説明上，DSE記述の構成を定めている（図3.8参照）。

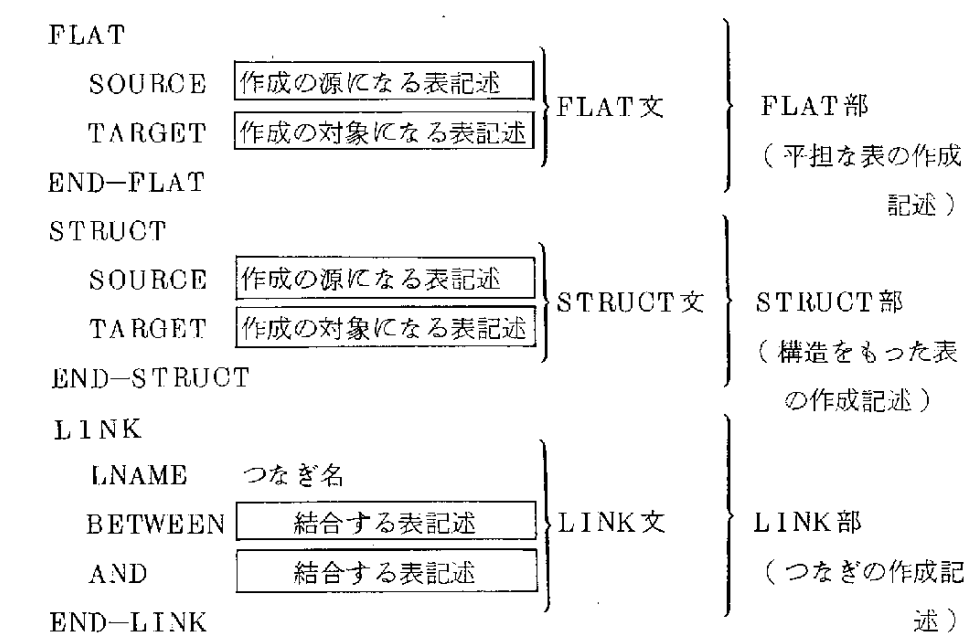


図3.8　DSE記述の構成

3) DSE記述例

本項では例を上げてDSEの記述力の一部を示す。なお，モード，ライフサイクルの指定は詳細が決定されていないので省略している。

以下の例で用いる概念レベルの平坦な表の表名，属性名のみを図3.9に示す。

表名	属性名			
従業員マスタ	番号	氏名	管理者番号	年齢
工場マスタ	番号	工場名	住所	
工場製品マスタ	工場番号	製品番号	生産量	
製品マスタ	番号	製品名	管理者番号	発売年
製品部品マスタ	製品番号	部品番号	使用量	
部品マスタ	番号	部品名	仕様	

ただし，従業員マスタ表で，自分自身が管理者である場合，番号属性値と管理者番号値は同一であるとする。

図 3.9　DSE記述例で用いる概念レベルの表

(a)　平坦な表の作成

(i)　例1.　工場表を作成する（図 3.10 参照）。

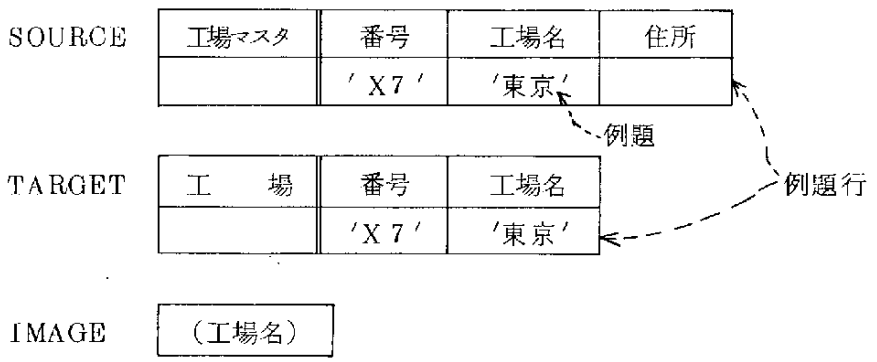


図 3.10　DSE記述例1

文字SOURCE，TARGETはそれぞれ続く表が表作成の源である対象であることを示す。また，文字IMAGEは，TARGET表中のその属性に対しアクセスパスIMAGEを生成せよという指示である。

ここで引用符'で囲まれた文字列は「例題」であり，値に意味はなく，同じものが指定されている属性と対応していることを示す。すなわち，工場表は工場マスタ表の番号属性，工場属性を取り出した表である。対応のないSOURCE表上の属性，例1の場合の住所属性の下は空白でもかまわない。

TARGETの表名はSOURCEの表名と異なるものでなければならない。TARGETの属性名は適当につけ直してもよい。

例1では工場表の工場名に対してIMAGE作成を指示している。このため工場名属性での検索は速く行なえるが，番号属性にはIMAGEが作成されないので検索は表全体に対して行なわれるようになる。IMAGEは省略してもよい。

また，例題の値に意味はないが直感的な理解のしやすさのために属性値に似たものが望ましい。

(ii) 例2. 管理者表を作成する（図3.11参照）。

SOURCE

従業員マスタ	番号	氏名	管理者番号	年齢
	'1'	'西田'	'1'	

TARGET

管理者	番号	氏名
	'1'	'西田'

IMAGE

（氏名）

図3.11　DSE記述例2

従業員マスタ表上の番号属性，管理者番号属性に同じ例題 '1' を指定することにより，管理者の行のみを選択できる。（従業員が管理者である場合，番号と管理者番号には同じ値がはいっている。）

管理者表は従業員マスタ表上で管理者の行を選択した後，番号属性と氏名属性を取り出した表である。

(iii) 例3. 若年担当者表を作成する（担当者とは管理者でない従業員をいう）（図3.12参照）。

SOURCE

従業員マスタ	番号	氏名	管理者番号	年齢
	'1'	'西田'	'2'	'24'

論理否定記号 ¬

管理者	番号	氏名
¬	'1'	'西田'

TARGET

若年担当者	番号	氏名	管理者番号	年齢
	'1'	'西田'	'2'	'24'

IMAGE （（番号），（氏名，年齢））

WHERE '24' ≦ 30

図3.12 DSE記述例3

30才以下の担当者からなる表を作成する。管理者表の論理否定記号¬は，従業員マスタ表から管理者表と同じ番号，氏名の行を除くことを示す。このように，作成した平坦な表をSOURCE中に指定してもよい。

IMAGEは次の2つのIMAGEを作成せよという指定である。

IMAGE（若年担当者；番号） …… （3.7）

IMAGE（若年担当者；氏名，年齢）…… （3.8）

WHEREは条件を示す。例4の場合,年齢属性に書かれた‘24’という例題に対し，‘24’≦30という条件をつけたので，30才以下という条件になる。

(b) 構造をもった表の作成

構造をもった表の作成は平坦な表の作成と見出しの書き方が異なるのみである。

例4として新製品表を作成する。新製品表は1980年に発売される製品とそれが使用する部品の情報を示す表である（図3.13参照）。

SOURCE

製品マスタ	番号	製品名	管理者番号	発売年
	'101'	'ラジオ'	'1'	1980

定数

製品部品マスタ	製品番号	部品番号	使用量
	'101'	'2500'	

部品マスタ	番号	部品名	仕様
	'2500'	'TR'	'2SC458'

TARGET

新製品	番号	製品名	管理者番号	部品		
				部品番号	部品名	仕様
	'101'	'ラジオ'	'1'	'2500'	'TR'	'2SC458'

IMAGE

（製品名，部品名）

図3.13　DSE記述例4

SOURCE製品マスタ表の例題行中の「１９８０」は引用符で囲まれていない。これは定数であり，発売年属性値が「１９８０」である行のみを選択する条件を示す。条件を示すのにWHERを使用しない書き方も許される。もし「１９８０年以降の」という条件に変更したければ「1980」を「≧１９８０」と書く。

またSOURCEの表がいくつかあり，それらが例題で関係づけられているときは表を結合することを示す。例４の場合，例題‘１０１’で製品マスタ表と製品部品マスタ表を結合し，さらに例題‘２５００’で部品マスタ表を結合する。すべての表を結合した後，必要な属性を取り出す。

表の結合に条件を指定してもよい。

(c) 表のつなぎの作成

表のつなぎは表の作成における表の結合と同様の書き方をする。

(i) 例5. つなぎMANAGEを作成する。

例２と例３で作成した管理者表と若年担当者表の間に「MANAGE」というつなぎを作成する（図3.14参照）。つなぎMANAGEにより，管理者表中の行からその管理者が管理（manage）する若年担当者表中の対応する行をたどることができる。

LNAME

MANAGE

BETWEEN

管理者	番号	氏名
	'1'	

AND

若年担当者	番号	氏名	管理者番号	年齢
			'1'	

図3.14　ＤＳＥ記述例５

また，逆に若年担当者から管理者をたどることもできる。

関係づけられる表は平坦な表でも構造をもった表でもよい。

関係づけに大小関係が必要であれば表作成の場合と同様にWHERE を使用したり，例題，定数の前に演算子をつけたりしてよい。

(ii) 例6. つなぎSHIPを作成する。

工場表と新製品表との間につなぎSHIPを作成する（図3.15参照）。つなぎSHIPにより工場が発送（ship）する新製品をたどることができる。

なお，工場製品表は既に作成されているものとする。

LNAME

SHIP

BETWEEN

工場	番号	工場名
	'X7'	

AND

新製品	番号	製品名	管理番号	部品		
				部品番号	部品名	仕様
	'101'					

USING

工場製品	工場番号	製品番号
	'X 7'	'101'

図 3.15 DSE記述例 6

例6のように属性値で直接関係づけられない表の間をUSINGで示される別の表を使って補うことによりつなぎを作成できる。

USINGで平坦な表，構造をもった表をいくつか指定してつなぎを

作成してもよい。

(4) アクセスパス解析

DDAはDSE記述を入力し，その記述で必要とする概念レベルの関係モデル上のアクセスパスを解析，生成する。DDAの処理は相Ⅰ，相Ⅱに分けられる。

1) 相Ⅰ

DDAの基本の相で，DSE記述で必要とするすべてのアクセスパスを生成する。

(a) LINK部の解析

(i) 必要ならばSELECTORを生成する。

(ii) LINKを生成する。

たとえば前節の例6からは，次のLINKが生成される。

LINK(工場，工場製品；番号＝工場番号)……(3.9)

LINK(工場製品，新製品；製品番号＝番号)…(3.10)

(b) FLAT−STRUCT部の解析

(ii) 必要ならSELECTOR，IMAGE，LINKを生成する。

(iii) TARGETの表上にアクセスパスが生成されていれば，それらをSOURCEの表上のアクセスパスに置き換える。

この操作をアクセスパスの分解と呼ぶ。

たとえば，式(3.11)のIMAGEは前節の例4のDSE記述の処理が終わったときには次の2つのIMAGEに分解される。

IMAGE(SELECTOR(製品マスタ；発売年＝1980)；番号，製品名，管理者番号)　……(3.12)

IMAGE(製品部品マスタ；製品番号)　……(3.13)

アクセスパスの分解により，外部モデル上でのアクセスパスを概念モデル上のものに置き換えていく。また，生成過程で重複するアクセスパスは1つにまとめられる。最終的にはあるアクセスパスはシステム内に1つだ

け生成，維持される。ユーザーはデータベース中のデータのみでなく，アクセスパスも共用することができる。

2) 相Ⅱ

システムのオーバーヘッドを低減するために相Ⅰで生成されたアクセスパスの最適な組み合わせを決定し，数を減らす相である。

相Ⅱの詳細なアルゴリズムは決定されていないが，次の基準を考慮する予定がある。

(a) 包含関係

全く同一のアクセスパスは相Ⅰで除去されているが，包含されるものも除くようにする。

たとえば式(3.15)のIMAGEは式(3.14)に含まれるので除くべきである。

IMAGE(従業員；番号，氏名) ……(3.14)

IMAGE(従業員；番号) ……(3.15)

(b) 関数従属性

関数従属性とはある属性値が決まると別の属性値が一意に決まる性質をいう。

この性質を利用すればLINKの条件並びの数，IMAGEの属性並びの数を減らすことができる。

(c) 使用頻度

データ処理中にアクセスパスの利用回数を数え，良く使われるものだけを残す。

(d) 処理の効果

処理の効果の大きいものを残す。たとえば行数の多い表上の数行を選択するSELECTORは処理の効果が大きい。

(e) 有効なアクセスパスの共用

既に生成されているアクセスパスを組み合わせて，新しいアクセスパスの代用にならないかをチェックする。

4. ビジネス自動化システム
ートリガー機構と画像インタフエイスー

4.1 トリガー機構

(1) はじめに

データベースを管理する上で，ある条件を満足した時に特定の処理をとらなければならないことがある。例えば，在庫管理を行なっているデータベースでは，ある品物が一定量以下になった場合は，最低限，それを知らせるメッセージを出すことが必要であろうし，さらに自動的にその品物の注文を出すような機構があると便利である。もちろん，データを変更するプログラムで，条件をチェックして，その処理をとるようにすればよいのだが，それではプログラムが，特別な条件のチェックのために，非常に複雑なものとなるであろう。上のような機構を実現するためには，「ある条件が満たされた時にだけ，呼び出される手続き」が定義できれば，便利であろう。そのような手続きは「トリガ」と呼ばれ，M．ZloofのQBA(Office and Business Automation)で実現されている。

トリガの基本的概念は，新しいものではなく，ハードウェアの割込機能や，PL/IのON文の考え方と同じである。しかし，QBAのトリガは，それを非常に一般化したものであり，検査すべき条件，行なうべき手続きも，利用者が自由に定義できるようになっている。それだけに，データ構造が不適当であると，オーバーヘッドが大きくなりすぎ，実用にはならないことになる。

本稿では，QBAのトリガの機構を形式的に定義した後，実用的なトリガを実現するために必要と思われるデータ構造，アルゴリズムを示すことにする。また，話を簡単にするため，元になるデータベースは，関係形式データモデルによるものだけに制限し，それの一般的な用語は，定義しないで使用する。

(2) 定　義

先ず，トリガの持つべき性質を示す。

1) トリガは，データベースの状態がある条件を満たさない間は，眠っていて，その条件が満たされると，起きる（あるいは，目覚める）ような一種の手続きである。

2) あるトリガが目覚めた影響として，普通の手続きが呼び出され実行されたり，別のトリガが目覚めたりする。

3) トリガは，普通の手続きとは違って陽に，呼ばれることはない。

トリガが目覚めた時，別のトリガが目覚めるというのを，能動的に，別のトリガを起こすと考えれば，「トリガを呼び出している」ことと同じであるが，利用者から見た場合，性質3)を満たしているようにしたい。

トリガが目覚めるかどうかは，データベースの状態にのみ依存することであるが，トリガを実用的なものとするためには，どうしても，「いつ，状態を調べるか」を指定する必要がある。

以上のことを考えると，トリガを定義する際は，次の3つを指定することが必要である。

1. トリガ名
2. 評価時
3. トリガ式

詳細は後述するが，トリガ名は，トリガを一意的に指定する識別子であり，トリガ式は，データベースの状態の満すべき条件を示す論理式であり，評価時は，トリガ式を評価する時刻を指定するものである。

トリガが目覚めた時の影響のうち，普通の手続きを呼び出す手段として，次の2つの要素をもつ文が，書けなくてはならない。

1. トリガ名
2. 動　　作

これは，トリガ名で指定された，トリガが目覚めた時，動作で指定された

処理が行なわれるという意味をもつ。これは，普通のプログラミング言語のＩＦ文に相当するもので

if（トリガ名） then（動作） ……（＊）

と書くことにする。しかし，ここで注意しなければならないのは，トリガ名を普通の意味の論理式と同一には扱えない点である。

トリガが目覚めている状態を真，眠っている状態を偽と対応させた時，いくつかのトリガの論理和をとることは意味があっても，論理積や，否定をとることは無意味である。トリガは，普通は眠っていて，特別な時にだけ目覚めるものであると期待すると，否定をとることが無意味なのは，明らかであろう。論理積が無意味なのは，無関係なトリガの論理積は一般に，ほとんど常に偽となるからである。従って，ここでは，論理和だけ考えることにし，トリガ名の所に，複数個のトリガ名を書くことを許す。意味は，トリガ名のうちどれか１つに相当するトリガが目覚めた時動作を行なうこととする。

以下，トリガの定義のための３つの要素について考えてみる。

1) トリガ名

トリガ名は，トリガを識別するための名前であり，システム内において，一意的でなければならない。しかし，多数の使用者がいる環境の下で，一意的な名前をつけることは難しいであろうから，システム内で一意的な領域で，一意的な名前であればよいとすることができる。そのような場合，一意的な領域の識別子と，使用者が指定したトリガ名の組を，完全トリガ名と呼ぶ。

完全トリガ名を用いれば，システム内のどのトリガをも使うことができ＊るが，そのトリガと同一の領域内で使う限りは，領域の識別子を省略してもよい。

＊　機密保護のために，他の利用者に対してトリガの使用を制限することも必要だが，その問題はここでは扱わない。

以下では，特にことわらない限り，トリガ名と言えば，完全トリガ名のことを表わすものとする。

2) 評価時

評価時とは，いつトリガ式を評価するかを定めるものである。これを指定することによって，無駄な評価を避けることができる。

評価時を指定するものは，次の3種類に分類される。

1. 修　正
2. 時刻・頻度
3. トリガ名

修正は，データベースに対する操作であり，最も基本的なものである。眠っているトリガが目覚めるのは，データベースの状態が変化した場合に限られるから，トリガ本来の目的は，これだけでも一応果すことができる。しかし，これだけを使ったのでは，やはり，オーバーヘッドが大きくなることが予想される。2.や3.をうまく使うことがシステムの効率を上げることになる。

時刻・頻度というのは，データベースの状態とは無関係に，外界の時刻や，1時間ごと，1日ごと等という頻度を指定するものである。

トリガ名は，他のトリガが目覚めた時，今定義するトリガ式を評価することを指定する。トリガAが目覚めた時，トリガBのトリガ式が評価されるということを「トリガAがトリガBを呼ぶ」という。

評価時に，修正を指定したものを「修正トリガ」と呼び，時刻・頻度を指定したものを「時間トリガ」，トリガ名を指定したものを「トリガ・トリガ」と呼ぶことにする。また評価時としては，上の3種類の指定を混ぜて1個以上並べて用いることができ，それらは論理和で結合されているとする。

3) トリガ式

トリガ式は，データベース中のデータの値により，真偽が定まる論理

式である。

これは，データベースの状態にのみ依存するならどんな式でもよい。例えば，ある部分の総和の値を何かの値と比較するという式であってもよい。実際QBAではこのような演算を許しているわけであるが，これを，修正トリガで計算した場合，修正が頻繁に行なわれると，非常な負担がかかるが，ここではその最適化については考えない。このような時間のかかる式を評価するには，時間トリガを使用するようにすれば比較的負担が少なくてすむであろう。この判断は，利用者に負せることにする。

(3) トリガ管理

トリガは，論理的には，トリガ名を1次キーとする関係形式で記録されていると思ってよい。実際，トリガの定義時には，この関係形式を使って，それ以前に定義された，トリガを探索してみる必要がある。しかし，実行時には，評価時をキーとして使わなければならず，それの高速・探索のために，トリガ個々に対するポインタを作っておくことが必要となる。

評価時による3種類のトリガ（修正トリガ，時間トリガ，トリガ・トリガ）はそれぞれ特有の構造をもって管理することが必要であるので，実現のために必要な構造の条件を考えてみる。

1) 修正トリガ

修正トリガは，関係形式の性質の1つとみなすことができる。対象となる関係形式，属性と，修正の種類（挿入，削除，修正）が定まった時，それに関連する修正トリガが直ちに参照できるデータ構造が必要である。また，参照されるトリガの個数は任意個であり，動的に変化することも考慮に入れなければならない（図4.1参照）。

2) 時間トリガ

時刻は，外界のものであるので，何等かの意味で「時計」が必要となる。普通は，OSが持っている時計を利用することになる。

時間トリガを働かせるのに必要なのは，次に評価されるべき時刻（T）

と，トリガへのポインタの組のリストで，Tによって昇順に並べられていた方が都合がよい。このリストを管理するプロセスは，一定時間おきに，ＯＳの目覚まし時計によって起きて，その時の時刻とTを比べ，時刻がTを過ぎているものについては，その組のポインタの指すトリガを評価し，その組を削除する。必要ならば，（評価時が，頻度である場合）次に評価すべき時刻を求め，その時刻と，トリガへのポインタを組にして，再びリストに挿入しておく（図4.2参照）。

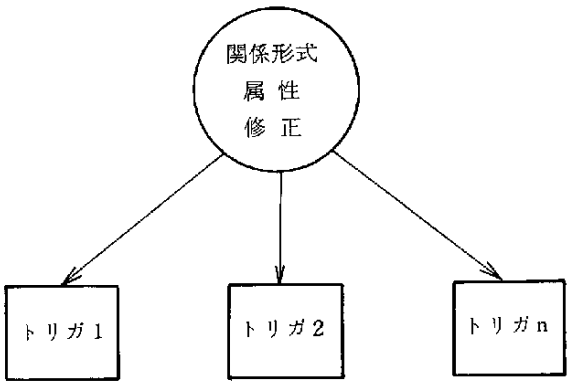


図 4.1　修正トリガ

3)　トリガ・トリガ

トリガＡが，トリガＢを呼ぶ場合を考える。

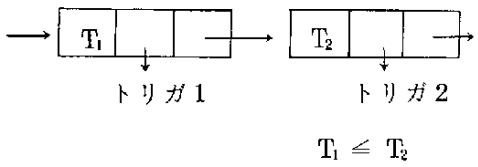


図 4.2　時間トリガ

トリガＡが目覚めた時，すぐにトリガＢを評価できなければならないので，トリガＡはトリガＢへのポインタを持っている必要がある。この場合も，１つのトリガに呼ばれるトリガは一般に複数個であり，動的に変り得るので，リストのような，可変長のデータ構造を持たせる必要がある。

4)　動　　作

トリガが，目覚めた時に処理される動作も，トリガを呼ぶ場合と同じように，動作へのポインタのリストを持っていなければならない。

3)，4)については，呼ぶ方のトリガは，定義される時には，何を呼ぶかは全く気にされない。呼ばれる方が，どのトリガから呼ばれるかを決定する

ように，利用者には見えなくてはならない。従って，呼ばれるトリガや動作が削除される場合も，呼ぶトリガには全く無関係に行なわれる。そこで動作や，トリガ・トリガには，それを呼ぶトリガへのポインタを残しておく必要がある。

以上のことと，評価時に，複数個の条件を書いてもよいということを考えると，トリガの本体は，次の6つの要素で構成されていなければならない。

1. トリガ名
2. 修正トリガとして，評価される時の関係形式の定義表へのポインタのリスト
3. 時間トリガとして，評価される時の時刻表へのポインタのリスト
4. 呼ばれるトリガへのポインタのリスト
5. 呼ぶトリガへのポインタのリスト
6. 呼ぶ動作へのポインタのリスト

ここで，ポインタというのは，直接アクセス可能な，アクセスパスのことである。トリガ名は，定義や削除の際，外部からトリガを参照するのに必要なキーである。

動作の主体として，必要な要素は次の3つである。

1. 動作のアルゴリズム
2. 呼ばれるトリガへのポインタのリスト
3. 動作中に修正する関係形式と属性

1.のアルゴリズムは，コンパイルされたコードであるか，文字のままかは問わない。3.は(4)で示すアルゴリズムで使用するものである。

図4.3に2つのトリガを定義したときの管理表のリストの例を示す。この例では，トリガ1は，時間トリガとして2通りに評価され，トリガ2は，トリガ1に呼ばれるトリガ・トリガと関係形式1の修正トリガとして定義されている。トリガ1は，直接呼ぶ動作はなく，トリガ2は動作2を呼ぶ。

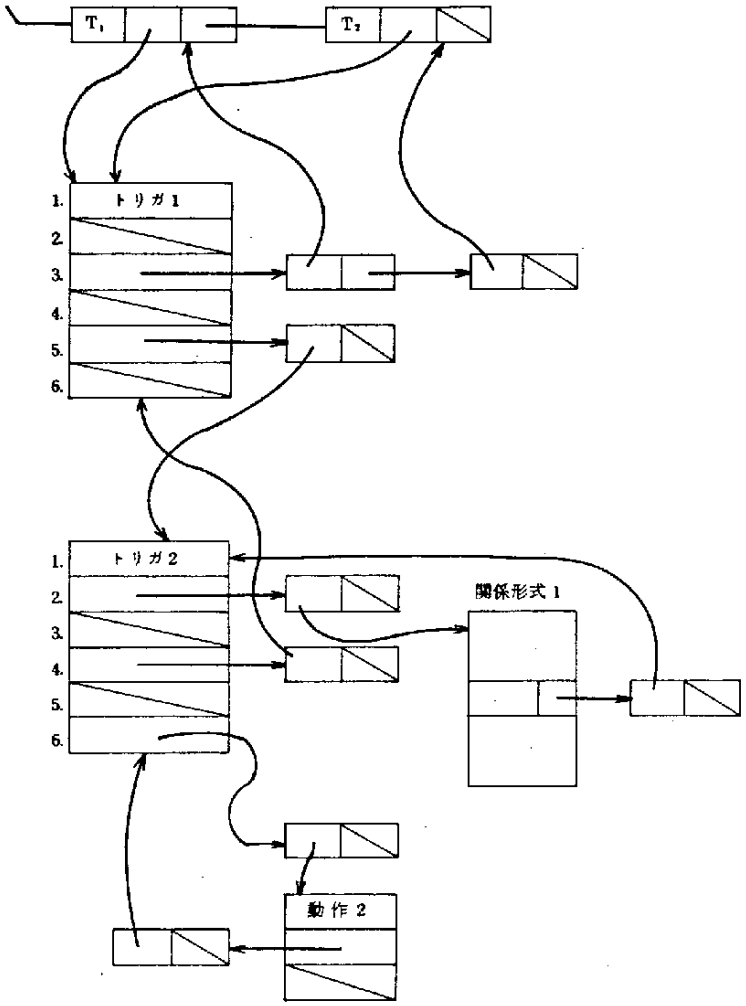


図 4.3　2つのトリガに対する管理表のリストの例

この図で関係形式1と書いてある箱は，関係形式の定義表であり，ここではその内部構造は述べない。ただ，付加的な欄として修正を受けると呼び出すべきトリガへのポインタのリストを持つことが必要である。

図 4.3 でリストは，2欄からなるセルをつなぐことで表現しているが，これも便宜的なものであり，これと同様の探索が可能なデータ構造ならど

んなものであってもよい。

(4) 無限ループの回避

トリガを節として，トリガAが評価され目覚めるとトリガBが評価されるという関係をA→Bの向きにもつ弧とする有向グラフを考える。（注：この関係は，直接トリガBを呼ぶ場合と，呼ばれた動作を実行するうちに修正トリガとして評価される場合がある。）この有向グラフに対して，次のことを要請する。

「上記の有向グラフは有向閉路を含んではならない。」

有向閉路を含むと，その閉路にそって評価を続けると無限ループに陥る可能性があるからである。ループを回っているうちにデータベースの状態を変化させ，うまく終了するというトリガと動作の列を考えることもできる。それは，通常のプログラミング言語における再帰的呼び出しに相当する。しかし，そのようなトリガでは一般に評価に必要な時間が長すぎると思われる。そもそも，Zloofがトリガ・トリガを導入したのは，いくつかのトリガの論理積に当るような計算を行なうためで，あまり技巧的な使い方をするためではないと思われる。また，トリガは効率の点からも，なるべく小さな仕事をするようなものを設計するべきであるという考えの下では，無限ループに陥る可能性のあるものは，実行前に調べておくことが必要である。

上の要請を満たすようにトリガを定義していくことが，それほど困難ではないことを以下に示す。

主張　有向閉路のない有向グラフGを考える。このグラフGに1つの節Nと，Nを端点にもつ弧E_1，E_2 …，Enをつけ加えた有向グラフをG′とする。

G′が有向閉路をもつならば，その有向閉路は節Nを含む。

証明　G′の有向閉路が節Nを含まないとすると，節Nを端点にもつ弧は有向閉路には含まれない。従ってグラフG′から節Nと弧E_1，E_2，…Enを除いたグラフ，つまりGが有向閉路を含むことになり，矛盾である。

非常に単純な主張であるが，この主張により図4.4に示すアルゴリズムによってトリガをするごとに有向閉路を含まないことを調べていけば，トリガのつくる有向グラフの中に有向閉路が含まれないことが保証される。

図4.4においてchildというのは，グラフとその節の集合を引数として，して，その節から出る弧の行き先の節の集合を値として返す手続きである。Gは，もともとあった閉路のないグラフに，節Nとそれを端点にもつ弧をつけ加えたグラフである。

```
procedure     circuit_check ( G, n ) ;
      var   M :   set of mode ;
begin       M←   child ( G, { n } ) ;
      while  M <> ∅  do
      begin  if   n  ε  M   then   return   false ;
         M ← child  ( G, M )
        end ;
        return   true
end ;
```

図4.4　有向閉路を検査する手続き

(5) 結　論

ZloofがQBAに導入したトリガの実現に必要と思われるデータ構造を考えてみた。速度の効率を要求される部分であるので，直接ポインタで結ぶことは絶対に必要である。しかし，この程度で実用に耐え得る程の速度が出るかは疑問である。また，本稿では，トリガ間のデッドロックの問題，最適化の問題については一際述べなかったが，実際の場合には，当然それらに対する考慮が必要である。

参考文献

1)Zloof, M.M. and Jong, P., "The System for Business Automation (SBA): Programming Language," CACM 20, 6 (June, 1977) pp. 385-396.

2)Zloof, M.M., "A LANGUAGE FOR OFFICE AND BUSINESS AUTOMATION" IBM Thomas J. Watson Research Center, Yorktown Heights, New York, 10598.

4.2 会話型ＣＡＤのための要求仕様

グラフィクデザイナーにとってデザインを行う際に，以前行われたデザインを会話的に参照出来ると非常に便利である。その為には以前のデザインは共有出来る形でグラフィックデータベースに蓄えられている必要があり，現在商用に存在するデータベースに比して必要とされる機能に大きな差がある。例えば，

1. 画像と文字との混在

   現存のデータベースが文字のみを扱うのに対しグラフィックデータベースは画像と文字が混在する。

2. 動的なスキーマ

現存のデータベースでは値のみが実行時にならないと決定されないのに対し，スキーマが実行時にならないと一般に決まらないので，データ記述言語（ＤＤＬ）をコンパイルする時点で決定できない。

3. ディスプレイ端末に対する柔軟性

グラフィックデータはデザイナの要求により種々に変化してゆく必要がある。カラーの画像を出力する為にはデザイナはラスタスキャン型のカラーディスプレイが必要となるだろうし，デザインが完了した後にはデザイナは印刷する為，より高解像度の白黒のベクター型又はラスタースキャン型のディスプレイを使用する。

4. 概念的又は物理的なズーミング

概略から詳細へと表現を変えてゆく事がダイアグラム又は画像に対して必要となる。現用のデータベースでは全てのデータを同等と考えるのでこの様な必要はない。概念的に行われる場合も存在するが，利用者に公開されているわけではない。

―― 構成法に対する要求仕様――

以上述べた分野に適応する為には如何なる機能を持つべきかを以下に述べる。

1. データの独立性の拡張

デザイナは如何なる順序で，画像や文字型データに対し操作を行うか予想出来ない為，それに対応する形でデータをデータベース中に蓄える必要がある。

2. 動的から静的への変換

動的なスキーマはデザインし終ると同時に静的な物に変化する。これに対応してスキーマに関連する値も同時に静的な物に変化しなければならない。この様な変換をも許す柔軟さを持ち合せなければならないが，それと同時に利用者に対する応答速度も十分に早くなければならない。

3. 出力端末に対する独立性

データは抽象化の高い形で蓄えられ，種々の端末に対して，それに合致す

る様に変換して出力出来ねばならない。

4. ズーミング可能なように拡張したＤＤＬ

データ記述言語（ＤＤＬ），データ操作言語（ＤＭＬ）は概念的又は物理的なズーミングを許す様に拡張されてなければならない。

――ＣＡＤデータベースの構成法――

以上述べてきた条件を満足させる構成法として次の物を提案したい。番号1～4は前項の番号に対応している。

1. フラットなデータ構造＋動的データ再構成

参考文献[1]に述べたように画像や文字を関係や関係の集合として表現する。フラットな表に変換され表現されたデータは，使用者の視点やデータスキーマの集合によって与えられた操作列に容易に対応出来る。スキーマは使用者により異なりうる。ある者は階層モデル[7]で用い，ある者はネットワークモデル[8]として使用するかもしれない。異なった型のデータ構造のスキーマはユニバーサルグラフォーマリズム（ＵＧＦ）[5]に基ずくユニバーサルデーターモデル（ＵＤＭ）と再帰グラフ理論（ＲＧＴ）[3]に基ずく再帰データモデル（ＲＤＭ）によって統一的に表現し扱う事が出来る。

現在議論しているデータベース構成法はＣＡＤ用に二つのレベルを持っている。上のレベルにはデータベースがあり，下のレベルにはグラフィックディスプレイシステムがある。二つの間にはデータベース中の表現から出力装置に依存した表現への変換を行うインターフェイスを介して結ばれている。

2. データのコンパイル／インタプリターによる変換の混在

デザインし終ったスキーマの一部及び関連するデータ値は動的にラベルづけされ，デザイナの相互作用を通じてインタープリットされる。そして結合は最大限の自由度を許す為実行時により異なる。残りはデザインし終った時点で静的に結ばれ，最大の効率が与られるようにコンパイル時に結合される。

3. 仮想画面構成法

データベース中のデータを直ちに端末に依存したデータ構造に変換するので

なく，ある種の中間的な論理的仮想表現に変換する事が出来る。端末として現在利用可能な物は全て基本的にはマトリックス状の平面と考えられる。例えばマトリックスをベクトルで表現することは離散数学の基本的な問題である。端末の解像度の差異は，端末に依存した表現にマトリックス表現を変換する際の抽出間隔を変化させる事に対応している。色とか色調はマトリックス表現の一つの要素に適当な値を代入することに対応している。マトリックス表示平面に仮想画面を仮定することにより，いかなる方式の端末にも，端末に依存しない内部表現から仮想画面への変換系を変更する事により容易に対応出来る。

4. 再帰グラフ理論によるスキーマ＋プロシジュアーベース

スキーマのズーミングの論理的構成法はＵＤＭ又はＲＤＭにより実行される。スキーマや操作は実際には拡張ＤＤＬやＵＤＭのＤＭＬにより記述される。ＵＤＭは他のスキーマと関係しているスキーマやエンティティを許している。グラフ理論で言えば与えられたグラフの弧や節は再びグラフとなり得ることを意味している。再帰グラフあるいはスキーマを扱う為の再帰グラフの演算，例えばズーミング，は実行時に画像に対する演算の手続きが呼ばれることを意味する。この手続きの集まりをプロシェジュアーベースと呼ぶが，利用者の目には見えないようになっている。

――仮想グラフィクスオペレーティングサブシステム（ＶＧＯＳ）――

円滑に又効率良く全ての画像データベースを動かす為ＶＧＯＳが管理している。ＶＧＯＳは３つの部分から成立している。

- グラフィクスモニタ
- グラフィクスアナライザー
- グラフィクススケジューラ

いくつかのグラフィック端末が接続されるとアナライザは端末を識別し，モニタはスケジューラに仮想表示端末を作るよう指令し，仮想画面と物理画面との写像を実行する。

結　び

我々は現存の商用データベースと全く異なる応用分野である会話型ＣＡＤ用のデータベースの構成法について述べた。種々の実験によりＣＡＤを利用する分野でこの方式が非常に有用である事がわかった。

参考文献

1. "A Relational Data Base Schema for Describing Complex Pictures with Color and Texture," T. L. Kunii, S. Weyl and J. M. Tenenbaum, Proceedings of the Second International Joint Conference on Pattern Recognition, 310 (Lyngby-Copenhagen, August, 1974). [Available also as Stanford Research Institute Technical Note 93, SRI Project 8721 (June, 1974), and reprinted in Policy Analysis and Information Systems, Vol. 1, No. 2, 127 (1978)]

2. "An Architecture for Evolutionary Database System Design," T.L. Kunii, J. C. Browne, and H. S. Kunii, The IEEE Computer Society's Second International Computer Software and Applications Conference, 382 (Chicago, 1978).

3. "RGT: The Recursive Graph Theory as a Theoretical Basis of a System Design Tool DESIGN - TOOL--With an Application to Medical Information System Design --," M. Harada, T. L. Kunii, and M. Saito, Proceedings of International Symposium on Medical Information System (Osaka, 1978).

4. "An Interactive Fashion Design System 'INFADS' ", T. L. Kunii, T. Amano, H. Arisawa and S. Okada, Computer and Graphics, Vol. 1, 297 (1975).

5. "The Universal Graph Formalism and Its Application to Data Models," H. S. Kunii and T. L. Kunii, in "bit" Computer Science Monograph Series, 1978, No. 8: Special Issue on Software Engineering, 242 (Kyoritsu Shuppan Co. Ltd. Tokyo, 1979).

6. "Hierarchical Strategy for Texture Discrimination and Its Application," N. Ohbo, K. Yokokawa and T. L. Kunii, Preprints of IFIP Working Conference on Modelling of Environmental Systems, 143 (Tokyo, 1976).

7. "A Data Management System for Engineering and Scientific Computing, Linda Elliott, H. S. Kunii and J. C. Brwone, Proceedings of a NASA Conference on Engineering and Scientific Data Management, 197 (Hampton, 1978). (NASA Conference Publication #2055. Proceedings available through NASA Langley Research Center, Hampton, Virginia.)

8. "Data Description for Computer-Aided Design," A. E. Bandurski and D. K. Jefferson, Proceedings of ACM-SIGMOD International Conference on Management of Data, 193 (San Jose, 1975).

4.3 グラフを基礎とした領域分析法

従来のデータベース・システムは殆ど，文字・数字型データのみを対象としてきた。しかしながら事務書類を初めとして，地図情報，医療情報等々，大量の画像情報が集積されながら，それらが有効に効率良く利用できるようなデータベース・システム（非文字・数字型データを処理可能なシステム）の開発が進んでいない。その理由の一つは，データベース・システムそれ自身が，文字・数字型データ処理に最適化されていることもあるが，それ以上に，従来画像処理研究が1枚ないし数枚の絵を処理する技術開発に焦点を絞ってきたことにある。既ち大量画像データを効率よく操作する問題が大きなボトルネックになっているのが現実である。

グラフを基礎とした領域分析法は，汎用で計算コストの低いアルゴリズムにより，画像を"意味のある領域"に分割する方式である。この方式は特に，画像データベース・システムに於て，画像のイメージ・データ（生のデータ）から，画像の意味論的記述（プレリミナリなもの）を，なるべく自動的に効率よく得ることを目的としている。

本方式では，画像イメージは，マトリックス形式のグラフで表現される。このグラフはアークに重みのついたグラフである。各頂点は各領域を表わす。初期状態では各領域は一ピクセルから構成されている。アークの重みはグレイレベルの微分値に対応する。領域の融合可能性のチェックされるアークの集合は，ＭＳＴ（Minimal Spanning Tree）構成アルゴリズムによって作られる。領域はＭＳＴの道に沿って順次に融合されてゆく。ＭＳＴでは，二頂点の融合は，対応する二つの領域が一つの領域に統合されることを示す。

本方法の有効な点は次の様である。

a) 計算速度が早い

b) グラフデータ構造を用いたことによる汎用性

c) non－semantic なスキームであるための汎用性

a)は大量画像データを扱うための最も重要な要求であり，b)，c)は多様な画像データを扱うために基本的な要求である。

このスキームは，東京大学大型計算機センターのHITAC8800／8700上でＰＬ／Ｉによりインプルメントされた。図4.4に画像データの例を，図4.5にその分割結果を示した。この例では，グレイ・レベルは7ビット，絵のサイズは128×75である。この絵の分割は約2秒で実行された。

図4.4　Original Image

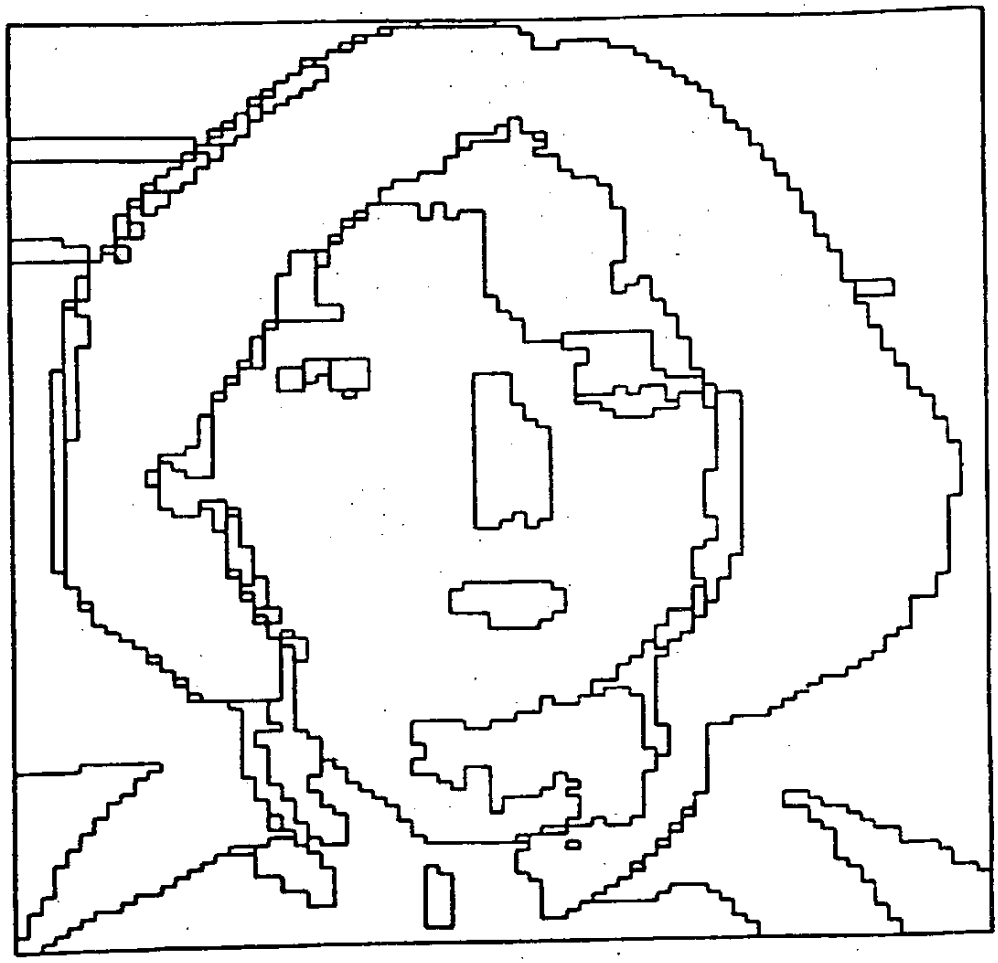


図 4.5　Result of Segmentation

昭和55年3月発行

発行所　財団法人　日本情報処理開発協会
東京都港区芝公園3－5－8
機械振興会館内
TEL　(434) 8211(大代表)

印刷所　山陽株式会社
東京都港区虎ノ門1－9－5
TEL　(591) 0248