58—S —002

高密度通信処理における分散情報統合利用システムに関する研究開発報告書

昭和59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５８年度に実施した「高密度通信処理における分散情報統合利用システムに関する研究開発」の成果をとりまとめたものであります。

序

今後の組織体等における情報処理システムは，大型コンピュータを中心とする大規模システムとは別に，パーソナル・コンピュータなどの小型コンピュータを中心とする小規模システムによって独自の処理，データの分散利用が急速に進むものと思われる。しかし，システムの有用性の観点からは，これらシステムが分散して持つ情報を相互に統合利用することが重要とされている。

このためデータ利用の基本ツールとなるデータベースに加え，これら小規模システムを有機的に結合するローカル・ネットワーク，各種情報処理機器の持つ非コンピュータ・リーダブルなデータ（文書，書類，メッセージ等）の取扱いなど，異種のシステムに分散する情報の統合利用について3年計画で研究し，ローカル情報システムとしてモデルを開発する。

本報告書は，初年度としてローカル情報システム（ＬＩＳ）全体の基本概念（機能，インタフェース，ＤＤ／Ｄ等）を確立し，これにもとづきローカル・ネットワークの基本通信方式，小型データベース（文書，書類等を扱う）の基本設計及びプロトタイプの開発，大型データベースと小型データベース間のデータ・メッセージの統合技術の基本機能概念についてまとめたものである。

本報告書がこの方面に興味をお持ちの方々に広く利用され，今後の情報処理技術向上の一助として寄与することができれば幸いである。

昭和５９年３月

目　　次

1. はじめに

ＶＬＳＩを中心としたハードウェア技術の進歩によって，従来の小～中型機並みの性能を有した超小型計算機が，事務所，工場等の各作業者に普及してきている。これによって，各作業者が，単に計算能力を持てるだけでなく，必要なデータを保有出来る様になってきている。一方，Ethernet〔METCR76〕等のローカルエリア網（ＬＡＮ）の発展によって，一つの組織体内の種々の計算機の相互結合が可能となってきている。この様なシステムでは，超小型計算機は，情報の送受信端末としてではなく，データベースシステム（ＤＢＳ）を中心とした作業用のワークステーションと考えられ，他の計算機のＤＢＳから，自分の計算機内にデータを導出し，蓄積して操作する為に用いられてくる。

複数のＤＢＳを通信網で結合し，統合的なＤＢＳ利用を行わせるシステムは，分散型データベースシステム（ＤＤＢＳ）と呼ばれている。同種の関係型ＤＢＳから成るＤＤＢＳとして，ＳＤＤ－1〔ROTHJ80〕，Polypheme〔ADIBM80〕，System R*〔WILMP83〕，ＤＤＭ〔CHANA83〕等が開発されている。関係型以外の異種のＤＢＳから成るＤＤＢＳとしては，ＪＤＤＢＳ〔TAKIM78－84〕，ＤＥＩＭＳ〔SUZUK80〕，Multibase〔SMITJ80〕が開発されてきている。当協会で，1977年より８３年３月まで開発を行ったＪＤＤＢＳ（Ｊipdec ＤＤＢＳ）は，従来のCODASYL ＤＢＳを，一つの関係型ＤＢＳに論理的に見せれるシステムである。
現在，Ｍ－１７０Ｆ内の四つのCODASYL ＤＢＳ（ＡＩＭ　ＤＢＳ）を，ＤＢ／ＤＣを利用した通信網で結合して実働している。ＪＤＤＢＳの利用者は，四つのCODASYL ＤＢＳを，一つの関係型ＤＢＳとして，データの所在，各ＤＢＳの操作方法（例,COBOL ＤＭＬ）を意識せずに利用出来る利点をもっている。

本プロジェクトでは，ＪＤＤＢＳの成果を生かして，新たに超小型計算機を加え通信網としてＬＡＮを用いたＤＤＢＳの研究開発を，３カ年計画で行っている。このＤＤＢＳを，ローカル情報システムＬＩＳとする。本年度は，プロジェクトの初年度として，ＬＩＳの全体モデル，ＬＡＮを用いた通信処理方式を検討し，超小型機用の関係型ＤＢＭＳの開発を行った。この結果，ＬＩＳの制御システムの通信プロトコルの概要を定め，超小型機用の関係ＤＢＭＳとしての Granada のプロトタイプを実現した。

第２章では，ローカル情報システム（ＬＩＳ）の全体モデルを述べる。第３章

では，ＬＩＳの制御システムである全体データベースプロセッサ（ＧＤＰ）のプロトコル階層と，各階層の位置づけ，インタフェースを与える。第４章と５章では，各々分散したデータに対する問合せとトランザクション処理の為のプロトコルを与える。第６章では，利用者に与えられる抽象データ型としての視野の管理について述べる。第７章では，Ethernet データリンク（ＥＤＬ）サービス上の基本通信プロトコルを与える。第８章では，超小型計算機用の関係ＤＢＭＳ Granada の概要を示す。９章に，ＬＩＳの機器構成を与える。最後に付記として，現在のＤＤＢＳにおける同時実行制御の方法をまとめておく。

2. ローカル情報システム（ＬＩＳ）

本章では，ローカル情報システム（ＬＩＳ）の論理構造とアーキテクチャについて述べる。

2.1 ＬＩＳの論理構造

本節では，複数の異種ＤＢＳを統合化したＤＤＢＳの論理構造を検討する。まず，データモデルは，データ構造と操作演算とで定まるとする。データ構造は，時間不変な論理構造を与えるスキーマと，これに実際のデータを与えたデータベースとから成る。操作演算には，ある目標スキーマを満足する目標集合をデータベースから導出する検索演算と，データベースを変化させる更新演算とがある。操作演算を表す言語をデータ操作言語ＤＭＬとする。実際に実現されたデータモデルには，関係，網（ＣＯＤＡＳＹＬ），階層の三種の型がある。各型を，スキーマとＤＭＬとの対によって特性化する。ＤＢＳとは，ある型のデータモデルを利用者に提供する処理システムである。ＤＢＳの異種性を，各データモデルの型（即ち，スキーマとＤＭＬ）の差とする。本論文では，異種ＤＢＳの統合化とは，各ＤＢＳのデータ構造の集合に対して，これと等価な一つのデータ構造とＤＭＬとを利用者に提供することとする。統合化によって，利用者はＤＤＢＳをあたかも一つのＤＢＳかの様に見れる利点がある。統合化は，同種化と，同種化されたＤＢＳの統合化との二つの手順から成る〔ＴＡＫＩＭ７８，７９〕。

2.1.1 同種化

異種ＤＢＳの同種化とは，各ＤＢＳの実際のデータモデルに対して，これと等価な共通の型のデータモデルを利用者に提供することである。文献〔ＴＡＫＩＭ８０，８３ａ，８４ａ〕では，各ＤＢＳに対して次の結果を示した。(i)等価な関係型のローカル概念（ＬＣ）モデルのデータ構造が存在し，(ii)この上の述語論理形式のＲＱＬ操作演算を満足する目標集合を導出するＤＢＳ操作演算が存在する。従って，従来のＤＢＳ上に，ＬＣモデルを提供するインタフェース〔ＴＡＫＩＭ８２ｂ〕を設けることが出来る。この様にして同種化されたＤＢＳを仮想ＤＢＳとする。仮想ＤＢＳを通信網で結合したシステムを，ローカル概念（ＬＣ）層のＤＤＢＳとする。

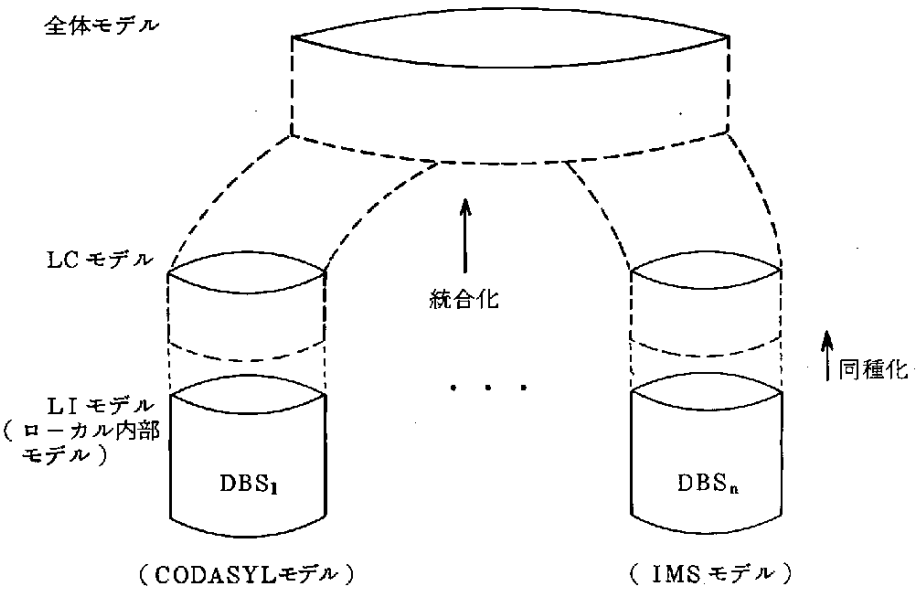


図2.1　ＬＩＳの論理構造

2.1.2　仮想ＤＢＳの統合化

A.　全体モデル

仮想ＤＢＳの統合化とは，各点に在る仮想ＤＢＳのデータ構造の集合に対し，これと等価なデータ構造と操作演算とを全利用者に提供することである。本論文では，各仮想ＤＢＳ内のデータ構造そのものの集合を全体データ構造と定め，これに作用する操作演算を全体操作演算とする。この両者の対で定まるモデルを全体モデルとする。全体モデルが利用者に提供される階層をＤＤＢＳの全体層とする。ここで，利用者には，全体モデルを提供する一つのＤＢＳが存在することになり，ＤＤＢＳは統合化されたことになる。この為には，仮想ＤＢＳ間の有効な通信処理システムの設計が重要となり，本論文で論じる。

B.　全体操作演算

全体モデルの操作演算としては検索と更新演算がある。

a.　検索演算

全体データベース上の検索演算は，ＲＱＬ［ＴＡＫＩＭ８３ａ，８４ａ］で次式の形式で表現される。

$$\underline{\text{var}}\ (x_1,\ X_1)\cdots(x_\ell,\ X_\ell)\ ; \qquad \cdots\cdots\cdots(2.1)$$

get into G (Λ) where Ψ ; ……(2.2)

X_i は全体データベース内の任意の関係で，x_i は X_i 内の組を値としてとる組変数である。(2.1)式は，X_i に対して組変数 x_i を定める。Ψ は検索条件式で，(i) x_i . a θ v と(ii) x_i . a θ x_j . b の形式の二種の基本式の積・和で与えられる。x_i . a は，x_i のとる組の属性 a の値を表す変数である（x_j . b も同様）。θ は比較演算子，v は定数である。Λ は目標関係 G のスキームを与える目標リストで，x_i . a の形式の変数の並びである。全体検索演算のＲＱＬ表現を全体問合せ（Λ, Ψ）とする。この意味を次式で与える。

$$G = \{ t \mid t = \Lambda(v) \wedge \Psi(v) \wedge v \in X_1 \times \cdots \times X_\ell \} \cdots (2.3)$$

b. 更新演算

全体データベースの更新演算としては，関係Ｘに対する消去，追加，更新演算がある。これらの演算は，ＲＱＬで次の様に表現される。以下，$Y_1, \cdots, Y_m$ は，全体データベース内の任意の関係とする。

(a) 消去演算

var (x,X) (y_1 ,Y_1) $\cdots$ (y_m,Y_m);
del x where Ψ ; ……(2.4)

は，$x, y_1, \cdots, y_m$ 上の条件式である。この意味は，Ｘから，$t \in X \wedge w \in Y_1 \times \cdots \times Y_m \wedge \Psi(t,w)$ なる組 t を除くことである。

(b) 追加演算

var (y_1 ,Y_1) $\cdots$ (y_n,Y_m);
app X (Λ) where Ψ ; ……(2.5)

Ψ と Λ は，$y_1, \cdots, y_m$ 上の目標リストと条件式である。この意味は，関係Ｘに，$w \in Y_i \times \cdots \times Y_m \wedge \Psi(w) \wedge t = \Lambda(w)$ なる組 t を加えることである。

(c) 置換演算

var (x,X) (y_1 ,Y_1) $\cdots$ (y_m,Y_m)
rep x (Λ) where Ψ ……(2.6)

Λ と Ψ は，$x, y_1, \cdots, y_m$ 上の目標リストと条件式である。この意味は，$t \in x \wedge w \in Y_1 \times \cdots \times Y_m \wedge u = \Lambda(t,w) \wedge \Psi(t,w)$ なる組 t を u で置換することである。

c. 全体トランザクション

全体トランザクションとは，全体データ構造に対する組単位の検索（read）演算と更新（write）演算とで定まる系列である。更に，全体トランザクションは，原子的な実行単位である。即ち，全体トランザクションは，正しく実行されるか，全く実行されないかのどちらかである。全体トランザクションは，正しい全体データベースを，正しい全体データベースに変化させるが，全体トランザクションの各演算の実行段階では，必ずしも全体データベースの状態は正しくない。

全体トランザクションの操作演算は，全体データ構造内の関係を組単位に操作する。操作演算は，各関係X毎の現在子（currency 又は，carsor）xに基づいている。xは，関係X内の組を指示している。関係Xに，複数の現在子を設けられる。以下に，基本操作演算を関係として与える。

(1) var（X）；

input X ＝全体データ構造内の関係名

output var ＝関係Xの現在子（へのポインタ）

例 crtx1 ＝ var（X）；

crtx2 ＝ var（X）；

crtx1とcrtx2を，関係Xの現在子とする。

(2) {first / last}（crtx）；

input crtx ＝関係Xの現在子

output {first / last} ＝ {OK if 成功（Xの現在子） / NO if 失敗（Xには組がない）}

関係Xの現在子crtxを，Xの格納順の先頭（first）又は最後（ℓast）の組を指示させる。

(3) {next / prior}（crtx）；

iuput crtx ＝Xの現在子

output {next / prior} ＝ {OK if 成功 / NO if 失敗（Xには組がない）}

関係Xの現在子crtxの指示する組で，一つ次（next）又は，一つ

前（prior）の組とする。組がないときにはＮＯがかえり，crtxの内容はＮＩＬとなる。

(4) read（crtx）；

input　crtx＝Ｘの現在子

output　crtx＝{ ＯＫ if 成功 / ＮＯ if 失敗

crtxの指示する組を，現在子内に読み出す。このとき，Ｘの任意の属性aに対して，以下を定める。

crtx.a　＝今読まれた組の属性aの値。

*crtx　＝現在子の指す組の識別子

(5) crtx.a＝v；

現在子crtxの属性aの値をvとする。

(6) del（crtx）；

input　crtx＝Ｘの現在子

output del＝{ ＯＫ if 成功 / ＮＯ if 失敗

現在子crtxの指す組をＸから除く。*crtxはＮＩＬとなる。

(7) app（crtx）；

現在子crtx内の組の値を，関係Ｘに加える。*crtxは，加えられた組の識別子となる。

(8) rep（crtx）；

現在子crtxの指す組の値を，crtx内の値に置き換える。

例　x＝var（Ｘ）；　first（x）；

x.ename＝"鈴木"；　x.age＝２８；

rep（x）；

Ｘの先頭の組のenameを鈴木に，ageを２８にする。

(9) abort（）；

トランザクションを異常終了（アボート）させる。

(10) commit（）；

トランザクションをコミットさせる。

(11) begin（）；

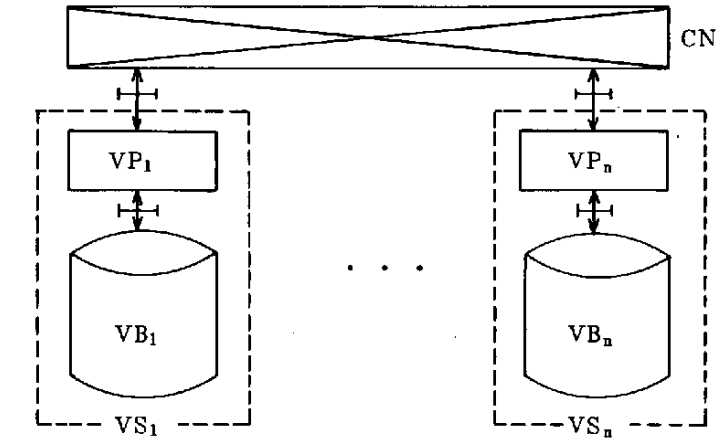
トランザクションを開始させる。

(12) end( );

トランザクションを終了させる。

2.2 ＬＩＳのアーキテクチャ

ローカル情報システム（ＬＩＳ）は，ローカルエリア網（ＬＡＮ）で結合させたｎ（≧１）個の仮想ＤＢＳ（ＶＳ）の集合としてモデル化出来る〔図2.2〕。



VS：仮想DBS　VP：仮想プロセッサ　CN：通　信　網

VB：仮想データベース　GP：全体プロセッサ

図2.2　ＬＩＳアーキテクチャ

2.2.1 通信網

通信には，一対一通信と放送通信との二種がある。一対一通信では，ある点から一つの受信点に情報が通信される。放送通信では，ある点から，複数の受信点に同一情報が通信される。このとき，以下を定義する。

〔定義2.1〕

$T_i\{j_1,\cdots,j_n\}(x) \triangleq$ 送信点ｉからｎ（≧１）個の受信点$j_1,\cdots,j_n$に情報Ｘ（情報量x〔ｂｉｔ〕）を通信する為に，通信網に送信される通信量（＝パケット量）〔ｂｉｔ〕。

$R_i\{j_1,\cdots,j_n\}(x) \triangleq$ 送信点iからn（≧1）個の受信点$j_1,\cdots,j_n$に情報X（情報量x〔bit〕）を，通信し始めてから終了するまでの時間〔sec〕。

$T_i\{j_1,\cdots,j_n\}(x)$を全通信量，$R_i\{j_1,\cdots,j_n\}(x)$を通信時間とする。□

このとき，以下の仮定を設ける。

〔仮定2.1〕

(1) 通信網は軽負荷で，衝突，輻輳等による待合せ遅延は生じない。

(2) 伝播遅延は無視し得る。

(3) 通信網の各通信路の伝送速度は同一とする。□

一対の点iとjの間で情報X（情報量x〔bit〕）を通信するとき，全通信量$T_{ij}(x)$と通信時間$R_{ij}(x)$は，次式で与えられるとする。

$$T_{ij}(x)=\alpha+\beta\cdot x\ [\text{bit}] \quad \cdots\cdots(2.7)$$

$$R_{ij}(x)=(\alpha+\beta\cdot x)/w\ [\text{sec}] \quad \cdots\cdots(2.8)$$

ここで，αとβは定数である。wは，通信網の伝送速度〔bit／sec〕である。即ち，$T_{ij}(x)$と$R_{ij}(x)$は，通信される情報量xのみに依存するとする。

次に，点iからn個の受信点$j_1,\cdots,j_n$（$n\geq 1$）に，一つのパケットの情報を通信する場合を考える。このとき，一対一網と放送網を以下の様に定義する。

〔定義2.2〕

送信点iが，各受信点j_h（$h=1,2,\cdots,n$）毎に，一パケットづつ，合計nパケットを送信せねばならない通信網を一対一網とする。一方，送信点iが，受信点の数nに係りなく一パケット送信することによって，全受信点にパケットを通信出来る通信網を放送網とする。□

従来のDDX，ARPANET，CYCLADES等の広域パケット交換網は，一対一網である。Ethernet〔METCR76〕等のローカルエリア網（LAN）と無線網〔ABRAN70〕の様に，一本の通信路を共有している通信網では，(i)一時に一点からのパケットを通信出来て，(ii)これを全点で

受信出来る。従って，これ等の通信網は放送網である。以上より，一対一網と放送網では，以下の式が成り立つ。

〔一対一網〕

$$T_i\{j_1,\cdots,j_n\}(x)=\sum_{h=1}^{n}T_{ij.h}(x)=n\cdot(\alpha+\beta\cdot x)\ 〔\mathrm{bit}〕\cdots\cdots(2.9)$$

$$R_i\{j_1,\cdots,j_n\}(x)=T_i\{j_1,\cdots,j_n\}(x)/(\gamma\cdot w)=n\cdot(\alpha+\beta\cdot x)/(\gamma\cdot w)〔\mathrm{sec}〕\cdots\cdots(2.10)$$

ここで，wは各通信路の伝送速度〔bit／sec〕，γは通信の並行度（即ち，同時に開設出来る通信リンク数）である。□

〔放送網〕

$$T_i\{j_1,\cdots,j_n\}(x)=\alpha+\beta\cdot x〔\mathrm{bit}〕\cdots\cdots(2.11)$$

$$R_i\{j_1,\cdots,j_n\}(x)=(\alpha+\beta\cdot x)/w〔\mathrm{sec}〕\cdots(2.12)$$

以上より，放送網は，放送通信を行うとき，一対一網に対して，以下の特徴を有している。

〔放送網の特徴〕

(i) 全通信量と通信時間の低減〔（2.9）～（2.12）式〕。

(ii) 同時到着性〔（2.16）式〕放送網の各通信路には，一時に一点からのパケットのみを送信出来るので，各点はパケットを送信順（ＦＩＦＯ順）に受信出来る。□

(i)は，本論文で述べる様に，ＤＤＢＳ処理の為の通信負荷を低減する。一対一網の同時実行制御では，各通信リンク毎の通信遅延の差によって，各点でのパケットの受信順は送信順とならない点が，制御を困難にしていた〔ＢＥＲＮＰ８１〕。しかし，一本の通信路から成る放送網では，(ii)の性質よりこの問題は生じない。

2.2.2 仮想データベースシステム（ＶＳ）

仮想ＤＢＳ VS_i は，仮想データベース DB_i と仮想プロセッサ VP_i とから論理的に構成される（$i=1,\cdots,n$），VB_i は，関係の集合である。

VP_iは，(i)VB_i内の関係に対して，ＲＱＬで表現された操作演算（検索，消去，追加，置換）を実行し，(ii)他の仮想ＤＢＳとの通信演算を行うプロセッサである。全体データベースは，各VS_iのDB_i内の関係そのものの集合となる。この全体データベース上の全体操作演算は，各VP_iの演算によって表現される必要がある。この為，トランザクションを以下に定義する。

［定義2.3］

トランザクションとは，次の基本操作演算で定まる系列である。

(i) 各VS_iのＲＱＬ操作演算。

(ii) 通信演算 send（X,i,j_1,…,j_n）（VS_iから情報Ｘを，ｎ個のVSj_1,…,VSj_nに放送する）。□

従って，全体データベース上のＲＱＬ操作演算を実行する為には，以下の二つの手順が必要になる。

(i) 全体操作演算Ｑを，トランザクションＴに変換する。

(ii) トランザクションＴの各基本操作演算を，各仮想ＤＢＳで実行させる。

(i)をトランザクション変換とし，(ii)をトランザクション実行とする。(i)と(ii)を併せて，ＤＤＢＳ処理とする。

仮想ＤＢＳには，格納しているデータと利用方法とにより，次の二種がある。

(i) 個人情報システム（ＰＩＳ）

(ii) 中央情報システム（ＣＩＳ）

個人情報システム（ＰＩＳ）は，各作業者専用の情報システムであり，各作業者の必要とするデータが格納される。ＰＩＳは，主に，超小型計算機上に実現される。ＰＩＳとなる計算機は，１００～２００ＭＢ程度の二次記憶装置と，関係型ＤＢＭＳ機能を有しているものとする。

一方，中央情報システム（ＣＩＳ）は，組織体の作業者間で共有して利用される情報システムであり，組織体全体の共有データが格納される。ＣＩＳ内のデータは，組織体の管理者によって，統合的（中央集権的）に管理される。ＣＩＳは，主に，ＰＩＳよりも大型の計算機によって実現される。ＣＩＳは，大容量の二次記憶装置と，従来の大型ＤＢＭＳ（例，CODASYL型のＤＢＭＳ）とを有しているものとする。

2.3 ＬＩＳの実現方針

2.2節で述べたローカル情報システム（ＬＩＳ）の実現について述べる。

2.3.1 放送網の実現－ＥＤＬサービス

放送網としてローカルエリア網（ＬＡＮ）を用いる。ＬＡＮとしては，標準的な Ethernet のデータリンク（ＥＤＬ）〔ＤＩＸ８０〕のサービスを提供するものとする。ＥＤＬ層では，上位層に対して，フレームの送信と受信サービスを次の二つの関数によって提供している。

(i) Transmit Frame

(ii) Receive Frame

(1) フレームの送信

上位層は，次の関数によってフレームは送信される。

proc Transmit Frame (dest, source, type, data)

input dest ＝目的点の番地

source ＝送信点の番地

type ＝タイプ（上位層が利用）（プロトコルの型）

data ＝送信すべきデータ

output Transmit Frame

＝ { transmit ＯＫ if 送信が成功

{ excessive Collision Error if トラフィックの輻輳か網の障害の為に過剰な衝突が起きた。

Transmit Frame は，フレームを目的とする点（又は点の集合）に送信する。フレームの送信が終了するまで，Transmit Frame から制御はもどってこない。網に対して送信が成功した時，Transmit ＯＫが，又，失敗した時には，excessive Collision Error が表示される。

(2) フレームの受信

上位層は，次の関数によって，ＥＤＬ層からフレームを受信出来る。

proc Receive Frame (dest, source, type, data)

input dest ＝自分の番地

output source＝受信したフレームを送信した点の番地

type ＝上位層が利用（プロトコルの型）

data ＝受信データ

Receive Frame

receive OK if	受信は成功
frame Check Error if	物理的なチャネル内での伝送誤りにより破壊フレームを受信した
alignment Error if	受信したフレームは破壊されていて，フレーム長はオクテート（８bit）の整数倍でない

Receive Frame によって，フレームを受信する。フレームの受信が終了するまで，制御はかえってこない。

(2) フレームの形式

図 2.3 は，ＥＤＬでの通信単位となるフレームの形式を表している。

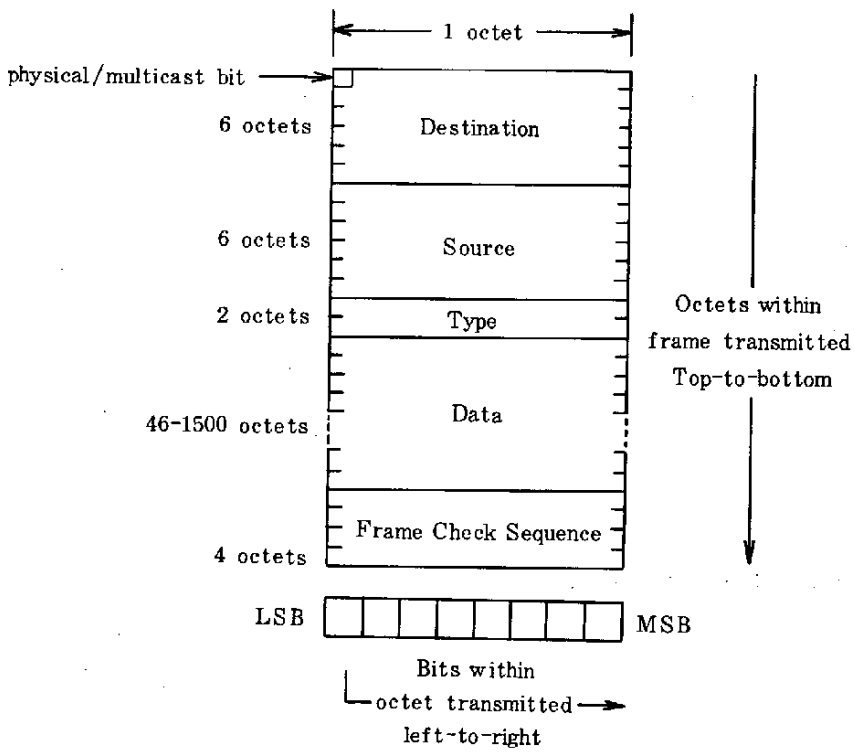


図 2.3　フレームの形式

ＥＤＬ層での番地には次の二種がある。

(i) 物理番地　Ethernet の各点に一対一に対応する番地である。任意の Ethernet 任意の点は，全く異なった番地を有している。

(ii) 同報（multicast）番地　Ethernet 内の複数の点に対応した番地であり，次の２種がある。

(α) 同報番地　上位層の取り決めにより，論理に関連づけられた点の群に対応した番地。

(β) 放送番地　ある Ethernet 内の全点を表す番地

フレームのＥＤＬ番地の先頭の１ビットによって，物理番地と同報番地のどちらを目的点の番地として選ぶかを決める。

番地 {
- 物理番地　if　先頭ビット＝０
- 同報番地　if　先頭ビット＝１
  - (α)同報番地　　残り４７ビットが全て１

タイプ・フィールドは，上位層で使用する為の領域（２オクテット）である。特に，上位層から見て，このフレームが何であるかを識別する為に使用される。

データ・フィールドは，４６オクテット以上1,500オクテット以下の任意のデータを含める。

ＦＣＳ（frame sequence check）フィールドは，４オクテットの巡回符号検査（ＣＲＣ）値を含んでいる。ＦＣＳフィールドの値は，送信番地と受信番地，タイプ，データ（ＦＣＳフィールドは含まない）のフィールドの内容Ｘの次の関数値である。

$$G(X)=X^{32}+X^{26}+X^{23}+X^{22}+X^{16}+X^{12}+X^{11}+X^{10}+X^{8}$$
$$X^{7}+X^{5}+X^{4}+X^{2}+X+1$$

2.3.2　仮想ＤＢＳ（ＶＳ）の実現

仮想ＤＢＳは，論理的には，図 2.2 に示す様に仮想プロセッサＶＰと仮想データベースＶＢとから構成される。本項では，これらの論理的な仮想ＤＢＳを実現する為のモヂュール構成について考える。仮想ＤＢＳは，図 2.4 に示す様に，以下のモヂュールから構成される。

(i)利用者インタフェース（ＵＩ）

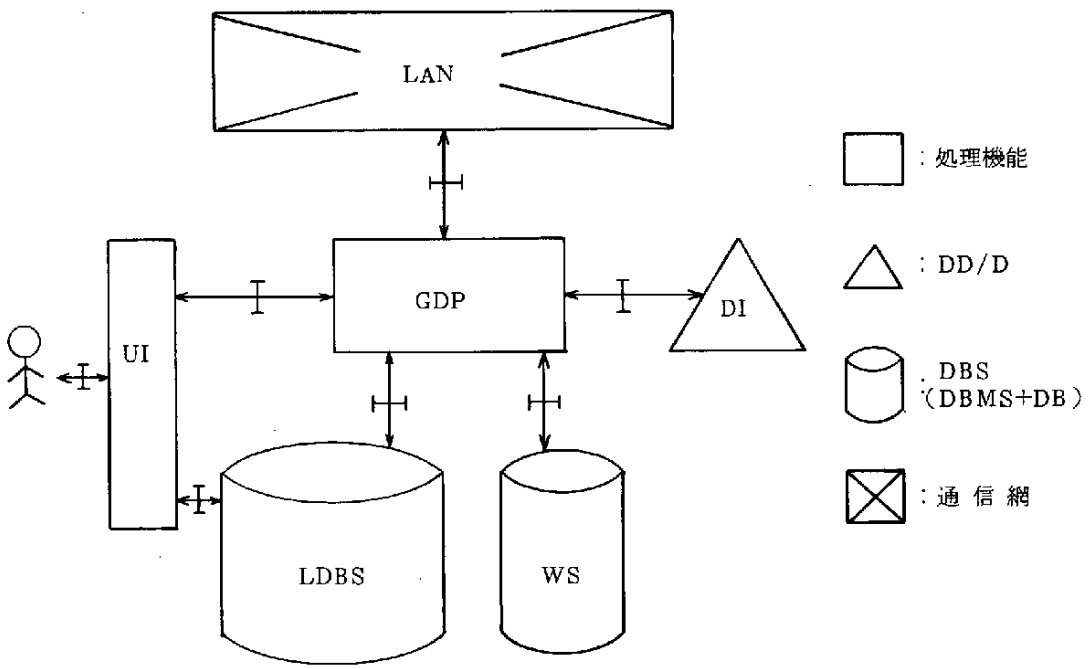
(ii)全体データベースプロセッサ（ＧＤＰ）

(iii)分散性情報（ＤＩ）

(iv)作業領域（ＷＳ）

(v)ローカル関係ＤＢＳ（ＬＤＢＳ）

ＬＤＢＳは，仮想ＤＢＳ内のローカルな関係を格納し，操作する為のシステムである。仮想ＤＢＳの計算機が，関係型のＤＢＭＳを持つならばＬＤＢＳは，このＤＢＭＳによって実現される〔図2.5〕。一方，計算機がRDBMSを持たず，関係型とは異種なＤＢＭＳ（例，CODASYL型のＤＢＭＳ）を持つならば，関係モデルを提供するインタフェースシステムＬＤＰ（ローカルデータベースプロセッサ）〔TAKIM80，82b，83a，84a〕を，異種ＤＢＳ上に設けて，異種ＤＢＳを関係型に同種化するものとする。〔図2.6〕。



ＧＤＰ　：全体データベースプロセッサ
ＷＳ　：作業領域
ＤＩ　：分散性情報
ＵＩ　：利用者インタフェース
ＬＤＢＳ：ローカルＤＢＳ

図2.4　仮想ＤＢＳの構成

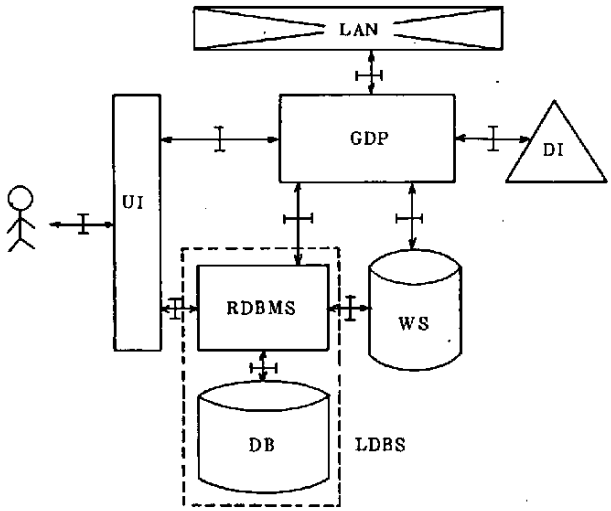


図 2.5　RDBMSによる仮想ＤＢＳの実現

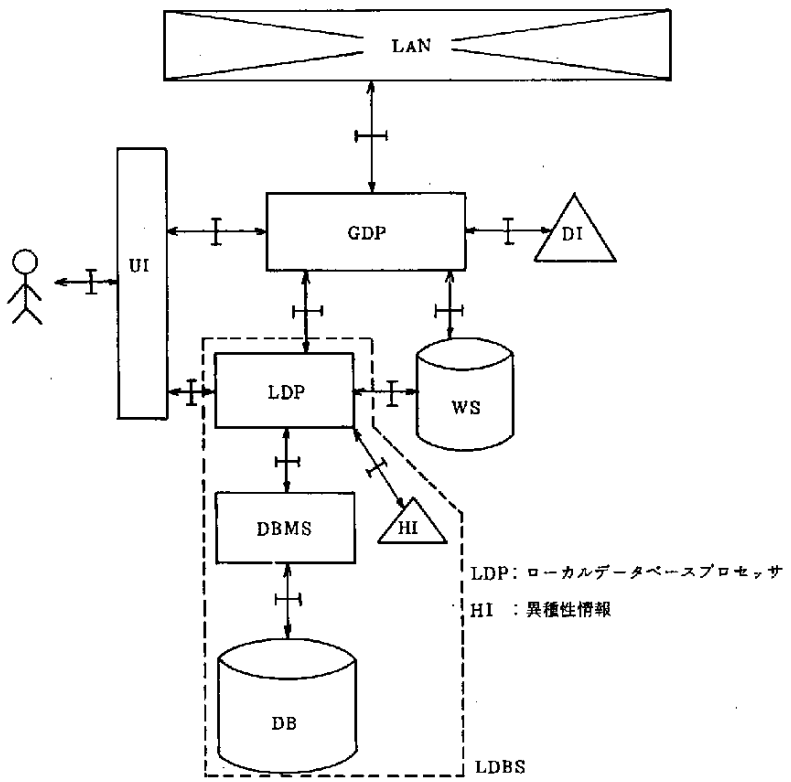


図 2.6　異種ＤＢＭＳによる仮想ＤＢＳの実現

作業領域（ＷＳ）は，ローカルなＤＢから導出された関係，他のＶＳから通信された関係，ＤＢに対する更新データを格納する為の領域であり，全体データベースプロセッサ（ＧＤＰ）の作業用の領域である。

ＧＤＰは，ＶＳ間の通信処理を行うプロセッサである。ＧＤＰは，トランザクションの基本演算を実行する。

利用者インタフェース（ＵＩ）は，高水準な利用者インタフェースである。更に，トランザクションの変換機能を有している。

3. 全体データベースプロセッサ（ＧＤＰ）

全体データベースプロセッサ（ＧＤＰ）は，ＤＤＢＳ処理を行う制御システムである。即ち，(i)全体データ構造上の操作演算を入力として，これをトランザクションに変換し，(ii)トランザクションに基づいて基本操作演算（ローカルなデータの操作演算と通信演算）を実行する。本章では，ＬＩＳの制御システムとなるＧＤＰの論理構造について述べる。ＧＤＰの具体的なＤＤＢＳ処理方法については，４，５，６章で述べる。

3.1 ＧＤＰの論理構造

ローカル情報システム（ＬＩＳ）は，実体（entity）の集合から成る。実体は，受動的なオブジェクト集合と，能動的なプロセスとから成る。プロセスは，オブジェクトに対して適用可能な操作演算を行う。利用者にとって，ＬＩＳ内の実体は，オブジェクトと，利用可能な操作演算の対とみられる。実体のオブジェクトは，この実体のプロセスを通してのみ操作出来る。従って，プロセスは，オブジェクトの保護者（guardian）〔SVOBL79〕でもある。

ＧＤＰは，ＬＩＳ内の実体間の通信を制御するシステムである。ＧＤＰは，ローカルエリア網（ＬＡＮ）の Ethernet データリンク（ＥＤＬ）サービスを，実体間の通信の基本としている。実体間の通信は，ＥＤＬサービス上のメッセージの送受に基づいている。ＬＩＳは，実体の集合である。実体は，階層構造を有している。ＬＩＳでの基本となる実体は，各仮想ＤＢＳ内の関係をオブジェクトとし，これに対する操作演算（検索，更新，視野定義）との対で定まる。このオブジェクト上に抽象的なオブジェクトとして，視野が定められる。視野は，抽象データ型〔ＬＩＳＫＢ７７〕であり，関係上に定義された抽象関係（視野関係）をオブジェクトし，これに許される抽象操作演算が定められる。この為に，ＧＤＰは，次の三階層から構成される。〔図3.1〕。

(i) 基本通信　（ＦＣ）層
(ii) ＤＤＢＳ処理（ＤＤ）層
(iii) 視野処理　（ＶＰ）層

最下位の基本通信（ＦＣ）層は，ＬＡＮのＥＤＬ（Ethernetデータリンク）サービスを用いて，ｎ（≧２）個の上位層の（ＤＤ）実体（（ＤＤ）-entity）間のメッセージ（パケット）の送受サービスを，上位層（ＤＤ層）に提供する。

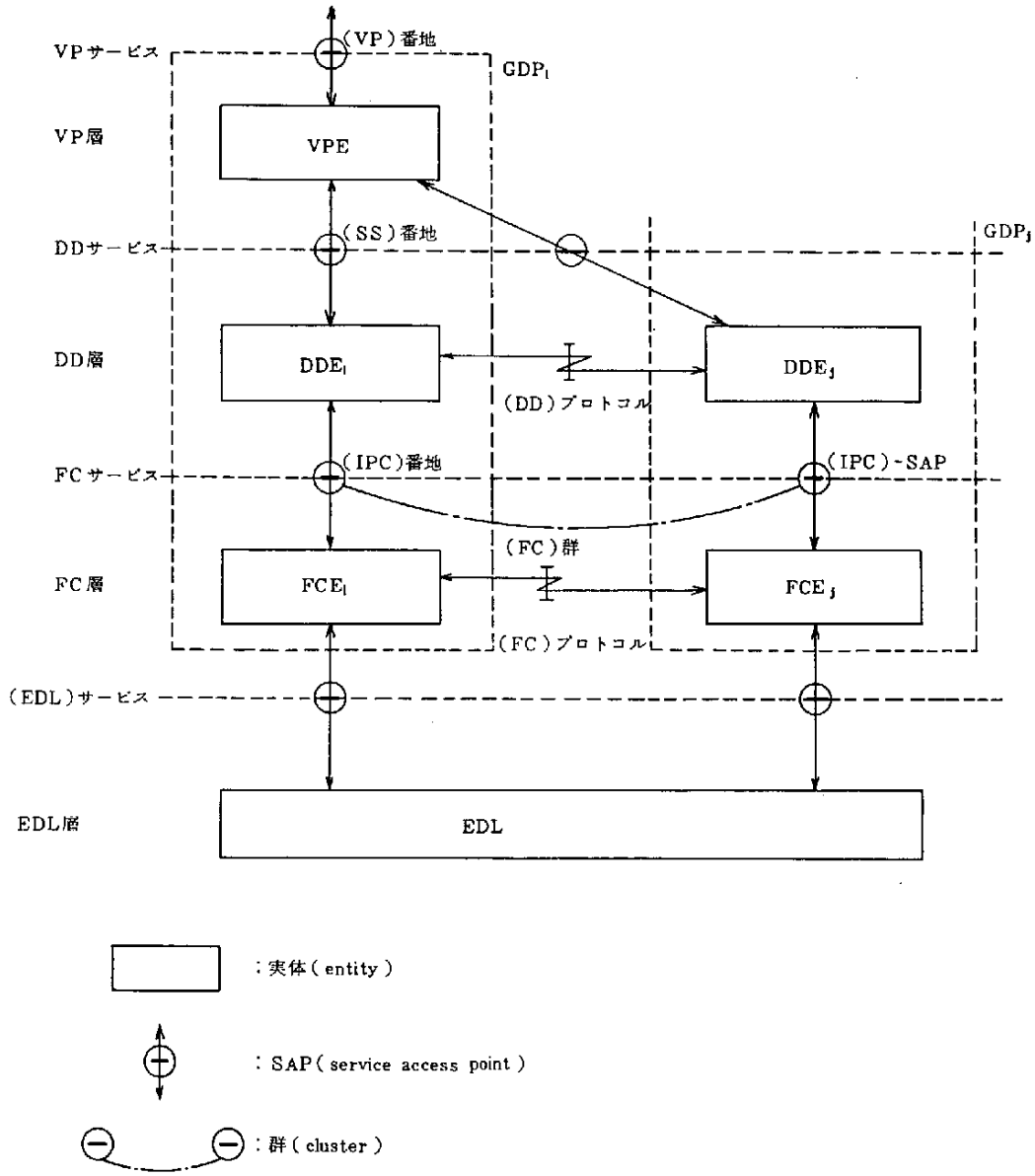


図 3.1　ＧＤＰの階層構造

ＦＣ層は，従来の広域パケット変換網の網及びトランスポート層に対して，複数実体間での多対多通信を基本としている特徴を持つ。

ＦＣ層の一つ上位にあるＤＤＢＳ処理（ＤＤ）層は，ＦＣ層のサービスを用いて，複数ＤＤ実体間での通信と協働により，ＬＩＳの全体データ構造（ＤＤ層のオブジェクトの集合）上の操作演算の処理サービスを，上位（ＶＰ）層に提供する。ＤＤ実体間の通信では，多対多通信が基本となる。ＤＤ層の上位層では，ＬＩＳを，あたかも一つのＤＢＳかの様にみて，操作を行える。

最上位の視野処理（ＶＰ）層は，ＤＤ層のＤＤサービスを用いて，全体データ構造上の視野に対する操作演算の処理サービスを提供する。ＶＰ層の上位層では，視野のオブジェクト（例，郵便箱）と，これに対する抽象的な操作演算（手紙の郵送と受け取り）とが提供される。

3.2　基本通信（ＦＣ）層

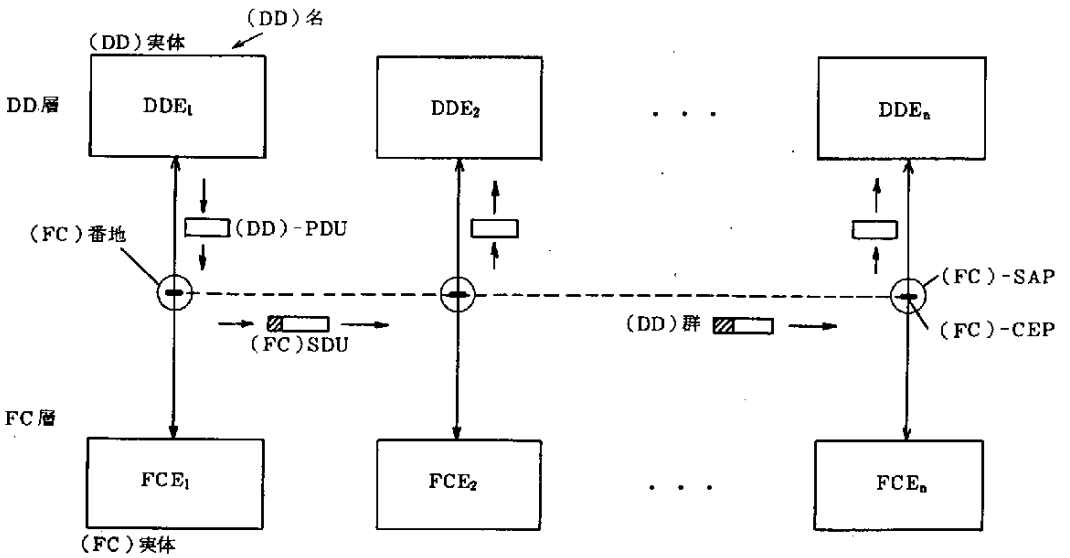
基本通信（ＦＣ）層は，ＥＤＬ（Ethernetデータリンク）層のフレームの送受信サービスを用いて，上位のＤＤ層の実体（ＤＤ実体）間での基本通信手段を提供する。

A. ＦＣサービス

ＤＤ層では，複数のＤＤ実体間での多対多通信が基本となっている。この為にＦＣ層ではＬＡＮの放送通信の特徴〔2.2.1項〕〔ＴＡＫＩＭ８１〕を生かし，広域網の様に一対の実体間の結合（connection）サービスではなく，ｎ（≧２）個の実体間の群（cluster）サービスを提供する。

a. 群サービス

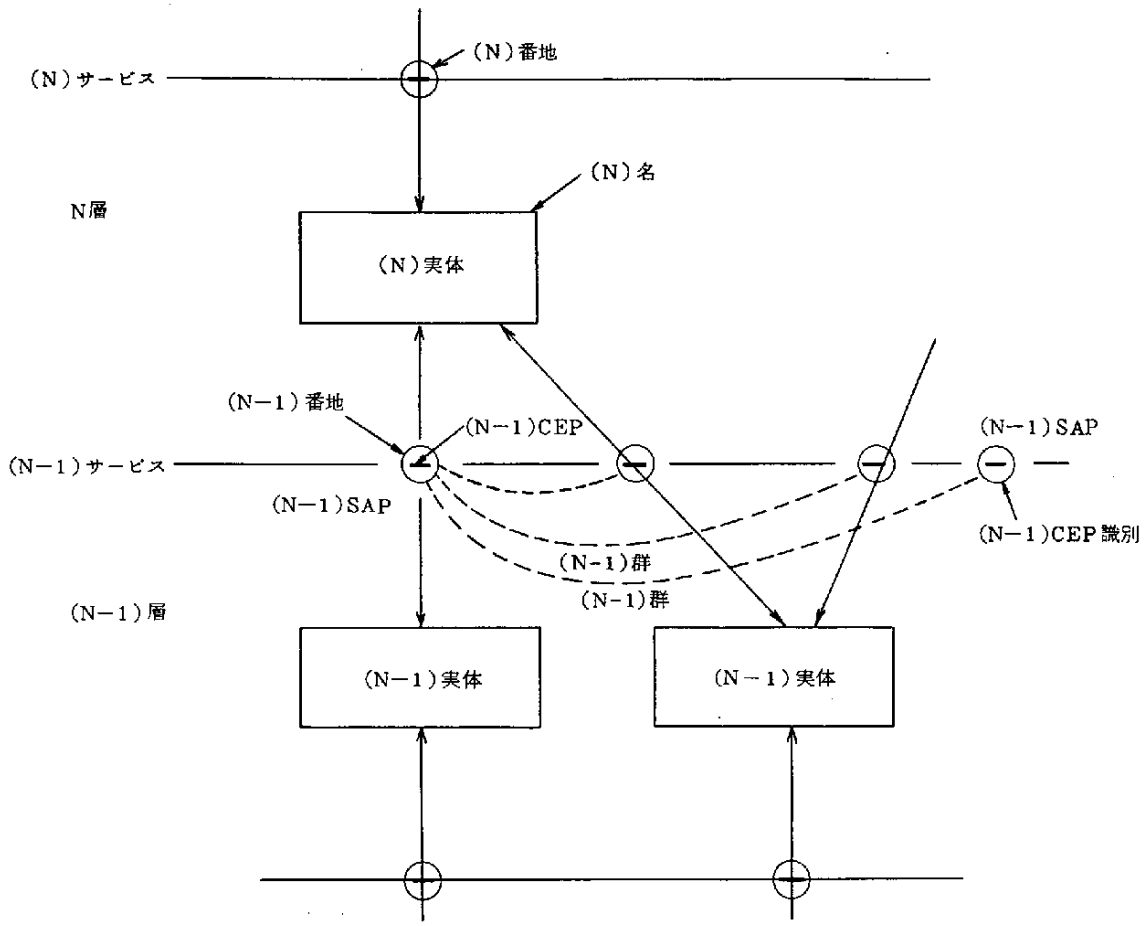
ＦＣ層は，複数の（ＤＤ）実体間に群を形成する。群とは，群内のある（ＤＤ）実体から送信されたＤＤプロトコルデータ単位（（ＤＤ）ＰＤＵ）を，群内の全点に届ける為の通信路である〔図3.2〕。この群を，（ＤＤ）群とする。



SAP＝サービスアクセス点　　PDU＝プロトコルデータ単位
CEP＝群　端　点　　SDU＝サービスデータ単位
DDE＝(DD)実体　　FCE＝(FC)実体

図 3.2　（ＤＤ）群

図 3.3 には，一般的な網の階層構造記述〔ＺＩＭＭＨ８２〕を表してある。



(N)名　　＝N層内で(N)実体を識別する名(title)

(N−1)番地　＝(N)層と(N−1)層の境界で，(N−1)SAPを識別するもの

(N)ディレクトリ＝(N)実体の(N)名と，(N)実体から到達可能な(N−1)番地との対応。
(N)番地に，(N)名は一つだけ対応する。

(N)群　　＝(N)実体間の通信路

図 3.3　通信階層の一般記述

（ＤＤ）群は，以下の性質を有している。

〔性質〕

(i) ＤＤ群に，ある（ＤＤ）実体から送信された複数の（ＤＤ）-ＰＤＵは，群内の全（ＤＤ）実体に，送信順に到着する。

(ii) ＤＤ群内の複数の（ＤＤ）実体，DDE_1，…，DDE_n から送信された各（ＤＤ）-ＰＤＵ P_1，…，P_n を，ＤＤ群内のどの（DD）実体も，群に送信された順に受信する。□

即ち，（ＤＤ）群は，一本のＦＩＦＯ通信路と考えられる〔図3.4〕。

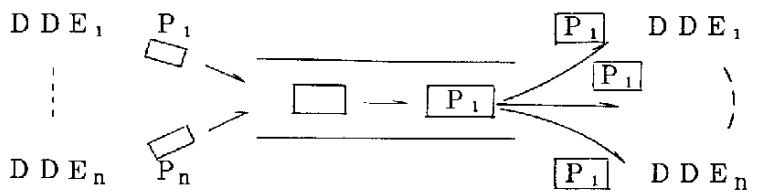


図3.4　群のＦＩＦＯ性

群サービスは，X.25の仮想回線サービスに対応したものと考えられる。（ＤＤ）群には，次の三種のモードがある。

(i) 動的／永続的

(ii) 到着保障／到着非保障

(iii) 集中制御／分散制御

(1) 動的／永続的群サービス

（ＤＤ）群には，群の存在期間によって，次の二種がある。

(i) 動的群（ＤＣ）サービス

(ii) 永続的群（ＰＣ）サービス

動的群（ＤＣ）は，サービスが必要な期間のみ，（ＤＤ）実体間で設定される。永続的群（ＰＣ）は，（ＤＤ）実体が加入したときから永続的に設定される。

(2) 保障／非保障群サービス

更に，（ＤＤ）群内での通信の信頼性について，次の二種がある。

(i) 保障群（ＧＣ）サービス

(ii) 非保障群（ＵＣ）サービス

保障群（ＧＣ）では，（ＤＤ）群内の任意の（ＤＤ）実体から送信された（ＤＤ）－ＰＤＵが，全（ＤＤ）実体で受信されたか，全く受信されないかを保障する。即ち，（ＦＣ）実体が，（ＦＣ）－ＰＤＵの受信に対して，ＡＣＫ（nowledgement）／ＮＡＫ（non－ＡＣＫ），及び無応答に対する時間切れ（time－out）と再送により，通信の信頼性を高める。ＡＣＫ／ＮＡＫ，再送による通信負荷を減じる為に，非保障群（ＵＣ）では，（ＤＤ）実体で送信された（ＤＤ）－ＰＤＵが全点で受信されたことの確認はなされない。この確認は，（ＦＣ）実体間ではなく，（ＤＤ）実体間で行われる。

(3) 集中／分散制御群サービス

群に対しては，以下の操作演算を考えられる。

(i) 群の開設

(ii) 群の終了

(iii) 群への参加

(iv) 群からの退去

これらの群操作演算の制御上，（ＤＤ）群には，次の二種がある。

(i) 集中群（ＣＣ）サービス

(ii) 分散群（ＤＣ）サービス

に対応して，次の二種がある。

(i) 保障データグラム（ＧＤ）サービス

(ii) 非保障データグラム（ＵＤ）サービス

ＧＤサービスでは，n（≧1）個の（ＤＤ）実体に送信された（ＤＤ）－ＰＤＵが，全（ＤＤ）実体で受信されたか，又は全く受信されなかったかを保障する。

集中群（ＣＣ）では，ある（ＤＤ）実体のみが，群操作を行う。この（ＤＤ）群の主（ＤＤ）実体とし，他の（ＤＤ）実体を従（ＤＤ）実体とする。主実体が，群を開設し，新たに実体を加入させ，退去させ，群を終了させる。

これに対して，分散群（ＤＣ）では，各（ＤＤ）実体は，群操作演算を対等に行える。（ＤＤ）群の最初の設定を行う（ＤＤ）実体を，発起実体とする。分散群（ＤＣ）は，自由参加のクラブに対応している。分散群では，群内に（ＤＤ）実体が一つ以下になった時自動的に消滅する。

b. データグラム（ＤＧ）サービス

ＦＣ層は，群サービスに加えて，任意の（ＤＤ）実体間でのデータグラム（ＤＧ）サービスを提供する。（ＤＤ）実体は，群の設定を行わずに，（ＤＤ）－ＰＤＵを，任意のｎ個の（ＤＤ）実体に送信出来る。群サービスに対して，ＤＧサービスでは，送信した（ＤＤ）－ＰＤＵの送信順は保障されない。ＤＧサービスにも，保障／非保障群サービス

B. 基本通信（ＦＣ）インタフェース

ＤＤ層の（ＤＤ）実体は，以下のＦＣインタフェースを用いて，ＦＣサービスを受けられる。

a. 群の開設と終了

群の開設と終了を行うＦＣインタフェースとしては，以下がある。

(1) 群の開設（ＯＣ）

proc　open_Cluster（source,destlist,mode）；

input　source　＝発起（ＤＤ）実体の（ＤＤ）名

destlist＝（ＤＤ）実体名のリスト

mode　＝群のモード

∈｛DC,PC｝×｛GC,UC｝×｛CC,DC｝

output open_Cluster＝
- ＯＫ if 成功
- （OK＝（DD）－CEP識別子）
- ＮＯ if 失敗

群の発起（ＤＤ）実体は，群に含まれるべき全（ＤＤ）実体に，群の開設を open_Cluster によって行う。成功すれば，ＯＫに（ＤＤ）－ＣＥＰ識別子（以降，（ＤＤ）群識別子とする）が表示される。失敗すれば，群開設に応じない（ＤＤ）実体名としての原因が表示される。集中群であれば発起実体が，群の主実体となる。

(2) 群の終了（ＣＣ）

proc　close_Cluster（source,Cluster id）

input　source　＝（ＤＤ）実体名

Cluster_id＝群識別子

output close_Cluster＝
- ＯＫ if 成功
- ＮＯ if 失敗

群の終了を行う。集中群であれば，主実体のみが群終了を行える。

b. 送受信

（ＤＤ）実体は，群内で，（ＤＤ）－ＰＤＵを次の関数によって行える。

(1) 送信（ＳＮＤ）

proc　send（source,clusterid,pdu）；

input　source　＝送信（ＤＤ）実体名

clusterid＝群識別子

pdu　＝（ＤＤ）－ＰＤＵ（送信データ）

output

send＝{ ＯＫ if 送信に成功，保障群のときは，データは，群内の全実体に受信される。
　　 ＮＯ if 送信に失敗。保障群のとき，ＮＯに，受信しなかった（ＤＤ）実体名がセットされる。

群内の（ＤＤ）実体から，群内の（ＤＤ）実体に，（ＤＤ）－ＰＤＵを送信する。送信が終了するまで制御は，かえされない。

(2) 受信（ＲＥＣ）

proc　receive（dest,clusterid ,source,pdu ）

input　dest　＝受信（ＤＤ）実体名

clusterid　＝群識別子

output　source　＝受信パケットの送信（ＤＤ）実体名

pdu　＝受信（ＤＤ）-ＰＤＵ（受信データ）

receive　＝{ ＯＫ if 成功
　　 ＮＯ if 失敗

c. 群への加入／退去

開設されている群に，新たに（ＤＤ）実体が加入する場合と，群から（ＤＤ）実体が退去する場合について考える。

(1) 群からの退去（ＲＴＲ）

(i) 分散群

proc　retire（source,clusterid）；

input　source　＝（ＤＤ）実体名

clusterid＝群識別子

output　retire = { OK if 成功 / NO if 失敗

分散群の場合には，（ＤＤ）群内の退去を行いたい（ＤＤ）実体が，退去することを群内の全実体に通知し，確認がとれれば，ＯＫとなる。

(ii) 集中群

proc　retire (source,entity_list,
nclusterid)

input　source と clusterid は(i)と同じである。
entity_list　＝退去させたい（ＤＤ）実体名のリスト
nclusterid　＝新しい群識別子

output　(i)と同じである。

集中群では，主実体が，退去させたい（ＤＤ）実体を指定する。このとき，群に新たな識別子を与える。

(2) 群への加入（ＪＯＩＮ）

群に対して，新たに（ＤＤ）実体を加入させる操作について考える。加入は，群内のある（ＤＤ）実体が，加入させたい（ＤＤ）実体に，加入要求することと，相手の受け入れ，及び群内の全（ＤＤ）実体の受け入れ確認によって行われる。

a) 呼びかけ加入

proc　joinus_cluster (source,destlist,clusterid);

input source　＝主（ＤＤ）実体名
destlist　＝加入される（ＤＤ）実体名のリスト
clusterid ＝群識別子

output

joinus_cluster = { OK if 成功 / NO if 失敗

集中群では，主（ＤＤ）実体が，加入させたい（ＤＤ）実体に，加入命令を出す。加入を受けいれ，これを群内の全（ＤＤ）実体が確認したときにＯＫとなる。

分散群では，（ＤＤ）群のどれかの（ＤＤ）主体が加入呼びかけを行う。

b) 加入（ＡＤＤ）

proc　add_me_cluster（source,clusterid）;

input　source　＝加入希望の（ＤＤ）実体名

clusterid　＝群識別子

output

add_me_cluster　＝｛ＯＫ　if　成功
　　ＮＯ　if　失敗

分散群では，その群識別子を知っている（ＤＤ）実体は，自由にその群に，この関数を用いて参加できる。群内の全（ＤＤ）実体が，加入を確認した時，ＯＫとなる。

3.3　ＤＤＢＳ処理（ＤＤ）層

ＤＤＢＳ処理（ＤＤ）層は，基本通信（ＦＣ）層のサービスを用いて，全体データ構造（オブジェクト集合）上の操作演算サービスを，上位のＶＰ層に提供する。視野処理（ＶＰ）層の実体は，ＤＤ層のサービスを用いて，全体データ構造内の各オブジェクト（関係）の所在を意識せずに，操作演算を行える。操作演算としては，(i)検索と更新，(ii)全体トランザクション，及び(iii)視野管理（ＶＭ）（視野の定義と消去）がある。各々の操作演算に対応して，以下のサービスがある。

(1)　分散問合せ処理（ＤＱＰ）サービス

(2)　分散トランザクション処理（ＤＴＰ）サービス

(3)　視野管理（ＶＭ）サービス

(1)は(i)を，(2)は(ii)を，(3)は(iii)の演算をＶＰ層に提供する。

3.3.1　（ＤＤ）実体

ＤＤＢＳ処理（ＤＤ）層の実体（（ＤＤ）実体）は，一つの仮想ＤＢＳ内に一つ存在している。（ＤＤ）実体は，オブジェクト集合と，これに対する操作演算を行うプロセスとから成る。オブジェクト集合は，仮想ＤＢＳのローカルな関係型データベース（ＤＢ）である。オブジェクト集合内の各オブジェクトは，ＤＢ内の関係である。プロセスは，オブジェクト集合に対する検索演算と更新演算を行え，他の複数の（ＤＤ）実体間での通信を行える。（ＤＤ）実体内のオブジェクト集合は，そのプロセスを通してのみ操作を行える。

3.3.2 分散問合せ処理（ＤＱＰ）サービス

分散問合せ処理（ＤＱＰ）サービスは，ＤＤ層のオブジェクト集合（全体データ構造）に対するＲＱＬ操作演算Ｑ＝（op，Λ，Ψ）（op ε ｛検索，消去，追加，置換｝）を，上位のＶＰ層に提供する。ＶＰ層は以下のＤＱＰインタフェースを用いて，ＤＱＰサービスを受けられる。

(1) 全体問合せ（ＲＱＬ）

```
proc  Dqp ( subject,objectlist,op,Λ,Ψ,dest ) ;
input  subject    ＝（ＶＰ）実体名
       objectlist ＝オブジェクト名のリスト
       op         ＝基本操作演算
                    ε｛検索，消去，追加，置換｝
       Λ          ＝目標リスト
       Ψ          ＝検索条件式
       dest       ＝目標集合（オブジェクト）を格納する（ＤＤ）
                    実体名
output
          ⎧ ＯＫ  if  全体問合せが起動した
   Dqp ＝ ⎨             （ＯＫ＝全体問合せ名）
          ⎩ ＮＯ  if  失敗
                       （ＮＯ＝失敗原因）
```

全体問合せＱ＝（op，Λ，Ψ）をＤＤＢＳ処理を開始した時に制御が返される。

(2) 実行状態の監視

```
proc    status ( subject, Query ) ;
input   subject ＝（ＶＰ）実体名
        Query   ＝全体操作演算の識別子
output  status  ＝全体操作演算の実行状態
                  ∈｛executing, normalend,
                     abnormalend｝
```

全体問合せＱ＝（op，Λ，Ψ）の実行状態を表示する。

(3) 実行停止（ＫＩＬＬ）

```
proc    kill ( subject, Query ) ;
input   subject ＝（ＶＰ）実体名
```

Query＝全体操作演算の識別子

output　kill＝{ OK　if 成功
　　NO　if 失敗

実行中の全体操作演算を強制的に終了させる。

3.3.3 分散トランザクション処理（ＤＴＰ）サービス

分散トランザクション処理（ＤＴＰ）サービスは，全体データ構造上の全体トランザクションのＤＤＢＳ処理サービスを，上位のＶＰ層に提供する。ＤＴＰサービスは，次のＤＴＰインタフェースを通して利用出来る。

(1) 全体トランザクション

proc　Dtp（subject，objectlist，transaction）

input　subject　＝（ＶＰ）実体名
　　objectlist　＝（ＤＤ）オブジェクト名のリスト
　　transaction＝全体トランザクション

output　Dtp＝{ OK if 全体トランザクションはコミットされた
　　NO if 全体トランザクションはアボートされた

3.3.4 視野管理（ＶＭ）サービス

視野管理（ＶＭ）サービスは，全体データ構造上に視野の定義，及び消去を行う。

A. 視　野

全体データ構造上の視野ｖは，抽象データ型である。視野ｖは，五字組（Ω_v，V_v，O_v，Q_v，R_v）として定義される。視野は，(i)抽象的なオブジェクト（抽象データ構造）（Ω_v，V_v），(ii)これに許される抽象操作演算の集合O_v，(iii)及び抽象データ構造の全体データ構造上の定義Q_v＝（Λ_v，Ψ_v）と，抽象操作演算の全体操作演算による実現R_vとから定められる。Ω_vを抽象データ構造の属性集合で，抽象関係スキームとする。V_vを，Ω_vに値を与えた抽象関係とする。Q_v＝（Λ_v，Ψ_v）は，（Ω_v，V_v）の定義式で，Λ_vはΩ_vを与える目標リスト，Ψ_vは条件式である。Λ_vとΨ_vは，ＲＱＬと同じ形式を持つ。

視野は，抽象データ構造をオブジェクトとし，抽象操作演算の実行機構をプロセスとした実体と考えられる。　ＬＩＳの利用者には，視野が

与えられ，抽象データ構造を，許される抽象操作演算によって操作が出来る。

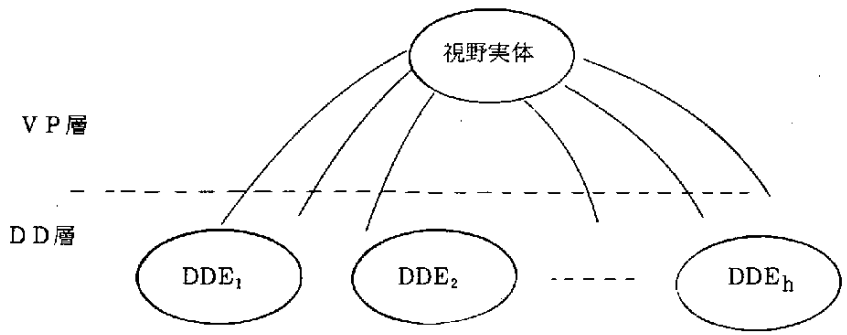


図3.1　視野実体

B. DMインタフェース

(1) 視野の定義（DEFV）

proc　def_View　（subject, objectlist, v）;

input　subject　＝（DP）実体名（VP層の管理実体の名前）

objectlist　＝（DD）オブジェクト名のリスト

v　＝視野定義

＝（Ω_V, O_V, （Λ, Ψ）, R_V）

output def_View＝{OK if 成功（OK＝視野識別子）
NO if 失敗

(2) 視野の消去（DRPV）

proc　drop_View（subject, View）;

input　subject＝（DP）実体名（VP層の管理実体の名前）

View　＝視野識別子

output drop_View＝{OK if 成功
NO if 失敗

3.3.5 （ＤＤ）実体の構成

以上述べてきた（ＤＤ）サービスを提供する（ＤＤ）実体の構成を図3.2に示す。

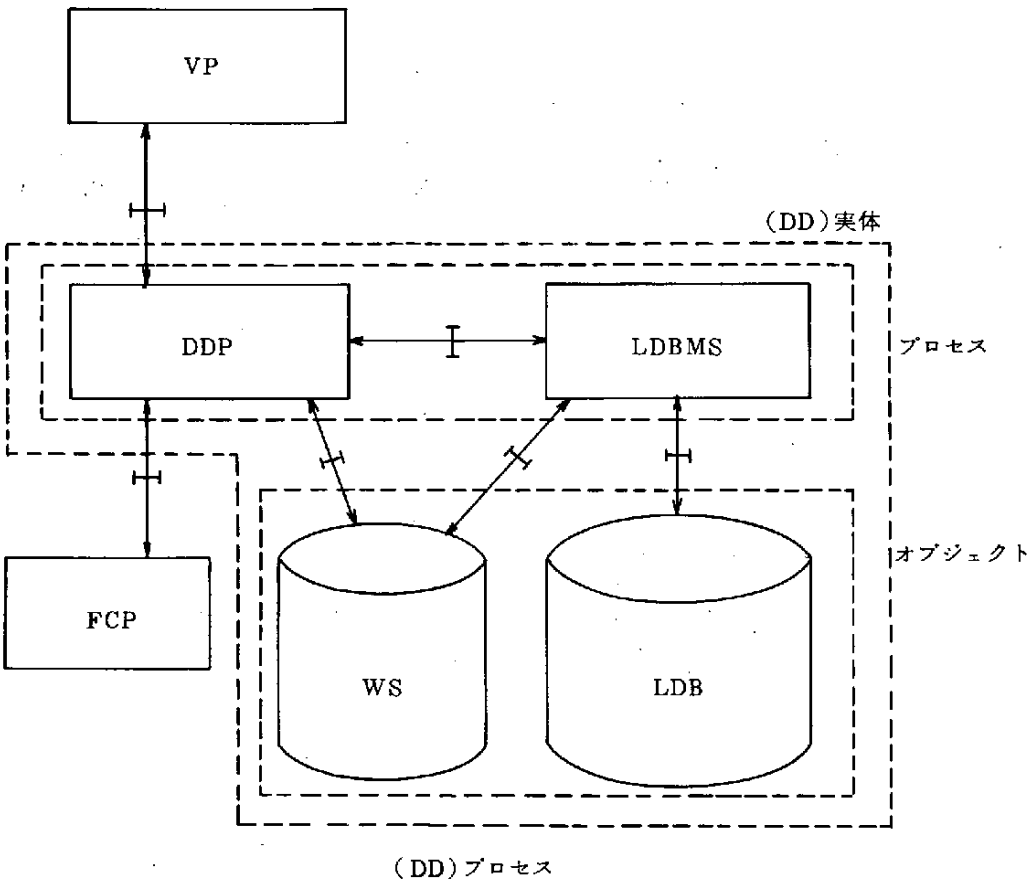


図3.2　（ＤＤ）プロセスの構成

ＤＤ実体は，関係をオブジェクトとしたオブジェクト集合と，この上の操作演算を実行出来るプロセスとからなる。仮想ＤＢＳのローカルＤＢＳは，ローカルな関係集合としてのＬＤＢと，これに対する操作を行うLDBMSから成る。他の（ＤＤ）実体との間でDDBS処理を行う為に，作業用の関係を格納する作業領域ＷＳと，通信とＷＳへの操作を行うＤＤＰ（ＤＤプロセス）がある。

3.4 視野（ＶＰ）層

視野（ＶＰ）層は，ＤＤ層のサービスを用いて，全体データ構造上の視野に対する操作演算を上位層に提供する。視野を，抽象データ型と考える。即ち，全体データ構造上の抽象データ構造と，これに対して許される抽象的な操作演算及びこれの全体モデルの操作演算での実現が定義されていて，利用者は，抽象データ構造に対して，抽象操作演算のみを適用出来る。この抽象データ構造を抽象関係とする。又，抽象関係と抽象演算，及びその表現（実現）を併せて視野とする。

視野は，ケーパビリティとも考えられる。ＬＩＳのサブジェクト（例，利用者，プログラム）は，自分の利用可能な抽象オブジェクト。O(即ち，抽象関係)と，利用可能な抽象操作演算の集合Ｐとの対（Ｏ，Ｐ）のリスト（ケーパビリティリスト又はＣ－リスト）を有している。即ち，自分がケーパビリティを有している抽象オブジェクトを，ケーパビリティ内の操作演算でのみ操作出来る。従って，視野を，安全性の管理の為にも用いられる。

以下に，ＶＰ層の上位層に対するインタフェースを与える。

(1) 視野へのログオン

```
proc    logon_View ( subject, capability )
input   subject    ＝ＬＩＳのサブジェクトの識別子
        capability ＝視野の識別子
output  logon_View ＝ { ＯＫ    if 成功
                      { error if 失敗
```

利用者は，視野への利用要求を出す。

(2) 抽象演算の実行

```
proc    apply ( subject, capability, operation )
input   subject    ＝ＬＩＳのサブジェクトの識別子
        capability ＝視野の識別子
        operation  ＝抽象操作演算
output  apply ＝ { ＯＫ    if 成功
                 { error if 失敗
```

視野の抽象操作演算を実行する。抽象操作演算が検索演算の時は，ＤＤ層のＤＱＰサービスが利用される。更新演算の時は，抽象演算の表現であるトランザクションが，ＤＤ層のＤＴＰサービスによって実行される。

(3) 視野への操作の終了

proc　logoff_View (subject, capability)

input　subject　＝上位層のサブジェクトの識別子

　　capability ＝視野の識別子

output　logoff_View ＝ { OK　if 成功
　　error　if 失敗

4. 全体問合せのＤＤＢＳ処理方法

本章では，全体問合せQの条件式Ψと目標リストΛとを満足する目標集合Gを求めるトランザクションを求め，実行するＤＤＢＳ処理方法について論じる。

4.1 ＤＤＢＳ処理の戦略

全体問合せQを（Λ，Ψ）とする（Λは目標リスト，Ψは検索条件式）。Qはｎ個の仮想ＤＢＳ，VS_1，…，VS_n内の関係を参照しているとする。条件式Ψは，次式の様に積正規化される。

$$\Psi = \bigwedge_{i=1}^{n} \Psi_i \bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^{n} \Psi_{ij} \quad \cdots\cdots (4.1)$$

ここで，Ψ_iは，VS_i内の関係についての条件式である。Ψ_{ij}は，VS_iとVS_j内の関係間の条件式である。又，Ψ_iを，目標リストΛ内のVS_i内の関係の目標属性と，各Ψ_{ij}内のVS_iの属性との集合とする。全体問合せQ＝（Λ，Ψ）を満足する目標集合Gを得る為には，以下の操作を行える必要がある。

(i) 各VS_iは，問合せ（λ_i，Ψ_i）を満足する目標集合X_iを導出する。この（λ_i，Ψ_i）を，全体問合せQ＝（Λ，Ψ）のローカル問合せとする。

(ii) 各VS_iは，条件式$\bigwedge_j \Psi_{ij}$を満足する関係X_i内の組集合G_iを導出する。

(iii) 各VS_iは，G_iを目標点のVS_0に送信する。VS_0は，VS_iから送信されたG_iを結合し，目標集合Gを得る。

(i)は，各VS_i毎に独立に行える。(i)の処理を初期ローカル問合せ処理（ＩＬＱＰ）〔HEVNA78，TAKIM82a〕という。(ii)と(iii)では，複数の仮想ＤＢＳ間の条件式を調べる為に，仮想ＤＢＳ間でのデータの通信が必要になる。この通信に伴う負荷が，全体問合せQのＤＤＢＳ処理の主要な負荷となることから，(ii)と(iii)を有効に行う方法が必要になる。ここでは，特に(ii)と(iii)を有効に行う方法について論じる。

4.2 基本定義

全体問合せのＤＤＢＳ処理手順を説明するうえで必要となる基本操作演算の定義を与える。

A．ビットマップ生成演算//

Xを全順序集合$<a_1, \cdots, a_n>$（ここで，$a_1 < \cdots < a_n$）とし，YをXの部分集合とする。Xに対するYのビットマップ生成演算Y//Xを次の様に定める。

$$Y/\!/X = \text{系列} < b_1, \cdots, b_n > \quad \text{ここで} \ b_i = \begin{cases} 1 & \underline{\text{if}} \ a_i \in Y \\ 0 & \underline{\text{otherwise}} \end{cases}$$

Y//Xを，YのXに対するビットマップとする。

以下に例を与える。

$X = <A, D, E, B, H>$

$Y = \{A, B, E\}$

$Y/\!/X = <1, 0, 1, 1, 0>$

B．ビットマップ制限演算*

AのXとYについて考える。$<X>_i$をa_iとし，BM＝Y//Xとする。R（A，B）を関係とし，AとBは各々属性集合とし，dom(A)＝$\{a_1, \cdots, a_n\}$とする。このとき，以下を定義する。

$$X^* BM = <<X>_{k_1}, \cdots, <X>_{k_l}> \quad (l \leq n)$$

ここで，$<BM>_{k_i} = 1$で，$k_i < k_j$（$i < j$）。即ち，ビットマップBMのビットオン（＝1）に対応した系列Xの値の系列である。

$$R^* BM = \{ t \mid t = <a, b> \in R \wedge a \in X^* BM \}$$

$X = <A, D, E, B, H>$，$Y = \{A, B, E\}$とする。R（A，B）を以下の関係とする。

R	A	B
	A	4
	A	6
	B	7
	C	8
	C	10
	D	2
	E	15
	G	2

このとき，$BM = Y /\!/ X = <1, 0, 1, 1, 0>$

$X^{*}BM = <A, E, B>$

$R^{*}BM =$

A	B
A	4
A	6
B	7
E	15

C．コード化演算∧

BのXとRを考える。σを置換$\{0/a_1, 1/a_2, \cdots, (n-1)/a_n\}$とする。このとき，以下を定義する。

$$R^{\wedge}X = \{<v, b> \mid <a, b> \in R \wedge v = \sigma a\}$$

$R^{\wedge}X$を関係$S(\tilde{A}, B)$とする（ここで，$dom(A) = \{0, 1, \cdots, n-1\}$）。$S(\tilde{A}, B)(= R^{\wedge}X)$を，$R(A, B)$の$X$によるコード化関係とする。又，$\tilde{A}$を，属性Aの$X$によるコード化属性とする。

Bの例を考える。このとき，$X = <A, E, B>$で$\sigma = \{0/A, 1/E, 2/B\}$となる。

$R^{\wedge}X = S(\tilde{A}, B)$

R	A	B
	A	4
	A	6
	B	7
	E	15

S	$\tilde{A}$	B
	0	4
	0	6
	2	7
	1	15

D．射影演算〔 〕

$R(A, B)$を関係とし，AとBを各々属性の部分集合とする。このとき，以下を定める。

$R〔A〕$＝ 関係RのAの値の集合（冗長値は含まない）。

＝ RのAについての射影。

E．結合演算

$R(A, B)$と$S(C, D)$を関係とし，A，B，C，Dを各々，属性の部分集合とする。$dom(A) = dom(C)$とするとき，以下を定義する。

$$R〔A \theta C〕S = \{<r, s〔D〕> \mid r \in R \wedge s \in S \wedge r〔A〕\theta s〔C〕\}$$

ここで，$\theta \in \{<, \leq, =, \geq, >, \neq\}$。

F．放送通信演算

(i) broadcast(X)； Xを，ある順序集合とする。Xを全点に，値順に放送する。

(ii) Z←rec( )；放送された情報を，送信順に受信し，順序集合Zとする。

G．組数と情報量

Rを関係とする。このとき，以下を定める。

|R|＝関係Rの組数(cardinality)。

‖R‖＝関係Rの全情報量(サイズ)。

＝|R|×(Rの組の長さ)〔bit〕。

4.3 単純問合せのDDBS処理方法

単純問合せを，次の様に定義する。

〔定義〕

全体問合せ$Q=(\Lambda, \Psi)$の検索条件式Ψが，一つの属性aに対して，各点iとjについて，$\bigwedge_{i=1}^{n-1} \bigwedge_{j=i+1}^{n} (x_i.a = x_j.a)$の形式を持つとき(ここで，$x_h$は，$VS_h$の関係$X_h$の組変数とする(h＝i，j))，Qを単純問合せとする。□

本節では，LANの放送通信〔2.2.1項〕の特徴を生かした単純問合せのDDBS処理方法を与える。より一般の問合せのDDBS処理方法については，次節で論じることにする。

A．DDBS処理方法としてB^2R通信手順

利用者からの全体(単純)問合せ$Q=(\Lambda, \Psi)$を受けた点の仮想DBSを，目標点0の仮想DBS VS_0とする。単純問合せQのDDBS処理の為のB^2R(Broadcast and Bit-map Responsing)通信手順〔TAKIM 81，82c，84b〕を以下に示す。

〔B^2R通信手順〕

(1) 目標VS_0は，利用者からの全体問合せ$Q=(\Lambda, \Psi)$を受けたならば，Qの参照する関係を有する仮想DBS間で，群(cluster)を設定する。Qを，群内の全仮想DBSに放送する。

(2) 群内の各仮想DBS VS_jは，Qを受信したならば，Qのローカル問

合せ$Q_j = (\lambda_j, \Psi_j)$〔4.1節〕を実行する。*) Q_jの目標集合X_jを導出する。$C_{ja} = |X_j〔a〕|$を求め，X_jの物理構造の情報を載せたPI_jメッセージを放送する。Λ_jをX_jの目標属性集合とする。

(3) 各VS_jは，全点VS_hからのPI_hを受信したならば，ある目標値を最少とする点VS_iと関係X_iとを求める。

(4) 〔送信関係の送信〕

送信点iのVS_iは，関係X_iから$Y_i = X_i〔a〕$を，VS_iのバッファB_iに読出し，ある順序で放送する。

```
Y_i  ← X_i〔a〕;
broadcast (Y_i);
BM_ii ← Y_i ∥ Y_i;   j ← i, go to (6);
```

(5) 〔送信関係の受信とビットマップの放送〕

各受信点VS_jは，VS_iからのY_iを受信し，X_jとaの値について結合を行う。

```
Z_j   ← rec( );
X'_j  ← X_j〔a=a〕Z_j;
BM_ij ← X'_j〔a〕∥ Z_j;
broadcast (BM_ij);
```

(6) 〔ビットマップの受信〕

群内の$\Lambda_j \neq \emptyset$なる全VS_jは，全ビットマップを受信し，積をとる。

```
if Λ_j ≠ ∅ then
{while (全ビットマップを受信していない)
    {BM ← rec( );   BM_ij ← BM_ij ∧ BM;}}
```

ビットマップBM_{ij}を用いて，関係X_iの制限を行う。

```
X_j ← X'_j * BM_ij;
```

---

*) CODASYL 型等の関係型と異種のDBSに対しては，LDP〔TAKIM 80, 82b, 82d, 83a, 84a〕によって，Q_iをCOBOL DMLプログラムに変換し，CODASYL DBS上でこれを実行させ，Q_iの目標集合X_iを導出する。

(7) 〔目標集合の放送〕

各VS_jは，目標集合$\Lambda_j \neq \emptyset$ ならば，X_jをVS_0に放送する。

if $\Lambda_j \neq \emptyset$ then broadcast(X_j)

VS_0は，X_jを結合して，Qの目標集合Gを求める。

```
        G ← rec ( );
while (Λj≠∅なる全Xjを受信していない)
  {   Gj ← rec ( );
      G ←G〔a=a〕Gj ; }
      G ←G〔Λ〕;        □
```

図4.1に，B^2R通信手順を状態遷移図で与える。

例を考える。関係X，Y，Zが各々VS_1，VS_2，VS_3にあるとする。

X	a	b
	A	4
	A	6
	B	7
	C	2
	C	3
	D	10

1

Y	a	c
	B	4
	C	8
	C	10
	D	12
	D	6
	D	11
	E	8
	F	9

2

Z	a	d
	A	2
	B	1
	C	5
	E	8
	E	9
	F	11
	G	16

3

このとき，次の単純問合せを考える。

```
var (x, X) (y, Y) (z, Z);
get into temp (x.b, y.c, z.d)
where x.a = y.a and y.a = z.a and z.a = x.a ;
```

これは，図4.2の問合せグラフで表される。図4.3は，B^2R通信手順によって，(1)VS_1の関係Xが放送され，(2)ビットマップの応答が放送され，(3)目標集合がVS_0に送出されることを表している。

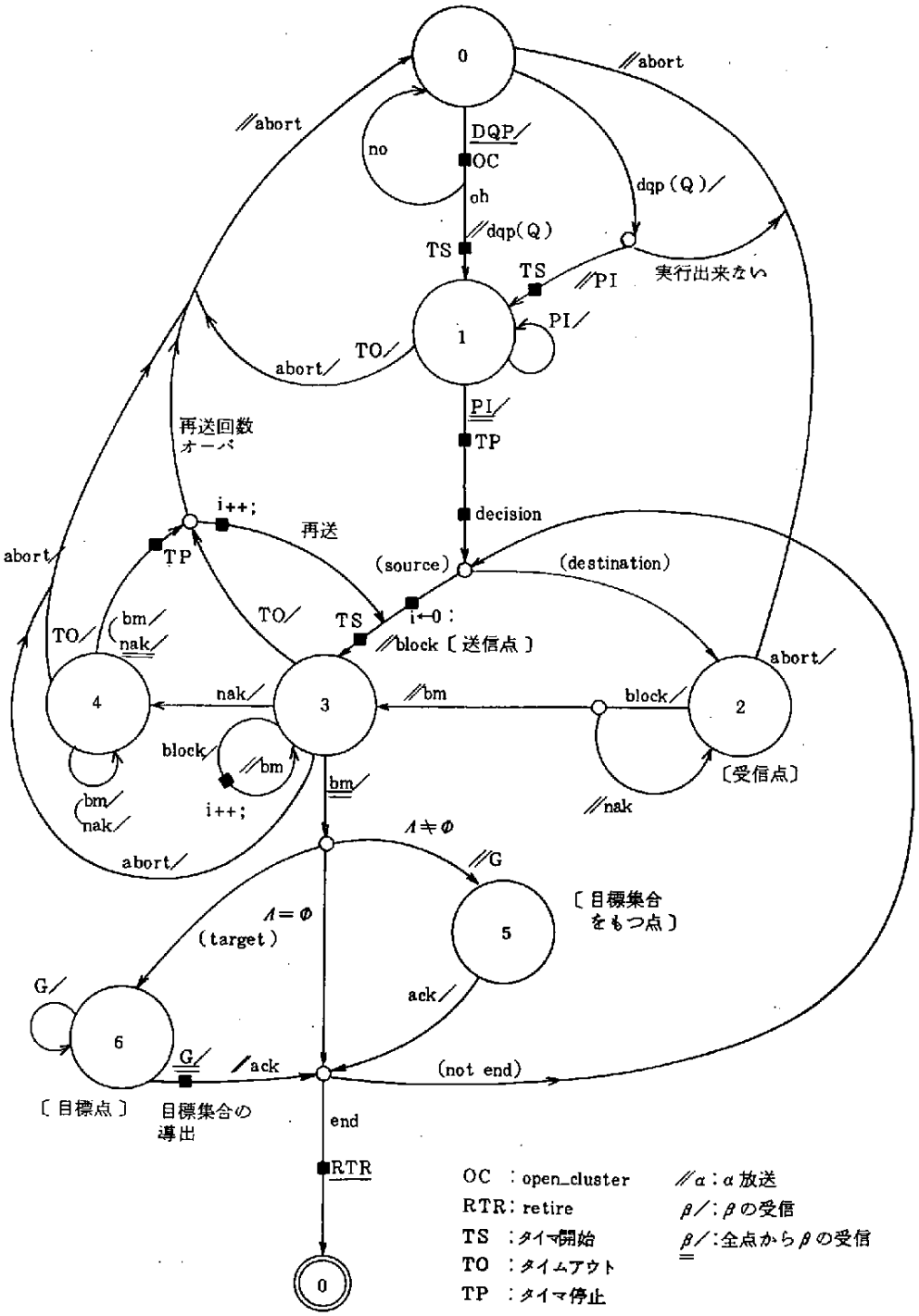


図 4. 1　B^2R 通信手順の状態遷移図

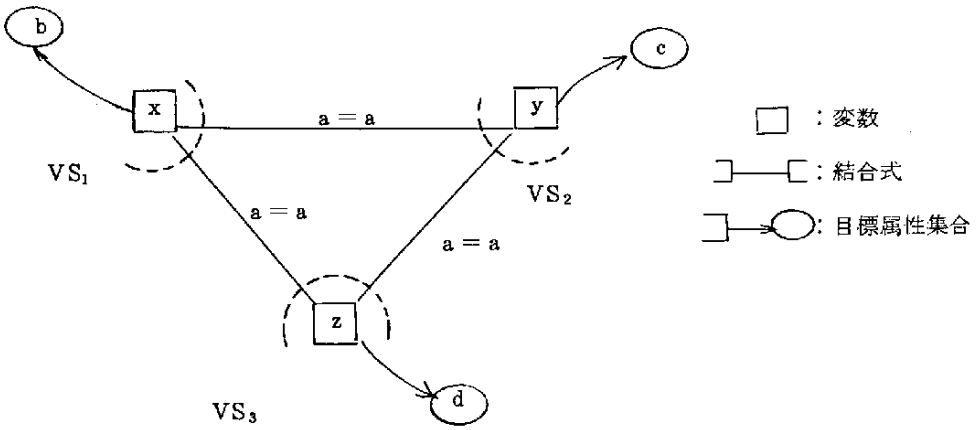


図 4.2　問合せグラフ

B．B^2R通信手順での全通信量と通信時間

B^2R通信手順での全通信量Tは，(i)のQの放送（T_1），(ii)の応答の放送(T_2)，(iii) Y_i の放送（T_3），(iv) BM_{ij}の放送（T_4），(v) G_h の送信（T_5）での通信量の和である。Qの情報量を‖Q‖〔bit〕とする。（2.11）式で$\alpha = 0$とすると，T_1～T_4 は，各々，次式で与えられる。

$$T_1 = \beta \cdot \| Q \| \quad \cdots\cdots (4.2)$$

$$T_2 = \beta \cdot n \cdot \ell og_2 | dom[a] | \quad \cdots\cdots (4.3)$$

$$T_3 = \beta \cdot \| X_i [a] \| \quad \cdots\cdots (4.4)$$

$$T_4 = \beta \cdot (n-1) \cdot | X_i [a] | \quad \cdots\cdots (4.5)$$

T_5 を考えるうえで，以下の仮定を設ける。

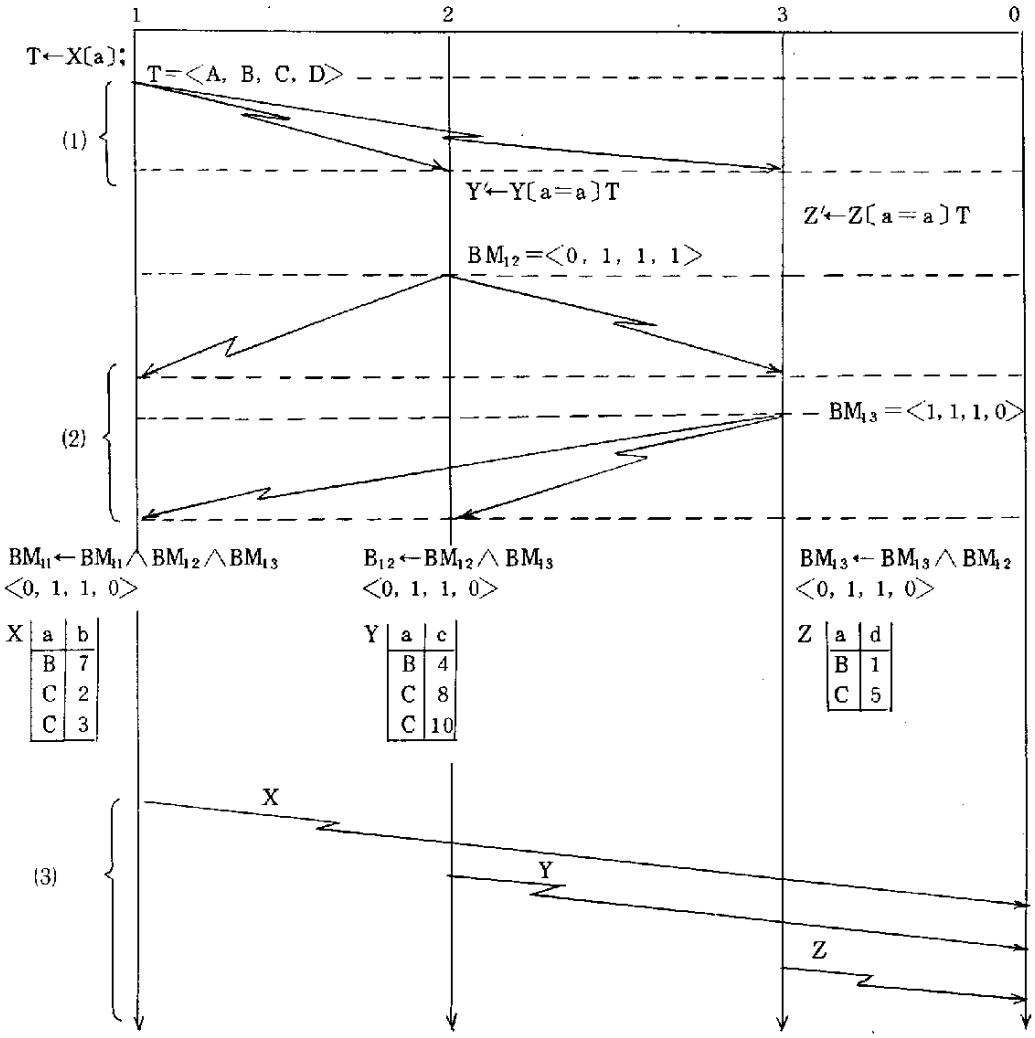
〔仮定 4.1〕

各関係X_hで，属性aの値の分布は均一でかつ独立である。□

この仮定のもとで，X_hの属性aの結合選択度σ_hを$|X_h[a]| / |dom(a)|$とする〔HEVNA78〕と，T_5は次式で与えられる（以降αは0とする）。

$$T_5 = \beta \cdot \sum_{h=1}^{n} [\varsigma_h \cdot \kappa_h \cdot \| X_h \|] [bit] \quad \cdots\cdots (4.6)$$

ここで，$\kappa_h = \sigma_1 \cdot \cdots \cdot \sigma_{h-1} \cdot \sigma_{h+1} \cdot \cdots \cdot \sigma_n$ $\cdots\cdots (4.7)$



X	a	b
	B	7
	C	2
	C	3

Y	a	c
	B	4
	C	8
	C	10

Z	a	d
	B	1
	C	5

G	a	b	c	d
	B	7	4	1
	C	2	8	5
	C	2	10	5
	C	3	8	5
	C	3	10	5

図 4.3　B^2R 通信手順例

$$\varsigma_h = \begin{cases} 1 & \underline{\text{if}} \quad \Lambda_h \neq \emptyset \\ 0 & \underline{\text{otherwise}} \end{cases} \quad \cdots\cdots (4.8)$$

従って，全通信量Tと通信時間Rは，以下となる。

$$T = \beta \cdot \{ \| Q \| + n \cdot \ell og_2 | dom(a) | + \| X_i〔a〕\| + (n-1) \cdot | X_i〔a〕| + \sum_{h=1}^{n} 〔\varsigma_h \cdot \kappa_h \cdot \| X_h \|〕\}〔bit〕 \quad \cdots\cdots (4.9)$$

$$R = T / \omega 〔sec〕 \quad \cdots\cdots (4.10)$$

ωは，網の伝送速度〔bit／sec〕である。

4.4. 複数属性問合せのＤＤＢＳ処理方法

次に，単純問合せよりも一般的な複数属性問合せについて考える。複数属性問合せを以下に定義する。

〔定義〕

全体問合せ$Q = (\Lambda, \Psi)$の検索条件式Ψが，VS_iとVS_j間の等結合式$x_i \cdot a_h = x_j \cdot a_h$の積で（$x_i$と$x_j$は各々，$VS_i$と$VS_j$の関係$X_i$と$X_j$の組変数）あり，複数の結合属性$a_h$が存在しているとき，Qを複数属性問合せとする。□

A．記号の定義

複数属性問合せ$Q = (\Lambda, \Psi)$に対して，以下を定義する。

$\Sigma = \{ X_1, \cdots, X_n \}$
＝Qの参照する関係の集合（ここで，X_iは，VS_i内の関係）。
A＝Qの結合属性の集合。

各関係$X_i \in \Sigma$に対して，

Λ_i＝関係X_iの目標属性集合，
A_i＝X_iの結合属性の集合。

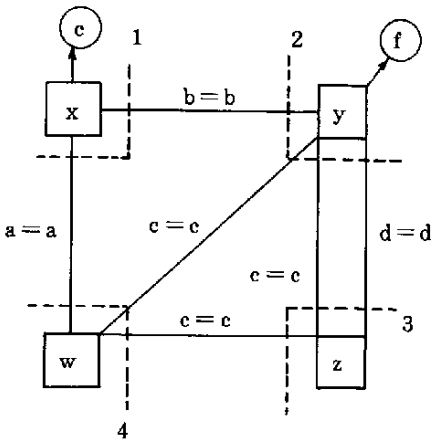
従って，$\Lambda = \bigcup_{i=1}^{n} \Lambda_i$，　$A = \bigcup_{i=1}^{n} A_i$である。

例として，関係X，Y，Z，Wが各々，VS_1，VS_2，VS_3，VS_4に存在して

いるとする。このとき，以下の全体問合せは，複数属性問合せである。又，この問合せグラフを図4.4に示す。

```
var  (x, X)(y, Y)(z, Z)(w, W);
get into Temp(x.e, y.f)   where
  x.a=w.a  and  x.b=y.b  and  y.c=w.c  and
  w.c=z.c  and  z.c=y.c  and  y.d=z.d;
```

ここで，



$\Sigma = \{X, Y, Z, W\}$

$A = \{a, b, c, d\}$

$\Lambda = \{e, f\}$

$A_x = \{a, b\}$　　$\Lambda_x = \{e\}$

$A_y = \{b, c, d\}$　$\Lambda_y = \{f\}$

$A_z = \{c, d\}$

$A_w = \{a, c\}$ である。

図4.4　問合せグラフ

B．DDBS処理方法 ― B^2C通信手順

複数属性問合せのDDBS処理の為のB^2C（Broadcast, Bit-map response, and Code）通信手順を以下に与える。

〔B^2C通信手順〕

(1) Qを受けたVS_0は，Qの参照する関係を有している全VS_iにQを放送する。

(2) Qを受けたVS_iは，初期ローカル問合せ処理（ILQP）を行い，目標集合X_iを求め，X_iの性能情報（組数，各属性値の数，X_iの物理構造）PI_iを放送する。

(3) 各点VS_iは，全点からPIを受信したならば，次の様に初期化を行う。

$\Sigma' \leftarrow \{X_1, \cdots, X_n\}$；　$\Sigma'' \leftarrow \{\phi\}$；　$A \leftarrow$　（全結合属性集合）；

各$X_h \leftarrow \Sigma'$に対して，$A_h \leftarrow$　（X_hの結合属性集合）；

(4) 〔送信VS_sと関係X_sの決定〕

全点VS_iは，Σ'のなかから，最適なX_sを選ぶ。但し，Aがコード化属性$\tilde{A}$を含むならば，$\tilde{A} \in A_s$なるX_sのなかから選ぶ。

(5) 〔X_sの放送〕

VS_sは，$X_s[A_s]$を読出して，一定順序で放送する。

$Y_s \leftarrow X_s[A_s]$; broadcast (Y_s); $BM_{ss} \leftarrow Y_s /\!/ Y_s$;
$X'_s \leftarrow X_s$; $j \leftarrow s$; go to (7);

(6) 〔X_sの受信〕

(i) $A_i \cap A_s \neq \phi$なる関係$X_j \leftarrow (\Sigma' \cup \Sigma'')$を有する全点VSは，$VS_s$からの$Y_s$を受信し，$Z_s$とする。ここで，$A_{sj} = A_s \cap A_j$

$Z_j \leftarrow$ rec ( );

Z_jと，X_jとをA_jについて結合し，$X'_j(A_s, A'_j, \Lambda_j)$を得る。

$X'_j \leftarrow X_j[A_{sj} = A_{sj}]Z_j$;
$BM_{sj} \leftarrow X'_j[A_s] /\!/ Z_j$;〔ビットマップ$BM_{sj}$の生成〕

(ii) if $X_j \in \Sigma'$ then broadcast (BM_{sj});

(7) 〔ビットマップの受信〕

全VS_j(ただし，$X_j \leftarrow (\Sigma' \cup \Sigma'')$)は，全ビットマップを受信する。

while (全ビットマップを，受信していない)

{ $BM \leftarrow$ rec ( ); $BM_{sj} \leftarrow BM_{sj} \wedge BM$; }

全ビットマップを受信し終えたならば，ビットマップによりX_jを制限し，コード化する。

$X''_j \leftarrow X'_j * BM_{ij}$; 〔ビットマップ制限〕
$X_j \leftarrow X''_j{}^{\wedge}(Z_j * BM_{sj})$;〔コード化，$X_j(\tilde{A}, A'_j, \Lambda_j)$〕

$A_h \cap A_s \neq \phi$なる全X_hに対して，

$A_h \leftarrow (A_h - \{A_{ij}\}) \cup \tilde{A}$;
$A \leftarrow (A - \{A_s\}) \cup \tilde{A}$;
$\Sigma' \leftarrow \Sigma' - \{X_s\}$;

(8) 〔不要関係の除去〕

$$\underline{\text{if}}\ (X_j \in \Sigma' \cup X_j = X_s)$$
$$\underline{\text{then}}\ \ \underline{\text{if}}\ \ A_j = \tilde{A}\ (\text{全コード化されている})$$
$$\underline{\text{then}}\ \ \{\Sigma' \leftarrow \Sigma' - \{X_j\};$$
$$\underline{\text{if}}\ \ \Lambda_j \neq \emptyset\ \ \underline{\text{then}}\ \ \Sigma'' \leftarrow \Sigma'' \cup \{X_j\}\};$$

各 VS_j で，$X_j \leftarrow (\Sigma' \cup \Sigma'')$ ならば，VS_j はQの処理を終了する。

(9) 〔目標導出処理〕

$\underline{\text{if}}\ |\Sigma'| \geq 2\ \underline{\text{then}}$ go to (4) 〔$|\Sigma'| = \Sigma'$内の関係数〕$\underline{\text{else}}$〔終了〕;

$\Lambda_h \neq \emptyset$ なる全 VS_h は，X_h ($\tilde{A}$, Λ_h) を VS_0 に送信する。VS_0 は X_h を受信する毎に $G \leftarrow G\,[\tilde{A}=\tilde{A}]\,X_h$ を行い，$G[\Lambda]$ を目標集合とする。□

図4.5には，B^2C 通信手順を，状態遷移図によって表す。

以上の B^2C 通信手順において，Σ' 内の関係を候補関係とし，Σ' を候補関係集合とする。候補関係は，送出関係となり得る関係である。Σ'' 内の関係を，非候補目標関係とし，Σ'' を非候補目標関係集合とする。Σ'' 内の関係は，候補関係ではないが，目標属性を持つものである。又，$\tilde{A}$ をコード化属性とする。$\tilde{A}$ は，最大 $|Y_s[\tilde{A}]|$ を表すだけのビット長があればよく，通常1～2Bあればよい。Y_s を送信系列，X_s を送信関係とする。コード化属性 $\tilde{A}$ を持つ関係をコード化関係とする。又，Σ' と Σ'' の関係をあわせて，受信関係とする。

図4.5には，B^2C 通信手順の(5)，(6)，(7)の操作を図示する。

図4.7には，図4.4の複数属性問合せの B^2C 通信手順によるDDBS処理の例を示してある。

(1) まず，VS_3 から，送信系列Z〔c，d〕が放送され，VS_2 と VS_4 で受信される。VS_2 と VS_4 では各々，Z〔c，d〕と関係YとWとの結合が行われ，ビットマップ放送の後，各々コード化されて，関係Y($\tilde{cd}$, b, f)，W($\tilde{cd}$, a) となる。関係Wは，候補関係集合 Σ' から除かれる。

(2) 次に，VS_2 からY〔$\tilde{cd}$, b〕が，VS_1 と VS_4 に放送される。VS_1 と VS_4 で各々，関係XとWとY〔$\tilde{cd}$, b〕と結合され，X($\tilde{bcd}$, a, e) と W($\tilde{bcd}$, a) が求められる。又，VS_2 には，Y($\tilde{bcd}$, f) が求められる。

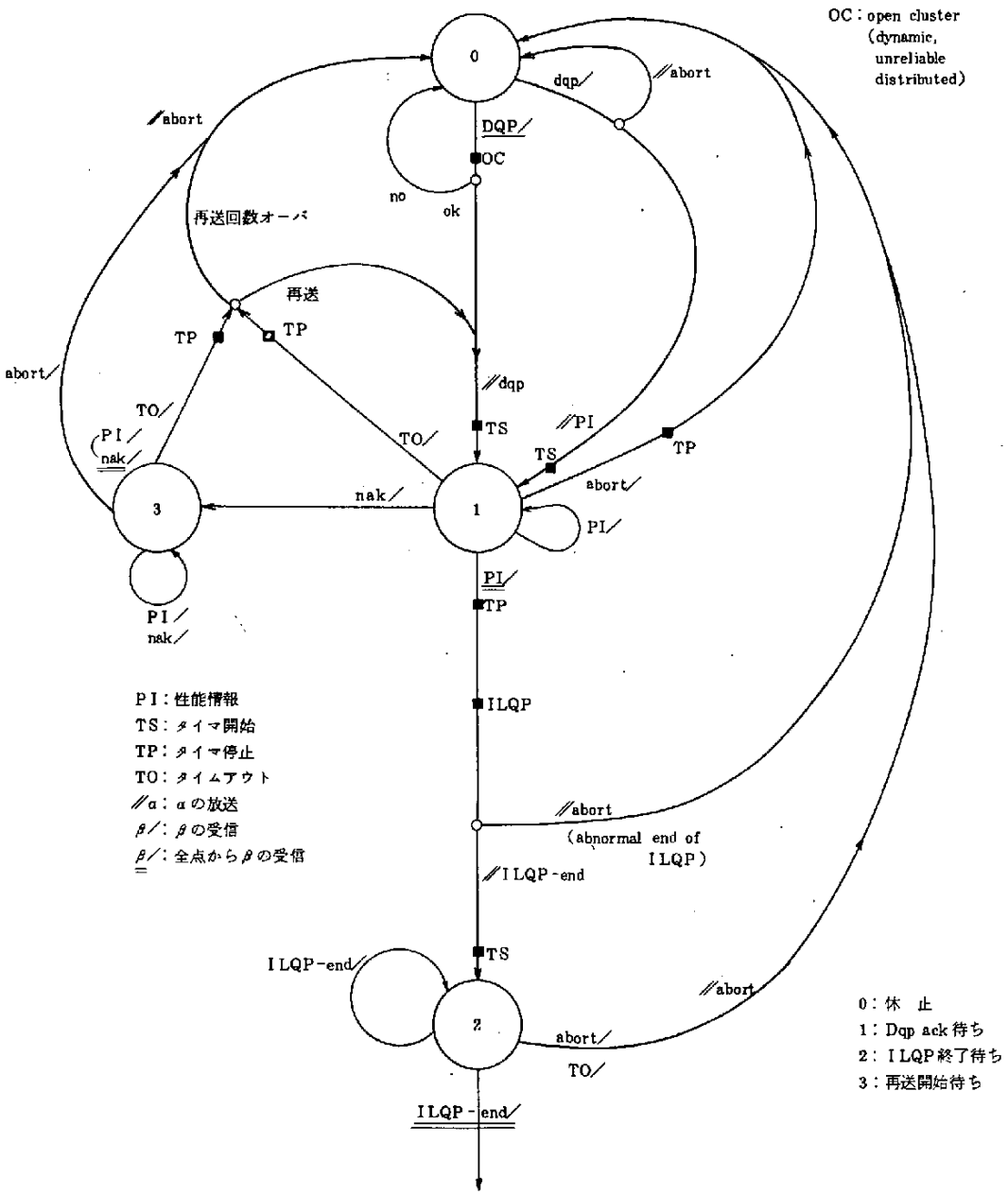


図 4. 5　B²C 通信手順 (a)

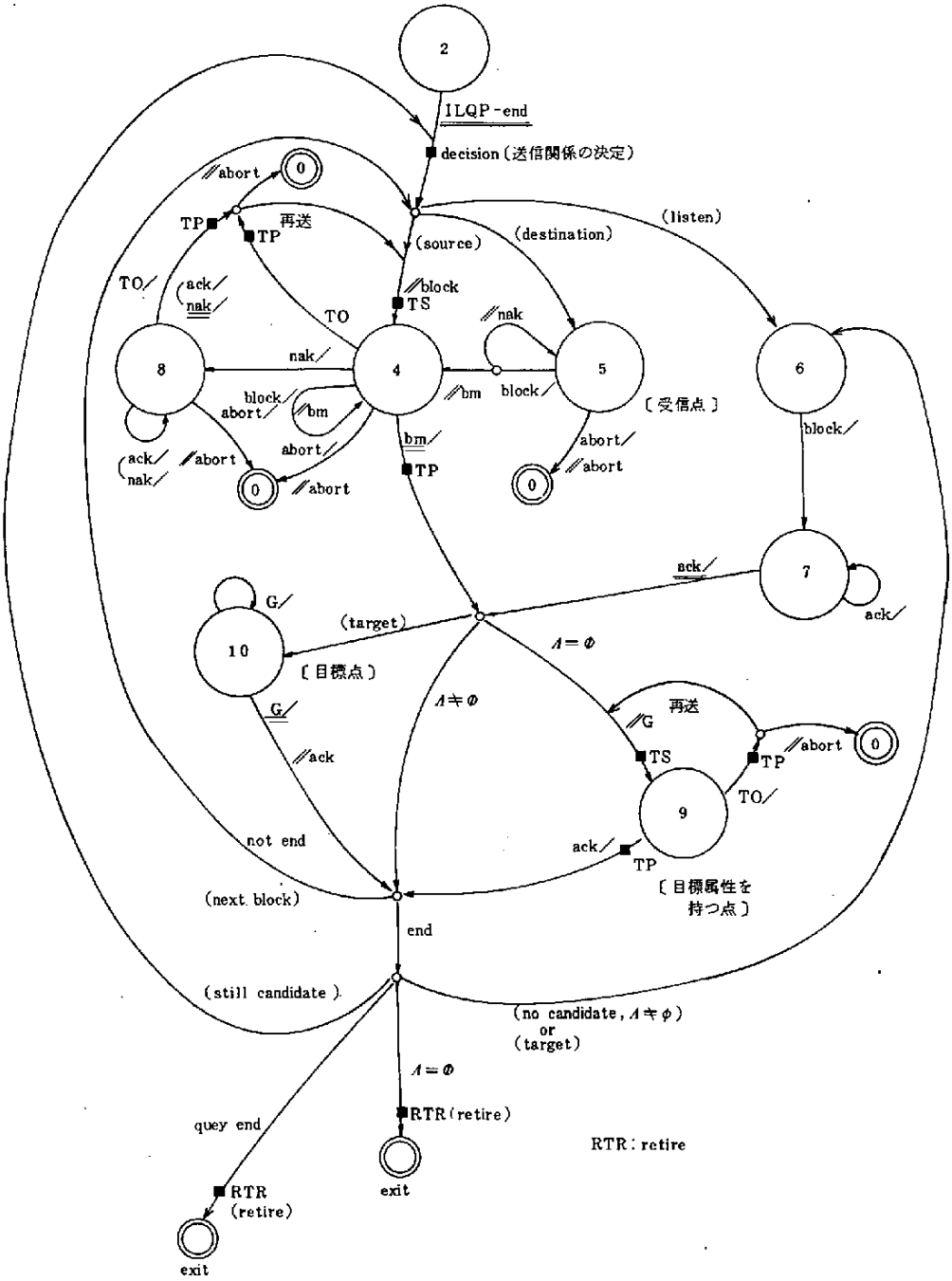


図 4.5　B²C 通信手順 (b)

送信系列

Y_S	a	b	c
(Z_S)	α	1	B
	α	2	A
	β	2	D
	γ	7	B

候補関係

X_2	a	b	d
	α	1	a
	α	1	b
	β	2	b
	γ	3	c
	δ	2	a

X_3	b	e
	1	X
	1	Y
	2	Y
	3	Z
	4	X
	5	Y
	6	X

〔結合〕　$X_2' \leftarrow X_2 \,〔\, ab=ab \,〕 Z_S$;

X_2'	a	b	c	d
	α	1	B	a
	α	1	B	b
	β	2	D	b

$X_3' \leftarrow X_3 \,〔\, b=b \,〕 Z_S$;

X_3'	a	b	c	e
	α	1	B	X
	α	1	B	Y
	α	2	A	Y
	β	2	D	Y

$BM_{12}=\langle 1010 \rangle$

$BM_{13}=\langle 1110 \rangle$

〔ビットマップ放送〕

$BM_{12}= BM_{12} \wedge BM_{13}$
$=\langle 1010 \rangle$

$BM_{13}= BM_{13} \wedge BM_{12}$
$=\langle 1010 \rangle$

〔ビットマップ制限〕

$X_2'' \leftarrow X_2 * BM$

X_2''	a	b	c	d
	α	1	B	a
	α	1	B	b
	β	2	D	b

$X_3'' \leftarrow X_3 * BM$

X_3''	a	b	c	e
	α	1	B	X
	α	1	B	Y
	β	2	D	Y

〔コード化〕

$X_2 \leftarrow X_2'' \wedge (Z_S * BM_{12})$

X_2	$\widetilde{abc}$	d
	0	a
	0	b
	1	b

$X_3 \leftarrow X_3'' \wedge (Z_S * BM_{13})$

X_3	$\widetilde{abc}$	e
	0	X
	0	Y
	1	Y

図 4.6　B^2C 通信手順の例

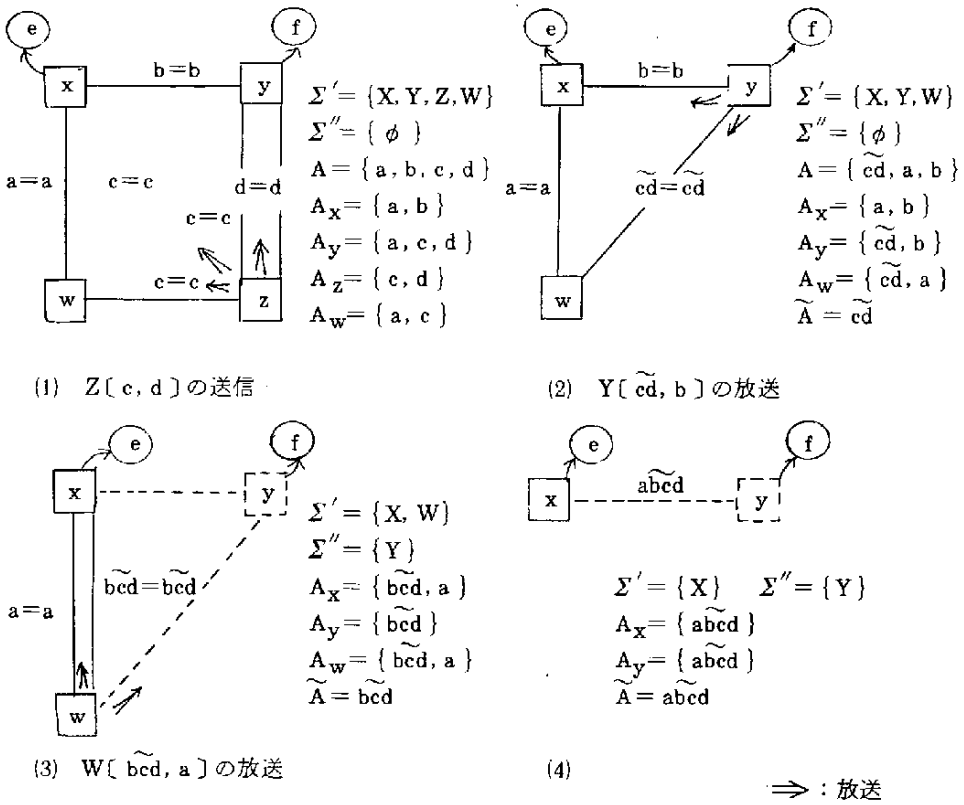


図 4. 7　B^2C 通信手順の例

Yは，Σ' からは除かれるが，$A_y = \{f\} \neq \emptyset$ なので，非候補目標集合Σ''に加えられる。

(3)　VS_3 から，$W〔\tilde{bcd}, a〕$が放送され，VS_1 と VS_2 で各々，関係XとYと結合され，$X(\tilde{abcd}, e)$ と $Y(\tilde{abcd}, f)$ が得られる。

(4)　VS_1 と VS_2 は，VS_0 にXとYを送信する。VS_0 では，$T \leftarrow X〔\tilde{abcd} = \tilde{abcd}〕Y$ を行い，$G = T〔e, f〕$ を目標集合とする。

C．B^2C 通信手順の全通信量と通信時間

B^2C 通信手順での全通信量Tと通信時間Rとを考える。B^2C 通信手順で，関係 X_s の Y_s ($=X_s〔A_s〕$) が放送されたとする。このとき，$\|Y_s\|$ と $|Y_s|$ を各々，Y_s の情報量と組数とする。又，n_s を，$A_s \cap A_j \neq \emptyset$ なる候補関係 X_j の数とする。Y_s の放送と，ビットマップの放送の為の全通信量 T_s と通信時間 R_s とは次式で与えられる。

$$T_s = \beta \cdot \{ \| X_s[A_s] \| + (n_s - 1) | X_s[A_s] | \} [bit] \cdots (4.11)$$

上式で第一項はY_sの放送と，第二項はビットマップの放送の為の全通信量である。（2.5）式で$\alpha = 0$とする。従って，このときの通信時間R_sは，次式で与えられる。

$$R_s = T_s / w \ 〔sec〕 \cdots (4.12)$$

wは，通信網の伝送速度〔bit/sec〕である。全体問合せQのDDBS処理の為のTとRは，B^2C通信手順に従って，各々T_sとR_sを加算したものとなる。

4.5 入出力時間量を考慮したDDBS処理方法

B^2R通信手順では，バッファを無限長とし，入出力時間を零としたが，最近の光ファイバ等による通信速度の向上によって，各点での処理時間は無視出来なくなっている。従って，全通信量とともに，各点での全処理時間量の最少化を図らねばならない。本章では，この為のDDBS処理方法を論じる。

4.5.1 仮想DBSの物理構成

仮想DBS VS_iは物理的には，実際のLDBS$_i$（DBS$_i$）と，DDBS処理の作業領域WS_iから構成される〔図2.4〕。WS_iには，(i)DBS$_i$から導出された目標関係と，(ii)DDBS処理の中間結果及び目標関係とが格納される。(i)は，（4.1）式に対するローカル問合せ$Q_i = (\lambda_i, \Psi_i)$の目標関係$X_i$をDBS$_i$から$WS_i$に導出することに対応している。最近の光ファイバ等による高速通信に対して，各点で有効な入出力を行うには，WS_iの物理レベルの入出力を考える必要がある。

A．作業領域WSの物理モデル

WSは，一定長(g)〔bit〕のページ集合Pと入出力ヘッドHから成る。P内のページP_iは，$P_1 < \cdots < P_q$と順序付けられている（qは全ページ数）。VPは主記憶VMを持ち，VMとHのいるページ間で，ページ単位のデータ転送を行う。WSからVMへのページ転送を読出し，逆を書込み，両者を併せて入出力とする。主記憶VMには，一ページ分の入出力バッファがあるとする。ページP_iの入出力は，次の二つの操作から成る。

(i) ヘッドHをページP_iの位置に移動し，

(ii) P_iとVM間でのデータ転送を行う。

ヘッドHがページP_iの位置にいるとき，ページP_j（$j \neq i$）を入出力するのに必要な時間を$I(i,j)$とすると，$I(i,j)=f(i,j)+t$となる。tは(ii)の転送時間である。$f(i,j)$は，(i)のヘッドHの移動時間で，次式で与えられる。

$$f(i,j) \triangleq \begin{cases} d & \underline{\text{if}} \quad j=i+1 \\ k+d & \underline{\text{otherwise}} \end{cases} \qquad \cdots\cdots\cdots\cdots (4.13)$$

kをシーク時間，dを待ち時間とする。$j=i+1$のとき連続入出力，$j \neq i+1$のときランダム入出力という。前者で，$r=d+t$を連続入出力時間，後者で$s=k+d+t$をランダム入出力時間とする。

B．関係のWSへの格納手続き

Xを関係，aをXの属性とする。Xを組の順序集合$\{t_1,\cdots,t_c\}(t_i<t_j(i<j))$とする。順序関係としては，aの値順と，任意の順とがある。Xの各組t_hのWS内のページへの割り当てについて以下で考察する。

各ページP_iは，ℓ個の連続したスロットから成り，各スロットには組を格納出来る。関係X内の組tの格納位置は，ページ番号$\pi(t)$とスロット番号$\gamma(t)$との対で与えられ，$\varsigma(t)=(\pi(t),\gamma(t))$を組tの組識別子とする。XのWSへの格納方法としては，以下の連続，リンク，ランダムの三種がある。

〔格納方法〕

(i) 連続格納　$\pi(t_{i+1})=\pi(t_i) \wedge r(t_{i+1})=r(t_i)+1 \quad \underline{\text{if}} \ r(t_i)<\ell$

$\pi(t_{i+1})=\pi(t_i)+1 \wedge r(t_{i+1})=1 \quad \underline{\text{if}} \ r(t_i)=\ell$

(ii) リンク格納　$\pi(t_{i+1})=\pi(t_i) \wedge r(t_{i+1})=r(t_i)+1 \quad \underline{\text{if}} \ r(t_i)<\ell$

$\pi(t_{i+1}) \neq \pi(t_i) \wedge r(t_{i+1})=1 \quad \underline{\text{if}} \ r(t_i)=\ell$

（$\pi(t_{i+1})$はランダムに定まる。但し，ページの重複はないものとする。）

(iii) ランダム格納　$\pi(t_{i+1})$と$r(t_{i+1})$は以下の二つの方法によりランダムに定まる。

(iii－1) ハッシュ格納　$\pi(t_{i+1})$は，t_{i+1}のある属性bの値のハッシュ関数値で定まり，$\gamma(t_{i+1})$はランダムに定まる（但し，重複はないものとする）。bをハッシュ属性とする。

(iii－2) ランダム格納　$\pi(t_{i+1})$，$\lambda(t_{i+1})$は全く任意に定まる（但し，重複はないものとする）。

関係Xが属性aで順序付けられていて，(i)又は(ii)の方法で格納されているとき，aを整列属性とする。この他に，関係Xのある属性cについて組を順序付けたいときには，WS内の組に次の順序の組へのポインタを与えて表せる。このcを順序属性とする。

関係Xのある属性aについて，組tの属性aの値vと識別子$\delta(t)$との対応表I_{xa}を，関係Xの属性aの索引表とし，aを索引属性とする。I_{xa}は，属性aの値で順序付けられているとする。I_{xa}は，(i)又は(ii)の方法で格納される。*) h_{xa}を，任意値vに対して，$a(t)=v$なる組$(v, \delta(t)) \in I_{xa}$を格納したページを読出すのに必要な平均ページ数とする。表4.1に，作業領域WSと関係Xに関するパラメータをまとめる。

C．WSの基本操作演算

VPは，次の二種の演算を行う。

(1) 〔VM内演算〕VM内のデータに対する演算（計算，比較，移動）。

(2) 〔入出力演算〕VMとWS間のページ単位の入出力演算。

Xを関係，aをXの属性とすると，(2)として以下の演算がある。

〔入出力演算〕

(i) D(X, δ)；関係Xから組識別子が$\delta=(\pi, \gamma)$なる組を読出す。

(ii) S(X)；関係Xから入出力された次の組識別子を持つ組を読出す。

(iii) SA(X, a)；属性aが整列又は順序属性のとき，Xから入出力された次の順序のaの値を持つ組を読出す。

(iv) SI(X, a)；aが索引属性のとき，索引表I_{xa}から入出力された次の格納順の組を読出す。

---

*) 関係X内の各組で属性aの値が異っているとき，I_{xa}を主索引表，そうでないとき副索引表，更に，aがXの整列属性のとき群索引表と呼ぶ。

表4.1　WSと関係Xのパラメータ

	パラメータ	説　　明
WS	g	ページの大きさ　〔bit〕
	k	シーク時間　〔msec〕
	d	待ち時間　〔msec〕
	t	一ページの転送時間　〔msec／page〕
	s	連続読出し時間　〔msec／page〕　(=d+t)
	r	ランダム読出し時間　〔msec／page〕 (=s+d+t)
関係X	c_x	X内の組の数（=｜X｜）
	p_x	Xを格納する為の全ページ数
Xの属性a	σ_{xa}	結合選択度：X〔a〕｜／｜dom(a)｜
	c_{xa}	X内のa のことなった値数｜X〔a〕｜
	p_{xa}	I_{xa}の格納する為の全ページ数
	h_{xa}	I_{xa}を検索するのに必要なページ数

(v)　I(X, a, v)；aが索引属性のとき，aの値がvなる組の識別子を読出す。

(vi)　H(X, a, v)；aがハッシュ属性のとき，aの値がvなる組を読出す。

(vii)　ST(X, t)；関係Xに組tを加える。

(viii)　DL(X, δ)；Xから，組識別子がδなる組を除く。□

主記憶内演算時間は，入出力演算時間に対して無視し得るので，後者のみを考えることにする。

D．全入出力時間量と全通信時間量の比率

入出力時間量と通信時間量の比率について考える。表4.2には，超小型機（PC−9800），小型機（VAX11），大型機（IBM308X）の代表的な固定ディスク装置の性能を示す。

表4.2　固定ディスク装置の性能

機種（容量）	ページ大きさ g (Byte)	平均シーク k 時間 (ms)	平均回転待ち時間 d (ms)	転送時間 t (ms／ページ)	ランダム読出し時間 r (msec／頁)	連続読出し時間 s (msec／頁)
超小型機 (PC-9800) PC-9801 (5MB)	256	117.0	8.3	0.256 (256B／1MB)	125.6	8.6
小型機 (VAX -11) RP-07 (516MB)	512	31.3	8.3	0.233 (512B／2.2MB)	39.8	8.5
大型機 IBM 3380 (1.26GB)	4096	16.0	8.3	1.365 (4096B ／3MB)	25.7	9.7

表4.3　入出力速度

入出力速度〔kbps〕

機種＼格納方法	ランダム	連続
超小型機	1.6	22.0
小型機	10.3	46.0
大型機	128.0	290.0

40KBのデータの入出力速度（＝転送量／入出力時間）〔bit／sec〕を表4.3に示す。表から，超小型機がRS232C（≦19.2kbps）と，小型機が48 kbpsの網と同程度の入出力速度を持つことがわかる。以上より，高速なLANに，低速度入出力装置（例，フロッピーディスク）をもった超小型機が結合されてくると，通信時間量以上に，各VS_hでの入出力時間量の減少が必要とされることがわかる。この為，DDBS処理では，通信時間量とともに，全入出力時間量の減少が必要となり，次項以降にこの問題を論じる。

4.5.2 B^2R通信手順における仮想ＤＢＳでの処理方法

各VS_iのバッファB_iはb〔bit〕なる一定長であり，その大きさをブロックとする。B^2R通信手順〔4.3節〕で，VS_iを送信点，VS_jを受信点とすると，以下の三つの処理から成る。

〔B^2R通信手順での処理〕

(i) 〔送信処理〕送信VS_iは，関係X_iの属性aの値をバッファB_iに順に読出し，これを放送する。

(ii) 〔受信処理〕各VS_j($j \neq i$)は，バッファB_jに受信した属性aの値と，X_jに対して，条件式$\Psi_{ij}(x_i.a = x_j.a)$を調べ，BM_{ij}(ビットマップ)を求め各点に放送する。

(iii) 〔目標処理〕各VS_h($\Lambda_h \neq \emptyset$)($h = 1, \cdots, a$)は，全BM_{ij}を受信したならば，問合せの条件式Ψを満足する関係X_hの組集合G_hを読出し，目標VS_0は，これらのG_hを結合し，目標関係Gを導出する。□

以下に，各々の三つの処理手順を与える。

A. 送信手順

送信点iでは，(i)関係X_iの属性a値を順に読出し，(ii)バッファB_i内の値を整列し，冗長値を除きながら，(iii)B_iが満杯になった時，B_i内の値を整列順にブロックとして放送する。

X_iの各属性aの値は，WS基本操作演算によって，ある順序でB_iに読出される。ここで，VS_iから放送された全a値の系列S_iを送信系列とする。S_iは，整列された部分系列に分割出来て，各部分系列を連とする。連長は，(i)～(iii)より，ブロックの整数倍となる。S_iに含まれるa値の総数をN_i，全ブロック数をL_i，連数をn_iとする。連毎の属性aの値とブロックの平均数を各々，$a_i = N_i / n_i$と$\ell_i = L_i / n_i$とする。S_iが一つの連のとき，整列系列とし，S_iが冗長値を含まないとき非冗長系列とする。整列系列は，非冗長系列である。

関係X_iから属性aの値の読出し方法E_iは，表4.4の9通りある。各々の入出力ページ数，入出力時間P_i'，S_iの連数n_iは関係X_iの格納構造に対応して定まり，これらも表4.4に示す。例えば，X_iが索引表I_{ia}を持つときには，I_{ia}をWS演算ＳＩで連続読出しすれば，連数1の整列系列が得られる。このとき，読出されるページ数はP_{ia}であり，入出力時間は，

表 4.4　送信手順と入出力時間

読出し方法 E_i	索引表 I_{ia}		格納方法			入出力ページ数	入出力時間 $P_{i'}$ (msec)	送信系列型	連数 n_i	説明
	索引属性	格納型	整列属性	順序属性	格納型					
(1)SI (X_i, a)	a	C				p_{ia} (p_{ia}:aの昭字)	$k_{ia} + p_{ia} \cdot s_i$	S	1	I_{ia}の検索
(2)SI (X_i, a)	a	L				p_{ia}	$p_{ia} \cdot r_i$	S	1	
(3)SA (X_i, a)			a		C	p_i (p_i:aの昭字)	$k_i + p_i \cdot s_i$	S	1	X_iをaの値順に検索
(4)SA (X_i, a)			a		L	p_i	$p_i \cdot r_i$	S	1	
(5)SA (X_i, a)			≠a	a		c_i	$c_i \cdot r_i$	S	1	
(6)S (X_i)			≠a		C	p_i	$k_i + p_i \cdot p_i$	N	r^*	X_iの連続読出し
(7)S (X_i)			≠a		L	p_i	$p_i \cdot r_i$	N	r	
(8)S (X_i) + sort (S)			≠a		C	$p_i + p_{si}$	$k_i + p_i \cdot s_i + p_{si} \cdot r_i$	S	1	X_iの連続読出しと整列
(9)S (X_i) + sort (S)			≠a		L	$p_i + p_{si}$	$p_i \cdot r_i + p_{si} \cdot r_i$	S	1	

C＝連続格納
R＝ランダム格納
L＝リンク格納

S＝整列
N＝非整列

p_{si}＝ 送信系列 S_i の整列の為の入出力ページ数

r^*＝連数＝ $|X_i| \cdot \|a\| / b$　(b はバッファの大きさ)

I_{ia} が連続格納されているとき $k_i + p_{ia} \cdot s_i$〔表 4.4(1)〕，リンク格納のとき $p_{ia} \cdot r_i$〔表 4.4(2)〕となる。

B．受信手順

各受信点 j は，以下の手順を行う。

(i)　VS_iからのブロックを B_j に受信。

(ii)　ビットマップ BM_{ij} の構成。

(iii)　BM_{ij} の放送。

送信されてきた連毎に，関係X_j内の全a値と照合される。このときのX_jから属性aの値の読出し方法E_jは，表4.5に示す10通りがある。各々の連毎の読出しページ数，入出力時間P'_jは，関係X_jの格納構造に対応して定まり，これらも表4.5に示す。例えば，X_jが属性aの値順に連続格納されているとき，各連に対して，X_jをWS操作演算のS又はSAによって一回連続読出しして，属性aの値の照合を行える。このときの入出力ページ数はp_iで，入出力時間は$k_i + p_i \cdot s_i$となる〔表4.5(6)〕。送信されたブロック毎に以上の処理がなされた後，バッファB_jには条件式Ψ_{ij}を満たす属性aの値vと，この値を持つ関係X_j内の組の識別子δの対（v，δ）を格納するとする。VS_jでの受信手順の全入出力時間は，$n_i \cdot P'_j$となる（n_iは送信系列S_iの連数）。

C．目標処理手順

目標処理手順は，以下から成る。

(i) 〔目標送信手順〕各VS_h（$\Lambda_h \neq \emptyset$）（$h=1, \cdots, n$）での目標組集合G_hをX_hから求め，VS_0に送信し，

(ii) 〔目標導出手順〕VS_0でG_hを結合し，目標集合Gを求める。

a．目標送信手順

各VS_h（$\Lambda_h \neq \emptyset$）は，ビットマップBM_{ih}のビットオンに対応した属性aの値vとバッファB_h内の組（v，δ）より，属性a値がvの組の識別子δを求め，D（X_h，δ）によってX_hを読出し，G_hを求める。この為の，X_hの読出し方法F_hを表4.6に示す。各方法の連毎の入出力ページ数と入出力時間P''_hは，送受信手順E_hとX_hの格納構造によって定まり，これも表4.6に示す。

b．目標導出手順

目標点0のVS_0は，VS_iからのブロックの放送毎に，各VS_h（$\Lambda_h \neq \emptyset$）からG_hを受信し，これらを属性aで結合して目標関係GをWS$_0$に格納する。全G_hを主記憶VM_0に格納出来るとすると，VS_0での入出力時間P_0は，目標関係Gの格納時間であり，次式で与えられる（但し，仮定4.1が成り立つとする）。

$$P_0 = \begin{cases} k_0 + s_0 \cdot \| G \| / g_0 & \text{（連続格納）} \\ \| G \| \cdot r_0 / g_0 & \text{（リンク格納）} \end{cases} \quad \cdots\cdots\cdots (4.14)$$

$$\| G \| = \| \Lambda \| \cdot \prod_{h=1}^{n} \sigma_h \cdot x_h \quad \cdots\cdots (4.15)$$

$$x_h = \begin{cases} | X_h | & \underline{\text{if}} \quad \Lambda_h \neq \varnothing \\ 1 & \underline{\text{if}} \quad \Lambda_h = \varnothing \end{cases} \quad \cdots\cdots (4.16)$$

D．最適なＤＤＢＳ処理手順の決定方法

B^2R通信手順での全処理時間量Pを，各VS_hの入出力時間P_hの和と定める（h＝0，1，…，n）。

$$P = P_0 + P_1 + \cdots\cdots + P_n \quad 〔\text{sec}〕 \quad \cdots\cdots (4.17)$$

P_h（h≠0）は，次式で与えられる。

$$P_i = P_i' + \delta_i \cdot n_i \cdot P_i'' 〔\text{sec}〕(VS_i \text{は送信点}) \quad \cdots\cdots (4.18)$$

$$P_h = n_i \cdot (P_h' + \delta_h \cdot P_h'') 〔\text{sec}〕(h \neq i)$$

P_h'とP_h''は表4.4～4.6より，δ_hは（4.8）式で与えられ，n_iは送信系列S_iの連数である。P_0は（4.14）式で与えられる。一方，全通信時間Tは，送信点iのX_iが与えられたとき，（4.10）式で与えられる。

$$T = R = T/w \quad 〔\text{sec}〕 \quad \cdots\cdots (4.19)$$

このとき，全通信処理時間量Tを，次式で与える。

$$T = P + T \quad 〔\text{sec}〕 \quad \cdots\cdots (4.18)$$

表 4.5　受信手順と入出力時間

読出し方法 E_j	索引表 I_{ja}		格納方法				入出力頁数 /連	入出力時間 P_{jr} /連	説明
	索引属性	格納型	整列属性	ハッシュ属性	順序属性	格納型			
(1)I (X_j, a, v)	a						$a_i \cdot h_{ja}$	$a_i \cdot h_{ja} \cdot r_j$	I_{ja}の直接読出し
(2)H (X_j, a, v)				a		R	a_i	$a_i \cdot r_j$	ハッシング
(3)SI (X_j, a)	a	C					p_{ja}	$k_j + p_{ja} \cdot s_j$	I_{ja}の連続読出し
(4)SI (X_j, a)	a	L					p_{ja}	$p_{ja} \cdot r_j$	
(5)I $(X_j, a, v$ +SI (X_j, a)	a	L					$h_{ja} + \frac{a_i - 1}{a_i + 1} \cdot p_j$	$(k_j + \frac{a_i - 1}{a_i + 1} \cdot p_j) \cdot r_j$	(1) と (2) I_{ja}は B* 木
(6)SA (X_j, a) S (X_j, a)			a			C	p_j	$k_j + p_j \cdot s_j$	X_jのaの値の連続読出し
(7)SA (X_j, a) S (X_j, a)			a			L	p_j	$p_j \cdot r_j$	
(8)SA (X_j, a)			≠a		a		c_j	$c_i \cdot r_j$	
(9)S (X_j, a)			≠a			C	p_j	$b_i \cdot (k_j + p_j \cdot s_j)$	X_jの格納順の連続読出し
(10)S (X_j, a)			≠a			L	p_j	$b_i \cdot p_j \cdot r_j$	

表 4.6　目標送信手順と入出力時間

読出し方法 F_h	読出し方法 E_h	格納構造 索引属性	格納構造 整列属性	格納構造 ハッシュ属性	格納構造 順序属性	格納構造 格納型	入出力ページ数／連	入出力時間 P_h・／連
(1) D (X_h, d)	SI (X_h, a)	a	a			C	κ	$k_h + \kappa \cdot s_h$
	SI (X_h, a)	a	a			L	κ	$\kappa \cdot r_h$
	SI (X_h, a)	a	≠a				$\kappa_i \cdot a_i$	$\kappa_i \cdot a_i \cdot r_h$
	SA (X_h, a) S (X_h)		a			C	κ	$k_h + \kappa \cdot r_h$
	SA (X_h, a) S (X_h)		a			L	κ	$\kappa \cdot r_h$
							$\kappa_i \cdot a_i$	$\kappa_i \cdot a_i \cdot r_h$
(2) SA (X_h, a) S (X_h, a)			a			C	p_h	$k_h + p_h \cdot s_h$
(3) SA (X_h, a) S (X_h, a)			a			L	p_h	$p_h \cdot r_h$
(4) SA (X_h, a)			≠a		a		c_h	$c_h \cdot r_h$
(5) S (X_h)			≠a			C	$b_i \cdot p_h$	$k_h + b_i \cdot p_h \cdot s_h$
(6) S (X_h)			≠a			L	$b_i \cdot p_h$	$b_i \cdot p_h \cdot r_h$
(7) H (X_h, a, v)				a		R	$\kappa_i \cdot a_i$	$\kappa_i \cdot a_i \cdot r_h$

$$\kappa_i = \sigma_{1a} \cdot \cdots \cdot \sigma_{i-1a} \cdot \sigma_{i+1a} \cdot \cdots \cdot \sigma_{na}$$

$$\kappa = \begin{cases} \kappa_i \cdot a_i & \text{if } \kappa_i \cdot a_i < p_h \\ p_h & \text{if } \kappa_i \cdot a_i \geqq p_h \end{cases}$$

次に，通信処理時間Rを求める。B^2R通信手順で，

並行して行われる処理は，受信処理と目標送信処理である〔図 4.7〕。

従って，通信処理時間量Rは次式で与えられる。

$$R = T + P_o + P'_i + [\max(\{P'_1, \cdots, P'_{i-1}, P'_{i+1}, \cdots, P'_n\}) + \max(\{\delta_1 \cdot P''_1, \cdots, \delta_n \cdot P''_n\})] \cdot n_i \quad \cdots\cdots\cdots\cdots (4.19)$$

このとき，Rが一定値ε以下で，Tを最少化することが望ましい。この目標を満足する様に，(i)送信点 i，(ii)各点 h での送受信手順E_hと目標送信手順F_h（h＝1, ⋯, n）が決定される必要がある。この決定

手順を以下に与える。

〔決定手順〕

各VS_hの関係X_hの格納構造が与えられたもとで，表4.4，4.5，4.6に示す可能な全手順の組合せを調べ，$R < \varepsilon$なる手順の中で，Tを最少となる手順（トランザクション）を求める。□

RQL問合せQを受けたVS_0は，以上の決定手順でトランザクションSを求め，$VS_1, \cdots, VS_n$に放送する。各VS_hは，Sに従って基本操作演算を実行する。

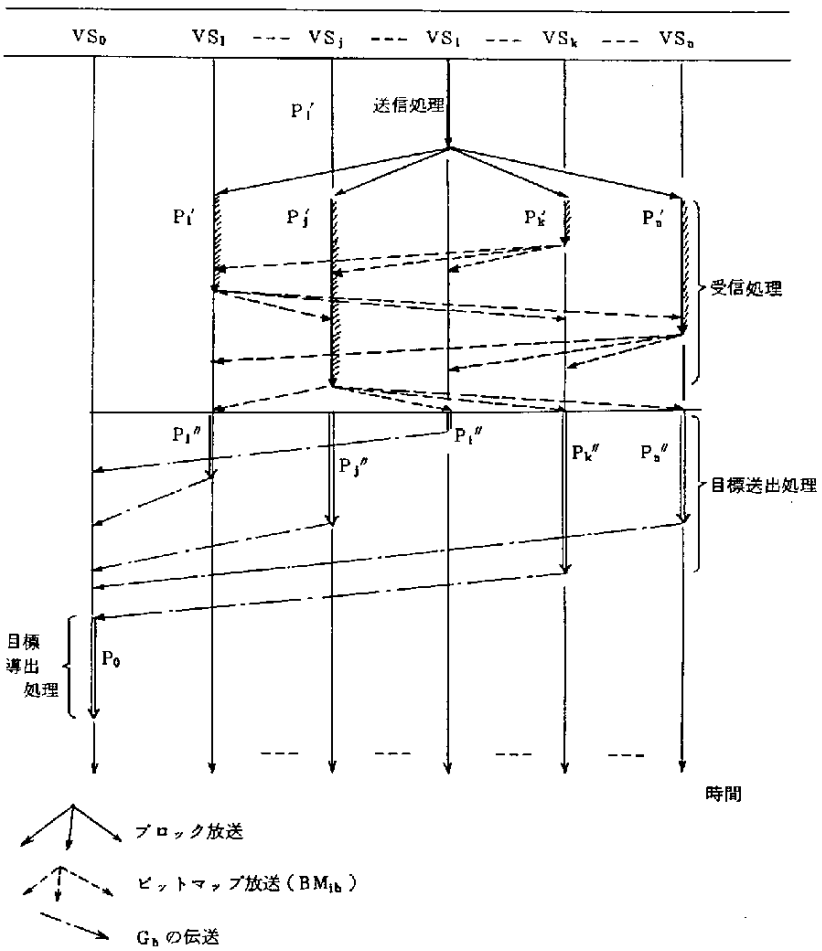


図 4.7　B^2Rの処理時間

4.5.3　B^2C通信手順における仮想ＤＢＳでの処理方法

複数属性問合せに対するB^2C通信手順における各仮想ＤＢＳの処理方法について考える。B^2C通信手順では，(i)候補関係X_sの全結合属性集合A_sの射影$X_s[A_s](=Y_s)$が一定順序で，ブロック単位に放送され，(ii)受信関係X_jと結合され，コード化され，X_sが候補関係でなくなる。候補関係が一つになるまで，(i)と(ii)が繰り返されることになる。

A．コード化関係の物理構造

コード化関係の物理構造について考える。まず，関係$X_i(\Omega_i)$を，B^2C通信手順の初期ローカル問合せ処理（ＩＬＱＰ）後に，各ＶＳに導出された関係とする。ここで，Ω_iは，X_iの属性集合とする。B^2C通信手順で，関係X_iから得られたコード化関係を$\widetilde{X}_i(\widetilde{A}, A_i', \Lambda_i)$とする。$\widetilde{A}$はコード化属性，$A_i$を$\widetilde{X}_i$の結合属性集合とすると，$A_i'=A_i-\widetilde{A}$とする。$\Lambda_i$を関係$X_i$の目標属性集合とする。関係$X_i$は，4.5.1項で述べた物理構造を有しているとする。$\widetilde{X}_i$は図4.8に示す様に，表$\widetilde{M}_i(\widetilde{A}, D)$と，$X_i(\Omega_i)$によって実現される。即ち，以下の様である。

(i)　$\widetilde{X}_i[\widetilde{A}]=\widetilde{M}_i[\widetilde{A}]$

(ii)　$\forall \widetilde{t}\in\widetilde{X}_i$

$\exists \widetilde{m}\in\widetilde{M}_i \quad \widetilde{t}[\widetilde{A}]=\widetilde{m}[\widetilde{A}] \wedge$

$\exists t\in X_i \quad \widetilde{m}[D]=\delta(t) \wedge$

$\widetilde{t}[A_i', \Lambda_i]=t[A_i', \Lambda_i]$

ここで，$\delta(t)$は，関係X_iの組ｔの識別子とする。$\widetilde{M}_i$は，コード化属性$\widetilde{A}$についての索引表である。$\widetilde{A}$は１～２Ｂ，Ｄは組識別子なので４Ｂであり，10000組に対しても６０ＫＢで$\widetilde{M}_i$を構成出来るので，$\widetilde{M}_i$を主記憶内に保てる。

B．コード化結合処理方法

送信系列$Y_s(A_s)$と，受信関係$\widetilde{X}_i(\widetilde{A}, A_i', \Lambda_i)$とについて，$B^2C$通信手順での次の処理手順の実現方法を示す。

$X_i' \leftarrow X_i[A_{si}=A_{si}]Y_s$;

$BM_{ij} \leftarrow X_i'[A_s] /\!\!/ Y_s$;

〔BM_{ij} の放送と，ビットマップ応答の受信と積〕

$X_i'' \leftarrow X_i' * BM_{ij}$ ；〔ビットマップ制限〕

$X_i \leftarrow X_i'' \vee (Y_s * BM_{ij})$ ；〔コード化〕

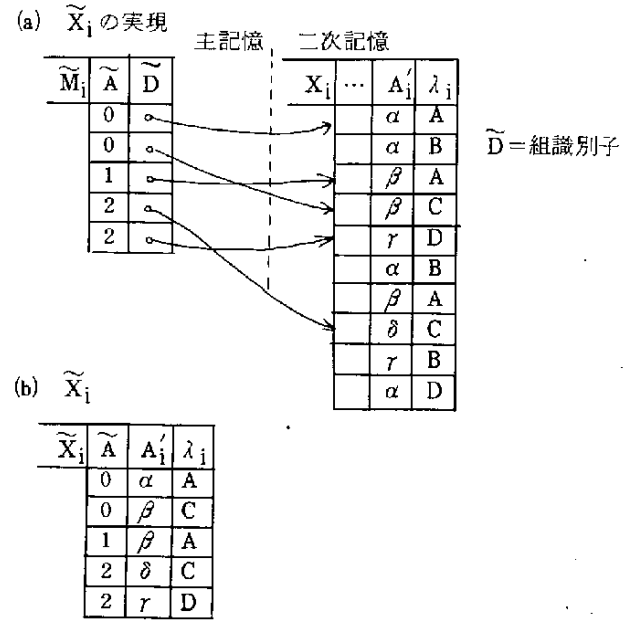


図 4.8　コード化関係 $\widetilde{X}_i$ の実現

以上を，コード化結合処理とし，この実現方法を以下に与える。まず，$\widetilde{M}_i$ は，$\widetilde{A}$，$\widetilde{D}$，n の三つの属性から成るとする。

〔コード化結合処理の実現方法〕

(1)　〔初期化〕

$bn \leftarrow 0$ ；

(2)　〔ブロック単位の Y_s の受信〕

Y_s を一定順序で一ブロックづつ受信する。これを主記憶内に $B_i(\widetilde{A}, A_s')$ として記憶する。各ブロックについて，以下の操作を行う。

(3)　〔結合処理〕

$B_s[\widetilde{A}=\widetilde{A}]\widetilde{M}_i$ を行う。ここで，各組 $b \in B_s$ に対して，# b を b の B_s 内の順番とする。

各 $b \in B_s$ に対して，$b[\widetilde{A}] = m[\widetilde{A}]$ なる各 $m \in M_i$ に対して，

$d \leftarrow m[\widetilde{D}]$；
$d = \delta(t)$ なる $t \in X_i$ を読み出し
if $t[A_{si}] = m[A_{si}]$
then $\{ m[n] \leftarrow -\#b$；
$<BM_{ij}>_{\#b} \leftarrow 1$；〔bit-on〕$\}$；

(4) 〔ビットマップの放送と積〕

BM_{ij} を放送し，他のビットマップの積を BM_{ij} とする。

(5) 〔ビットマップ制限とコード化〕

$\widetilde{M}_i$ をサーチして，$h \leftarrow -1$；
$\forall m \in \widetilde{M}_i$ ($m[n] < 0$) に対して，
$\{ k \leftarrow -m[n]$；
if $<BM_{ij}>_k = 1$ then $m[n] \leftarrow k + bn$；
$h \leftarrow$ if $h > k + bn$, then $k + bn \}$；$bn \leftarrow h$；go to (2)；

(6) 〔$\widetilde{M}_i$ の再構成〕

$\widetilde{M}_i$ から，$m[n] = 0$ なる全組 $m \in \widetilde{M}_i$ を除く。

図 4.9 に，以上のコード化結合の例を示す。

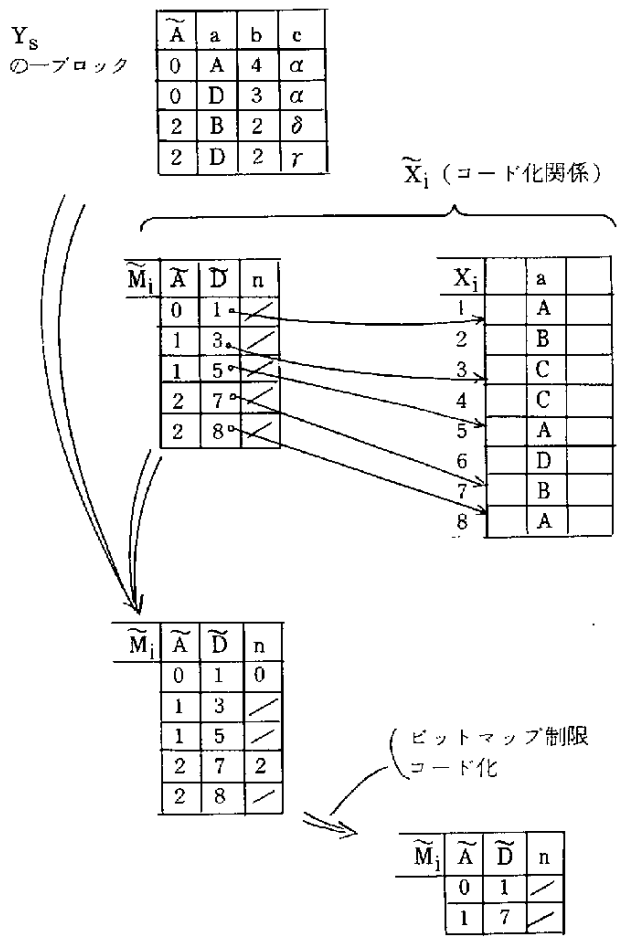


図 4.9　コード化結合の例

C．決定手順

B^2C通信手順で、次の送信関係を決定する問題がある。通信網が放送網であることから、一つの送信関係についてコード化結合が終了した時点で、候補関係X_iを持つ各仮想VS_iは、自分のコード化関係X_iの情報量$\|X_i\|$〔bit〕についての情報を放送する。これによって、各VS_iは、他のVS_j内の候補関係X_iの情報量$\|X_j\|$を知ることができる。

コード化結合処理手順で見てきた様に、どの関係X_iに対しても、索引$\widetilde{M}_i$の組数分だけのページ数が入出力される。コード化されていない関係に対しては、送信されてきた系列に対して、表4.5に示した様な手順と入出力時間が必要である。決定手順は、各コード化結合処理が終了する毎に、全候補関係を持つVS_iによって、同時に同一決定がなされる。

〔決定手順〕

候補関係を持つ各仮想DBSは、各候補関係X_iを送信関係とした時、可能な送信手順に対して、最少時間量の受信手順を求め、全通信処理時間量と全通信処理時間を求める。この中で、全通信処理時間が一定値以下で、全通信処理時間が最少のものを求める。□

4.6 更新演算のDDBS処理方法

RQL更新演算は$Q=(op, \Lambda, \Psi)$と表せて、基本更新演算$op \in \{del, app, rep\}$で、Λは目標リスト、Ψは条件式である。更新演算の意味は、(Λ, Ψ)を満足する目標集合に基本更新演算opを作用させることである。利用者のいる仮想DBSをVS_0とする。

A．消去演算

$X, X_1, \cdots, X_m$を全体データ構造内の関係とする。Xに対する消去演算はRQLで次の様に表せる。

$\underline{var}\ (\chi, X)(\chi_1, X_1)\cdots(\chi_m, X_m);$

$\underline{del}\ \chi\ \underline{\omega here}\ \Psi_\chi;\quad (\chi=\{\chi, \chi_1, \cdots, \chi_h\})$

Ψ_χは、$\chi, \chi_1, \cdots, \chi_h$上の条件式である。この意味は、次の様である。

(1) $\underline{var}\ (\chi, X)(\chi_1, X_1)\cdots(\chi_m, X_m);$

$\underline{get}\ T[d=\chi.@X]\ \underline{\omega heve}\ \Psi;$

(2) 全$\delta \in T$に対して、$@X(t)=\delta$なる組$t \in X$を消去する。

ここで、@Xは、tの組識別子である。

〔消去演算の為の通信手順〕

(1) TをB²C通信手順でVS$_i$を求める。VS$_i$はXの存在する仮想DBSである。

(2) VS$_i$で、T内の全組をXから消去出来れば ack をVS$_0$に返す。一つでも出来なければ、更新を無効として、abort を返す。□

B．追加演算

X, X$_1$, …, X$_h$を全体データ構造内の関係とする。追加演算はRQLで以下の様に表せる。

var (χ$_1$, X$_1$)…(χ$_h$, X$_h$) ;

app X〔Λχ〕 ωhere Ψχ ; (χ={χ$_1$, …, χ$_h$})

Λχ とΨχ は各々、χ$_1$, …, χ$_h$上の目標リストと条件式である。この意味を以下に与える。

(1) var (χ$_1$, X$_1$)…(χ$_h$, X$_h$) ;

get T(Λχ) ωhere Ψχ ;

(2) 全t∈Tに対して、tをXに加える。

〔追加演算の為の通信手順〕

(1) Tを、B²C通信手順により、Xの存在する仮想DBS VS$_i$に求める。

(2) VS$_i$は、TをXに加える。成功すればackをVS$_i$に返し、失敗すればabort を返す。□

C．置換演算

X, X$_1$, …, X$_h$を、全体データ構造内の関係とする。Xに対する置換演算は、RQLで次の様に表せる。

var (χ, X)(χ$_1$, X$_1$)…(χ$_h$, X$_h$) ;

rep χ (Λχ) ωhere Ψχ ; (χ={χ, χ$_1$, …, χ$_h$})

Λχ とΨχ は各々、χ上の目標リストと条件式である。この意味を以下に与える。

(1) var (χ, X)(χ$_1$, X$_1$)…(χ$_h$, X$_h$) ;

get T(χ.@X, Λχ) ωhere Ψχ ;

(2) 全 (δ, u) ε T に対して、@X (t) = δ なる t ε X を u に置換する。

〔置換演算の為の通信手順〕

(1) Tを、Xの存在するＶＳ$_i$ に求める。

(2) ＶＳ$_i$は、Tをもとに X を置換する。成功すれば ack を、失敗すれば abort をＶＳ$_0$ に送信する。□

5. 全体トランザクションのDDBS処理方法

本章では、全体データ構造上の全体トランザクションのＤＤＢＳ処理方法について論じる。全体問合せのＤＤＢＳ処理方法と同様に、放送網としてのＬＡＮの放送通信を用いた方法を与える。

5.1 トランザクション

全体トランザクションとは、ＬＩＳの全体データ構造内の関係に対する検索演算と更新演算とで定まる系列である。更に、全体トランザクションは、原子的な実行単位であり、正しい（インテグリティ条件の保れれた）全体データベースの状態を、正しい状態に変化させる。しかし、全体トランザクションの各演算の実行後では、必ずしも全体データベースを正しい状態に保つとは限らない。

これに対して、分散トランザクションとは、2.2.2項の定義2.3で述べた様に、(i)各仮想ＤＢＳの操作演算と、(ii)仮想ＤＢＳ間での通信演算で定まる系列である。又、分散トランザクションは、原子的な実行単位である。

全体トランザクションのＤＤＢＳ処理は、次の二つの操作から成る。

(i) 全体トランザクションを分散トランザクションに変換する。

(ii) 分散トランザクションを実行する。

(i)をトランザクション変換、(ii)をトランザクション実行とする。

全体トランザクションをＤＤＢＳ処理する為には、次の制御が必要となる。

(i) コミットメント制御

(ii) 同時実行制御

(iii) 障害回復制御

分散トランザクションは、一般に、複数の仮想ＤＢＳ内のオブジェクトに対する更新を行う。トランザクションの原子性を保つ為には、複数の仮想ＤＢＳ内のオブジェクトに対する更新演算が原子的であること、即ち、全て更新されたか、又は全く更新されていないかどちらかであることを保障する必要がある。この為の制御をコミットメント制御とする。

一方、ＬＩＳ内のオブジェクトのインテグリティを保つとともに、ＬＩＳ全体のスループット（単位時間当りに処理されるトランザクション数）を向上させる必要がある。この為には、各仮想ＤＢＳで、複数の分散トランザクション

の競合する演算（例、同一オブジェクトに対する検索と更新、更新と更新演算）の実行順序を一定に保たねばならない。この為の制御を同時実行制御とする。同時実行制御の目的は、ＬＩＳ内のオブジェクトのインテグリティを保つことと、ＬＩＳ全体のスループットの向上である。

更に、ＬＩＳ内の種々の障害に対して、ＬＩＳ内のオブジェクトのインテグリティを保つ必要がある。この為の制御を、障害管理とする。障害としては、各仮想ＤＢＳの障害、通信網の障害がある。通信網の障害では、網内のメッセージの紛失、重複、破壊等の他に、仮想ＤＢＳ又はそのグループの孤立化が問題となる。

全体トランザクションＧは、Ｇの入力された仮想ＤＢＳ（VS_0とする）によって、分散トランザクションに変換される。この変換手続きを以下に与える。

〔トランザクション変換手順〕

(1) 全体トランザクション内の各基本操作演算（first，last，next，prior，read，del，app，rep）を、そのまま各仮想ＤＢＳの基本操作演算とする。

(2) 割り当て演算$x.\alpha = \exp(y.\beta)$（expは、$y.\beta$の式）に対して、xとyが異った点にあれば、送信演算send（x,y）とする。□

全体トランザクションを、各仮想ＤＢＳ毎の基本操作演算に加えて、仮想ＤＢＳ間の通信演算を加えたものが分散トランザクションとなる。

5.2 コミットメント制御方法

ＤＤＢＳでは、ｎ個の実体間で、全員一致の場合のみ合意（コミット）が得られたとするコミットメント制御が、ＤＤＢＳ処理の基本となっている。従来の一対一網では、〔ＧＲＡＹＪ８１〕に示される様に、(i)ある点iから、ｎ個の点一つづつに同一の request を送信し、(ii) request の応答を受信し、(iii)全点が合意したかしないかの結果をｎ点に送信しなければならない。この為には３ｎ回の通信が必要になる。ＬＡＮの放送通信を用いると、〔ＴＡＫＩＭ８２Ｃ〕に示すように、ｎ＋１回の通信で合意が取れる。

〔コミットメント手順〕

(i) ある点 i から、全点に request を放送する。

(ii) request を受けた各点 j は、ＯＫ又はＮＯの応答を放送する。

(iii) 全点（ i と j ）は、他の点からの応答を待ち、全応答がＯＫであれば合意が得られ、一つでもＮＯのときは合意が得られなかったとする。□

図 5.1 は、上記の手順を図示している。

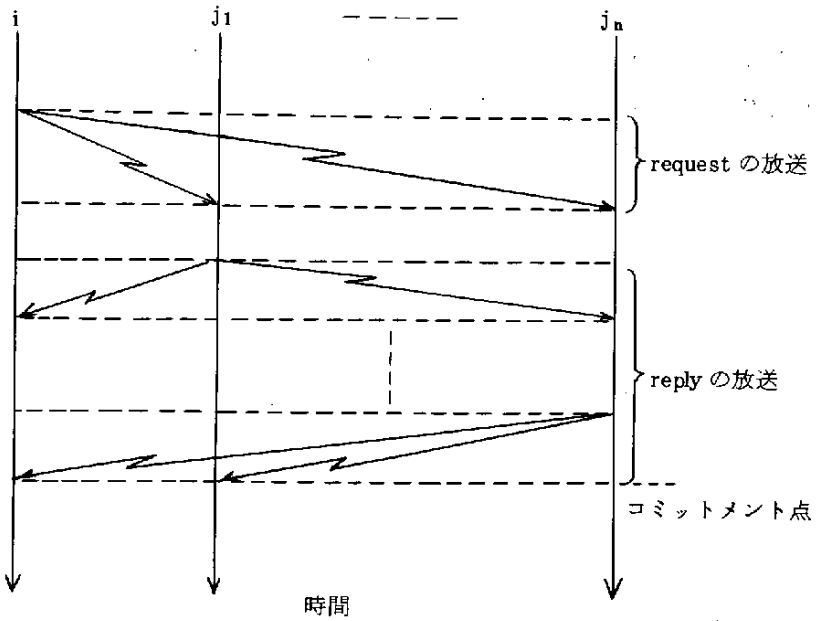


図 5.1　コミットメント手順

以上の手順を、トランザクションがｎ個の仮想ＤＢＳ内のデータを更新する場合に適用すると、従来の二相コミット手順〔ＧＲＡＹＪ８１〕が４ｎ回の通信が必要となるのに対して、２ｎ＋１回の通信で行える〔ＴＡＫＩＭ８１、８４ｂ〕。

〔Ｂ２Ｐ通信手順〕

(i) トランザクションＴを受けたＶＳ$_0$に、更新データをＢ^{2}Ｃ通信手順で求める。求まった更新データをｐｒｅｃｏｍｍｉｔメッセージとして、更新対象の全ＶＳ$_1$，ＶＳ$_2$，…，ＶＳ$_n$に放送する。

(ii) 各VS_iは、precommitを受信したならば、ログlog_iに更新データを退避させる。完了したならば、precommittedを放送する。

(iii) 各VS_iは、他の全VS_jからのprecommittedを受信する。全て受信したならば、log_iを用いてDB_i内のデータの物理更新を行う。完了したならば、ack(nowledgement)を放送する。

(iv) 各VS_hは、全ackを受信したならば、トランザクションを終了させる。

(v) (iii)で、全precommittedを受信出来ない時、トランザクションをアボートさせる。□

図5.2には、B2P通信手順を時間軸を入れて示してある。又、図5.3には、状態遷移図によって、B2P通信手順を示す。

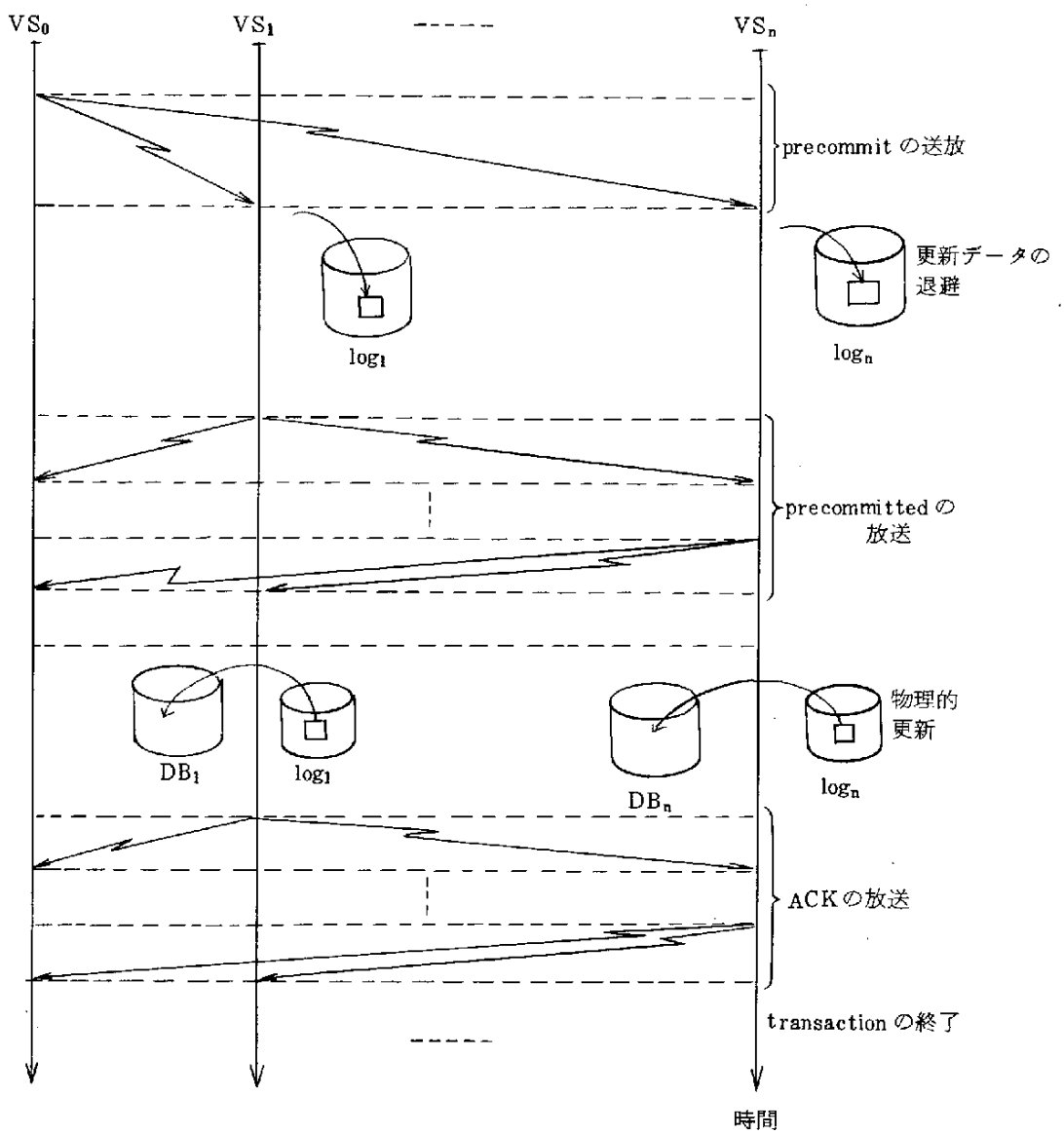
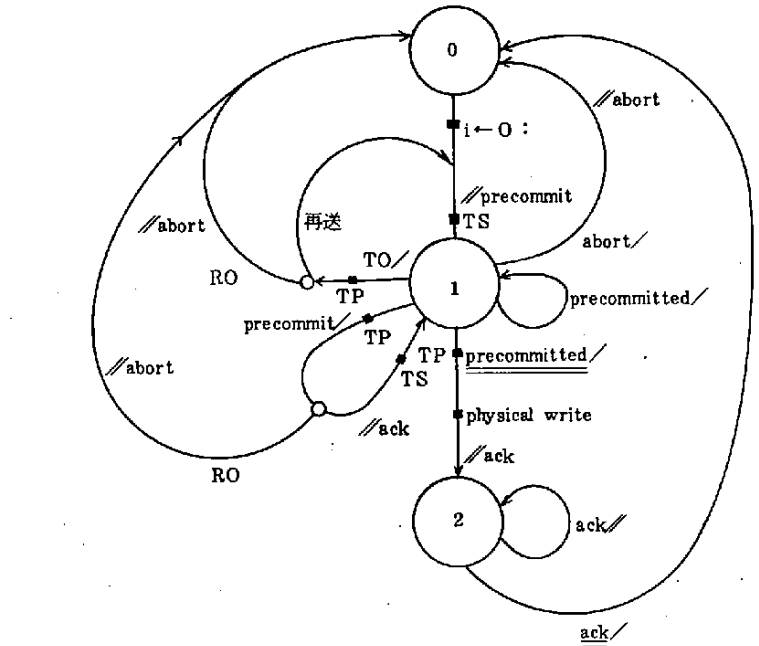
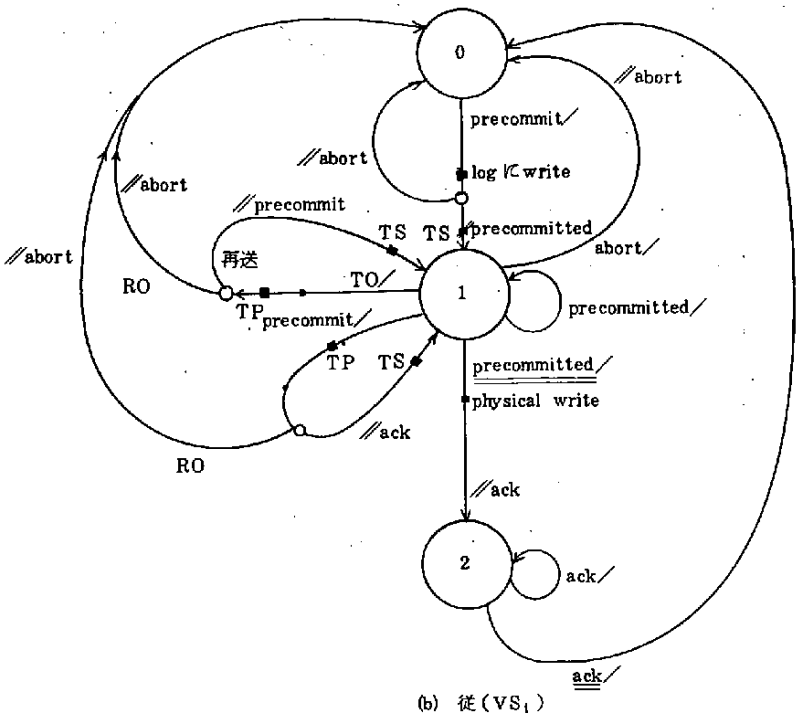

VS0
VS1
VSn
precommit の送放
更新データの
退避
log1
logn
precommitted の
放送
DB1
log1
DBn
物理的
更新
logn
ACK の放送
transaction の終了
時間

図 5.2　B 2 P 通信手順



(a) 主（VS_0）



(b) 従（VS_1）

図 5.3　B 2 P 通信手順の状態遷移図

5.3 同時実行制御方法

分散トランザクションを実行させる為には、同一のオブジェクトに対する複数の分散トランザクションからの演算に対して、全体データ構造のインテグリティを保つ必要がある。同時実行制御とは、以下の目標を達成する為の制御である。

(i) ＬＩＳの全体データ構造（オブジェクト集合）のインテグリティを保つ。

(ii) ＬＩＳ全体のスループットを向上させる。

(ii)の目標を達成する為には、各仮想ＤＢＳで、複数の分散トランザクションの基本操作演算をインタリーヴして実行させ、トランザクション実行の多重度を高める必要がある。個々の分散トランザクションの実行が正しい、即ち、単独で実行した場合には、インテグリティを保つとする。このとき、(i)は、複数の分散トランザクションを直列に実行させることによって、目標を達成出来る。各仮想ＤＢＳでの基本操作演算の実行順序を変えると、演算が競合するときには、結果が異ってくる。競合する演算とは、同一のオブジェクトに対する検索（ read ）と更新（ write）演算、及び更新と更新演算である。従って、同時実行制御とは、複数のトランザクション間の競合する演算の実行順序を、各仮想ＤＢＳで一定に保つという制約下で、トランザクション実行の多重度を高めることである。

従来の一対一網を用いた分散型データベースシステム（ＤＤＢＳ）での同時実行制御の問題は、ルーティング、エラー処理等の影響で通信遅延が各通信リンク毎に異なっている点である。従って、ある点 i から、二点 j_1 と j_2 に、あるメッセージＸが送信されたとすると、j_1 と j_2 には一般に同時に到着しない。従って、各点に到着するトランザクション（の基本演算）の順序は異っている。この為に、トランザクションに時刻印〔 BERNP81 〕を設けて、受信点で、競合するランザクションの演算を時刻印順（送信順）に実行する方法を用いている。一対一網では、通信遅延が通信リンク毎に異なるので、ある時刻印のトランザクションの基本演算を実行するためには、これ以前の時刻印のトランザクションの全基本演算が到着していることの保障が必要になる。基本演算の実行前に、一定時間（通信リンクの最大遅延時間）待つ等の手順が必要になる。

以上見た様に、一対一網では、各通信リンクの通信遅延の相違により、その制御が複雑なものとなっている。これに対して、ＬＡＮ等の放送網では、ある点から送信された情報（パケット）は、同時に全点に到着する。よって、通信

遅延の差による問題は生じないので、放送網を用いると、同時実行制御を容易なものとすることが出来る。放送網では、分散トランザクションSに変換した点のVS_0が、関連する仮想DBSのVS_1, …, VS_nにSを放送した時、全点で、この時に受信されたか、全く受信されなかったか、即ち、コミットされたかされないかを保障すればよい。

同時実行制御を考えるうえで、以下の仮定を設ける。

(i) 各仮想DBSは、DBS内での同時実行制御機能を有している。

(ii) 各仮想DBSは、関係の組単位（実際には、ページ単位）の排他制御機構として、検索演算に対する共有ロックとしてのRロックと、排他的なWロックとを設ける。

(iii) トランザクションの基本演算readに対してRロックが、del, app, rep に対してWロックが設けられ、トランザクションの終了時にロックは解除される。□

〔同時実行制御の通信手順〕

(1) 全体トランザクションを受けた仮想DBSをVS_0とする。VS_0は、全体トランザクションを分散トランザクションSに変換する。

(2) Sの参照する関係を持つ、仮想DBS VS_1, …, VS_nの（DD）実体間で群（動的、集中、非保障群）を形成する。VS_0が主実体となる。

(3) VS_0は、Sを放送する。コミットメント制御により、全VS_1, …, VS_nに、同時にSを受信させる。

(4) 各VS_iは、Sの受信がコミットされたならば、Sの基本演算を起動する。

(5) 各VS_iは、Sに対する全更新データを、自分のログ内に求める。まとまったならば、precommitted メッセージを放送する。

(6) 全点からprecommittedを受信したならば、ログ内のデータを用いて、物理的更新を行う。更新が終了したならばackを放送する。

(7) VS_0は、全ackを受信したならば、トランザクションを終了し、群を終了させる。□

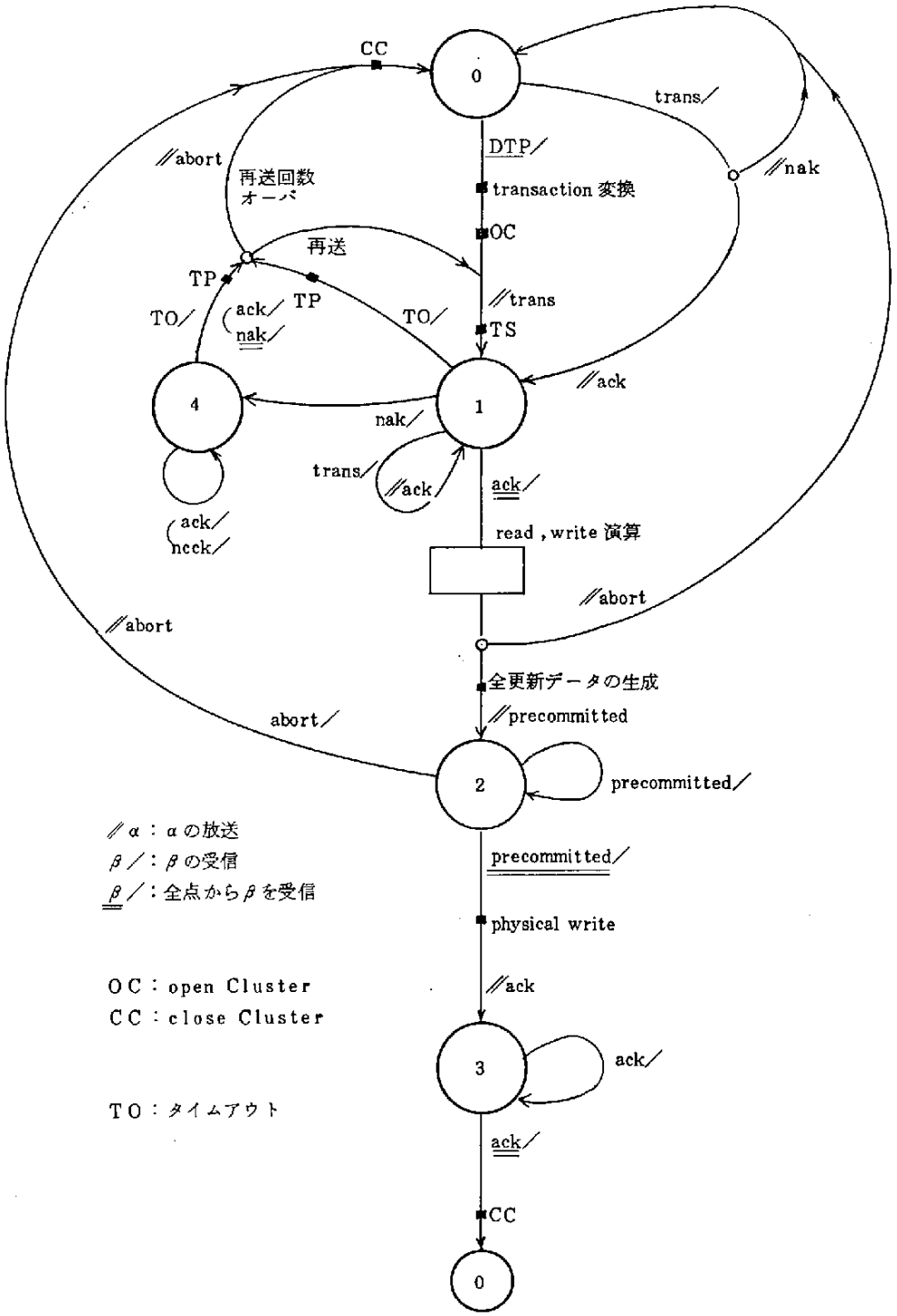


図 5. 4　同時実行制御の通信手順

5.4 障害管理方法

分散トランザクションの処理では、システムの種々の障害に対して、全体データ構造のインテグリティを保たねばならない。この為の制御を障害管理とする。ＬＩＳでの障害としては、次の２種がある。

(i) 各仮想ＤＢＳの障害

(ii) 通信網の障害

まず、(ii)の通信網の障害について考える。従来の一対一網では、網の通信路が障害を起こすことによって、ある点又は、点のグループが孤立化してしまう可能性がある。これを、網分断障害とする。しかし、ＬＡＮでは、通信路が一本だけなので、網が障害を起こして時には、全点が通信不能となってしまい、網分断問題は生じない。

各仮想ＤＢＳの障害について考える。仮定として、各仮想ＤＢＳは、単独の障害については、障害回復機能を有しているものとする。各仮想ＤＢＳの障害に対しては、分散トランザクションをバックアウト（コミットせずに、異常終了（アボート）させる）させる。プレコミット後の障害に対しては、回復時に、ログより物理更新を行ってから、復旧するものとする。

6. 視野処理(VP)サービス

本章では，ＬＩＳの応用としての視野サポートサービスについて述べる。

6.1 超小型計算機の役割

超小型計算機の記憶容量の増大を中心とした高性能化と，ＤＢＭＳを中心とした高機能化とによって，従来，大型機内に形成されてきたＤＢＳを，各個人単位の超小型計算機上にも実現出来る様になってきている。このことは，自分の計算機内のデータを，ローカルに処理出来ることを意味している。次に，この様な超小型計算機が，ローカルエリア網(ＬＡＮ)によって，他の超小型計算機，又従来の大型計算機と結合されることの意味を考えてみる。従来の計算機の端末は，単に，データの入力と，計算機から導出されたデータの表示だけに用いられてきた[図6.1]。しかし，ＤＢＳ機能を有した超小型機がＬＡＮ

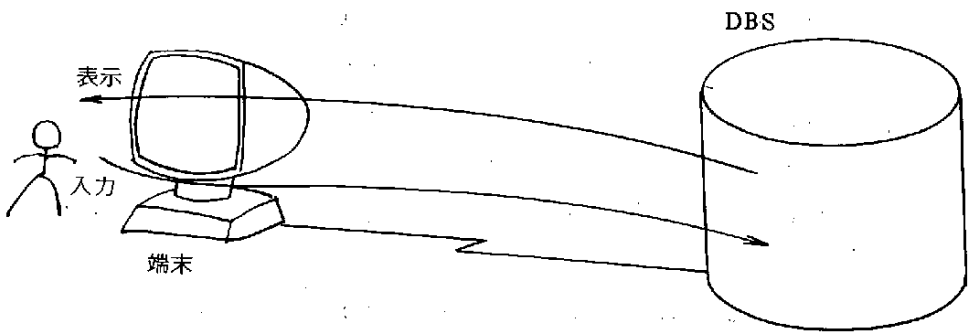


図6.1　従来の端末によるＤＢＳ利用

で結合されてくると，他のＤＢＳ内のデータを導出して，その結果を表示するだけでなく，結果を格納しておき，種々の処理をローカルに行うことが出来る[図6.2]。従って，応用が毎回必要とするデータを，ＤＢＳ又は，ＤＤＢＳから毎回導出する必要はなくなり，自分の計算機内のデータをローカルに利用出来ることになる。各ＤＢＳ及び，通信網の負荷を減少出来，かつ各利用者のデータ獲得性を高めることが出来る利点をもたらす。一方で，ＤＢＳ内のデー

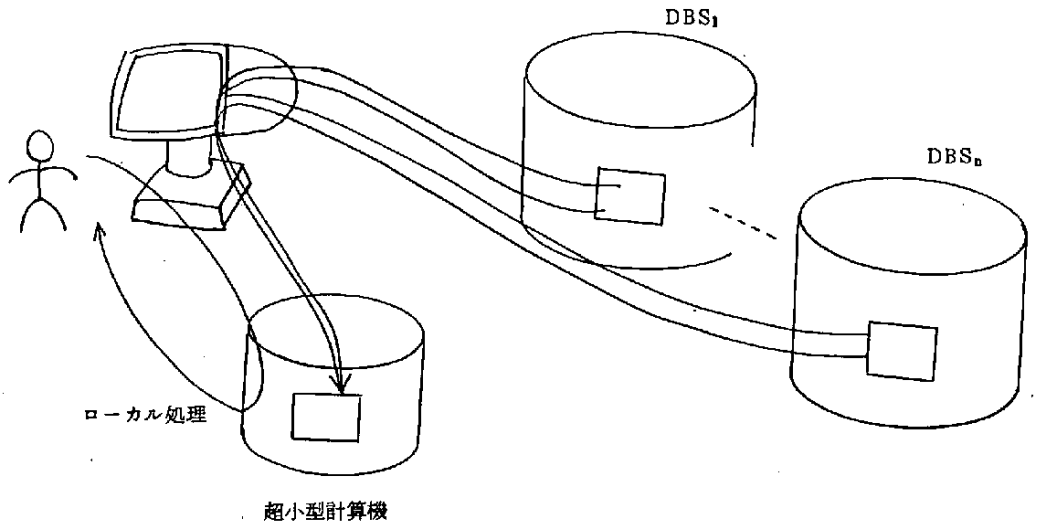


図 6.2　ＬＡＮのもとでの超小型機の利用形態

タが更新されていくことによって，超小型計算機内のデータは，〃古い〃ものとなってしまう。最新の，又は比較的新しいデータを必要とする応用にとってはこの状態は望ましくない。従って，超小型計算機内のデータと，ＤＢＳのデータ間での，ある程度の一致性の保持が必要となってくるものと思われる。

ＬＩＳでは，各作業者の必要とするデータは自分の超小型機内に導出しておいて，ローカルに利用するものとする。更に，ＬＩＳでは，これらのデータと，元のデータ間で一致性の必要のものの保持を行うことを目指している。

6.2　視　　野

ＬＩＳの一般利用者には，視野が与えられる。視野は，全体データ構造上の抽象的なデータ構造と，これに対して適用を許される抽象操作演算の集合とによって定まる。この抽象的なデータ構造（抽象オブジェクト）を，抽象関係構造とする。利用者は，抽象操作演算を用いてのみ抽象関係構造の操作を行える。即ち，視野は，抽象データ型［ＬＩＳＫＢ７７］とみなすことが出来る。

視野Ｖは，四字組（Ω_V，V_V，O_V，M_V，）として与えられる。（Ω_V,V_V）は，抽象関係構造を与える。Ω_V は，属性集合で，抽象関係スキームとする。

V_vは，Ω_vに実際に値を与えたもので，抽象関係とする。O_vは，抽象関係に適用可能な抽象操作演算の集合である。$M_v=(Q_v, R_v)$であり，全体データモデルと視野V間の写像である。$Q_v=(\Lambda_v, \Psi_v)$は，抽象関係構造を全体データ構造に与えるQ_vを，定義式とする。Λ_vとΨ_vは，ＲＱＬと同一形式の目標リストと条件式である。R_vは，O_v内の抽象操作演算を，全体データモデル上の操作演算で表現（又は実現）したものである。この表現は，分散トランザクションである。

例として，次の二つの関係が全体データ構造内に存在したとしよう。

EMP　（eno, ename, age, sal, sex, dno）

DEPT　（dno, dname, nanager）

経理部門では，従業員の給料についてのみ操作を行う。この時，以下の視野accountng を設ける。

```
accountng－view （name, sal, dnane）：
definition var （e, EMP）（d, DEPT）：
        def accounting－view
        （name＝e. ename, sal＝e. sal, dname＝
        d. dname）
        where d. dno＝e. eno；
operations
(i) 昇　給　　promote（who, percent）；
        begin
                rep e（sal＝e. sal＊（1＋percent/100））
                where e. eneme＝〝who〞；
        end；
(ii) 部と部員の消去
                erase（dept）；
        begin
                del e where d. dno＝e. dno and d. dname＝
                dept；
                del d where d. dname＝dept；
        end；
```

利用者は，抽象関係accounting－viewに，promoteとeraseの二つの操

作演算のみを適用出来る。

視野は，安全性の制御用にも利用出来る。利用者は，与えられた視野を利用出来るだけであり，視野は抽象関係をオブジェクト，抽象演算をオブジェクトの利用権としたケーパビリティと考えれる。ケーパビリティを有している利用者が，ケーパビリティ内の操作演算に基づいて抽象関係を操作出来る。

6.3 視野の実現

視野は，オブジェクトとしての抽象関係と，オブジェクト上の抽象演算とから成っている。抽象関係Vが，全体データベース内の関係X_1，…，X_nから，(Λ, Ψ)によって定義されているとする。即ち，$V=\{u \mid u=\Lambda(t) \wedge \Psi(t) \wedge t \in X_1 \times \cdots \times X_n\}$である。視野の抽象関係Vについて考える。Vとしては，次の点について考察出来る。

(i) Vの仮想性

(ii) Vの所在

(iii) VとX_1，…，X_n間の一致性

A 抽象関係の仮想性

抽象関係Vとしては，次の二種が考えられる。

(i) Vとして実体がある。

(ii) Vは仮想的である。

(i)は，関係X_1，…，X_nに対して，$V=\{u \mid u=\Lambda(t) \wedge \Psi(t) \wedge t \in X_1 \times \cdots \times X_n\}$なるVが物理的に存在している場合である。このときのVを，実現化抽象関係とする。一方，(ii)は，Vとしての実体は存在しない。このときのVを仮想抽象関係とする。これは，従来のＤＢＳでの視野，又は外部スキーマに対応して考えれる。

B 抽象関係の所在

抽象関係Vが実現化されているとき，Vがどこに存在するかが問題となる。Vの存在場所としては，以下の場合が考えられる。

(i) 利用者のいる超小型計算機（ＰＩＳ）内。

(ii) 利用者のいるＰＩＳ（個人情報システム）以外の仮想ＤＢＳ内。

ＬＩＳでは，特に，(i)について考える。各利用者は，視野の抽象関係を自分のＰＩＳ内に格納し，ローカルに処理出来る。

C　抽象関係の一致性

次に，実現化抽象関係Vと関係X_1，…，X_n間の一致性について考える。一致性の度合としては，以下を両極端として，この間に種々の度合が存在する。

(i)　VとX_1，…，X_n間には全く一致性は存在しない。VとX_1，…，X_nは互いに独立である。

(ii)　VとX_1，…，X_n間には，常に一致性が存在しなければならない。

(i)では，Vは関係X_1，…，X_n　から検索演算（Λ，Ψ）で導出された目標関係である。従って，VとX_1，…，X_nとは，互いに独立したものとなる。(ii)の型の視野を，独立型視野とする。一方，一致性の存在する視野を，一致性視野とする。以降，単に視野と述べる時には，特にことわらない限り，一致性視野を表すものとする。抽象関係Vと関係X_1，…，X_n間の一致性については，以下の仮定を設ける。

［仮定6.1］

抽象関係Vに対して更新がなされるときは，VとX_1，…，X_n間にある度合の一致性が存在するならば，必ずX_1，…，X_nは更新される。　□

6.4　抽象関係への更新波及方法

全体データベース内の関係に対する更新を，ＰＩＳ（個人情報システム）内の抽象関係とこれらの関係との一致性を保つ為に波及される方法について考える。

A　更新波及方法

X_1，…，X_nを全体データベース内の関係とする。Vを，X_1，…，X_n上の抽象関係とする［図6.3］。関係X_1，…，X_nの変化に対して，抽象関係Vの一致性を保つ為の方法としては，以下の方法が考えられる。

(1)　即時更新法　ある関係X_iが更新されたならば，直ちに，X_iを参照している抽象関係Vの更新を行う。

(2)　再検索法　Vの視野Vの定義式（Λ_V，Ψ_V）を検索演算として，X_1，…，X_n上で再実行させ，この目標集合をVとしてＰＩＳに導出する。

(3)　差更新法　関係X_iに対する更新によるX_iの変化分を記録しておき，この変化分だけVを一括して更新する。　□

VとX_1，…，X_n間に常に一致性が保たれる必要がある時には，(1)の即時

更新法が用いられる。一致性として，常ではなく，一定期間（時間毎，日毎等），利用者が要求した時必要な場合には，(2)の方法又は，(3)の方法が用い

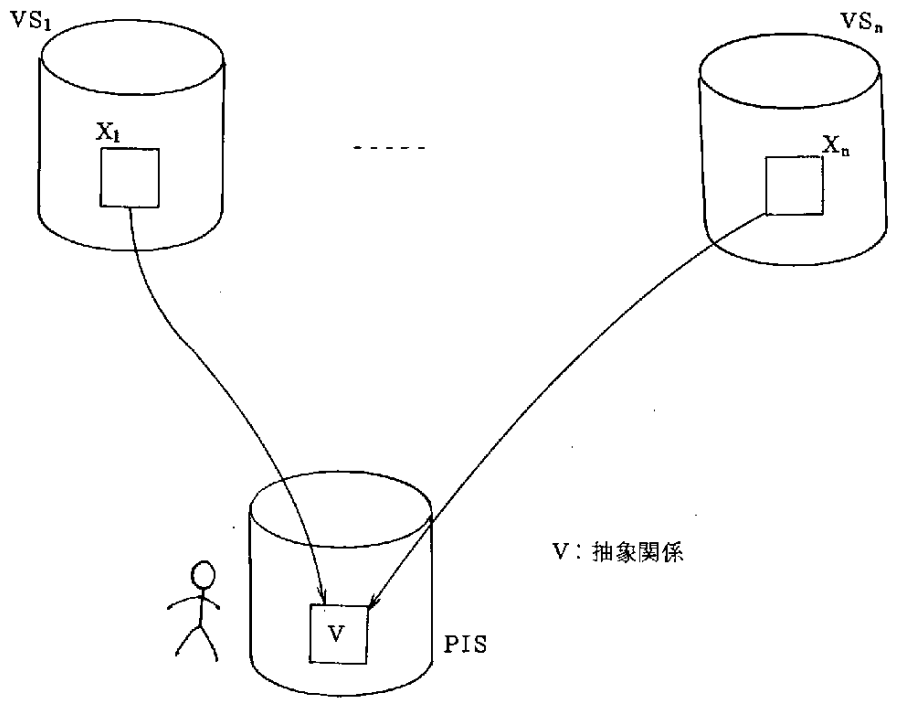


図 6.3　ＰＩＳの抽象関係

られる。ＬＩＳでは，ＶとX1，…，Xn 間に即時的な一致性が求められない場合には，再検索法を用いることにする。以降，即時性が要求される場合について考察する。

B　抽象関係の定義式

視野Ｖの抽象関係Ｖの定義式$Q_v=(\Lambda, \Psi)$は，図 6.4 に示すグラフとして表せる。x_i は，ＶＳi 内の関係Xi の組変数である。π_{ij}は，$x_{i,j}$間の結合条件式，ρ_i はx_i の制限条件式，λ_i はx_i の目標属性集合である。ここで，$\Lambda=\bigcup_{i=1}^{n}\lambda_i$ であり，$\Psi=\bigwedge_{i=1}^{n}\rho_i\bigwedge_{i=1}^{n-1}\bigwedge_{y=i+1}^{n}\pi_{ij}$である。$\Lambda_i$ をx_i の結合属性の集合とする。このとき，抽象関係Ｖは，Ｖ（Λ_1，…，Λ_n）となる。このＶに対して，関係$\widetilde{V}$（Λ_1，…，Λ_n，D_1，…，D_n，A_1，…，A_n）を次の様に定める。

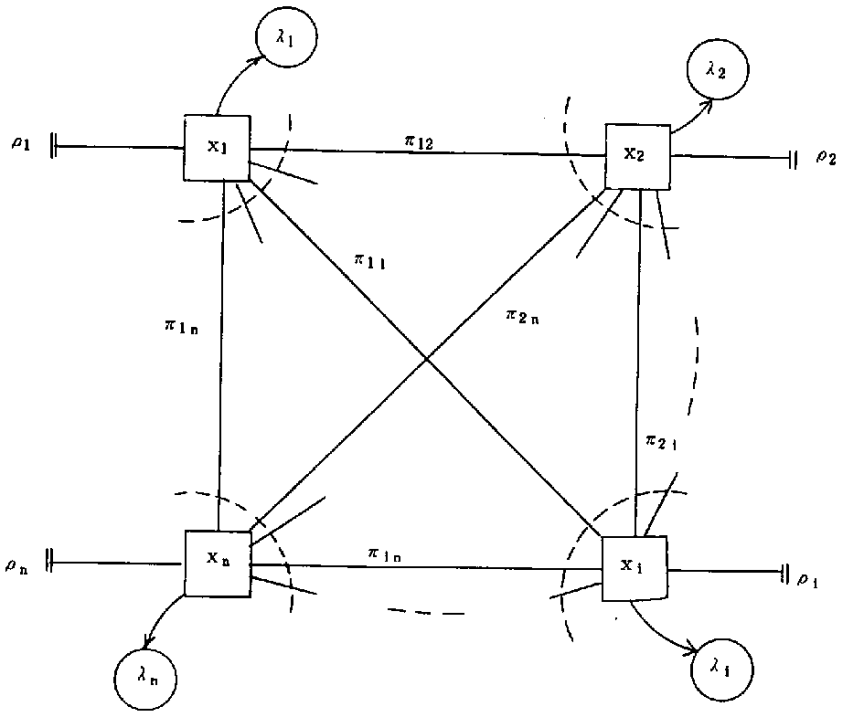


図 6.4　視野定義式のグラフ表現

$$\widetilde{V} = \{\, t \mid t = (A_1(w), \cdots, A_n(w), D_1(t_1), \cdots, D_n(t_n), A_1(w), \cdots, A_n(w)) \wedge \Psi(w) \wedge w = (t_1, \cdots, t_n) \in X_1 \times \cdots \times X_n \,\}$$

ここで，$D_i(t_i)$は，組 $t_i \in X_i$ の組識別子とする。この関係$\widetilde{V}$を，抽象関係の基底抽象関係とする。Vを，X_i の変化に対して即時更新する為に，ＰＩＳ内に$\widetilde{V}$を保持する。

又，即時更新を行う為には，ＰＩＳの（ＤＤ）実体と，各仮想DBS VS_i 間には，永続的な群が設定されている必要がある。

C　消去演算

関係X_i に対する次の更新演算を考える。

var（x_i, X_i）（w_1, W_1）…（w_m, W_m）：

del x_i where Ψ_i ；

ここで，W_1，…，W_mは，全体データ構造内の任意の関係である。上記の消去去演算に対して，Vの一致性を保つ為の通信手順を以下に与える。

［消去波及の通信手順］

(1) VS_i で，次の集合 T_i を求める。

$$T_i = \{ d_i \mid d_i = D_i(t_i) \wedge t_i \in X_i \wedge u \in W_1 \times \cdots \times W_m \wedge \Psi_i (t_i, u) \}$$

即ち，T_i は，消去される組の識別子の集合である。これらの組を消去する。

(2) VS_i は，T_i を放送する。

(3) T_i を，PISは受信する。PISは，各 $d_i \in T_i$ に対して，$d_i = D_i(u)$ なる組 u を $\tilde{V}$ から全て除く。

(4) 終了したならば，done メッセージを放送する。

VS_i は，全ての done を受信したならば，T_i を消去する。 □

以上の手順を図 6.5 に与える。消去される組の識別子の放送通信によって一回の通信で，X_i を参照する全ての抽象関係内の組を消去出来る。

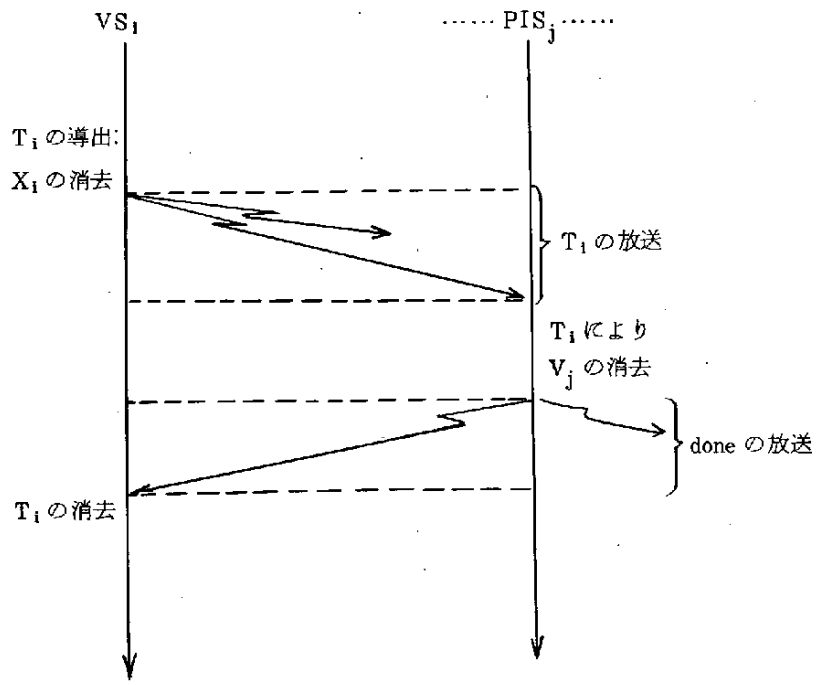


図 6.5　消去波及手順

D　追加演算の波及手順

次に，関係X_iに対する以下の追加演算を考える。

var　（w_1，W_1）……（w_m，W_m）；

app　X_i（Λ_i）　where　Ψ_i；

このとき，以下の集合を考える。

$$T_i' = \{ t \mid t = \Lambda_i(w) \wedge \Psi_i(w) \wedge w \in W_1 \times \cdots \times W_m \}。$$

追加演算は，関係X_iに集合T_i'を加える。このとき

$$T_i = \{ < d_i, t > \mid d_i = D_i(t) \wedge t = \Lambda_i(w) \wedge \Psi_i(w) \wedge w \in W_1 \times \cdots \times W_m \}。$$

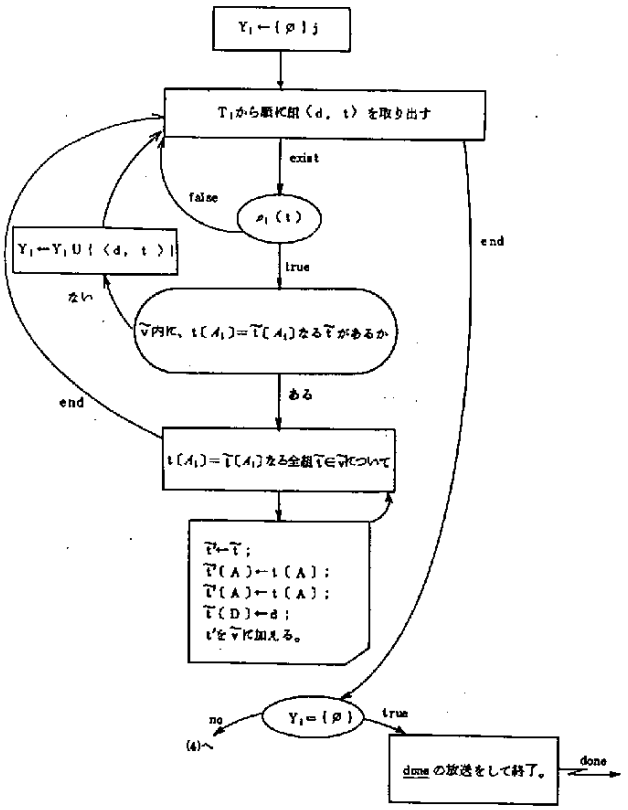
以下に，追加の波及の為の通信手順を示す。

［追加の波及手順］

(1)　VS_iは，集合T_i'を求め，X_iに追加する。このとき，T_iも求める。

(2)　VS_iはT_iを放送する。

(3)　PISはT_iを受信する。



(4) Y_i をＰＩＳの関係として，（Λ，Ψ）を再実行する。B^2C通信手順を用いる。目標集合を，Ｖに加える。 □

Ｅ　置換演算の波及手順

関係X_iに対する以下の置換演算を考える。

var （x_i，X_i）（w_1，W_1）……（w_m，W_m）；

rep x_i（Λ_i）where Ψ_i；

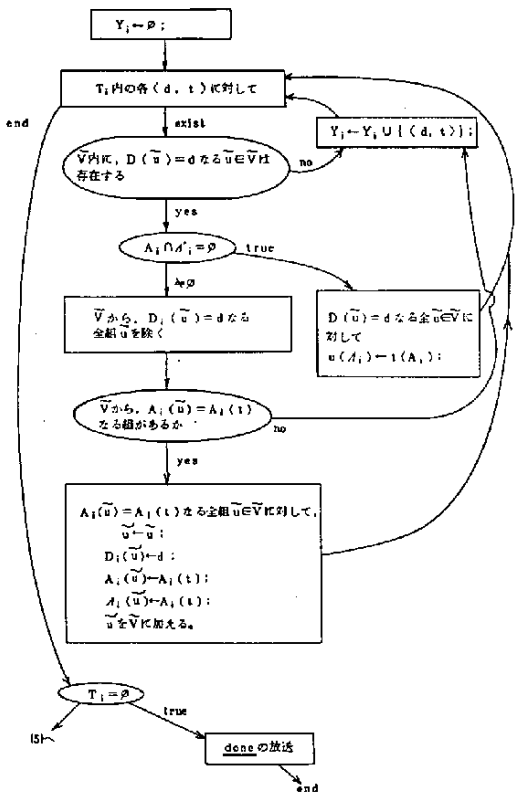
ここで，置換される属性の集合をΛ_i''とする。

［置換の波及手順］

(1) VS_iは，集合$T_i = \{ < d, t > \mid d = D_i(t_i) \wedge t_i \in X_i \wedge w \in W_1 \times \cdots \times W_m \wedge \Psi_i(t_i, w) \wedge t = \Lambda_i(w) \}$を求め，$X_i$の置換を行う。

(2) T_iの放送。

(3) ＰＩＳは，T_iを受信する。$\Lambda_i \cap \Lambda_i'' \neq \emptyset$又は，$\Lambda_i \cap \Lambda_i'' \neq \emptyset$でなければ，done の放送。



(4) X_i をＰＩＳ内のY_iとして，（Λ，Ψ）の再実行を行い，目標集合を求めX_iに加える。 □

6.5 視野を通しての関係の更新

視野として，抽象関係Ｖに対して許される抽象操作演算が定められている。従って，利用者は，抽象操作演算を通してのみ抽象関係を操作出来る。抽象操作演算ＶＯＰが抽象関係に適用されると，ＶＯＰの表現としての分散トランザクションが起動されて，Ｖ及び，Ｖの参照する関係への更新がなされる。

7. 基本通信（ＦＣ）層のプロトコル

ＦＣ層は，ＥＤＬ層のフレームの送受信サービス（ＥＤＬサービス）を用いて，ＦＣサービスをＤＤ層に提供する。本書では，ＦＣ層のプロトコルを示す。

ＦＣインタフェースの各関数を実現する通信手順を与える。通信手順は，状態遷移図で与え，ここで用いる記法とその意味を以下に示す。図7.2～7.5に，FCインタフェースの各関数に対するプロトコルを表す状態遷移図を与える。

：　状態

：　状態遷移

A／　：　上位置（NH）実体からのシステムコールA

／B（A）　：　システムコールAに対して，値Bを（Ｎ＋１）実体に表示して復帰

／α　：　（N）－ＰＤＵ α の放送

β／　：　（N）－ＰＤＵ β の受信

β／（二重）　：　（N）－ＰＤＵ β を群内の全（N）実体から受信

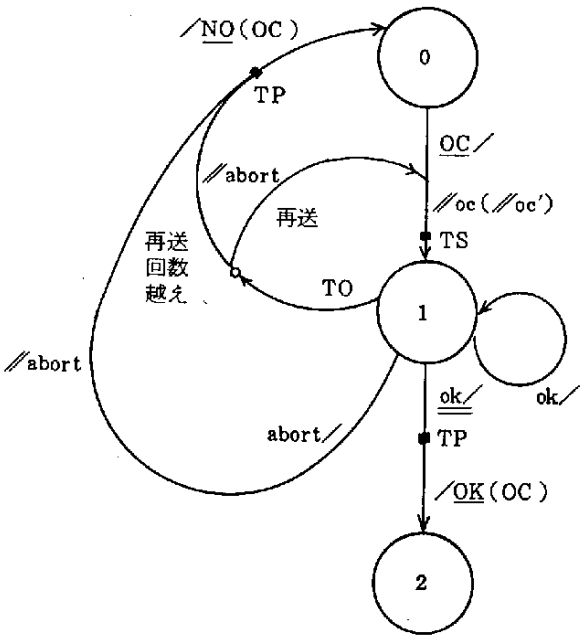
TO／　：　時間切れ

：　動作

ＴＳ　：　タイマ開始

ＴＰ　：　タイマ終了

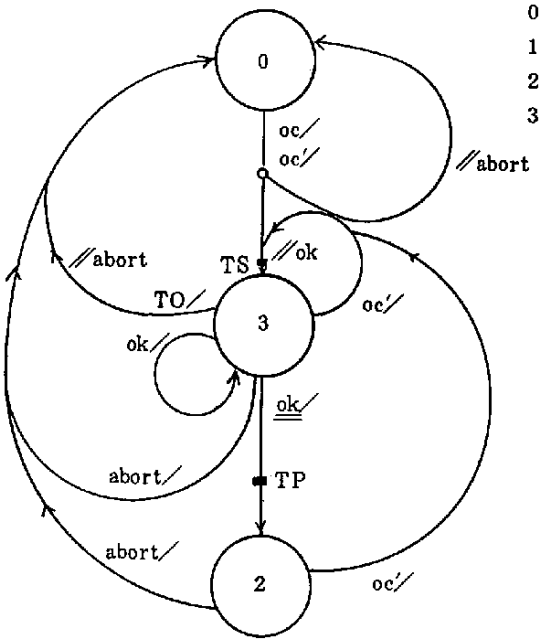
図 7.1　状態遷移図の記法



OC = open cluster

oc' = oc の再送

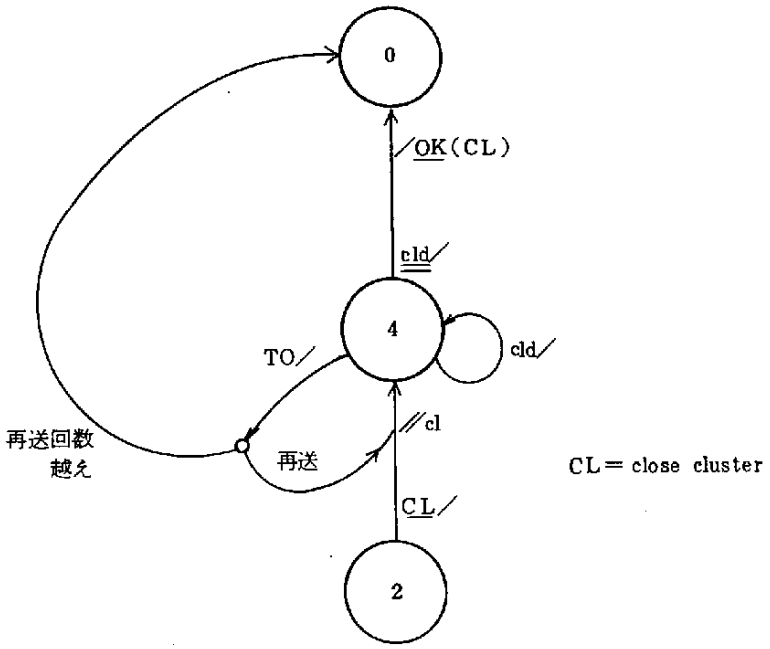
(a) 発起実体



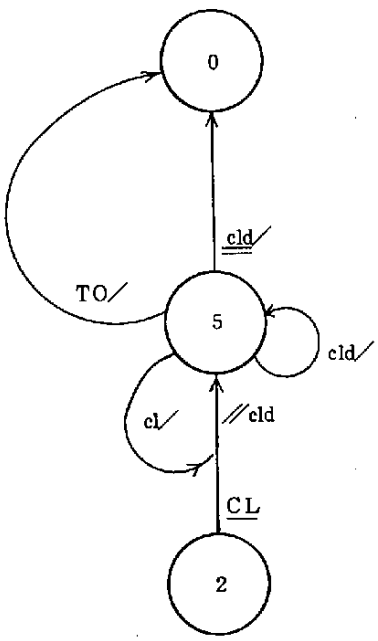
0 : 休止
1 : ok 待ち(群開設中)
2 : 群
3 : ok 待ち(群開設中)

(b) 他

図 7.2 群の開設

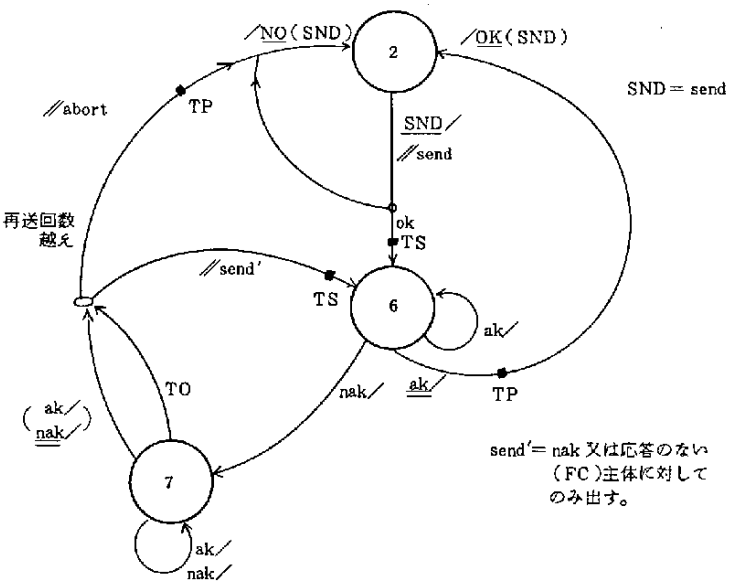


(a) 終了の発起主体

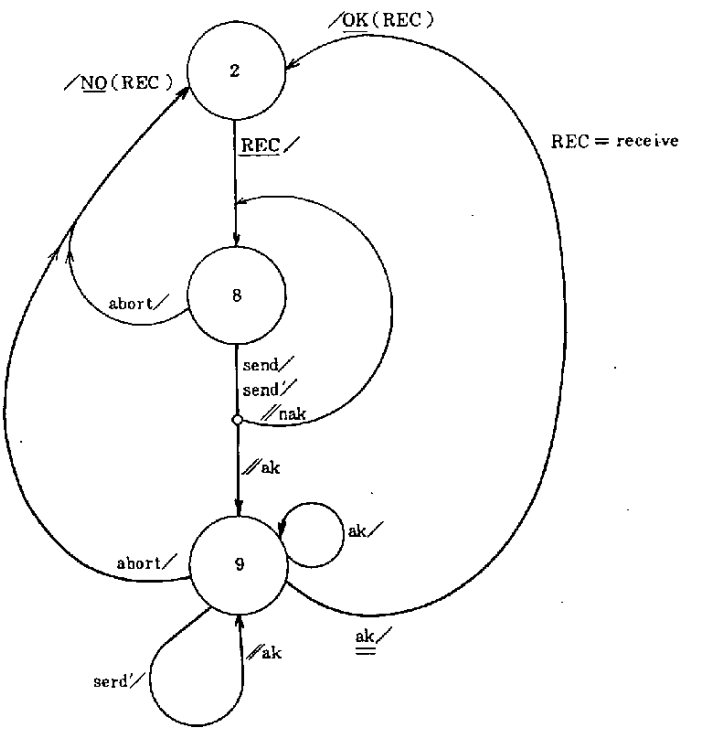


(b) 他

図 7.3　群の終了

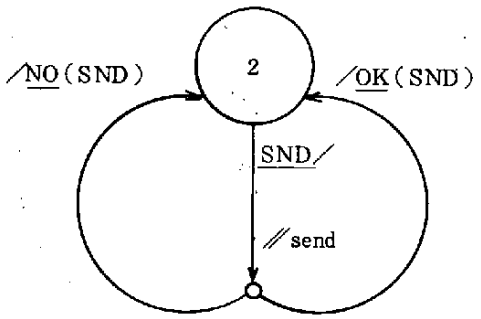


(a) 送 信

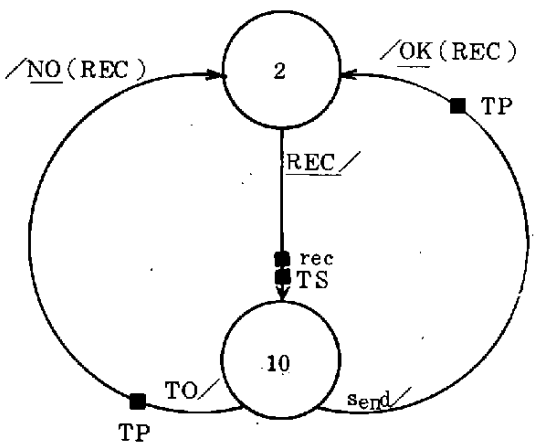


(b) 受 信

図 7.4　保障群での送受信



(a) 送　信



(b) 受　信

図 7.5　非保障群での送受信

8. 超小型計算機用の関係ＤＢＭＳ－Granada

超小型計算機の高性能化に伴い，従来大型機に構成されてきたＤＢＳを，各個人単位に形成出来る様になってきている。この様な超小型機上のＤＢＳをパーソナルＤＢＳと呼ぶ。パーソナルＤＢＳは，外部の大型ＤＢＳからのデータ及び各個人で形成したデータを蓄積し，加工する為に用いられ，通信網で結合されることにより今後の情報システムの主要な要素となっていくものと思われる。本論文では，当協会で開発しているＬＡＮで，超小型機と大型機を結合した分散型データベースシステムＬＩＳ(Local information system)の主要ＤＢＳとしてのパーソナルＤＢＭＳ Granadaの設計と実現について述べる。超小型機上に，従来のメインフレーム並みの機能を有した関係ＤＢＭＳを実現することが，Granadaの目標である。

8.1 Granada 概要

Granadaは，以下の機能を利用者に提供する関係ＤＢＭＳである。

(I) 関係スキーマの定義・変更

(II) 述語論理形式の非手続的操作演算（ＲＱＬ）検索・更新（追加・消去・置換）

(III) セキュリティ管理（アクセスリスト方式）

(IV) 視野（view）の定義・検索・更新

(V) フォーム形式の利用者インタフェース

(VI) コマンドのprocedure化

(VII) 物理構造の変更

(VIII) リカバリ機能

Granadaは次の三つの階層構造から成る［図8.1］。

(I) 外部層－視野（view）

(II) 論理層－論理データ構造とＲＱＬ演算

(III) 物理層－物理構造とアクセスパスとtuple単位の操作演算

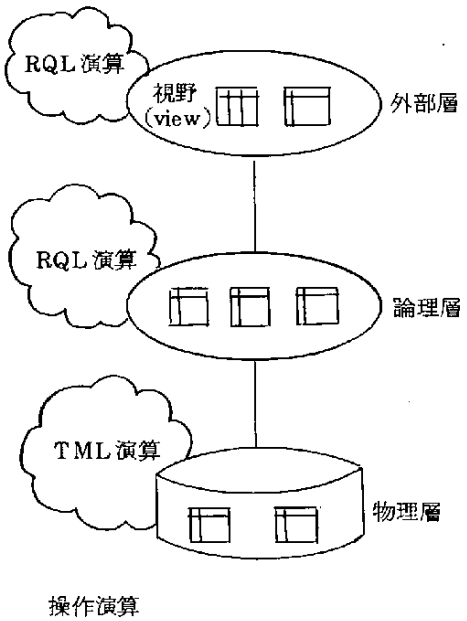


図 8.1　Granada の階層構造

8.2　データ構造

A　論理データ構造

論理データ構造は，関係R（a_i，…，a_m）（$m \geq 1$）の集合から成る。Rの各属性a_iには，以下の特性が定義される。

$dom(a_i)$＝　a_i の定義域

t_i ＝　タイプ（文字（英字，日本字），整数，実数）

l_i ＝　バイト長

n_i ＝ { 1　空値が許される
　　{ 0　空値が許されない

u_i ＝ { 1　R内でユニーク
　　{ 0　R内でユニークでない。

B　物理データ構造

二次記憶媒体は，一定長（1024B）のページから成る。関係Rの各組はページ内に格納され，各組はページ番号とオフセットで定まる組識別子

(tid)を持つ。関係の格納方法としては,(I) ある属性aの値順(aを整列属性とする),(II) 追加順,(III) ハッシングの三種がある。各組内の文字列は,右側の空白を圧縮して格納される。

typ	PP	NP	rel-id	NT	fo	TPL_1	TPL_2	・・・	ptt_2	ptt_1

typ - ページのタイプ
PP - 関係Rの前のPageへのポインター
NP - 関係Rの次のPageへのポインター
rel-id - 関係Rの識別番号

図 8.2 物理ページフォーマット

Rの任意の属性aに索引を設けれる。a値がR内でユニークなとき主索引そうでないとき副索引とする。又,aがRの整列属性のとき,群索引とする。索引は,B^+木で実現されているが,更新時の木内の節の分割,併合は以下の条件のとき行う。

(1) 節(ページ)がオーバフローしたとき,分割する。

(2) 節(ページ)内の利用率が30%以下のとき併合する。

ハッシュ格納は,線型ハッシングを用いる。ハッシュ格納された関係に対しても,索引を設けさせて,連続読出しも可能にしている。

8.3 利用者言語RQL

Granadaの利用者言語は,次の2つから成る。

(I) データベース管理コマンド(DBAC)

(II) データベース操作コマンド(DBMC)

DBACは管理用コマンドで,次の様な機能を持つコマンドである。

a) 論理データ構造の定義,変更,消去

b) 物理データ構造の定義,変更,消去

c) 利用権の管理

DBMCは利用者用コマンドで,次の様な機能を持つコマンドである。

a) 関係の検索

b) 関係の更新(追加,置換,消去)

a) とb) とは，ＲＱＬで表される。ＲＱＬ(relational query language) は，Quel［STONM７６］と類似した形式の述語論理式の問合せ言語である。ＲＱＬは，var，get，del，app，rep 文から成る。

(1) var文

var (<var>, <relation>) {,(<var>, <relation>)} ;

var 文は，<relation> に対して，組変数<var> を定める。組変数の数は，最大１６である。

(2) get 文

```
get [<result>][[<tarzet-list>]]
    [<quel>] :
    <result>  ::= <relation>
<target-list> ::= [<attribute>] <exp>
       <exp>  ::= <vatt> | <constant> |
                  <exp> <aop> <exp> | −<exp> | (<exp>) |
                  <afunc> | <gfunc> | <var>. all
      <vatt>  ::= <var>. <attribute>
       <aop>  ::= + | − | * | / | ^
     <afunc>  ::= <afunction> (<exp>{, <exp>})
<afunction>   ::= sin | cos | log | ……
     <gfunc>  ::= <gfunction> <exp> [, <vatt>{, <vatt>}]
                  [by <vatt> {by <vatt>}] <qual>)
<gfunction>   ::= sum | max | mim | count | aver | any | exist
      <qual>  ::= <conjunct> | <conjunct> and <qual>
  <conjunct>  ::= <cpredicate> | <cpredicate> or
                  <conjunct>
<cpredicate>  ::= <exp> <cop> <exp>
       <cop>  ::= < | <= | ¬= | >= | >
```

ここで，[[と]]は各々，記号" [" と "]"である。

get文は，検索を行う。例えば，関係 emp(eno, ename, …, dno), dept(dno, dname…) から成るデータベースに対して，dept に所属する従業員の名前，番号を求める検索コマンドは次の様である。

var (e, emp) (d, dept) ;

get [dname=d.dname ; eno=e.eno ; ename=e.ename] e.dno = d.dno ;

(3) del 文

del <var> [<qual>] ;

関係から組の消去を行う。

(4) app 文

app<relation> [[<target-list>]]
[<qual>] ;

関係に組の追加を行う。

(5) rep 文

rep <var> [[<target-list>]] [<qual>] ;

関係の組を置換する。

他に，以下の文がある。

(6) srt（整列）文

srt <relation> [on{<attribute>}]
[in <mode>] [to <relation>] ;
<mode> ::= D | A

srt 文は，関係を昇（A）順又は降（D）順に，整列する。

(7) prt（表示）文

prt <relation> [(<attfom>{, <attfom>})] ;
<attfom> ::=<attribute> [/[<type>] <length>]

prt 文は，関係を表示する。

(8) def 文

def <view> [[<target-list>]] [<qual>] ;

視野の定義を行う。

(9) drop 文

drop <view> {, <view >} ;

視野を消去する。

(10) mod 文

mod <relation> <mode> ;
<mode> ::= T | P

関係を一時的（T）又は，永続的（P）とする。検索結果は，一時的である

一つのセッションが終了すると消えてしまう。

8.4 最適化

Granadaでは，ＲＱＬ検索を，以下の目標値を達成する様な物理的Ｉ／Ｏ手順に変換する。

(I) 中間結果（ファイル）数の最少化

(II) 入出力ページＩ／Ｏ回数の最少化

ＲＱＬ問合せは，図8.3に示すグラフとして表せる。点は組変数，辺は結合式を表す。矢は目標属性を表す。このとき，全目標点（目標属性を持つ点）を

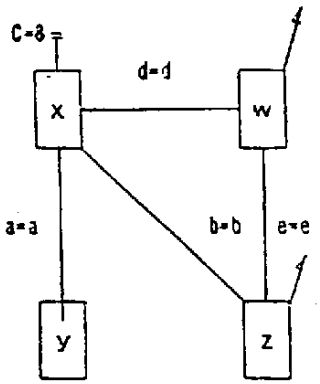


図 8.3

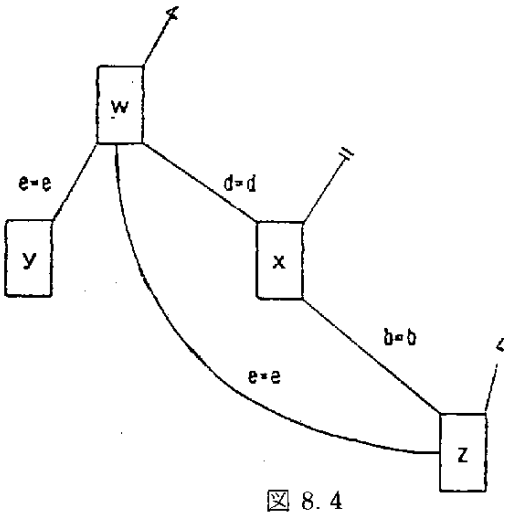


図 8.4

最右の枝にもつ木（巡航木）に変形する[図 8.4]。巡航木に対して，一つの出力ファイルに対して，ある一定順序で目標集合を出力出来ることは，文献[ＴＡＫＩＭ８４ａ]で知られている。Granadaでは可能な全ケースを調べて，入出力ページ数の最少の巡航木を求めている。一回の検索で参照する変数は，数個であると思われるので全面サーチを行っている。

8.5 DD/D

DD/Dは，以下の二重の関係から成る。

ＲＥＬ（rel-id, rel-name, deqree, …）

ＡＴＴ（rel-id, att-no, att-name, type, length, …）これらの関係は，通常の関係として操作できる。

8.6　Granadaの実現

Granadaは，PC－9800のMS／DOS下の応用プログラムとして，Lattice Cを用いて開発されている。Cで作成されている為に，他のシステム（Unix，CP／M，MS／DOS）への移植は容易である。図8.5に，Granadaの構成に示す。

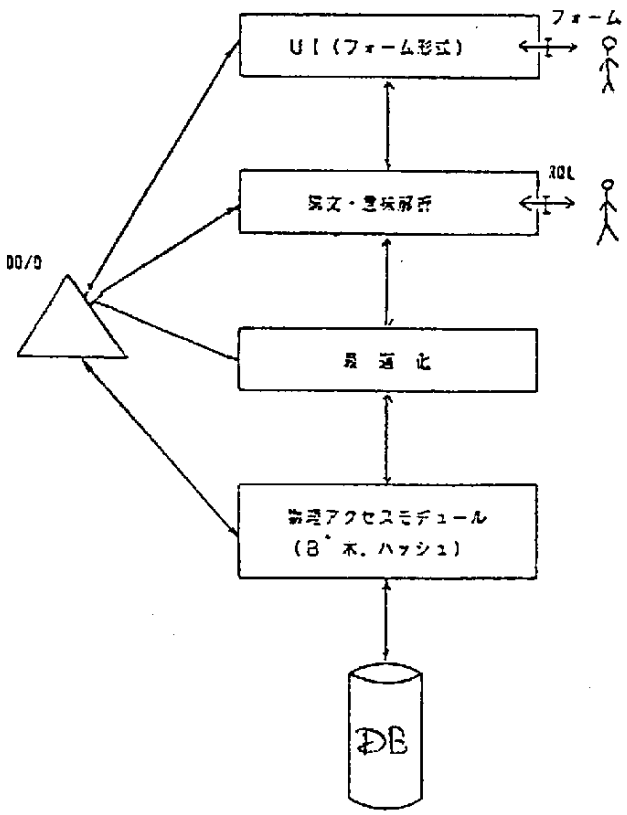


図8.5　Granadaの構成

8.7 Granada使用例

以下に，Granadaの使用例を示す。

A Granadaの起動

```
C>htest3 =4096
**********************************************************
*                                                        *
*                                                        *
*                                                        *
*                                                        *
*      G R A N A D A   ver. 1.0                          *
*                                                        *
*      C o p y r i g h t ( c )      J I P D E C          *
*                                                        *
*      R e l e a s e          9 9 . 9 9 . 9 9            *
*                                                        *
*                                                        *
*                                                        *
*                                                        *
**********************************************************
データベース名   :  prdbs
利用者名         :  yoko
ドライブ番号（終了＝ｚ）：c
ドライブ番号（終了＝ｚ）：z
作業ドライブ番号：  c
```

図8.6

図8.6は，Granadaを用いて，データベースPRDBS（ドライヴC内）を起動した例である。作業結果（検索の結果関係）も，ドライヴC内に求めめられる。

B PRDBS内の関係

図8.7は，PRDS内の関係を表している。PRDBSには以下の関係数がある。

```
EMPLOYEE (eno efname ename jname esex hyear
                byear  position ext dno)
PROJECT  (pno  pname ptype syear eyear status budget)
DEPT     (dno  dname)
PELNK    (pno  eno)
DELNK    (dno  eno)
```

```
#00010 var   dsp rel;
*---- 関係名   R E L   ----*
```

rel	rel_name	deg	c_date	card	width	c	n	s	h_ptr	t_ptr	pcd	pat	owner
20	employee	10	84/01/17	150	81	1	1	0	601	521	13	1	yoko
25	pelnk	2	84/01/18	67	10	0	0	0	2401	2467	1	0	yoko
24	delnk	2	84/01/18	0	4	0	0	0	65535	65535	0	0	yoko
22	dept	2	84/01/21	23	52	1	1	0	901	923	1	1	yoko
21	project	7	84/01/21	79	71	1	1	0	2201	2114	3	1	yoko

```
#00020 dsp att;
*---- 関係名   A T T   ----*
```

rel	an	att_name	tp	na	le	in	un	cl	ha	indpt	fptr	lptr	he	card	pcad
20	1	eno	5	1	2	1	0	1	0	4	4	4	1	140	1
20	2	efname	3	0	20	0	0	0	0	65535	65535	65535	0	0	0
20	3	ename	3	0	20	0	0	0	0	65535	65535	65535	0	0	0
20	4	jname	4	0	20	0	0	0	0	65535	65535	65535	0	0	0
20	5	esex	3	0	1	0	0	0	0	65535	65535	65535	0	0	0
20	6	hyear	5	0	2	0	0	0	0	65535	65535	65535	0	0	0
20	7	byear	5	0	2	0	0	0	0	65535	65535	65535	0	0	0
20	8	position	4	0	10	0	0	0	0	65535	65535	65535	0	0	0
20	9	ext	5	0	2	0	0	0	0	65535	65535	65535	0	0	0
20	10	dno	5	0	2	0	0	0	0	65535	65535	65535	0	0	0
25	1	pno	3	1	8	0	0	0	0	65535	65535	65535	0	0	0
25	2	eno	5	1	2	0	0	0	0	65535	65535	65535	0	0	0
24	1	dno	5	1	2	0	0	0	0	65535	65535	65535	0	0	0
24	2	eno	5	1	2	0	0	0	0	65535	65535	65535	0	0	0
22	1	dno	5	1	2	1	0	1	0	8	8	8	1	23	1
22	2	dname	4	0	40	0	0	0	0	65535	65535	65535	0	0	0
21	1	pno	3	1	8	1	0	1	0	20	20	20	1	45	1
21	2	pname	4	0	50	0	0	0	0	65535	65535	65535	0	0	0
21	3	ptype	3	0	4	0	0	0	0	65535	65535	65535	0	0	0
21	4	syear	5	0	2	0	0	0	0	65535	65535	65535	0	0	0
21	5	eyear	5	0	2	0	0	0	0	65535	65535	65535	0	0	0
21	6	status	3	0	3	0	0	0	0	65535	65535	65535	0	0	0
21	7	budget	5	0	2	0	0	0	0	65535	65535	65535	0	0	0

図 8.7　DD/D

C　検索例

図8.8は，PRDBSに対する検索例を表している。＃３０の var 文は，EMPLOYEE 関係に対して，組変数 e を定めている。検索演算「30才より若い職員の名前（日本語）と年齢を求めよ」をＲＱＬで表した例が＃４０である。＃５０は，このＲＱＬ問合せの実行である。＃６０は，問合せ結果を，年数で昇順に整列させ，整列結果をTEMP1とすることを表している。＃又０は，整列結果TEMP1の表示を表している。この時，jname は一行１０文字（２０Ｂ），age は５文字で出力することとしている。

```
#00030 var ( e, employee );
#00040 get   temp [e.jname; age = 83 - e.byear ] 83 - e.byear < 30;
#00050 exe;
#00060 srt temp on age to temp1;
#00070 prt temp1 ( jname/10, age/5 );
```

jname	age
西野　佐和子	23
鳥居　鉱太郎	23
佐藤　公子	23
香　礼子	23
林　美千予	23
堤内　晶子	24
桑原　由美	24
高畠　綾子	24
成海　洋	24
荒木　朋代	24
星　昌宏	24
水島　まゆみ	24
松浦　早苗	25
津田　芳伸	25
亀田　光子	25
伊藤　寿一	26
飯田　修久	26
檜爪　修	26
土川　潤	26
福井　寛隆	26
三宅　隆裕	27
稲垣　治子	27
中村　貞子	27
竹内　英二	27
黒田　学	28
野口　守	28

図 8.8　検　索　例

図 8.9 のは，検索演算「開発部開発第２課の従業員番号と名前を求めよ」のＲＱＬ表現とその実行結果を表している。＃１４０は，更に検索結果をenoで整列させて，＃１５０で結果を出力させている。

```
#00080 var ( d, dept );
#00090 get temp2
>00100 [ d.dname; e.eno; e.jname ]
>00110 d.dno   = e.dno        and
>00120 d.dname   = '開発部開発第二課';
#00130 exe;
#00140 srt temp2 on eno    to temp3;
#00150 prt temp3 ( dname/20, eno/5, jname/10 );
```

dname	eno	jname
開発部開発第二課	86	加藤　泰宏
開発部開発第二課	109	佐竹　紀男
開発部開発第二課	116	東　吉郎
開発部開発第二課	152	浜中　栄治
開発部開発第二課	181	日高　哲郎
開発部開発第二課	244	川瀬　博光
開発部開発第二課	247	妙泉　正隆
開発部開発第二課	250	三木　良治
開発部開発第二課	253	片岡　幸一
開発部開発第二課	255	滝沢　誠
開発部開発第二課	270	小池　博之
開発部開発第二課	272	横塚　実
開発部開発第二課	324	星　昌宏
開発部開発第二課	325	三宅　隆裕
開発部開発第二課	348	飯田　修久

図 8.9　検索例（結合）

図8.10を説明する。＃160のels　文は，今までの実行文を表示している。＃170のedl　文は，＃40のget文を消去することを表している。＃180のein文は，＃115にe.byear＞48の条件式を行ごと追加することを表している。＃200のerp　文は，＃90のget Temp 2をget Temp 4で置換することを表している。＃210は，以上の様に修正した＃90のget　文の実行を示している。

```
#00160 els;
#    40:get temp [e.jname; age = 83 - e.byear ] 83 - e.byear < 30;
#    70:prt temp1 ( jname/10, age/5 );
#    90:get temp2
#   100:[ d.dname; e.eno; e.jname ]
#   110:d.dno   = e.dno       and
#   120:d.dname   = '開発部開発第二課';
#   150:prt temp3 ( dname/20, eno/5, jname/10 );
#00170 edl 40;
#00180 ein 115;
 00115 e.byear  > 48        and
#00190 els;
#    70:prt temp1 ( jname/10, age/5 );
#    90:get temp2
#   100:[ d.dname; e.eno; e.jname ]
#   110:d.dno   = e.dno       and
#   115:e.byear  > 48         and
#   120:d.dname   = '開発部開発第二課';

#00200 erp 90;
    90 get temp2
 00090 get temp4
#00210 exe 90;
**  GBC END    # of used nodes =    65
**  GBC END    # of free nodes =    36

#00220 prt temp4 ( dname/20, eno/5, jname/10 );
+--------------------+----+--------------+
|dname               |eno |jname         |
+--------------------+----+--------------+
|開発部開発第二課    | 152|浜中　栄治   |
+--------------------+----+--------------+
|開発部開発第二課    | 181|日高　哲郎   |
+--------------------+----+--------------+
|開発部開発第二課    | 244|川瀬　博光   |
+--------------------+----+--------------+
|開発部開発第二課    | 247|妙泉　正隆   |
+--------------------+----+--------------+
|開発部開発第二課    | 250|三木　良治   |
+--------------------+----+--------------+
|開発部開発第二課    | 253|片岡　幸一   |
+--------------------+----+--------------+
|開発部開発第二課    | 255|滝沢　誠     |
+--------------------+----+--------------+
|開発部開発第二課    | 270|小池　博之   |
+--------------------+----+--------------+
|開発部開発第二課    | 272|横塚　実     |
+--------------------+----+--------------+
|開発部開発第二課    | 324|皇　昌宏     |
+--------------------+----+--------------+
|開発部開発第二課    | 325|三宅　隆裕   |
+--------------------+----+--------------+
|開発部開発第二課    | 348|飯田　修久   |
+--------------------+----+--------------+
#00230 end;
```

GRANADA終了

図8.10　修正と実行

D フォームインタフェース

Granadaは,文字列を入力することに加えて，フォームインタフェースを有している。図8.11はフォームの形式の定義例である。図8.12はフォームに対する入力データ例である。図8.13は図8.11と8.12より導出されたフォームである。

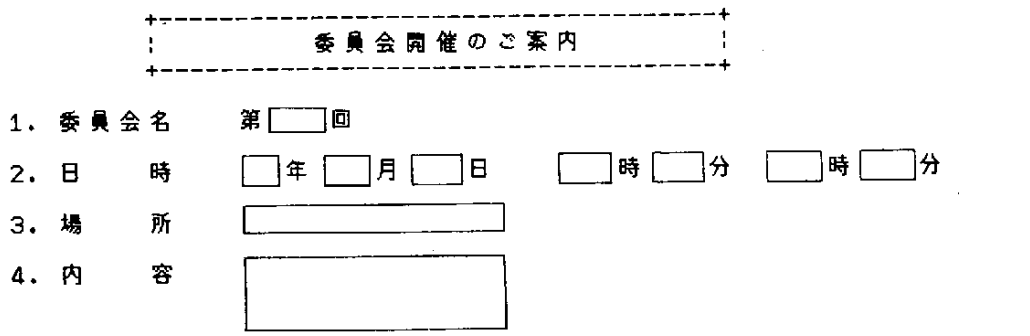


図8.11 フォームの定義例

```
#00070 prt temp ( no, title/20, yy,mm,dd );
+--+--------------------------------------+--+--+--+
|no|title                                 |yy|mm|dd|
+--+--------------------------------------+--+--+--+
| 1|分散型データベース調査研究委員会      |84| 3| 9|
+--+--------------------------------------+--+--+--+
#00080 prt temp (mm1,ss,mm2,ss2,place/10,gidai/10 );
+--+--+--+--+--------------------+--------------------+
|mm|ss|mm|ss|place               |gidai               |
+--+--+--+--+--------------------+--------------------+
|  |  |  |  |当協会第二会議室    |分散型データベースに|
|  |  |  |  |                    |おける放送通信機能に|
|  |  |  |  |                    |ついて              |
|10| 0|16| 0|                    |                    |
+--+--+--+--+--------------------+--------------------+
```

図 8.12　データ例

```
+------------------------------------+
|          委員会開催のご案内       |
+------------------------------------+
```

1. 委員会名　　第 1 回　分散型データベース調査研究委員会
2. 日　　時　　84 年 3 月 9 日　　10 時 0 分　16 時 0 分
3. 場　　所　　当協会第二会議室
4. 内　　容　　分散型データベースにおける放送通信機能について

連絡先

日本情報処理開発協会 分散情報システム開発室

担当　横塚

TEL（03）434－8211 内線 213

次のデータ　前のデータ　最始　最後　終了

図 8.13

9. ローカル情報システム（ＬＩＳ）の機器構成

本章では，以上述べてきたローカル情報システム（ＬＩＳ）の機器構成について述べる。

9.1 ＬＡＮ－Net/one（アンガコンパス社）

ＬＩＳは，ローカルエリア網（ＬＡＮ）に結合された超小型計算機と大型機から成る。ＬＡＮとしては，以下の理由で，アンガマンバス社のNet/one を採用した。

(1) 異種のメーカの機器の系統が容易である。

(2) Ethernetのデータリング（ＥＤＬ）サービスを提供する。

(3) E-DL層より上位のプロトコルを，利用者がNet/one 内に組み込める。

(4) （仮想回線サービス，ネーミングサービス等の上位プロトコル）を提供している。Net/oneは，１０Mbpsの同軸ケーブルとＮＩＵ（network interface Unit)とから成る。NIUには，NIU-150AとNIU-2Aとの二種があり，前者はZ80と64KB主記憶のボード一枚，後者は四枚から成る。各ボードには，種々の入出力ポートをつけられる。（例，シリアル６ポート，シリアル４ポートとパラレル２ポート，ＧＰＩＢ ポート）。通信手順（例，ＢＳＣ，ＨＤＬＣ）は，各ボードに，ダウンロード時に，それ用の制御プログラムをロードすればよい。図9.1に，現在のNet/one の構成を示す。

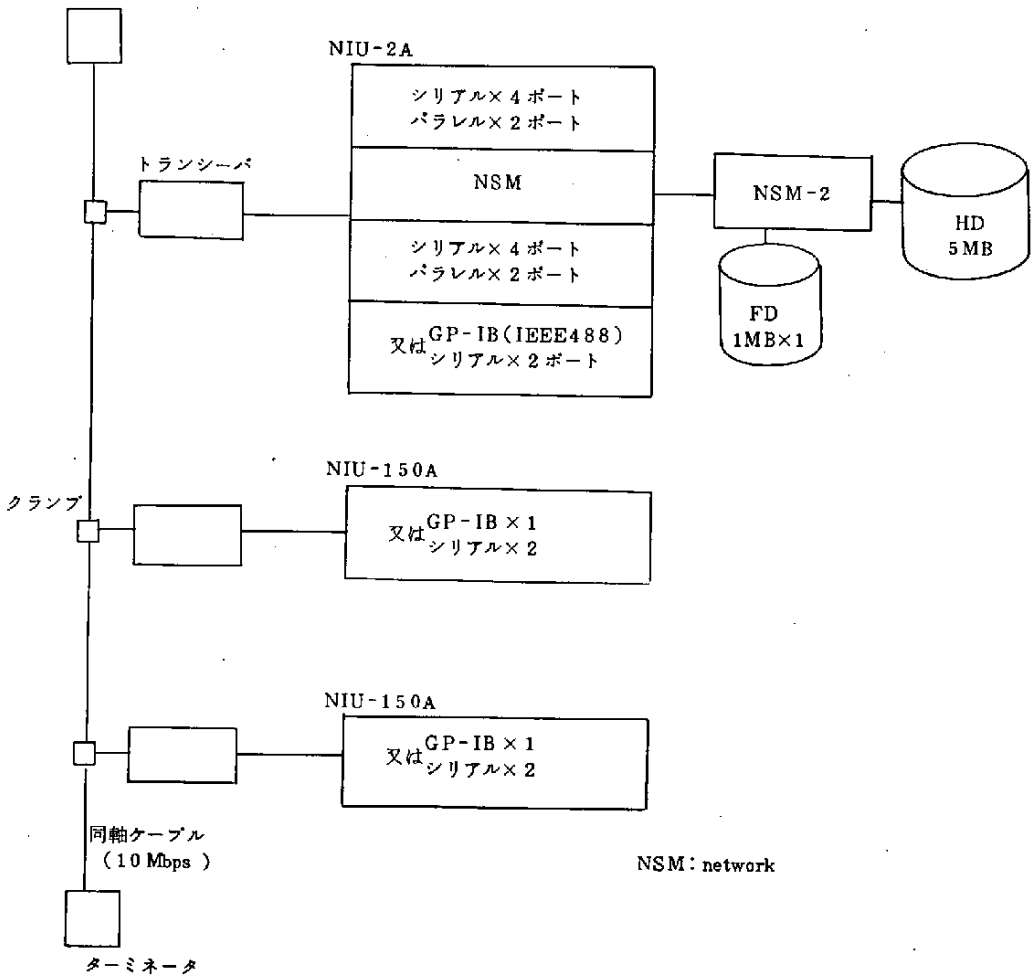


図 9.1　Net/one の構成

9.2 超小型計算機

ＬＩＳは，超小型計算機として，以下のものを用いる。

(1) Sun 2／１２０
（Sun マイクロシステム会社）
〔伊藤忠データシステム〕

ＣＰＵ	68010
ＭＥＭ	２ＭＢ
テープ	¼ inch
Ｄｉｓｋ	１６８ＭＢ（フォーマット １３０ＭＢ）
Ｄｉｓｐｌａｙ	1152×900 モノクロビットマップディスプレイ
ＯＳ	Ｕｎｉｘ ４.２ ｂｓｄ
Ｐｒｉｎｔｅｒ	デージーホイール
ソフト	Multi-window Sun core graphics, C, Pascal, Fortran, 68010マクロアンブラ

(2) U－１２００
（富士通）

ＣＰＵ	（１６ bits）
ＭＥＭ	１ＭＢ
テープ	¼ inch
Ｄｉｓｋ	１３０ＭＢ
ポート	BSC×1
	ｔｔｙ×４（うちF9450Ⅱ×２）
Ｐｒｉｎｔｅｒ	ＬＰ×１
ＯＳ	Ｕｎｉｃｕｓ(Ⅲ＋４.１ ｂｓｄ)
ソフト	Ｃ
ＤＢＭＳ	Ｉｎｇｒｅｓ

9.3　全体構成

大型計算機としては，M－１７０Ｆを用いる。図9.2には，ＬＩＳの機器構成を与える。

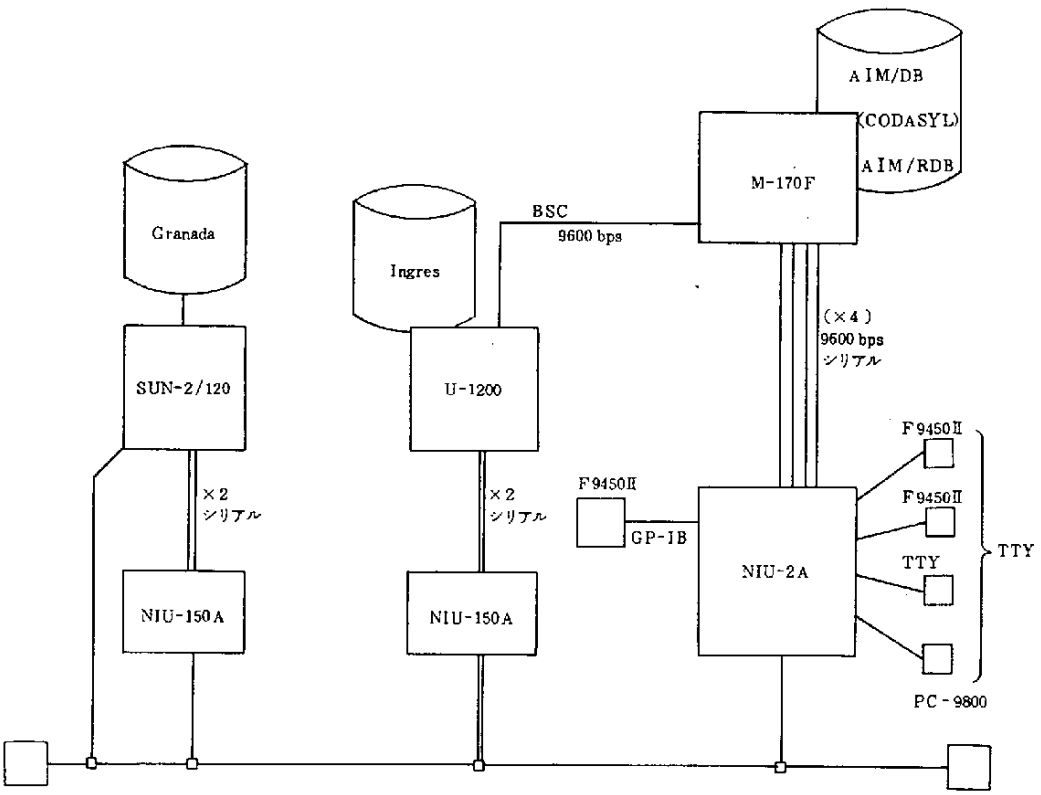


図 9.2　ＬＩＳの機器構成

10. おわりに

本年度は，ＬＩＳを実現する為に，(i)ＬＩＳの全体モデル，(ii)ＬＩＳの通信処理の制御システム，全体データベースプロセッサＧＤＰの基本設計，及び(iii)超小型機用の関係ＤＢＭＳ Granadaのプロトタイプ開発を行った。ＬＩＳの全体モデルでは，従来の大型機に対して，ＬＡＮで新たに結合された超小型計算機の位置づけを検討した。超小型機は，従来の端末に対して，単にデータの表示，入力だけでなく，他のＤＢＳ内のデータを導出してきて蓄積し，操作する為に用いられる。各超小型機内では，導出されたデータはオブジェクトとして，これに許される抽象的な操作演算を通してのみ利用される。超小型機内のオブジェクトと，元のオブジェクト間の一致進は，必要ならば得たれねばならない。

以上の為に，ＬＩＳの制御システムＧＤＰの通信処理階層を考察した。ＧＤＰは，(i)基本通信（ＦＣ），(ii)ＤＤＢＳ処理（ＤＤ），(iii)視野処理（ＶＰ）の三つの階層から成る。ＦＣ層は，Ethernetデータリンク（ＥＤＬ）サービスを用いて，各ＤＢＳ（仮想ＤＢＳとする）間の多対多の通信を提供する。これを群サービスとする。ＤＤ層は，ＦＣ層の群サービスを用いて，上位のＶＰ層に，データの所在と独立なデータ操作（検索，更新，トランザクション）を提供する。ＶＰ層は，利用者に，抽象データ型としての視野を提供する。ＤＤ層の分散問合せ処理，分散トランザクション処理，及び視野と元のデータ構造との一致性保持の為の処理の為の放送通信に基づいたプロトコルを与えた。

超小型機の関係ＤＢＭＳとしてのGranadaを，ＰＣ-９８００上にそのプロトタイプを完成させた。従来のミニコン並みの非手続的なデータ操作を提供できるＤＢＭＳである。

来年度は，本年度の成果を生かして，ＧＤＰ及び関係ＤＢＭＳの本格的な実現を行う予定である。

参考文献

〔ABRAN 70〕 Abramson, N., "ALOHA System-Another Alternatives for Computer Communications, "AFIPS Conf. Proc., 1980, pp. 281-285.

〔ADIBM 80〕 Adiba, M., et al., "An Overview of the Polypheme Distributed Management System," Proc. of the IFIP '80, 1980, pp. 475-479.

〔BERNP 81〕 Bernstein, P. A., "Using Semi-Joins to Solve Relational Queries," JACM, Vol. 28, No. 1, 1981, pp. 25-40.

〔BILLH 76〕 Biller, H., et al., "On the Semantics of Data Manipulation Language,"
Proc. of the IFIP TC-2, 1976, pp. 239-267.

〔CHAMI 75〕 Chambelin, D, D., et al., "Views, Authorization, and Locking in a Relational Data Base System,"
AFIPS Conf. Proc., Vol. 44, 1975, pp. 425-430.

〔CHANA 83〕 Chan, A., et al., "Overview of an Ada Compatible Distributed Database Manager," Proc. of the ACM SIGMOD '83, 1983, pp. 228-237.

〔CHENP 76〕 Chen, .P, -S., "The Entity-Relationship Model-Toward a Unified View of Data, "ACM TODS, Vol. 1, 1976, pp. 9-36.

〔CODAS 73〕 Codasyl DDL Committee, "CODASYL Data Description Language," Journal of Development, 1973.

〔CODAS 78〕 Codasyl DDL Committee, "Report of the CODASYL Data Definition Language Committee,"
Information Systems, Vol. 3, No. 4, 1978, pp. 247-320.

〔CODDE 70〕 Codd, E. F., "A Relational Model of Data for Large Shared Data Bank,"
CACM, Vol. 13, No. 6, 1970, pp. 337-387.

〔CODDE 82〕 Codd, E, F., "Relational Databases : A Practical Foundation for Productivity," CACM, Vol.25, pp. 109-117.

〔DATEC 81〕 Date, C.J., "Introduction to Database Systems," (2nd. ed.), Addison-Weley, 1981.

〔DAYAU 78〕 Dayal, U., and Bernstein, P, A., "On the Updatability of Relational View," Proc. of the 4th VLDB, 1978, pp. 368-377.

〔DAYAU 82〕 Dayal, U., et al., "Query Optimization for CODASYL Database Systems," Proc. of the ACM SIGMOD, 1982, pp. 138-150.

〔GOODN 79〕 Goodman, N., et al., "Query Processing in SDD-1 : A System for Distributed Databasc, "CCA-79 - 06, CCA, 1979, pp. 1-57.

〔GOUDM 81〕 Gouda, M., et al., "Optimal Semi-Join Schedule for Query Processi in Local Distributed Database Systems,"
Proc. of the ACM SIGMOD, 1981, pp. 164-175.

〔GRAYJ 81〕 Gray, J., "Notes on Database Operating Systems," Operating Systems : Advanced Cource, Springer-Verlac, 1981.

〔HELDG 76〕 Held, G., et al., "INGRES-A Relational Database Systems," AFIPS Conf. Proc., 1976, pp. 409-416.

〔HEVNA 78〕 Hevner, A., et al., "Query Processing on a Distributed Databases," Proc. of the 3rd Berkeley Workshop, 1978, pp. 91-107.

〔HOPCJ 69〕 Hopcroft, J, E., and Ullman, J.P., "Fornal Languages and Their Relation to Automaton," Addison -Wesley, 1969.

〔KAMBY 81〕 Kambayashi, Y., "圧縮型準結合による分散型データベースシステムの質問処理," 電子通信学会AL 81-54, 1981.

〔KNUTD 73〕 Knuth, D., "Sorting and Searching,"
The Art of Computer Programming, Vol. 3, Addison -Wesley, 1973.

〔LEBIJ 80〕 Le Bihan, J., et al., "SIRIUS : A French Nationwide Project on Distributed Data Base," Proc.of the

6th VLDB, 1980.

〔METCR 76〕 Metcalfe, R. M., et al., "ETHERNET Distributed Packet Switching for Local Commputer Networks," CACM, Vol. 9, 1976, pp. 395-404.

〔NOGUS 83〕 Noguchi, S., et al., "情報ネットワークの理論," 岩波講座 情報科学 Vol. 5, 1983.

〔OHSUS 83〕 Ohsuga, S., "CAD/CAMと情報処理," bit (共立出版), 1983.

〔ONUEE 83〕 Onuege, E., et al., "Local Query Translation and Optimization in a Distributed System," AFIPS Conf. Proc., 1983, pp. 229-239.

〔ROTHJ 80〕 Rothnie, J. B., et al., "Introduction to a System for Distributed Databases (SDD-1)," ACM TODS, Vol. 5, 1980, pp. 1-17.

〔SMITJ 77〕 Smith, J, M., et al., "Data Abstraction : Aggregation and Generaliazation, " ACM TODS, Vol. 2, 1977, pp. 105-133.

〔SMITJ 81〕 Smith, J. M., et al., "Multibase-Integrating Heterogeneous Distributed Database Systems," AFIPS Conf. Proc., 1981, pp. 487-499.

〔SUZUK 79〕 Suzuki, k., et al., "DEIMS-2について, " IPSJ DBMS 研 11-3, 1979.

〔TAKIM 78〕 Takizawa, M., et al., "Resource Integration and Sharing on Heterogeneous Resource Sharing Systems," Proc. of the ICCC'78, 1978, pp. 253-258.

〔TAKIM 79〕 Takizawa, M., and Hamanaka, E. "The Four-schema Concept as Gross Architecture of Distributed Databases and Heterogeneous Problems," JIP(IPSJ), Vol. 2, No. 3, 1979, pp. 133-142.

〔TAKIM 80〕 Takizawa, M., and Hamanaka, E., "Query Translation in Distributed Databases, " Proc. of the IFIP' 80, 1980, pp. 451-456.

〔TAKIM 81〕 Takizawa, M., "分散型データベースシステムJDDBS-IIの通信処理", 電子通信学会AL81-22, 1981.

〔TAKIM 82a〕 Takizawa, M., "Distribution Problems in Distributed Database," JIP (IPSJ), Vol. 5, No. 3, 1982, pp. 139-147.

〔TAKIM 82b〕 Takizawa, M., Yokotsuka, M., et al., "CODASYLデータベースシステムに対する関係インタフェースシステム(LDP-V 1.5)の設計と実現," 情報処理学会 論文誌, Vol. 23, No. 3, 1982, pp. 665-675.

〔TAKIM 82c〕 Takizawa, M., "放送通信機能を用いた分散型データベースシステムの通信処理の実現について," 情報処理学会 DBMS研31, 1982.

〔TAKIM 82d〕 Takizawa, M and Yokotsuka, M. "ローカルデータベースプロセッサ(LDP)の評価," データセマンティクスの理論と実際に関する研究, 京大数理解析研講究録461, 1982, pp. 127-150.

〔TAKIM 83a〕 Takizawa, M. and Noguchi, S., "CODASYLデータベースシステムに対する非手続的更新インタフェースの基本概念," 情報処理学会 論文誌, Vol. 24, No. 6, 1983, pp.

〔TAKIM 83b〕 Takizawa, M., "Distrbuted Database System-JDDBS," JARECT Vol. 7, Computer Science and Technologies (Kitagawa, T., ed.), Ohmsha and North-Holland, 1983, pp. 262-283.

〔TAKIM 83c〕 Takizawa, M., Yokotsuka, M., and Noguchi, S., "分散型データベースシステムに於ける放送通信を用いた通信処理方式," 情報処理学会「ローカルエリアネットワーク」シンポジウム報告書, 1983, pp. 211-218.

〔TAKIM 84a〕 Takizawa, M. and Noguchi, S., "CODASYL DML上の非手続グラフ問合せの設計と実現," IPSJ論文誌, 1989.

〔TAKIM 84b〕 Takizawa, M., and Noguchi, S., "放送通信を用いた分散型データベースシステムの統合化手法," 投稿中, 1984.

〔TOANN 81〕 Toan, N. G., "Distributed Query Management for a Local Network Database Systems, 1981, pp. 188-196.

〔VASSY 80〕 Vassiliou,Y., et al,"DBMS Transaction Translation," Proc. of the COMPSAC80, 1980, pp.86-96.

〔WILMP 83〕 Wilmes,P, F.,et al., "I wish I were over there: Distributed Execution Protocols for Data Definition in R*," Proc. of the ACM SIGMOD, 1983, pp. 238-242.

〔WONGE 77〕 Wong, E., "Retrieving Dispersed Data from SDD-1 :A System for Dispersed Databases, "Proc. of the 2nd Berkeley Workshop, 1977, pp. 217-235.

〔YAMAH 82〕 Yamazaki, H., et al., "A Proposal for Broadcast Architecture Network (BANET)," Proc. of the 1CCC' 82, 1982.

〔YAO S 76〕 Yao, S. B., "Attribute-based Model,, "ACM TODS, 1979.

〔ZANIC 79〕 Zaniolo, C., "Design of Relational View over Network Schemas," Proc. of the ACM SIGMOD, 1979, pp. 179-190.

〔DIX 80〕 Dec. Intel, Xerox, "The Ethernet:A Local Area Network:Data Link Layer and Physical Layer Specification," Version 1.0, Sept. 1980.

〔SVOBL 79〕 Svobodova, L., Liskov, B., Clark, D., "Distributed Computer System: Structure and Semantics" MIT/LCS/TR-215, LCS, MIT, 1979.

〔LISkB77〕 Liskov. B., et al., "Abstraction Mechanisms in CLU, " CACM, Vol. 20, No. 8, Aug 1977, pp. 564-576.

付　記

分散型データベースシステムにおける同時実行制御

概　要

データベースシステム（ＤＢＳ）の同時実行制御は主に，

(1)　多数の利用者に対して同時にデータベースを使える環境を与え，

(2)　さらに，その時，利用者に対して，他の利用者を意識することなく，自分一人がシステムを扱っているかのように思わせ，

(3)　しかも，システムが無矛盾の状態を保つ

ことが定義であり，目的である。(3)はデータベースが集中化されていれば，比較的，解決しやすいが，データベースが分散化されている場合，いわゆる分散型データベースシステム（ＤＤＢＳ）に関しては難かしい。理由としては，

(1)　利用者がＤＤＢＳの多くの計算機内のデータをアクセスすること

(2)　同時実行制御機構も各計算機ごとに分散化され，各々の同時実行制御機構同志の通信が容易でない。

ことがあげられる。集中化データベースシステムにおける同時実行制御の問題は大分理解されており，標準的な解決法として，２相ロックがある。これは数学的にもかなり解析されており，最適化などの理論も展開されている。これに対して，分散型の場合は，多くの同時実行制御アルゴリズムが提案されているが，決め手となるものが，今だに存在しない状態である。そこで，ここでは主流となっているアルゴリズムを概説し，考察を加えることにする。アルゴリズムを解説するにあたっては，ＤＤＢＳの環境などの諸定義から始め，アルゴリズムの正当性を証明するための解析理論について述べ，各アルゴリズムの内容を説明する。

1.　諸　定　義

分散型データベースシステム（ＤＤＢＳ）の定義としてトランザクション処理モデル〔Bernstein ８１〕を用いる。

分散データベースシステム（ＤＤＢＳ）とは，

網でつながったサイトの集合体

として定義する。ここでサイトとは

(1)　トランザクション管理（ＴＭ）

利用者の要求をトランザクションとして管理し，スケジューラ間とやりとりを行なう。

(2)　スケジューラ

ＴＭから送られた命令の順番を調整し，ＤＭに対して命令，データ，確認等のやりとりを行なう。

(3) データ管理（ＤＭ）

実際のデータベースを管理する。

の３つのモジュールが動く計算機と定義される。網とは，このサイトーサイト間の通信システムのことである。

利用者がＤＤＢＳとやりとりするにはトランザクションを用いる。これは特別に用意されたデータ操作言語で書かれたもの，または汎用のプログラム言語で書かれた，オンラインのデータ検索，又はデータ更新プログラムである。利用者が見たデータベースは論理データ要素（Ｘ，Ｙ，Ｚで表わす）の集合体である。論理データベース状態とは各論理データ要素の値の割当て状態である。論理データ要素は実際には１つのサイト，又は冗長に複数のサイトにまたがって貯えられたデータを格納データ要素，又はデータ要素（データ）と呼ぶ。Ｘのデータをx_1……，x_mと書き，通常は適当な１つを指してxと書く。格納データベース状態とはデータベース状態とはデータベース中のデータの値の割当て状態のことである。

以上述べてきたようにＤＤＢＳは５つの要素，トランザクション，ＴＭ，スケジューラ，ＤＭ，データにより構成される（図1.1）。次にＤＤＢＳの内部の動きを述べる。

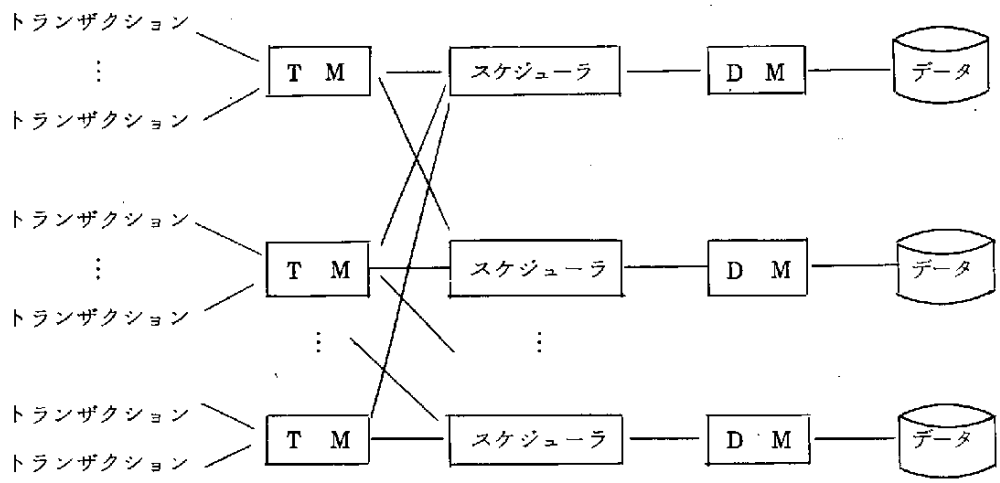


図1.1　分散データベースシステム

トランザクションは，理想的な形として，始まりにはBEGIN，終わりにはENDをおき，間にはRead，Writeの列を含むものとして表される。TMはトランザクションを受けると，スケジューラにRead，Writeを送る（アルゴリズムによってはBEGIN，ENDも送る）。スケジューラは同時実行制御アルゴリズムに従って動く。すなわち，DMに送るRead，Writeの列の送出を制御し，すぐに出力するか，それを保って，遅らせるか，又は取り消す。ここで大事なことは，(1)トランザクションはシステムが無矛盾の状態で始まり，終了した後も無矛盾性を保つこと（無矛盾性）と，(2)取り消された命令を発行したトランザクションは全体が取り消されること（原子性）である。さらに取り消されたトランザクションが書いた値を読んだトランザクションも取り消されることがある。この現象を並列取り消しという。DMはスケジューラからRead，Writeを受けると，すぐに実行する。Readの場合は自サイトのデータベース内を見に行って要求のあった値を返す。Writeの場合は更新して，確認を返す。DMがスケジューラに値や確認を返すとスケジューラはTMに継ぎ渡す。DMは命令の順番に一切関知しない。スケジューラとDM間のやりとりはハンドシェイキングを用いる。例えばスケジューラがRead(X)の後にWrite(X)を行ないたければ，まずRead(X)をDMに送り，DMの応答を持ち，それからWrite(X)を送る（図1.2）。

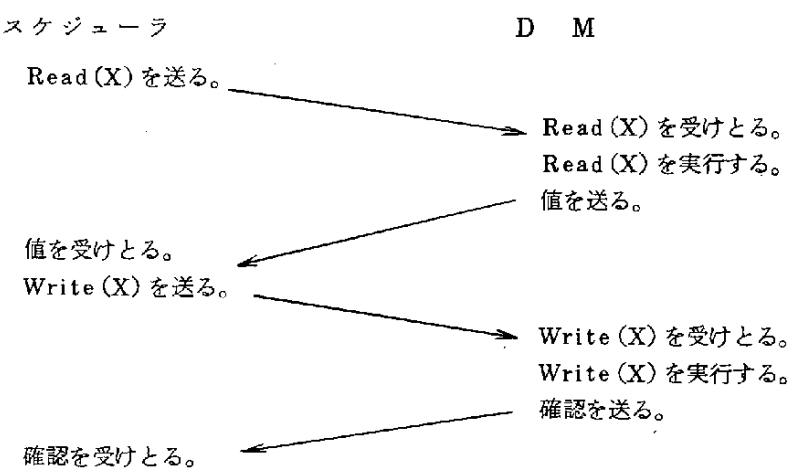


図1.2　スケジューラーDM間のハンドシェイキング

2. 同時実行制御の正当性

同時実行制御のアルゴリズムの正当性を解析するものとしてシリアライザビリティ（直列性）理論がある〔Casanova 79〕〔Bernstein 82〕。

2.1 ログ

シリアライザビリティ理論では各トランザクションのふるまいを記述したRead，Write操作の列であるログを用いる。トランザクションログとは単一トランザクションの許される実行を記述したものである。形式的には半順序集合（poset）で，$T_i = (\Sigma_i, <_i)$ で表わされる。ここで Σ_i はトランザクション i で発行されるRead，Writeの集合であり，$<_i$ はその操作が実行される順番を表すことにして，ダイアグラムとして表せる。

例えば

$$T_1 = \begin{matrix} r_1[x] \\ r_1[z] \end{matrix} \searrow\nearrow \; w_1[x]$$

T_1 はデータ x と z を同時に読み込み，その後 x を書くトランザクションを表わしている。

トランザクション T_i により発行された読み込み（書き込み）を $r_i[x]$（$w_i[x]$）と書く。

以下，ある1つのデータに対しての読み込みと書き込みはトランザクションに対して高々1回という仮定を設ける。Tをトランザクションログの集合 $\{T_0, \cdots, T_n\}$ とする。T上のDBSログ（又は単にログ）とは $T_0, \cdots, T_n$ の実行を表わしたもので，半順序集合 $L = (\Sigma, <)$ である。ここで

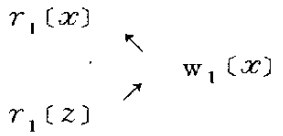
(1) $\Sigma = \cup_{i=0}^{n} \Sigma_i$　　(2) $< \supseteq \cup_{i=0}^{n} <_i$

(1)は $T_0, \cdots, T_n$ での操作を過不足なく実行することを表わしており，(2)は各ログの操作の順序関係を守ることを意味している。例えば2.1節で述べた T_1 の上での可能なログは

(1) $r_1[x] \downarrow r_1[z] \searrow\nearrow w_1[x]$　　(2) $r_1[x] \uparrow r_1[z] \searrow\nearrow w_1[x]$

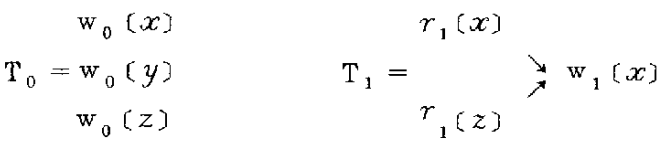
(3) $r_1[x] \searrow\nearrow r_1[z] \; w_1[x]$　　$(< = <_i)$

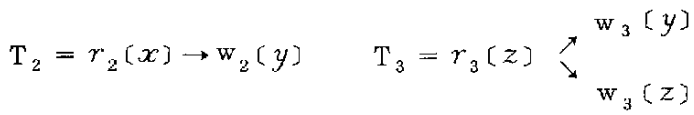
である。ここで注意する点は必ずしもＤＤＢＳが$r_1[x]$，$r_1[z]$を並列に行なう必要がないことである。しかし，順番を逆にしたり，削除したりすることはできない。例えば次のようなものは許されない。

(4)


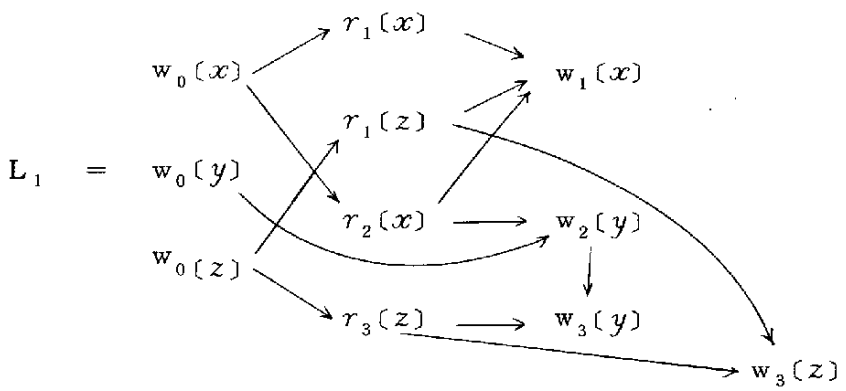
ログの上でさらに問題となるのは競合である。同じデータに対しての操作が２つあり，少なくとも１つがWriteの場合，これらの操作は競合しているという。さらに競合はReadとWriteが競合するrw競合とWriteとWriteが競合するww競合がある。競合している操作に対しては順序づける必要がある。これはＤＢＳログだけでなくトランザクションログにも適用される。

次のトランザクションログが与えられた時





T_0，…，T_3上のログの例として



が考えられる。（推移性はダイアグラムに書かない。例えば$w_0[y] \to w_3[y]$ etc）

2.2 ログの等価性

ログの等価性はreads－from関係とfinal－writesを用いることにより定義される。ログLにおいて$r_j(x)$が$w_i(x)$に対してreads－from関係にあるというのは，$w_i(x) < r_j(x)$であり，$w_i(x)$と$r_j(x)$の間に$w_k(x)$がないことをいう。例えば

$w_0(x) \rightarrow r_1(x) \rightarrow w_2(x) \rightarrow r_3(x) \rightarrow r_4(x)$において$r_1(x)$は$w_0(x)$に対してreads－from関係にあり，$r_3(x)$と$r_4(x)$は$w_2(x)$に対してreads－from関係にある。

ログにおいて$w_k(x)$以後xを書くWriteがない場合$w_k(x)$をfinal－writeという。final－writeは各データに対して存在する。すべてのfinal－writeを集めた集合をfinal－writesという。例えば

$$w_0(x) \rightarrow w_1(x) \rightarrow w_2(y) \rightarrow r_2(y)$$

においてfinal－writesは$w_1(x)$と$w_2(y)$である。

2つのログが

(1) 各ログとも同じreads－from関係の集合をもつ。

(2) 各ログとも同じfinal－writesをもつ。

の時，2つのログは等価であるという。例えば，次のログL_2は2.1節のログL_1と等価である。

$$L_2 = w_0(x) \rightarrow w_0(y) \rightarrow w_0(z) \rightarrow r_2(x) \rightarrow w_2(y) \rightarrow r_1(x) \rightarrow r_1(z) \rightarrow w_1(x) \rightarrow r_3(z) \rightarrow w_3(y) \rightarrow w_3(z)$$

2.3 正しいログの基準

シリアルログとはログのすべての各トランザクションの組T_i, T_jに対してT_iの操作がT_jのすべての操作の前にくる（又はその逆）ような場合をいう。例えば2.2節のログL_2はシリアルログである。つまり，あるトランザクションが終了しなければ別のトランザクションが始まることができないようなものである。これは正しい実行の基準となる。（各トランザクションの前後でシステム状態の無矛盾性が仮定されているので）したがってログが正しいことの基準はそのログと等価なシリアルログが存在することである。このようなログはシリアライザブル（ＳＲ）であるといい，シリアライザブルなログをシリアライザブルログ（ＳＲ－ログ）という。例えば2.2節のログL_2と等価な2.1節のログL_1はＳＲ－ログである。

2.4 シリアライザビリティの定理

ＳＲーログはシリアルログと等価なものと定義されたが，ここではあるログがシリアライザブルかどうかを有向グラフを用いて解析する手段を説明する。Ｌを｛T_0, …, T_n｝上のログとして，Ｌの

シリアライゼイショングラフＳＧ（Ｌ）とは，

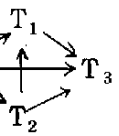
T_0, …, T_n をノードとし，$T_i \rightarrow T_j$ をエッジとする有向グラフである。ここで $T_i \rightarrow T_j$ とは，ある x に対して

(i) $r_i[x] < w_j[x]$ 又は

(ii) $w_i[x] < r_j[x]$ 又は

(iii) $w_i[x] < w_j[x]$

の関係が１つでもある場合である。例えば2.1節のログ L_1 のシリアライゼーショングラフＳＧ（L_1）は

$SG(L_1) =$ 

$T_0 \rightarrow T_1$ は $w_0[x] < r_1[x]$　　$T_2 \rightarrow T_1$ は $r_2[x] < w_1[x]$

$T_2 \rightarrow T_3$ は $w_2[y] < w_3[y]$　　$T_0 \rightarrow T_2$ は $w_0[x] < r_2[x]$

$T_1 \rightarrow T_3$ は $r_1[z] < w_3[z]$　　$T_0 \rightarrow T_3$ は $w_0[z] < r_3[z]$

シリアライザビリティの定理とは

ログＬのＳＧ（Ｌ）がサイクリックでない時

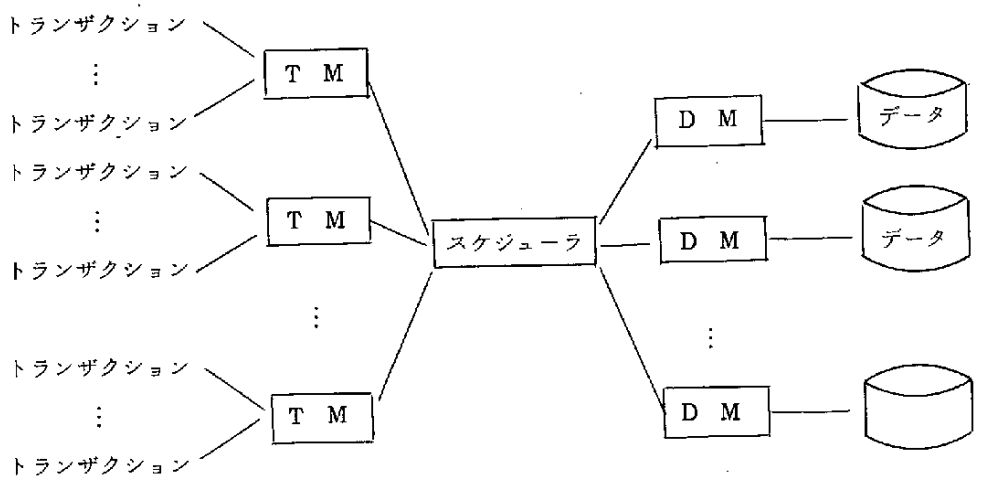
Ｌはシリアライザブルである。

というものである。

したがってＳＧ（Ｌ）を調べることによりログがシリアライザブルかを決定できる。この計算量はＮＰ完全である〔Bernstein 83〕。

3. スケジューラ

同時実行制御のアルゴリズムは数多く提案されている。しかし，すべてが正当というわけではなく，正当性が証明されているものとなると限られてくる。ここでは代表的な２相ロック，タイムスタンプ，ＳＧチェッキング，サーティファイアの４つのスケジューラについて説明する。話を簡単化するために，ここでは，3.1図のようにスケジューラがシステム全体に１つだけあるように仮定する。



3 1.図　集中化スケジューラＤＤＢＳ

さらにスケジューラはrw 競合，ww 競合の２つの場合に分けられて制御を行なうことがある。例えば rw 競合に対しては２相ロッキングを用い，ww 競合に対してはタイムスタンプを適用するという具合である。したがってスケジューラの説明も rw と ww の処理を分けてあるところがある。

スケジューラが分散化された時の，各アルゴリズムの問題は4章のスケジューラの分散化で述べる。

3.1　２相ロッキング

２相ロッキング（２ＰＬ）のスケジューラは次の３つの規則により定義される〔Eswaran 76〕。

(1)　$r_i[x]$（又は $w_i[x]$）を出す前に read ロック（又は write ロック）を

xに対してセットする。このロックは適当なDMに対して操作が実行されるまで保持されなければならない。

(2) 異なったトランザクションは同時に競合ロックを保持できない。2つのロックの競合とはそれらが同一データ上に対してかかっており，少なくとも1つがwriteロックの時である。rw と ww のスケジューリングが別々に行なわれるのならば，競合の定義は少し変更される。rw スケジューリングに対しては，同一データ上の2つのロックが競合を起こすのは1つのみが確実にwriteロックの時である。writeロック同志が競合を起すことはない。 wwスケジューリングの場合は必ず両方のロックがwriteロック時である。

(3) トランザクションが1つでもロックを解いたならばトランザクションは二度とロックはかけられない。

規則3がいわゆる2相の作法と呼ばれるもので，ログは

- 昇　相：ロックをかける相
- 降　相：ロックをはずす相

の2つに分離される。規則3を実現するには，よくトランザクションが終わるまですべてのロックを保持することで行なわれる。

規則2を守るには，競合するロックを持つトランザクションの操作を遅らせることで行なうが，そうするとデッドロックを導く可能性がある。例えば $r_1(x)$，$r_2(y)$ に対して read ロックをかけた後，$w_1(y)$，$w_2(x)$ がくると，T_1 は x に対して readロックをかけており，$w_2(x)$ (T_2) が待たされる。逆に T_2 は y に対して readロックをかけており，$w_1(y)$ (T_1) が待たされる。したがって，両方とも永久に動かなくなってしまう。デッドロックはwaits－forグラフで知ることができる。waits－forグラフはトランザクション $T_0, \cdots, T_n$ をノード，トランザクション T_i が T_j を待つ時に書く $T_i \rightarrow T_j$ をエッジとした有向グラフである。デッドロックが存在するのは，waits－forグラフがcyclicである時である。前述した例では

となっている。

デッドロックの処理はwaits－forグラフを作って定期的にcycleがないかを探す方法が一般的に行なわれている。デッドロックがあった場合は

cycle内のトランザクションの１つを取り消して再起動させる。

２相ロッキング方式の正当性は，これがいつもＳＲ－ログを作り出すことで証明されている〔 Eswaran　７６ 〕〔 Bernstein　８２ 〕。

3.2　タイムスタンプ－Time　stamp　Ordering（Ｔ／Ｏ）

タイムスタンプ（ Time　stamp　Ordering Ｔ/Ｏ ）では，各トランザクションはＴＭにより全体でユニークなタイムスタンプが打たれる。ＴＭはトランザクションのすべての操作にタイムスタンプを打つ。Ｔ／Ｏスケジューラは単一規則により定義される。競合する操作同志の組はタイムスタンプの順番に従って出力する。（出力された競合操作の組は出力された順に従ってＤＭにより実行される。）

Ｔ／Ｏスケジューリングはいくつかのバリエーションがある。基本Ｔ／Ｏスケジューラは基本的には最初に来たのを最初に処理する順で操作をＤＭに出力する。さらに，

◦スケジューラが $r_i[x]$ を受けた時

今まで受けた x を書く操作の中で一番大きいタイムスタンプが T_i のタイムスタンプ（ＴＳ(i)）より大きい時

$r_i[x]$ を取り消す。

そうでない時は $r_i[x]$ を受けて，より小さいタイムスタンプを持つ x を書く操作の確認がすべてとれた時点でＤＭに出力する。

◦スケジューラが $w_i[x]$ を受けた時

今まで受けた x を読む，又は書く操作のうちで一番大きいタイムスタンプがＴＳ(i)より大きい時

$w_i[x]$ を取り消す。

そうでない時は $w_i[x]$ を受けて，より小さいタイムスタンプを持つ x を書く操作の確認がすべてとれた時点でＤＭに出力する。

それに対して保存Ｔ／Ｏスケジューラは，現在の操作が後の操作を取り消す心配がなくなるまで操作を遅らせる。したがってスケジューラが，Read，Writeをタイムスタンプの順で受けとることが必要。さらにスケジューラは内部に入力キューが必要である。保存Ｔ／Ｏスケジューラはさらにバリエーションを持っている。詳しくは〔 Bernstein　８１ 〕。

以上述べたように，基本Ｔ／Ｏは操作を待たせるのは非常に短かいが，多

くの操作を取り消す傾向がある。それに対して，保存T／Oは取り消しは行なわないが，非常に長い期間待たせる傾向がある。この2つは両極端であり，この中間のスケジューラは現在，提案されていない。

トーマスの書き込み規則（Thomas'write rule TWR）

基本T／Oとの組み合せであり，wwスケジューリングに新規性がある〔Thomas 79〕。

スケジューラが$w_i[x]$を受けると，

もし今まで受けたxを読む操作のうちで一番大きいタイムスタンプがTS(i)より大きい時，

$w_i[x]$を取り消す。 ……（基本I／O）

そうでない時，もし今まで受けたxを書く操作のタイムスタンプで一番大きいものがTS(i)より大きい時，

$w_i[x]$を実行したふりをする。 ……（TWR）

そうでない時，$w_i[x]$を受けて，より小さいタイムスタンプを書く操作の確認がすべてとれた時点でDMに出力する。

保存T／OとTWRの組み合せも存在する。詳しくは〔Bernstein 81〕を参照されたい。

T／Oスケジューリングの正当性も，これがいつもSR－ログを作り出すことで証明される。

3.3 シリアライゼイショングラフチェッキング（SGチェッキング）

スケジューラがシリアライゼイショングラフを作ってゆく。サイクルがないかをチェックし，あれば取り消す。

SGチェッキング

(1) T_iが始まった時点でSGにノードT_iを加える。

(2) T_iからRead又はWriteを受けた時，それと競合する操作がT_jからすでに出力されている時，

$T_j \rightarrow T_i$なるエッジを加える。

(3) もし(2)が終わった時点でSGがサイクリックでないなら操作を出力する。（DMは出力された順に操作を実行する。）

(4) もし(2)のあとSGがサイクリックなら操作を取り消す。ノードT_iを削除し，$T_i \rightarrow T_j$，$T_j \rightarrow T_i$なるエッジを消す。

当然，ＳＧチェッキングはいつもＳＲ－ログを作り出すので，正当なアルゴリズムである。

トランザクションが終了してもそのノードをＳＧから取り除けるとは限らない。取り除けるのはそれがソースノード（入るエッジがない。すべて出るエッジを持つノード）の場合である〔Casanova 79〕。

3.4 サーティファイア（確認者）

サーティファイアはスケジューリングの規則というのではなく，ある種のスケジューラである。すなわち，操作を受け取ると，内部に一旦格納し，以前の競合している操作の終了が確認された時点で出力する。（ここまでは通常のスケジューラと同じである。）トランザクションが終了すると，ＴＭはEnd操作を送る。サーティファイアは独自に内部に格納しておいた情報（アルゴリズムによって異なる）のチェックを行ないシリアライザブルに実行されたかを確認する。大丈夫ならば，そのトランザクションをサーティファイ（確認）し，終了を許可する。さもなければ，トランザクションを取り消す。

サーティファイアはそのチェック方法に前述したアルゴリズムの組み合せが考えられる。例えば，ＳＧチェックサーティファイアはチェックにSGチェックサーティファイアはチェックにＳＧチェッキングアルゴリズムを用いたものである。通常のＳＧチェッキングと同様にして，操作が来た時点でＳＧグラフを作るが，その時サイクルのチェックは行なわない。ＴＭからEnd操作が送られた時点で，チェックを行なう。サイクルがあればそのトランザクションを取り消し，なければ，サーティファイする。

２ＰＬサーティファイア〔Thomas 79〕〔Kung 79〕。

あるトランザクションの最初の操作をサーティファイアが受けてから，End操作を受けるまでの間，そのトランザクションはアクティブであるという。サーティファイアはアクティブなトランザクションT_iに対して２つの集合をもつ。

T_iのreadset　$RS(i)=\{x \mid$ サーティファイアが出力した$r_i[x]\}$

T_iのwriteset　$WS(i)=\{x \mid$ サーティファイアが出力した$w_i[x]\}$

サーティファイアは操作を受けとると，これらの集合を更新する。そして，End_i（トランザクションT_iのEnd）を受けると次のことを行なう。

$RS(active)=\cup(RS(j), T_j$がアクティブただし$i \neq j)$

$WS(active) = \cup (WS(j),\ T_j$ がアクティブただし $i \neq j)$

とした時，

もし　$RS(i) \cap WS(active) = \phi$ かつ

$WS(i) \cap (RS(active) \cup WS(active)) = \phi$

ならばトランザクション T_i をサーティファイする。

そうでないなら T_i を取り消す。

これはトランザクションが，そのreadsetとwritesetに対して想像上のロックを持つようなことと等しい。トランザクション T_i が終了した時点で，サーティファイアが，T_i の想像上のロックが他のアクティブなトランザクションが保持しているロックと競合しているかをチェックする。競合していれば，T_i を取り消し，していなければ，サーティファイする。

ＳＧチェックサーティファイアも２ＰＬサーティファイアもＳＲログを作り出すので，正当である。他に，Ｔ／Ｏサーティファイアも考えられるが，現在まだ提案されていない。

4. スケジューラの分散化

スケジューラの分散化，すなわち１つのＤＭに対して１つのスケジューラを仮定〔図1.1〕した場合，前述したアルゴリズムに対して，様々な問題が生じる。

分散化された２ＰＬースケジューラは全体のwaits－forグラフが必要。又，分散化されたＳＧチェックスケジューラは全体のＳＧが必要である。

分散化２ＰＬスケジューラ

２ＰＬスケジューリング規則に従うと，(1)，(2)はローカルなことで，問題がないが，規則(3)はスケジューラ，ＴＭ同志の協力が必要である。トランザクション T_i が終了する時，そのＴＭは T_i のReadとWrite が確認されるまで待つ。ＴＭが T_i のロックがセットされ，無事にロックをはずすことができることを知った時点で，ＴＭは End_i をスケジューラに送り，トランザクション T_i のロックをはずす。デッドロックは別サイト同志で起こる可能性がある。各ローカルなwaits－forグラフの合成が必要である。

分散化Ｔ／Ｏスケジューラ

Ｔ／Ｏスケジューラは分散化がやさしい。Ｔ／Ｏスケジューリング規則は本質的にローカルである。

分散化ＳＧチェッキングスケジューラ

ＳＧは本質的に大域的であり，分散化するのは難かしい〔Casanova 79〕

分散化サーティファイア

２ＰＬの(3)の規則のような同期化が少し必要である。トランザクションT_iのすべてのread，writeが確認された時点で，ＴＭはEnd_iをサーティファイアに送る必要がある。それ以外のことは前述したアルゴリズムに従う。例えば２ＰＬの場合には，各ローカルのwaits－forグラフが必要…… etc。

5. マルチバージョン

マルチバージョンデータベースではWrite $w_i[x]$ によりxの新しいバージョンを作り出す。これをx_iと書く。したがってxはバージョンの集合となる。Read $r_i[x]$ に対しては，スケジューラは適当なxのバージョンを選び出す。Writeはお互いにうわ書きしないのでReadはどのバージョンを読むことも可能である。したがってスケジューラはより柔軟にRead，Writeの順番を制御することができる。

データベースがマルチバージョンであっても，利用者にとっては，各データのバージョンが１つしかないように見えるようにする必要がある。例えば，

$w_0[x_0] \rightarrow r_1[x_0] \rightarrow w_1[x_1 y_1] \rightarrow r_2[x_0 y_1] \rightarrow w_2[y_2]$

はシリアルログである。しかし，バージョンは１つだけではない。

シリアルログが1バージョンシリアルであるというのは，各$r_i[x_j]$に対してx_jが一番最近に書いたバージョンである場合である。すなわち$w_j[x_j]$と$r_i[x_j]$の間に別のxに対するWrite操作がない時である。例えば前の例は１バージョンシリアルではない。なぜなら，$w_0[x_0] < w_1[x_1] < r_2[x_0]$である。ログが１シリアライザブル（１－ＳＲ）というのは，１バージョンシリアルログと等価の場合である。

マルチバージョン同時実行制御のアルゴリズムが正当かどうかを解析するにはマルチバージョンシリアライゼーショングラフ（ＭＶＳＧ）を用いる。

ＭＶＳＧはバージョンの順番を表わす$<<$と，ログＬで定義される。（MVSG（Ｌ,$<<$）と書く。） 通常のＳＧに次のエッジを加えたものである。

Ｌ中の各$r_i[x_j]$，$w_k[x_k]$に対して，

もし$x_k << x_j$ならば　$T_k \rightarrow T_j$

でなければ　$T_i \rightarrow T_k$

マルチバージョンログの正当性はＭＶＳＧを用いて解析できる。すなわち〔Bernstein 83〕では次の定理が証明されている。

〔マルチバージョンの定理〕

マルチバージョンログが１－ＳＲであることと，バージョン順序≪が存在し，ＭＶＳＧ（Ｌ，≪）がサイクリックでないこととは等価である。

5.1 マルチバージョンＴ／Ｏ

Ｔ／Ｏスケジューリングのマルチバージョン版である。トランザクションT_iは実行される直前にＴＳ(i)なる一意なタイムスタンプがつけられる。その時T_iの各Read，Write およびバージョンはそのタイムスタンプをもつ。バージョンの順番はタイムスタンプの順番，すなわち，ＴＳ(i)＜ＴＳ(j)ならば$x_i \ll x_j$ となる。操作は最初に来たものが最初に処理される。Readに対するバージョンの選択は今までで一番大きいタイムスタンプを持つx_kになる$r_i[x_k]$。Writeに対しては２つの場合がある。すなわちＴＳ(k)＜ＴＳ(i)＜ＴＳ(j)においてすでに，$r_j[x_k]$が処理されてしまっている時にはそのWriteは取り消される。そうでなければ独自のバージョンをWrite $w_i[x_i]$する。

マルチバージョンＴ／Ｏスケジューリングは，そのＭＶＳＧがサイクリックでないことから，いつも１－ＳＲログを産み出す。したがって正当なアルゴリズムとされている。

5.2 マルチバージョンロッキング（ＭＶＬ）

マルチバージョンロッキング（ＭＶＬ）はサーティファイロックというロックを用いるアルゴリズムである。各トランザクションとバージョンの状態として，

サーティファイド（certified）とアンサーティファイド（uncerfified）の２つを設ける。これはそれぞれサーティファイという操作により，サーティファイドの状態になる。トランザクションは始まる状態ではアンサーティファイドである。バージョンに対してはそのバージョンが書かれた直後の状態はアンサーティファイドが仮定されている。ＭＶＬではトランザクションT_0，…，T_nのログ上にいくつかの操作を設ける。

c_i　：トランザクションT_iをサーティファイする

$cl_i[x_i]$：トランザクションT_iが書くx_iのデータxに対してサーティファイロックをかける。

$c_i[x_i]$：x_iをサーティファイする。これによりデータxに対してかかっていたサーティファイロックが解除される。

アルゴリズムは次の規則に従う。

〔アルゴリズム〕

1.1　各トランザクションT_iに対して，c_iはT_iのRead，Writeの並びの最後に置く。

1.2　各トランザクションが書く各x_iに対して

$$cl_i[x_i] < c_i < c_i[x_i]$$

（各トランザクションは，その読み書きが終った後にサーティファイされ，サーティファイロックはトランザクションがサーティファイされる前にかけ，各バージョンのサーティファイはトランザクションがサーティファイされた後に行なう。）

2.1　各$cl_i[x_i]$と$cl_j[x_j]$は順序関係（$<$）にあること。

2.2　各x_i，x_jに対して

$cl_i[x_i] < cl_j[x_j]$ ならば $c_i[x_i] < cl_j[x_j]$

（複数のトランザクションが同時に同一のデータに対してサーティファイロックをかけられない。）

3.1　各$r_k[x_j]$と$c_i[x_i]$は順序関係（$<$）にあること

3.2　各$r_k[x_j]$，$w_i[x_i]$，$i \neq j$において，

もし$c_i[x_i] < r_k[x_j]$かつ$c_j[x_j] < r_k[x_j]$

ならば$c_i[x_i] < c_j[x_j]$

（Readに対して，どのバージョンを読むかを決める規則である。$r_k[x_j]$において，すでにx_jがサーティファイされているなら，それは最も最近にサーティファイされているバージョンである必要がある。）

4.1　各$r_k[x_j]$ $k \neq j$において$c_j[x_j] < c_k$

4.2　各$r_k[x_j]$，$w_i[x_i]$，$i \neq j$において

もし$r_k[x_j] < c_i[x_i]$かつ$c_j[x_j] < c_i$

ならば$c_k < c_i$

（T_kがx_jを読む場合にはx_jはサーティファイされている必要がある。T_iがx_iを書く時，各xのバージョンはすでにサーティファイされて

おり，それらを読むトランザクションはすべてサーティファイされていなければならない。

ＭＶＬも１－ＳＲログを産み出すので，正当なアルゴリズムである。しかし，２ＰＬと同様にしてデッドロックが起こる可能性がある。このデッドロックの発見は，前述したwaits－forグラフを用いて，サイクルを探すことで行なうことができるもの〔Bayer 80〕とタイムスタンプをベースにしたもの〔Stearn 81〕などがある。

6. まとめ

ここではＤＤＢＳに対する同時実行制御のアルゴリズムについて説明した。前述したように，数多くのものが提案されており，決定版といわれるものは，今だに存在しないようである。スケジューリングに一長一短があって，例えばロッキングスケジューリングでは，トランザクションの取り消しは行なわれないが，デッドロックにより長期間にわたって待たされる可能性がある。又，スケジューラを分散化するのが非常に難しい。一方，Ｔ／Ｏスケジューリングではデッドロックは起らないが，トランザクションの取り消しが行なわれやすい。スケジューラの分散化は簡単である。したがって，ＤＢＳの性質，通信方式，通信量の点から，最適なものを選び出すのがよいかも知れない。さらに組み合せとして，ｒｗスケジューリングとｗｗスケジューリングをそれぞれ別のアルゴリズムを用いることが考えられる。ある方式を固定して使わずに，組み合せるのもよい方法であろう。

参考文献

〔Bayer 80〕 Bayer, R., E. Elhardt, H. Heller, and A. Reiser. "Distributed Concurrency Control in Database Systems," Proc. Sixth Int. Conf. on Very Large Data Bases, IEEE, N.Y., 1980, pp. 275－284.

〔Bernstein 81〕 Bernstein, P.A. and N.Goodman, "Concurrency Control in Distributed Database Systems," Computing Svrveys 13, 2(June 1981), pp. 185－221.

〔Bernstein 82〕 Bernstein, P.A., "A Sophisticate's Introduction to Distributed Database Concurrency Control," Proc. Eighth Int. Conf. on Very Large Data Bases, 1982, pp. 62－76

〔Bernstein 83〕 Bernstein, P.A. and N.Goodman, "Multiversion Concurrency Control－Theory and Algorithms," ACM Trans. on Database Systems 8, 4 (Dec. 1983), pp. 465－483

〔Casanova 79〕 Casanova, M.A. The Concurrency Control Problem of Database Systems, Lecture Notes in Computer Science, Vol. 116, Springer－Verlag, 1981 (originally published as TR－17－79, Center for Research in Computing Technology, Harvard University, 1979).

〔Eswaran 76〕 Eswaran, K.P., J.N. Gray, R.A. Lorie, and I.L. Traiger. "The Notions of Consistency and Predicate Locks in a Database System," Comminications of the ACM 19, 11 (Nov. 1976).

〔Kung 79〕 Kung, H.T. and J.T. Robinson. "On Optim-

istic Methods for Concurrecy Control," Proc. 1979 Int. Conf. on Very Large Data Bases, Oct. 1979.

〔Stearns 81〕 Stearns, R.E. and D.J. Rosenkrantz. "Distributed Database Concurrency Controls Using Before－Values," Proc. 1981 ACM-SICMOD Conf., ACM, N.Y., pp. 74－83.

〔Thomas 79〕 Thomas, R.H. "A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases," ACM Trans. on Database Systems 4, 2 (June 1979), pp. 180－209.

昭和５９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（代表）

印刷所　株式会社　三　州　社
東京都港区芝大門１－１－２１
TEL　（４３３）１４８１