資料 4

見 本

第５世代のコンピュータ

システム化技術研究分科会

―分散データベースシステム―

昭和５６年２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５５年度に実施した「第５世代の電子計算機に関する調査研究」の成果をとりまとめたものであります。

序

わが国における社会経済は，資源，エネルギー問題を始めとして国際的な変動と，不確実性の流れのなかにある。同時に，的確な情報の加工利用が重要視される情報化社会の形成が指向されている。

コンピュータは，われわれの情報活用においてすでに不可欠なツールとなっているが，今後１０年間には多くの諸問題を解決するため，更に高度な技術が要求され，新たな理論・技術にもとづくコンピュータ・システム実現が望まれるであろう。

このため，当協会では「第５世代コンピュータ調査研究委員会」を設置し，１９９０年代に実用化されるべきコンピュータ・システム（第５世代コンピュータ）はどのようなものになるか，またその開発プロジェクトはどのように進めていくべきかについての調査研究を，昭和54年度から２ヶ年の予定で開始した。

昭和５５年度は，本委員会のもとの３分科会（システム化技術，基礎理論，アーキテクチャ）および多数のワーキング・グループによる調査研究活動，内外の大学・研究所への研究委託，米国への技術調査等により，第５世代コンピュータのイメージ及び研究開発課題を明確化し，さらに，その研究開発計画・体制について検討した。

本資料は，これらの調査研究結果のうち，各ワーキング・グループの活動成果をとりまとめたものである。

最後に，調査研究にご協力いただいた第５世代コンピュータ調査研究委員会を始め，関係各位に厚く御礼申し上げる次第である。

昭和５６年２月

財団法人　日本情報処理開発協会
会　長　上　野　幸　七

システム化技術研究（S）分科会DDBSワーキング・グループ（DDBS-WG）

	氏名	所属
委員	児西清義	日本電信電話公社横須賀電気通信研究所データ通信部データ通信方式研究室室長補佐
〃	田中英彦	東京大学工学部電気工学科助教授
〃	田畑孝一	京都大学情報処理教育センター助教授
〃	増永良文	東北大学電気通信研究所助手
〃	溝口徹夫	三菱電機（株）開発本部計算機研究部研究員主事
〃	山崎晴明	沖電気工業（株）電気本部研究所複合システム研究部システム第二研究室主任
〃	後藤龍男	日本電気（株）情報処理官庁システム事業部システム部課長
〃	山本欣子	（財）日本情報処理開発協会開発部長
〃	鍛治勝三	〃 開発部課長代理
〃	滝沢誠	〃 開発部主任

目　　次

1.DDBS-WGの目的と方針

1. ＤＤＢＳ　ＷＧ の目的と方針

1.1 ＤＤＢＳ－ＷＧの目的

第５世代（５Ｇ）コンピュータシステムの主要アプリケーションとして，ＣＡＥ／ＣＡＤ，オフィス情報システム（ＯＩＳ）（又はＯＡ），ＤＳＳ，知能ロボットが考えられる。このため，システム化技術研究（Ｓ）分科会では，これらのアプリケーションを調査研究し，５Ｇコンピュータシステムに対する要求を明らかにしようとしている。これらのアプリケーションを実現するための共通技術として，分散データベースシステム（ＤＤＢＳ）技術の確立が必須であることは，昨年度のＳ分科会の報告書にも述べられている。しかし，昨年度行なったミニデルファイ調査においても，ＤＤＢＳの必要性は全般に強調されてはいるが，その具体的イメージは明らかとなっていないのが現状である。

このため，ＤＤＢＳ－ＷＧは，次の２点を目標として，Ｓ分科会のもとにつくられた。

a) ＤＤＢＳのイメージの明確化

b) ＤＤＢＳから５Ｇコンピュータシステムに対する要求の明確化

1.2 ＤＤＢＳ－ＷＧとしての問題意識

ＤＤＢＳの概要を述べる以前に，まず，本ＷＧとしての５Ｇコンピュータシステムに対する問題意識を明らかにしたい。

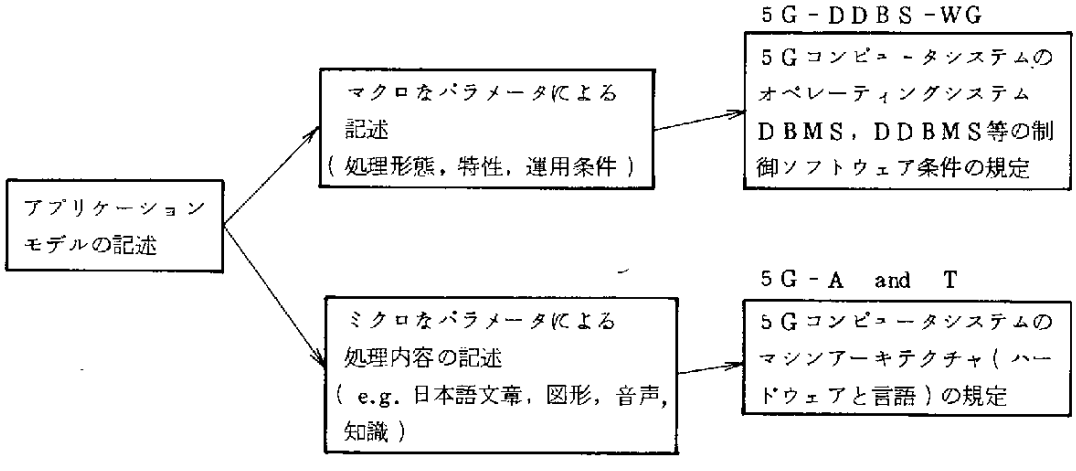


図 1.1　ＤＤＢＳ－ＷＧの問題意識

本ＷＧの問題意識は，図 1.1 の様に図示できる。即ち，基礎理論研究（Ｔ）分科会及びアーキテクチャ研究（Ａ）分科会のアプローチは，主として，処理対象の性質と処理内容とに着目して，マシン・アーキテクチャ（即ち，ハードウェア及び問題の記術系（言語））を規定しようとするものと考えられる。

これに対して，本ＷＧでは，システムのマクロなレベルで，アプリケーションからシス

テムに要求される処理の形態，特性，運用条件を明らかにして，５Ｇコンピュータシステムの制御ソフトウェアの条件を規定しようとするアプローチをとっている。使い易さ，柔軟性，進化性，環境との親和性等の５Ｇコンピュータシステムに対する基本的要求を考えると，分散システムが５Ｇシステムのもっとも基本的な形態と考えられる。このことから，通信ネットワーク，データベースシステム，処理の一体化した分散情報システム（distributed information system ： ＤＩＳ）〔図1.2〕は，５Ｇシステムの１つの中心と考

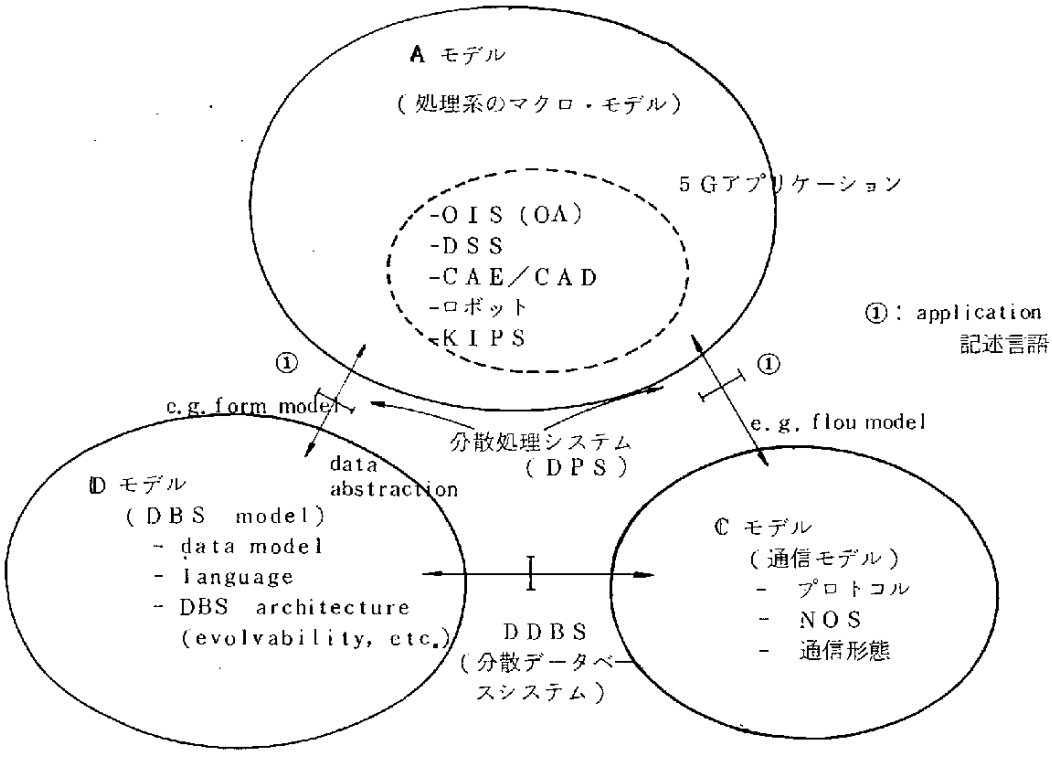


図 1.2　分散情報システム（ＤＩＳ）

えられる。このため，次の点を本ＷＧの最終的目標としたい。

a) ＤＩＳ制御ソフトウェアの条件の設定

b) ＤＩＳ上で，ＯＩＳ，ＤＳＳ，ＣＡＥ／ＣＡＤ等の種々の５Ｇアプリケーションを実現するための条件の明確化

第２の大きな問題意識は，従来の文字，数値データに加えて，図形，画像，文章，program 等の新しいデータをどのように情報システム内に組み込んでいくかである。

1.3　検討方針

ＤＩＳは，システムの目的に応じて，処理系のマクロモデル（アプリケーションの記述）（ ai ∈ 𝔸 ），データベースシステム（ＤＢＳ）モデル（ dj ∈ 𝔻 ），通信モデル

(ck ∈ C)の3つのモデル(ai, dj, ck)を，最適に統合することによって(i. e. システム化技術によって)実現される。このため，本WGでは，A，D，Cの3つのモデルの適合性を，代表的アプリケーション毎に研究し，DIS制御ソフトウェアの条件を明らかにし，5Gコンピュータシステムに反映させるために，次のステップに基づいて検討を進めている。

1. DDBS固有の研究開発課題の明確化
2. 処理系のマクロモデル(アプリケーション)の性質の規定

   (具体的には，分科会の他WGの成果を，分散情報システムの立場から検討する)
3. 分散情報システム(DIS)制御ソフトウェア条件の規定
4. 知識情報処理を含む高度応用とDISとの関連の研究

各stepの中で，図形，画像等のnew dataについても検討する。

この方針のもとに，本WGでは，7月末より本年度前半の活動を行ない，現在第1ステップを終了し，オフィス情報システム(OIS or OA)を中心に第2ステップ以降の検討〔例えば表1.1〕を始めようとしている。本年度後半以降のWG活動として，第2ステップから第4ステップの検討を行なっていきたいと考えている。

表1.1　アプリケーションの処理特性・形態とDBSの条件 *)

特性＼代表例＼形態			集中 ← B軸 → 分散	
定型 ↑ A軸 ↓ 非定型	RTS			
	TSS			
	New アプリケーション	OA		
		DSS		
		CAD/CAE		
		〃		

*) 表の□内を，DDBSのパラメータによって記述する。

1.4　DDBSの必要性

現在のデータベースシステム(DBS)は統合化(integration)を最も重要な概念としている。これは，関連する個々のアプリケーションが必要とする全てのデータを，1つのデータ集合に，論理的に一体化(unification)することである。このことは，データを個々のアプリケーションによって管理させるのではなく，関連するこれらのアプリケーションから成る共同体(community or enterprize)によって管理させることを意味して

いる。共同体を代表するデータベース管理者（ＤＢＡ）による，データの論理的集中管理によって，次の様な利点がもたらされた。

1) データの物理的及び論理的冗長性の低下

2) データの無矛盾性

3) 全てのデータに対するアクセスの一様性

4) アプリケーションのインプリメンテーションの容易性

こうした統合化は，一方では，共同体全体の最適性（i.e.中央集権化）を目指すものであることを指摘できる。これは，ある場合には，個々のアプリケーションの利益が犠牲にされ得ることでもある。今後，データベースシステムのアプリケーションは多様化し，高度化していくものと思われる。この様な情況に対処するためには，アプリケーションに対するＤＢＳの柔軟な適応性が求められる。しかし，統合化ＤＢＳは，その管理の集中化によって，次の様な欠点をもっている。

1) アプリケーション分野（domain）や，データ利用形態の変化への適応性の悪さ。

2) 全体のパフォーマンスを最大化すべく，ＤＢＳのシステム構成が決められている。しかし，全体最適性は，個々のアプリケーションにとって，最適とは限らない。

3) アプリケーションの必要とするデータの論理構造と，ＤＢＳの論理構造が一致しない。

4) セキュリティ／プライバシー問題を解決できない。各ユーザは，一般に，自分の情報を，他人（即ちＤＢＡ）の管理化に置きたがらない。集中的な管理権を持ったＤＢＡの正しさを保障する方法はない。

上述した統合ＤＢＳの問題点に対する解は，論理的かつ物理的なデータ冗長性をもたらすＤＢＳの分散化である。即ち，集中して管理されるべきデータと，そうでないものとを切り分け，各アプリケーションの必要とするデータはなるべく個々に適した様に管理させることである（即ち，分権化）。アプリケーションごとに共有されるデータは，そのデータに対する同時実行の度合，必要とされる一致性の程度（例えば，一致性が乱されていていいのは１秒以内か，１時間以内か）によって集中管理の度合が決まる。例えば，非常に厳しい一致性と，更新頻度の高いデータは，物理的にも集中管理される。しかし，種々のアプリケーションで利用され，あまり更新されない（されても，さほど一致性の条件が厳しくない）データは，データを必要とするアプリケーションのもとに置くことができる。こうした分散化の背景には，ＶＬＳＩ技術を中心としたコンピュータハードウェアの高性能化と低価格化がある。これによって，データを物理的に冗長して保持できるようになってきたことが，ＤＢＳの分散化への１つの大きな要因となっている。

更に，各ＤＢＳが相互に通信が可能となったことによって，１つのＤＢＳ（即ち，１つの共同体）の枠を越えた新たなアプリケーション（例えば，ＤＳＳ）の処理が可能となってくる。後述するＤＤＢＳの統合型設計技術がこれである。

以上のことより，ＤＢＳ利用の高度化，多様化は，ＤＤＢＳによって始めて解決できるものであり，５Ｇコンピュータシステムを考えるうえでの必須の技術と結論できる。

ＤＤＢＳの長所を，以下にまとめる。

1) １つの共同体（i.e.ＤＢＳ）を越えた新たなアプリケーション（e.g. ＤＳＳ，ＣＡＥ／ＣＡＤ）に対処できる。

2) 利用組織との整合性。処理を行なう場所に，必要なデータを置くことができる。

3) システムの柔軟性と適応性。新たなアプリケーションの付加と消去。新たなデータ要求，利用要求への対応が良い。

4) パフォーマンスの改良。データを冗長に管理させ，各アプリケーションに適した物理構成させることによって，データのavailabilityが向上する。

5) 高信頼化。データの冗長化によって，１つのサイトの障害が，全システムの障害とならない。

6) セキュリティ／プライバシー。自分のデータは，自分のもとで管理できる。

2.DDBS問題

2. ＤＤＢＳ問題

2.1 ＤＤＢＳの概要

分散データベースシステム（ＤＤＢＳ）とは，通信ネットワークを介して，互いに通信可能な複数のデータベースシステム（ＤＢＳ）を，どのＤＢＳに何のデータがあるか（分散問題），又各ＤＢＳをどの様にアクセスするか（異種性問題）を意識することなく統合的に利用できるシステムである。

ＤＤＢＳの基本構成要素は，ＤＢＳと通信ネットワークである。通信ネットワークとしては，ＤＢＳ相互の通信を行なわせるもので，ARPANET 等のパケット交換全体ネットワーク又は，光ファイバー，無線バス等を用いた（共有媒体）ローカルネットワークがある。ＤＤＢＳは，これらのネットワークのブラックボックス化された通信機能を用いて，複数ＤＢＳを１つの論理的ＤＢＳに仮想化したものである。即ち，ＤＤＢＳでは，既に存在する複数のＤＢＳと通信ネットワークとから，ある目的（e.g.高応答性，高信頼性，１つのＤＢＳの枠を越えた新たな処理要求）を達成する統合システムをつくり出すシステム化技術が大きな柱となる。又，ＤＢＳはデータの集合とプロセシング機能との対であり，これを通信ネットワークによって結合したものがＤＤＢＳであることから，ＤＤＢＳとデータベースマシン（ＤＢＭ）とは密接に関連したものとなってきている。この点は，我々の主要な問題意識の１つでもある。

ＤＤＢＳの問題は，前述したように，次の２つの問題を解決することである。

a) 異種性問題 （ＤＢＳのモデル，言語が異なっていることによって生じる問題）

b) 分散問題 （ＤＢＳが通信ネットワーク上に分散していることによって生じる問題）

この問題を解決するためには，図2.1に示すような四層スキーマ構造を，ＤＤＢＳの基本アーキテクチャ（全体アーキテクチャと呼ぶ）とすることが必要である。ローカル内部スキーマ（ＬＩＳ）は，各ＤＢＳに対応している。ローカル概念スキーマ（ＬＣＳ）層は異種性問題が解決された層であり，各サイド単位に共通データモデルと言語とがサポートされている。全体概念スキーマ（ＧＣＳ）は，ＤＤＢＳ全体のデータについての一意な記述である。このレベルで，ＤＤＢＳの２つの問題，即ち，異種性と分散問題は解決されたことになる。最後の外部スキーマ（ＥＸＳ）は，アプリケーション固有のデータ記述である。

この全体アーキテクチャ（gross architecture）に基づいた，分散データベースシステム（ＤＤＢＳ）の技術体系を図2.2に示す。

先述したように，ＤＤＢＳ問題は

a) アプリケーション（ＯＩＳ，ＤＳＳ等）からの要求　（Ａモデル記述）

例　concurrency と consistency の度合，アプリケーションのライフサイクル，応答速度，スループット，新たな処理要求，トランザクションのパターン　等

b) データベースシステム（既にデータを持っているか（bottom-up），ないか（top-down）　（Ｄモデル記述）

c) 通信ネットワーク　（Ｃモデル記述）

の３つが与えられたもとで，最適なシステムをつくり出すシステム化技術ということができる。

本報告では，b)とc)，特にb)中心になっている（a)については第２ステップ以降検討していきたい）。

ＤＤＢＳの問題の多くは，ＤＢＳ問題である。ＤＤＢＳの設計には，各ＤＢＳの状態（e.g.モデル）とともに，ＤＤＢＳ全体での共通モデルに大きく依存することになる。問合せ（検索と更新）の処理も，ＤＢＳにおけるものに対して，その複雑度（complexity）が大きくなっていることが特徴である。これは，処理が通信を介して行なわれねばならない（通信処理と呼ぶ）ことに依っている。

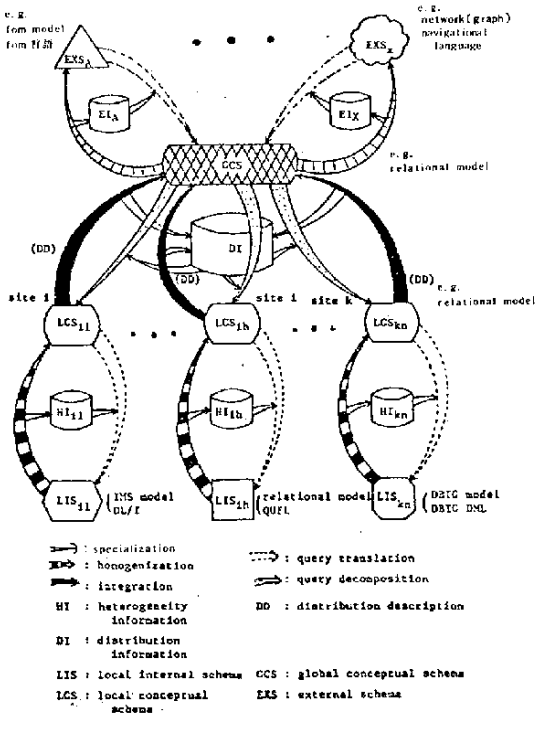


図 2.1　Four-Schema structure （ＦＳＳ）〔Takizawa,M〕

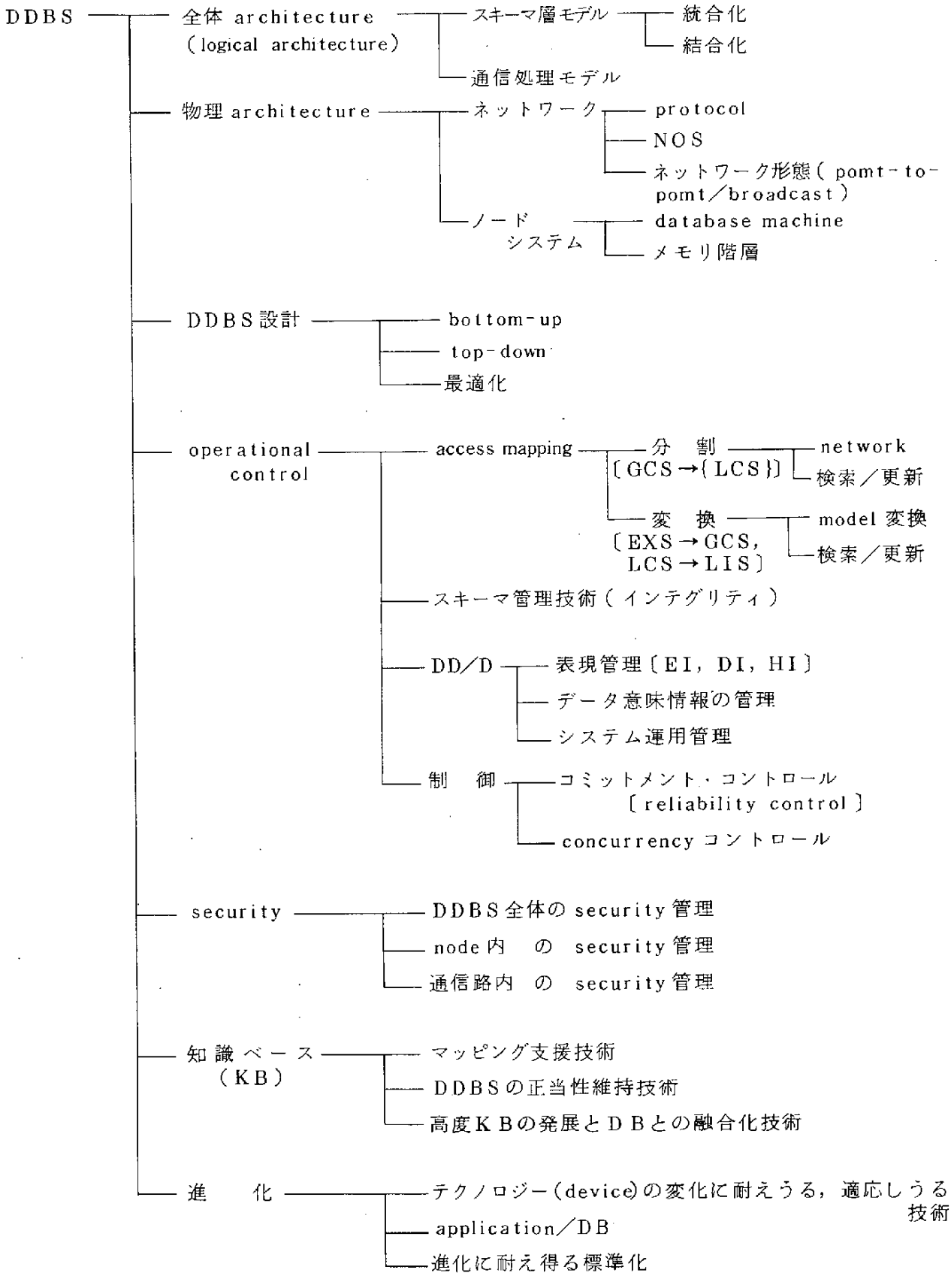


図 2.2 DDBSの技術体係

2.2 研究動向

ＤＤＢＳの研究開発は，欧米を中心に１９７６～１９７７年に始まり，１９７９～１９８０年にかけて，第１phase の implementation を終了している。これらの例としてはＳＤＤ－１（ＣＣＡ　米国），ＬＡＤＤＥＲ（ＳＲＩ，米国），ＰＯＬＹＰＨＥＭＥ（グルノーブル大学　仏国）等をあげることができる〔表2.2〕。これらの特色は，同種（homogeneous）ＤＢＭＳを，パケット交換 global network（e.g. ＡＲＰＡＮＥＴ，ＣＹＣＬＡＤＥＳ）によって結合し，ユーザに対してデータの所在を意識せずにアクセスできるような統一的view のサポートを行なっている点である。さらにＣＣＡ（Computer Corporation of America）社によって開発されたＳＤＤ－１は，１９７９年後半にそのインプリメンテーションを終了するとともに，これの商用 product としての distributed model ２０４システムを，１９８１年後半（第４四半期）までに出荷することになっている。

１９７９～１９８０年にかけて，上述したシステムの多くは，その第２phase ＤＤＢＳの研究開発を開始している。第１phase ＤＤＢＳが，コンピュータネットワークを用いてデータベースリソースを統合利用できることの可能性（i.e. general な methodology の確立）を明らかにするためのプロトタイプであったのに対して，第２phase ＤＤＢＳは，具体的なアプリケーション（i.e. office information system）への適用を目指している。即ち，分散データベースシステム（ＤＤＢＳ）と分散処理（distributed processing or ＤＰ）とが結合された分散システム（ＤＳ）を目標としていることを指摘できる。多様な構成要素に適応するために第２phase では異種ＤＢＭＳの統合が大きなテーマとなってきている。又，第１phase ＤＤＢＳの主要なパフォーマンス bottle neck となった global network にかわって，loop／bus といった local network を，各データベースシステム／処理システムを結合するために用いてきている。この点は，ＤＤＢＳとデータベースマシン（ＤＢＭ）とをより密接に関連したものにしてきている（e.g. ＩＮＧＲＥＳ－ＭＵＦＦＩＮ〔UC. Berkeley〕，ＤＩＡＬＯＧ〔Purdue Univ.〕，ＲＡＰ－３）。

表 2.2　第 1 phase　ＤＤＢＳ

		SDD-1	LADDER	POLYPHEME
開発サイト（年）		CCA (U.S.A.) (1976 ∿ 1979)	SRI (U.S.A.) (1976 ∿ 1978)	Univ. of Grenoble (1976 ∿ 1979)
Design		Top-down		Bottom-up
Application		NAVY の軍艦情報	NAVY の軍艦情報	
Network		ARPANET	ARPANET	CYCLADES
ユーザインタフェース (EXS level)	Model	Relational Model	Attribute Model	Relational Model
	言語	Data language (Procedural)	自然言語	Relational Algebra
分散 level (GCS level)	Model 言語	Relational Model QUEL (non-procedural)	Relational Model Relational Calculus	MOGADOR (Binary relational) Relational Algebra
	分解問合せ処理	∘ yes	∘ yes	∘ yes
	Con-currency 制御	Pre-analysis / time stamp	No update	No update/no concurrency
	Redundant copy	∘ Relation の horizontal と vertical decomposition	∘ backup copy (DB単位)	no redundancy
ローカルレベル (LCS level)	Model language DBMS	Relational Data language (procedural) Data computer (DEC)	Relational Data language Data computer (DEC)	Relational Algebra URANUSシステム (SOCRATE DBMS の relational system

このように，第2 phase ＤＤＢＳ／ＤＰＳの特徴は，次の様にまとめられる。

1) 異種ＤＢＭＳの統合

data model
language（transactionも含む）　の相違の解決

e.g　SDD-1 （CCA）
　　LADDER（SRI）
　　UC. Berkeley

2) 具体的アプリケーション＝office information system（ＯＩＳ）（分散処理）

e.g　SIRIUS-DELTAとKAYAK（INRIA）
　　DIALOG　（Purdue Univ.）
　　MICROBE-SCOT（POLYPHEME）(Univ.of Grenoble)

3) システムの効率化　　通信bottle neckの解消

local networkを用いようとしている。

データベースマシン（ＤＢＭ）化

e.g. DIALOG　（Purdue Univ.）
　　INGRES-MUFFIN　（UC.Berkeley）
　　SIRIUS-DELTA　（INRIA）
　　MICROBE　（Univ of Grenoble）

4) ユーザ　friendliness

マンマシンインタフェースの高度化（自然言語インタフェース，ガイダンス，対話技法）

e.g. LADDER　（SRI）

5) ＤＢＳの高度化

Semanticsの取り込み　KBS化

さて，国内の研究開発は，ほとんどが設計以前の段階にある。国内の研究とは，次のものをあげることができる。

1) ＪＤＤＢＳ　（ＪＩＰＤＥＣ）

－異質ＤＢＭＳの場合
－ＤＤＢＳのgross architectureとしての四層スキーマ構造（ＦＳＳ）
－分散問合せ処理（問合せ分割）
－問合せ変換（QUEL→DBTG）
－model変換（DBTG→relational）
－統合化　（local → global）

2) 京　大

－問合せ処理　（GCS→｜LCS｜）

3) 沖電気

－ concurrency

－ reliability control　（network partition failure 処理）

4) ＮＥＣ

－ concurrency contorl　（logical timestamp ）

5) 東　大

－ data の値の表現問題(DD／D)

ＤＤＢＳの個々の技術は，各サイドで種々に研究されているが，これらを統合して，１つのシステムとしてのＤＤＢＳにまとめあげるまではいっていない。欧米において，第１phaseのＤＤＢＳインプリメンテーションを終了し，その経験，反省を踏まえて，第２phase ＤＤＢＳ（／ＤＰＳ）の開発を開始しているのをみると，我国の研究開発は，５～６年の遅れがあるように思う。この遅れをとりもどすためには，現在のＤＢＳ，通信技術を統合した（システム化した）DDBS／DPSの prototypeを，数年の内に開発する必要がある。

本報告では，まず第３章において，ＤＤＢＳの全体アーキテクチャを論じる。この中で，ＤＤＢＳの全体アーキテクチャとして，四層スキーマ構造を示す。

第４章では，四層スキーマ構造に基づいたＤＤＢＳ設計問題を，スキーマ層設計（論理設計）と，管理システム（ＤＤＢＭＳ）設計と分けて論じる。

第５章では，アクセス要求のマッピングについて論じる。

第６章では，ＤＤ／Ｄについて論じる。ＤＤ／Ｄは，ＤＤＢＳ設計時に生成され，アクセス要求（例・問合せ）の処理等の運用時に用いられる。

第７章では，ＤＤＢＳの運用におけるシステムの制御問題について論じる。

第８章では，ＤＤＢＳにおける処理と通信との関連について論じ，通信処理概念を示す。

第９章では，ＤＤＢＳを実現するための問題点を論ずる。

最後に第１０章では，知識ベースとＤＤＢＳの関連について論じる。この中で，将来のＤＤＢＳが，知識ベースと密接に関連していくことを述べる。

3.全体アーキテクチャ

3. 全体アーキテクチャ（gross architecture）

3.1 序

分散型データベースシステム（ＤＤＢＳ）を考えるうえで，システムが何に基づいて，どのように階層化されるかを明らかにすることが必要である。ＤＤＢＳが，原則的には複数のデータベースシステム（ＤＢＳ）の統合されたシステムであることから，ＤＤＢＳの各層は，データベース記述，即ちスキーマによって特徴づけられる。

よって，ＤＤＢＳの全体アーキテクチャとして，次の３点を明確にする必要がある。

1） スキーマ層の位置づけ

－スキーマを記述するモデルと言語，記述されたデータの意味

2） スキーマ層間のマッピング

－スキーマ層の設計とアクセス

3） スキーマ層記述とマッピング情報

－data dictionary／directory（DD／D）

3.2 仮 定

ＤＤＢＳ検討のために，次の様な仮定を設ける。

1） ＤＤＢＳの各サイトはデータベースシステム（ＤＢＳ）とする。ＤＢＳとは，次の３つの要素から成っているとする。

i） データベース

ii） 管理システム（ＤＢＭＳ）

iii） ＤＤ／Ｄ

ファイル・システムに対して，ＤＢＳは，複数アプリケーション／ユーザによるデータベースの共有を可能とするとともに，より高度な（データ独立な）アクセス／記述インターフェースをサポートしている。ファイル・システムは，他のユーザと共有のない様な個人用ファイル・システムとして今後も存在していくと思われるが，インターフェースの高度化は進むものと思われる。

ＤＤＢＳから視た各サイトは，ＤＤＢＳ共同体のなかで共有可能なＤＢＳである。共有の存在しないファイル・システムは，ＤＤＢＳアプリケーションの処理，即ち，分散処理（ＤＰ）の問題としてある。この点については，後述するアプリケーション・モデルにおいて論じたい。

2） ＤＤＢＳを構成する各ＤＢＳは，ANSI／X3／SPARC による三層スキーマ構造をもっている。各スキーマ層は，あるデータモデルと言語とによって特徴づけられる。

ＤＤＢＳから視たＤＢＳとは，ＤＤＢＳのもとで共同利用可能なデータを記述したスキーマ層を意味している。よって，ＤＢＳはあるデータモデルと言語，及びこれらによって記述されたスキーマとによって特徴づけられるブラックボックスと考えられる。

3) 従って，各ＤＢＳの異種性（heterogeneity） とは，次の３点の各々の相違とする。

a) データモデル

b) a)に基づいたアクセス／記述言語

c) 格納されたデータベースの意味（これは，a)とb)とによって記述される）

ＤＢＳの存在するホストコンピュータの異種性（ハードウェア，ＯＳ等）は，DDBSからは視えないとする。個の値の表現（e.g. ２進１０進表現）の相違も，通信ネットワークレベル又はコード変換マシンによって解決されているとする。

値の単位（e.g. グラムとトン，分と秒）の相違，同一の実体を異なった値で表わしている問題は，データモデルとＤＤ／Ｄによって解決するとする。

4) 巨大情報システム（e.g. ＪＩＣＳＴ，ＮＩＳＴ，ＤＩＡＬＯＧ）は，ＤＤＢＳの外部（foreign)ＤＢＳとする。これらは，比較的簡単な構造をしたデータを大量に保有し，検索だけをサービスしている。巨大情報システム内のデータは，必要なものをＤＤＢＳ内に輸入した後に，分散処理するものとする。

5) 通信ネットワークは，高信頼性を持ったトランスペアレントな基本通信手段を提供するとする。ネットワークとしては，次の２種がある。

a) パケット交換ネットワーク

b) 共有媒体ネットワーク

a)では，通信実体間の１：１の通信が提供されるだけなので，高信頼な１：ｎ通信（e.g. 同時実行制御，guaranteed delivery）をネットワーク上に実現せねばならない。

b)では，物理的に１：ｎ通信をサポートされる。

3.3 ＤＢＳに対するＤＤＢＳ問題

ＤＤＢＳは，ネットワーク上に物理的に分散されたＤＢＳを，論理的な１つのＤＢＳに集中したシステムである。このことから，従来の集中型ＤＢＳに対して，次の２点がＤＤＢＳ実現上の問題点となる。

a) 各ＤＢＳのモデルと言語の各々の相違を解決する問題

b) 複数のＤＢＳと，１つの論理的ＤＢＳ（即ち，ＤＤＢＳ）との関係を明らかにする問題（各ＤＢＳ内の格納データベースの意味の相違の解決）

ここで，前者を異種性問題（heterogeneity problem），後者を分散問題（distribution problem）と呼ぶ。ＤＤＢＳでは，この２つの問題を解決する必要がある。

3.4 四層スキーマ構造 （four－schema structure）

四層スキーマ構造は，ＤＤＢＳの異種性と分散との２つの問題を解決することを目標としている。四層スキーマ構造は，次の３つの要素から成っている。

a) ４つのスキーマ層

b) 隣接スキーマ層間のマッピング

c) スキーマ層記述とマッピング情報

3.4.1 スキーマ層

四層スキーマ構造は，次の４つのスキーマ層から成っている。

a) ローカル内部スキーマ（ＬＩＳ）

－ local internal schema

b) ローカル概念スキーマ（ＬＣＳ）

－ local conceptual schema

c) 全体概念スキーマ（ＧＣＳ）

－ global conceptual schema

d) 外部スキーマ（ＥＸＳ）

－ external schema

ローカル内部スキーマ（ＬＩＳ）層は，既存ＤＢＳのスキーマ又はサブスキーマ層に対応している。ＬＩＳは，ＤＤＢＳ環境下で，そのＤＢＳの中で使用（共有）可能なデータの記述である。2.で述べたように，ＤＤＢＳから視たＤＢＳ（i.e.LIS）は，あるモデルと言語と，スキーマＬＩＳを備えたブラック・ボックスである。

ローカル概念スキーマ（ＬＣＳ）は．ＤＤＢＳ全体で共通なデータモデルと言語とによってそのサイトのＬＩＳを記述したものである。このスキーマ層においては，各ＤＢＳの異種性問題は視えない。ＬＣＳ層は，後述するＧＣＳ層とＬＩＳ層との中間にあり，共通記述系（モデルと言語）によるデータ記述とアクセス・インタフェースを提供する。

全体概念スキーマ（ＧＣＳ）は，ＤＤＢＳ全体の，ある共同体にとって意味のある論理データのユニークな記述である。このＧＣＳ層において，ＧＣＳが実際にどのようなＬＣＳから構成されているかという分散問題は視えない。即ち，ＧＣＳ層において，ユーザは，ＤＤＢＳを構成する各ＤＢＳの異種性と分散とを意識することなく，ＧＣＳ言語とＧＣＳを用いて必要なデータをアクセスできる。ここにおいて，ＤＤＢＳは，１つの論理的ＤＢＳに仮想化されたことになる。

外部スキーマ（ＥＸＳ）は，共同体のなかの各アプリケーションに適したモデルと言語とによって，ＧＣＳデータベースのサブセットを記述したものである。ＥＸＳは，

ANSI／SPARCの外部スキーマと等価である。何故なら，ＤＤＢＳは，既にＧＣＳ層において１つのＤＢＳに仮想化されているからである。よってＥＸＳ層は，ＤＤＢＳ固有の問題ではない。しかし，ＤＤＢＳのアプリケーション・モデルを考える時，先述したファイル・システムとともに重要となる。

各スキーマ層に，どのようなモデルと言語を設定するかは，最も重要な問題になる。

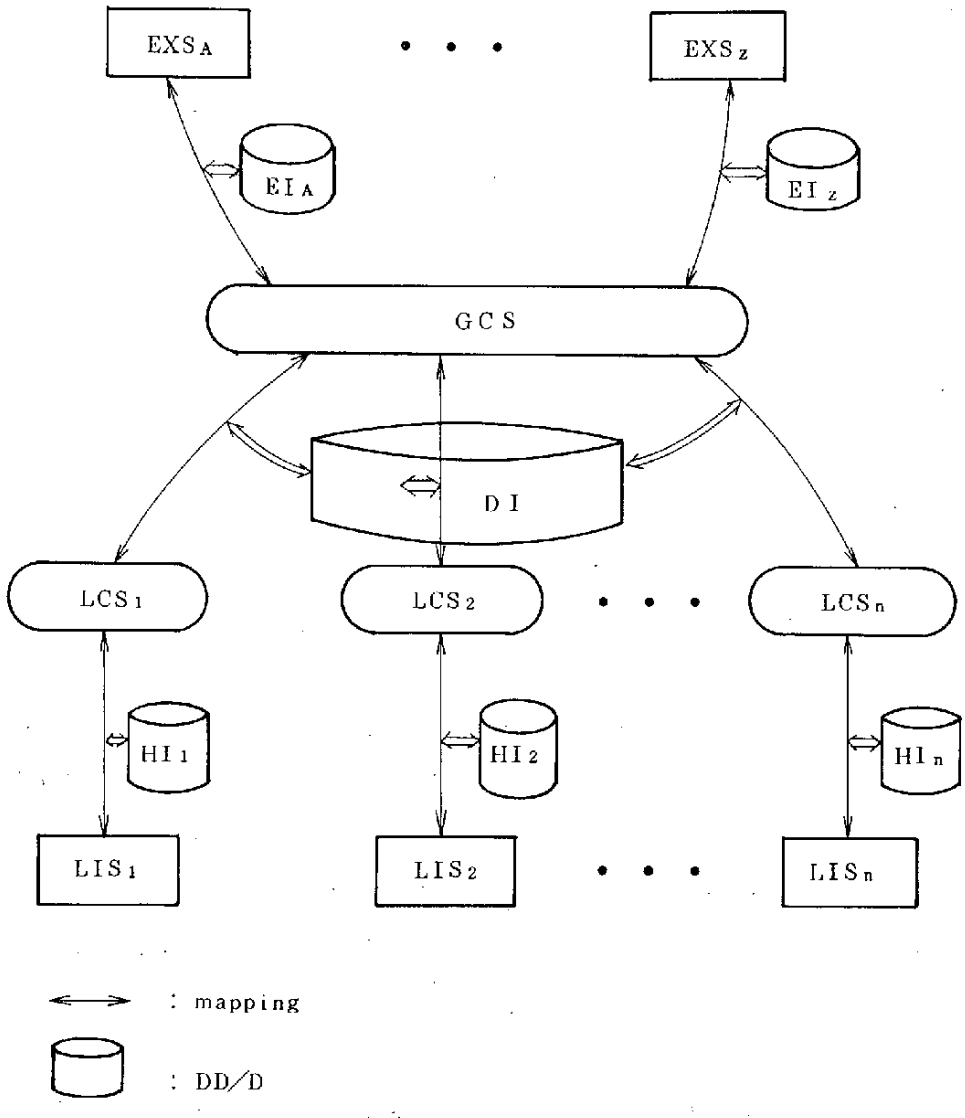


図 3.1　四層スキーマ構造

3.4.2　スキーマ層間マッピング

3.4.1で述べた４つのスキーマ層に対して，下記のような３つのマッピングを考えねばならない。

a)　ＬＩＳとＬＣＳ

b)　ＬＣＳとＧＣＳ

c)　ＧＣＳとＥＸＳ

スキーマ層間のマッピングとは，次の２種がある。

a) スキーマ層の設計マッピング

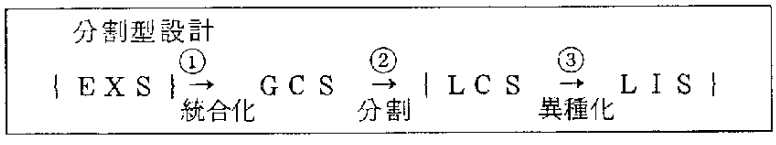
b) アクセス・マッピング

設計マッピングは，スキーマ層をどのように生成するかを示している。一般的に，ＤＤＢＳの初期設計時には，次の２種の設計アプローチがある。

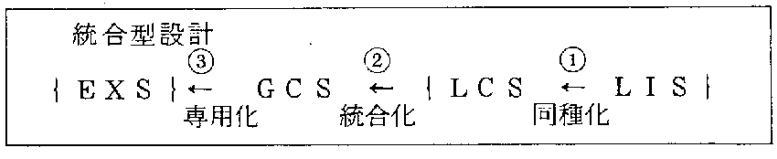
i) 分割型設計　　上位層から下位層へ

ii) 統合型設計　　下位層から上位層へ

分割型設計では，上位層がまず決定され，その後，ある目的関数のもとに下位層が決定される。これによると，次のようにスキーマ層は決定される。



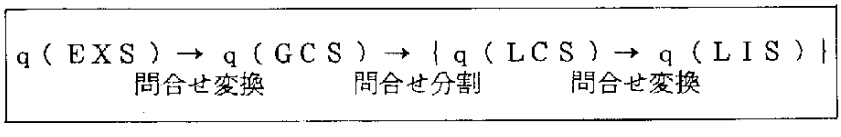
一方，分割型設計では，下位層が先に決定されて，その後上位層が決定されることになる。



実際のＤＤＢＳは，この２つのアプローチをミックスしてなされる。

アクセス・マッピングとは，上位層に基づいたアクセス要求（問合せと呼ぶことにする）を，下位層のアクセス要求に変換し，必要な結果を上位層に返すことである。ＤＤＢＳにおけるアクセス要求としては，検索と更新とがある。

アクセス・マッピング



3.4.3 ＤＤ／Ｄ

ＤＤ／Ｄは，各スキーマ層の記述と，スキーマ層間のマッピング情報（対応情報）とから成っている。これらの情報は，スキーマ層の設計時に，対応する管理者（administrator）／設計者によって定義される。

ＤＤ／Ｄとしては，次の３種類の情報がある。

a) 外部情報（ＥＩ）

— external information

b) 分散情報（ＤＩ）

— distribution information

c) 異種性情報（ＨＩ）

— heterogeneity information

外部情報（ＥＩ）とは，外部スキーマ自身と，ＥＸＳとＧＣＳとの対応情報である。ＥＩは，ＥＸＳのアプリケーション管理者（ＡＡ）によって定義される。

分散情報（ＤＩ）は，ＧＣＳ自身の記述と，ＧＣＳとＬＣＳとの対応情報である。ＤＩは，全体管理者（ＧＡ）によって定義される。

異種性情報（ＨＩ）は，ＬＣＳ及びＬＩＳ自身の記述と，ＬＣＳとＬＩＳとの対応情報である。ＨＩは，ローカル管理者（ＬＡ）によって定義される。

これらの情報は，アクセス・マッピングにおいて，下位層のアクセス要求に変換するために用いられる。このため，ＤＤ／Ｄの管理は，ＤＤＢＳのパフォーマンス上重要で

top-down design：

bottom-up design：

アクセス マッピング：

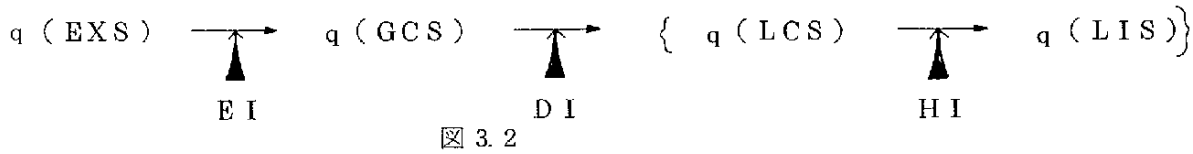


図 3.2

ある。このため，次の点を考える必要がある。

a) ＤＤ／Ｄの管理モデル

－ＬＣＳと同一モデルがよい。

dictionay 情報に対しては，より多くのセマンティクスを表わせる設計モデルが必要となる。

b) ＤＤ／Ｄの分散形態

－ＥＩ，ＤＩ，ＨＩごとに，分散形態を考える必要がある。

3.5 全体アーキテクチャ記述 〔TAKIM80a, c〕

四層スキーマ構造に基づいたＤＤＢＳ全体アーキテクチャは，前述してきたように，次の機能から成り立っている。

1) 設計マッピング

2) アクセス要求マッピング

3) ＤＤ／Ｄ

4) ＤＤＢＳ管理者

5) ＤＤＢＳユーザ

6) ＤＢＳ

これらの相互関連を示した図を，図3.3に示す。この記述法は，〔ANSIX75〕に基づいている。図中で，四角の箱は処理機能を，六角形は管理者とアプリケーション・ユーザを，三角形はＤＤ／Ｄを表わしている。更に，これらの機能間のインタフェース(i.e. ⟷)を示している。ＤＤＢＳ全体アーキテクチャは，これらの機能と，機能間のインタフェースとを明確にすることによって達成できる。

A・設計過程

図3.3の上半分は，統合型の設計マッピングを示しており，下半分はアクセス要求マッピングを示している。まず設計マッピングについて考えてみよう。

各サイトのデータベース管理者（ＤＢＡ）は，自分のデータベースのなかで，ＤＤＢＳ環境下で共有可能なデータ記述を，インタフェース１０（以降，インタフェースｉを Ii として表わすことにする）を介してＬＩＳＰ（ＬＩＳ processor）に入力する。ＬＩＳＰは，ＤＤ／Ｄにローカル内部スキーマ（ＬＩＳ）のオブジェクト形式を，異種性情報（ＨＩ）の一部として格納する（I_{11}）。

第２に，各ＤＢＳサイトのローカル管理者（ＬＡ）は，ＬＩＳＰの表示インタフェース（I_9）を通して，ＬＩＳの表示形式を得る。ＬＡは，これからローカル概念スキーマ（ＬＣＳ）と，ＬＩＳとＬＣＳとの対応記述とのソース形式をつくり，ＬＣＳＰ(LCS processor)に入力する（I_7）。ＬＣＳＰは，これからＬＣＳのオブジェクト形式と，ＬＩＳとＬＣＳとの対応情報のオブジェクト形式とを，ＤＤ／Ｄの異種性情報として格納する（I_8）。

第３に，全体管理者（ＧＡ）は，全てのサイトのＬＣＳＰの表示形式を I_6 を通して得て，全体概念スキーマ（ＧＣＳ）を設計する。ＧＡは，ＧＣＳと，ＧＣＳとＬＣＳとの対応記述とのソース形式をＧＣＳＰ（ＧＣＳ processor）に I_1 を通して入力する。ＧＣＳＰは，ＧＣＳとＧＣＳ／ＬＣＳ対応情報とのオブジェクト形式を生成して，分散情報（ＤＩ）としてＤＤ／Ｄに格納する（I_2）。

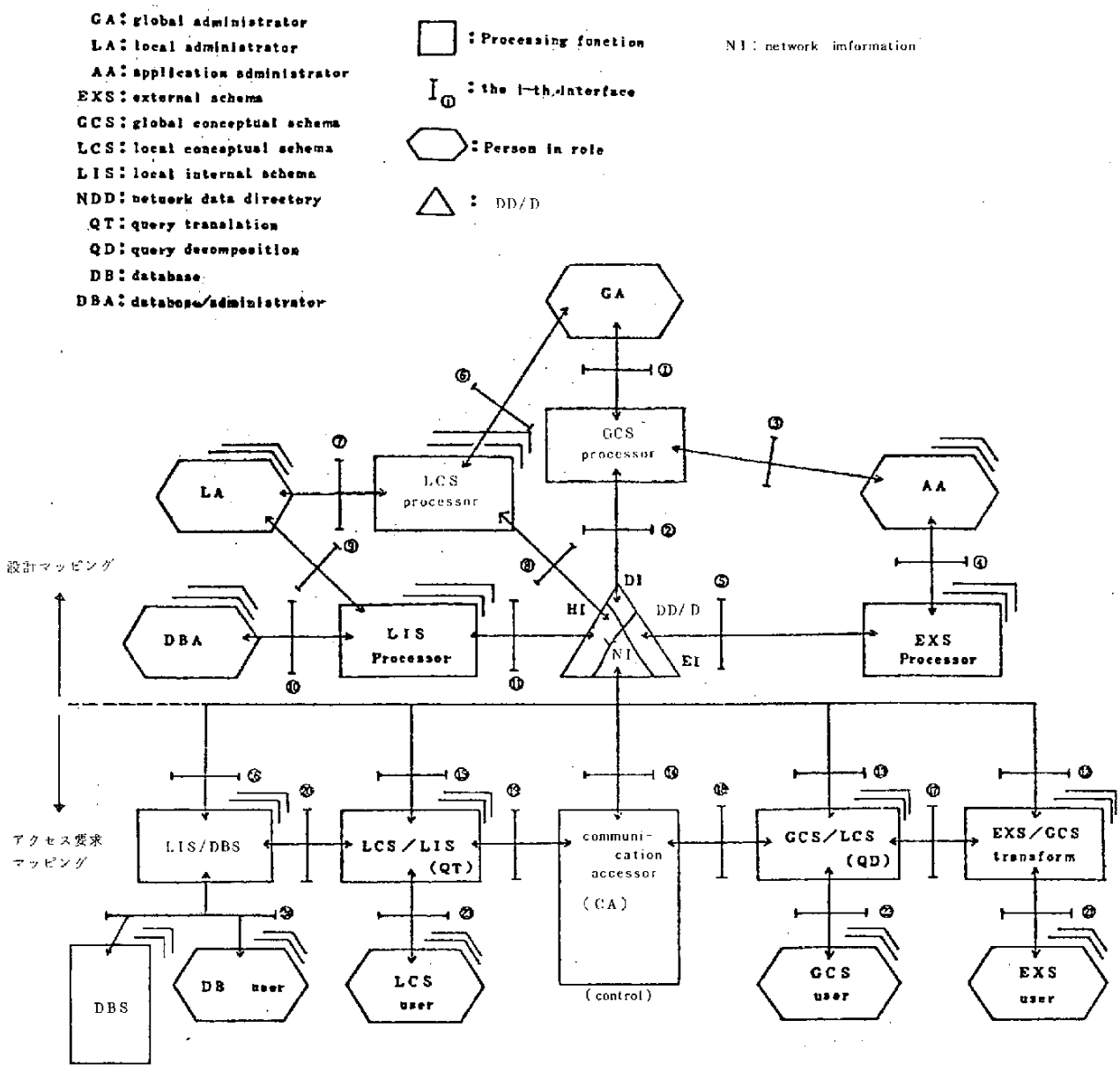


図 3.3　分散型データベースシステムの処理機能と機能間インタフェース

第4に，各アプリケーションの管理者（ＡＡ）は，ＧＣＳＰの表示インタフェース（I_3）を介してＧＣＳの表示形式を得て，必要な外部スキーマ（ＥＸＳ）を設計する。ＡＡは，ＥＸＳとＥＸＳ／ＧＣＳ対応情報とのソース形式を，ＥＸＳＰ（EXS processor）に，I_4 を通して入力する。ＥＸＳＰは，これらのオブジェクト形式を，外部情報（ＥＩ）としてＤＤ／Ｄに格納する（I_5）。

このようにして，ＤＤ／Ｄには，異種性情報，分散情報，外部情報の３種の情報が格納されたことになる。

分割型設計の場合は，上述した過程の逆をとることになる。この時の設計過程を，図3.4に示す。インタフェース番号の意味は，図3.3と同じである。ここで，インタフェ

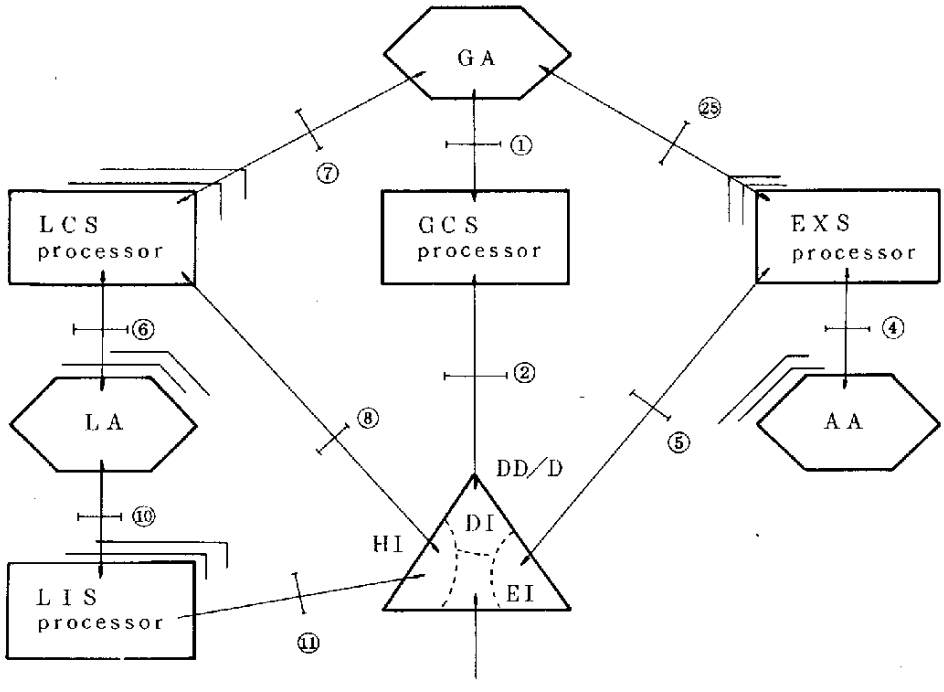


図3.4　統合型設計過程

ース25（I_{25}）は，ＥＸＳＰの表示インタフェースである。分割型設計の場合には，まず第1に，共同体内の各アプリケーションごとに，必要なデータ記述，即ちＥＸＳがＡＡによって定義される（I_4）。

次に，ＧＡは，これらのＥＸＳを統合してＧＣＳをつくる（I_1）。更に，ＧＡは，ある目的関数のもとで，ＧＣＳスキーマモデルを，各サイトに分割し，ＬＣＳをつくる（I_7）。

最後に，ＬＡは，自分のＤＢＳで実行可能なＬＩＳを生成する（I_{10}）。

B　アクセス要求マッピング

図3.3の下半分は，ＤＤＢＳが設計された後の運用におけるアクセス要求の変換過程を示している。この部分は，設計方法が分割型か統合型かには依存しない。

1) まず，ＥＸＳ層のユーザは，ＥＸＳアクセス言語インタフェース（I_{21}）を用いて，自分のアクセス要求を記述する。ＥＸＣ／ＧＣＳ問合せ変換システム（ＥＸＳ／ＧＣＳ transform）は，これを入力として，ＤＤ／ＤからI_{12}を通して得られる外部情報（ＥＩ）を用いて，ＧＣＳアクセス要求を生成する（I_{17}）。

2) 次に，ＧＣＳアクセス要求は，I_{17}又はI_{22}を介して，ＧＣＳ／ＬＣＳ問合せ分

割システムに入力される。これは，ＤＤ／Ｄ内の分散情報（ＤＩ）をI_{13}を通して用いて，ＬＣＳ層のアクセス要求に変換するとともに，必要な通信処理を行なう。各ＬＣＳサイトへのアクセス要求は，I_{18}を介して通信ネットワークアクセスシステム（ＣＡ）にわたされ，必要なサイトに届けられる。サイト間の通信処理も，ＣＡを用いて行なわれる。

3) ＬＣＳに対するアクセス要求は，ＬＣＳ／ＬＩＳ問合せ変換システム（ＬＣＳ／ＬＩＳ transform）にわたされ（I_{19} or I_{23}），ＤＤ／Ｄ内の異種性情報（ＨＩ）を用いて，ＬＩＳアクセス要求に変換される（I_{20}）。

4) ＬＩＳアクセス要求は，ＤＢＳで実行され，その結果は，I_{20}を通してＬＣＳ／ＬＩＳ問合せ変換システムに返される。

5) 更に，ＬＣＳアクセス要求の解は，I_{19}，I_{18}を通して，ＧＣＳ／ＬＣＳ問合せ分割システムに返される。

6) 最終的に，I_{17}を通してＥＸＳ／ＧＣＳ問合せ変換システムに解はわたされ，ＥＸＳユーザにI_{21}を通して出力される。

上述したように，ＤＤＢＳの設計，運用において，ＤＤ／Ｄは中心的な役割を持っている。ＤＤＢＳのスキーマ層，スキーマ層間のマッピングについての情報は，設計時に生成されＤＤ／Ｄに格納される。ＤＤＢＳの運用においては，この情報を用いて，アクセス要求の処理を行ない，必要な結果をユーザに返すことができる。

図3.3，3.4における各機能とインタフェースの概要は，表3.1，3.2に各々まとめてある。

表 3.1　ＤＤＢＳ機能－(1)　処理機能

処理機能	説　　明
ＬＩＳＰ	ＬＩＳのソース形式からオブジェクト形式及び表示形式の生成
ＬＣＳＰ	ＬＣＳと，ＩＣＳ／ＬＩＳ対応情報のソース形式からオブジェクト及び表示形式の生成
ＧＣＳＰ	ＧＣＳとＧＣＳ／ＬＣＳ対応情報とのソース形式から，オブジェクト形式と表示形式の生成
ＥＸＳＰ	ＥＸＳと，ＥＸＳ／ＧＣＳ対応情報とのソース形式から，オブジェクト形式と表示形式の生成
EXS／GCS transform	ＥＸＳアクセス要求を，ＥＩを用いてＧＣＳアクセス要求に変換する。 ＧＣＳアクセス要求の結果からＥＸＳアクセス要求の結果を生成する。
GCS／LCS transform	ＧＣＳアクセス要求を，ＤＩを用いてＬＣＳアクセス要求に変換するとともに，必要な通信処理を行なう。 ＬＣＳアクセス要求結果から，ＧＣＳアクセス要求に対する結果を生成する。
LCS／LIS transform	ＬＣＳアクセス要求を，ＨＩを用いてＬＩＳアクセス要求に変換する。 ＬＩＳアクセス要求結果から，ＬＣＳアクセス要求結果を生成する。
LIS／DBS transform	ＬＩＳアクセス要求をＤＢＳアクセス要求に変換する。 ＤＢＳアクセス要求結果からＬＩＳアクセス要求結果を生成する。
ＤＢＳ	データベースシステム データベースに対する物理的なアクセスを行ない，結果を生成する。

表 3.1　ＤＤＢＳ機能－(2)　ＤＤ／Ｄ

ＤＤ／Ｄ情報	説　　明
ＥＩ	ＥＸＳとＥＸＳ／ＧＣＳマッピング情報
ＤＩ	ＧＣＳとＧＣＳ／ＬＣＳマッピング情報 ＬＣＳとその所在情報
ＨＩ	ＬＩＳ，ＬＣＳ及びＬＣＳ／ＬＩＳマッピング情報

表 3.1　ＤＤＢＳ機能－(3)　管理者

管理者	設計アプローチ	
	統合型設計	分割型設計
ＡＡ	ＧＣＳから，ＥＸＳの生成 ＥＸＳ，及びＥＸＳ／ＧＣＳ対応情報の定義	アプリケーションの必要とするＥＸＳの生成
ＧＡ	ＬＣＳの集合から，ＧＣＳを統合する。 ＧＣＳ，及びＧＣＳ／ＬＣＳマッピング情報の定義	ＥＸＳの集合から，ＧＣＳを統合する。 ＧＣＳ，及びＥＸＳ／ＧＣＳマッピング情報の定義 ＧＣＳをＬＣＳの集合に分割する。 ＧＣＳ／ＬＣＳマッピング情報の定義．
ＬＡ	ＬＩＳからＬＣＳの生成 ＬＣＳ，及びＬＣＳ／ＬＩＳマッピング情報の定義	ＬＣＳからＬＩＳの生成 ＬＩＳ，及びＬＣＳ／ＬＩＳマッピング情報の定義
ＤＢＡ	ＬＩＳの定義	ＬＩＳに基づいて，ＤＢを生成

表 3.2　ＤＤＢＳインタフェース (1)

インタフェース番号	説明
1	ＧＣＳ ソース形式
2	ＧＣＳ オブジェクト形式
3	ＧＣＳ 表示形式
4	ＥＸＳ ソース形式
5	ＥＸＳ オブジェクト形式
25	ＥＸＳ 表示形式
6	ＬＣＳ 表示形式
7	ＬＣＳ ソース形式
8	ＬＣＳ オブジェクト形式
9	ＬＩＳ 表示形式
10	ＬＩＳ ソース形式
11	ＬＩＳ オブジェクト形式
12	ＥＩ
13	ＤＩ
14	ＮＩ
15	ＨＩ

表 3.2　ＤＤＢＳインタフェース (2)

インタフェース番号	設明
16	ＤＢＳ　ＤＤ/Ｄ
17	ＧＣＳ アクセス言語
18	通信処理記述言語
19	ＬＣＳ アクセス言語
20	ＬＩＳ アクセス言語
21	ＥＸＳ アクセス言語
22	ＧＣＳ アクセス言語
23	ＬＣＳ アクセス言語
24	ＤＢＳ アクセス言語

4.DDBS設計問題

4. ＤＤＢＳ設計問題

4.1 ＤＤＢＳの設計とは

ＤＤＢＳ設計のための gross architecture として四層スキーマ構造（ＦＳＳ）を想定する。

次の３つのデザイン objectives が考えられる。

(1) ＤＤＢＳでサポートされるデータベース（分散ＤＢ）の論理設計

(2) ＤＤＢ management system の設計

(3) data communication network の設計

以下，これら３項目につき，各項毎により詳しくそのアウトラインを述べる。

4.2 分散ＤＢの論理設計

ＤＤＢＳには大別して次の２つの発生形態がある。

(1) top-down 型

ＤＢが巨大化することにより極端に低下する manageability を救うため（ＶＬＤＢに対する反省）やデータの中央集権化を嫌う humanity からの要求として，ＤＢを分散しようとする形態である。

(2) bottom - up 型

独立して存在していたＤＢを統合して新たなＤＢを構築し，従来なし得なかった新しいサービスに対処したいという要求から生れた形態である。

以下，これから２つの形態の各々の場合についてＤＤＢの論理設計法の概略を示す。

4.2.1 top-down 型

全体的イメージとしては最初ＧＣＳが論理設計され，次いでＬＣＳ，ＬＩＳが論理設計される。一方，ＥＸＳが application oriented に設計される。このとき各スキーマ間の構造的，操作的写像関係が定義されねばならない。

step 1* 構築しようとする実世界（組織体）の意味的構造を記述するにふさわしいデータモデル（例えば，Entity - Relation ship モデルかもしれないし，他のモデルでもよい）をえらび，これを使ってモデル化する。（このモデル化は単独のデザイナーが行なうかもしれないし，あるいは複数のユーザが view を提示し（そのモデルで書かれている），それらを全体として矛盾がないよう統合して得られるかもしれない。）

＊ すでに組織体にあるデータベースが出来上っていて，これを単に分散したいというケースではこのステップと次のステップ2は省略される。

これをmeta ＧＣＳ，略してＭＧＣＳと呼ぶことにする。

step 2　ＧＣＳにどのようなデータモデルを採るか**（例えば，relational model）を決定する。ＭＧＣＳをそのモデルに変換してＧＣＳを得る。

step 3　ＧＣＳをsite-orientedに（重複を許して）分割する。その結果｛ＬＣＳ$_i$｝が出来上がる。このときＧＣＳと各ＬＣＳ$_i$間の構造的，操作的変換写像が確立する。

step 4　各ＬＣＳ$_i$について，サイトｉで許されるデータモデル，例えばrelational network，あるいはhierarchical モデル等，に変換し，ＬＩＳ$_i$を定義する。このとき各ｉについてＬＣＳ$_i$とＬＩＳ$_i$間の構造的，操作的変換写像が確立する。

step 5　applicationのためのＥＸＳ（view）を決め，user interface としてそれをＧＣＳ上に設計する。ＥＸＳ（一般には複数種類ある）とＧＣＳ間の構造的，操作的変換写像が確立する。

4.2.2　bottom－up 型

この場合，各ＬＩＳ$_i$がgivenであると考えられる。これらからＧＣＳを作りあげ，そのＧＣＳ上にＥＸＳを作りあげることがＤＤＢの論理設計である。各スキーマ間の構造的，操作的変換写像を確立しなければならないことはtop－down 型の場合と同様である。

step 1　ＬＣＳ$_i$およびその統合としてのＧＣＳを記述するためのデータモデルを選ぶ。***

step 2　上のステップで選ばれたモデルでＬＩＳ$_i$をＬＣＳ$_i$に変換する。（同種化）。ＬＩＳ$_i$とＬＣＳ$_i$間の構造的，操作的変換写像が確立する。

step 3　｛ＬＣＳ$_i$｝を統合して，ＧＣＳを作り上げる（統合化）。このとき各ＬＣＳ$_i$とＧＣＳ間の構造的，操作的変換写像が確立する。

step 4　applicationのための｛ＥＸＳ$_j$｝を決定し，それらをＧＣＳ上にサポートするようＧＣＳと各ＥＸＳ$_j$間の構造的，操作的変換写像を確立する。

＊＊　ＧＣＳのデータモデルには，実際システムがサポート可能なモデルでなければならない。現在relational モデルはSYSTEM Ｒや　INGRES　等が実現されているが，Ｅ－Ｒモデルをサポートするシステムは現存しない。

＊＊＊　ＬＣＳｉとＧＣＳは同一のデータモデルで記述される。このモデルはＧＣＳ，ＬＣＳｉがＤＤＢＭＳで管理されるために，machine 上でサポートされうるモデルであることが必要である。そうでなければ，そのＧＣＳをサポート可能なＧＣＳに変換（丁度，4.2.1節のstep 1 に相当）するステップを新たに必要としてしまう。

4.3 ＤＤＢ論理設計上の今後の課題

top - down 型，bottom - up 型ともに抱えている課題の主なるものは次の通りである。

(1) データモデル，データベースの（等価）変換技術の確立

(2) データモデル，データベースの分解，統合技術の確立

(3) application モデルをサポートするための view サポート技術の確立

(4) 設計のためよい言語，よいモデルの開発

4.4 ＤＤＢＭＳの設計

分散データベース管理システムが管理する対象を大別すると次の２つである。

(1) ＥＸＳ$_j$，ＧＣＳ，ＬＣＳ$_i$ 等のスキーマの管理

(2) 各種制御システムの管理

以下，これら２つについて更に詳しく設計対象を述べる。

4.4.1 スキーマ管理機構の設計

(1) ＥＸＳ$_j$，ＧＣＳ，ＬＣＳ$_i$，ＬＩＳ$_i$ に関する各種スキーマ情報の管理機構を設計する。より具体的には各スキーマの定義，スキーマ間の構造的，操作的変換写像，付随するスキーマの保全制約（integrity）の静的および動的（query 処理時）な管理を行なう機構の設計。

(2) ＥＸＳ$_j$ に発せられる query を各サイトへの subquery に分割する機構の設計。

(3) Directory（分散情報）の管理機構の設計。

4.4.2 各種制御管理機構の設計

(1) Commitment control 機構の設計。これはたとえＤＤＢＳに１個の application program しか存在しない場合でも考察しなければならない分散システムに固有の管理機構である。

(2) Concurrency control 機構の設計。これはＤＤＢＳに複数の application programs があるとき問題となる。勿論，分散ではなく集中ＤＢＳでも問題となるが，データが物理的あるいは論理的に分散していることにより，従来に比べ問題の複雑さが大きくなっていると考えられる。

(3) 障害回復機構の設計

(4) Security と privacy 管理機構の設計。ＤＤＢＳでは，特に不特定多数のユーザが入り込む余地があることから，単一の集中ＤＢＳに比べ次の諸点に留意する必要がある。

(a) アクセス権の明確化

(b) データベース全体のSecurity

(c) データ伝送路でのsequrity

4.5 ＤＤＢＭＳの技術課題－パイロットシステム開発の提言－

各種スキーマの管理，あるいは各種制御を如何程精密に実行しなければならないかは，

(a) その管理システム開発に要する年月

(b) その管理システム開発に要する金額

(c) application model 等によると思われる。

現在のところ，例えばConcurrency controlの方法にしても数多くの提案がなされているが，一体どのようなapplicationのもとでどの方式で十分なのかといった具体的な見通しは何も得られていない。他の管理，制御機構についても同様な次元にあると思われる。設計されるＤＤＢＭＳは対象とした組織体の分散データベースシステムを適正に運用するにあたり，過少な能力しか持ち合わせていないのでは困るし，過大な能力を持ち合わせていれば資源の無駄使いである。

このような立場から，次の２つのいづれかを今後の課題として提言する。

(1) ＤＤＢＳシミュレーション，エミュレーション・システムの開発

(2) パイロットＤＤＢＳの開発

最後に，これらの課題を通して，ＤＤＢＳデザインaidの開発要求が出されることが十分予想される。

4.6 Data communication networkの設計

ＤＤＢＳを通信サイトからサポートするdata communication network の次の２点をＤＤＢＳ全体にかかるapplicationとしての負荷や各種管理のためのデータ伝送量の特性を吟味してデザインする。

(1) network topologyと回線容量の決定

(2) 各サイトの機能的構成の決定

4.7 その他－標準化－

top - down 型にしろ bottom - up型にしろ，ＤＤＢを構築するということは，データベースの異種性（heterogenity）をスキーマ間写像により吸収している。この写像は，実際システム運用時にはその実行がoverhead を意味する。ＤＤＢＳ全体として効率の良い運用をはかるには出来るだけ個々のサイトのデータベースおよびデータベース管理システ

ムが標準化されていることが望ましい。

また，実際にデータベースを作成する際も標準的入力装置の開発を行ない，専門家でない人々からも容易に誤りなくデータベースの作成が行なえることが，特に他人のデータをあてにするＤＤＢＳでは大事であろう。

末節ながら，ＤＤＢの論理設計，directoryの設計，あるいはＤＤＢＳデザインaidの開発にあたり，知識工学的手法の導入が大いに期待されることを記して本章を終える。

5.アクセス要求マッピング

5. アクセス要求マッピング

5.1 序

アクセス要求マッピングとは，四層スキーマ構造（ＦＳＳ）において，上位層に対して発せられたアクセス要求を，これの下位層のアクセス要求に変換することである。ここでアクセス要求とは，データベースに対する検索更新（追加，消去，挿入）といった問合せとする。

四層スキーマ構造のもとで，外部スキーマ（ＥＸＳ）に基づいた問合せ（外部問合せと呼ぶ）は，まず，全体概念スキーマ（ＧＣＳ）に基づいた問合せ（ＧＣＳ問合せと呼ぶ）又は，これのシーケンスに変換される。次に，ＧＣＳ問合せを，各サイト又はサイト間で実行可能なローカル概念スキーマ（ＬＣＳ）問合せのシーケンスに分割される。各サイトに出されたＬＣＳ問合せは，そのサイトで実行可能なローカル内部スキーマ（ＬＩＳ）問合せ又はそのシーケンスに変換される。〔図 5.1〕

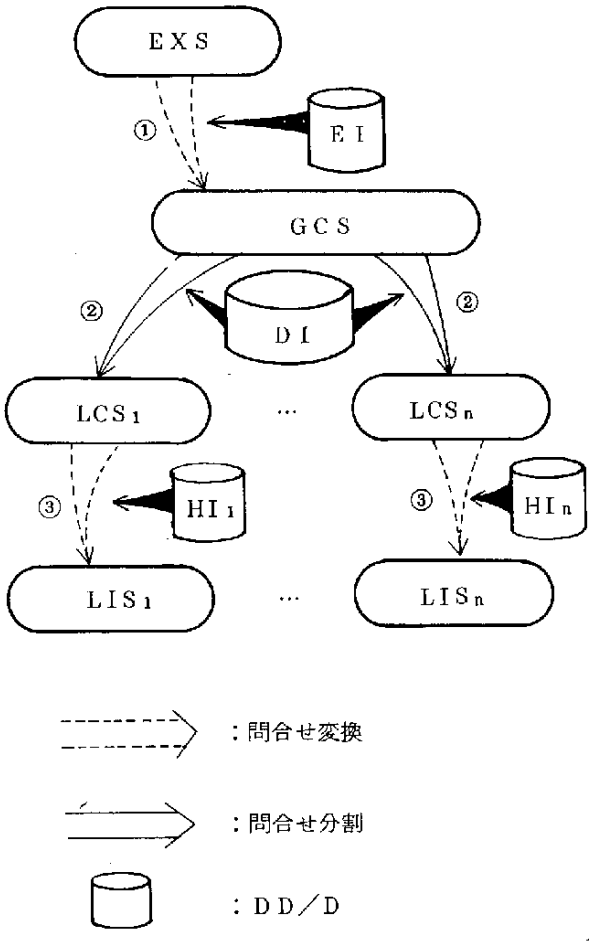


図 5.1　アクセスマッピング

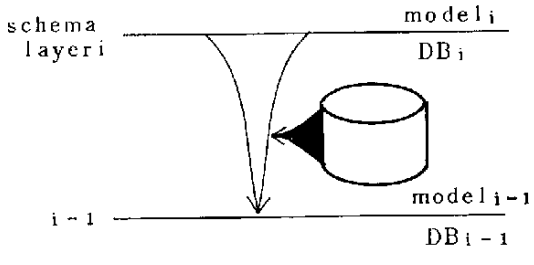


図 5.2　アクセスマッピング

ここで，２つの隣接したスキーマ層ｉとｉ－１があり，ｉからｉ－１層への問合せのマッピングを考えてみよう。modelα（ここでα＝ｉ or ｉ－１）をスキーマ層αを特徴づけるデータモデルとし，ＤＢαをこれのデータ（extension）とする。アクセス要求マッピングでは，次の２点を考えねばならない。

1) $model_i$ と $model_{i-1}$ の関係

－同じか違うか。

2) DB_i と DB_{i-1} との関係

$DBi \subseteq DBi\text{-}1,2 \cup DBi\text{-}1,2 \cup \cdots \cup DBi\text{-}1,n$

i.e. DBiは，複数のDBi-1,j（ｊ＝1,…，ｎ）から成っている。

－$DBi \subseteq DBi\text{-}1$

i.e. DBiは，DBi-1だけから成っている。

	マッピング	model	ＤＢ
問合せ変換	① q(EXS) → q(GCS) EI	$model_{EXC} \neq model_{GCS}$	$DB_{EXS} \subseteq DB_{GCS}$
	③ q(LCS) → q(LIS) HI	$model_{LCS} \neq model_{LIS}$	$DB_{LCS} \equiv DB_{LIS}$
問合せ分割	② q(GCS) → q({LCS}) DI	$model_{GCS} \neq model_{LCS}$	$DB_{GCS} \subseteq \bigcup_i DB_{LCS_i}$

図 5.3　アクセス・マッピング

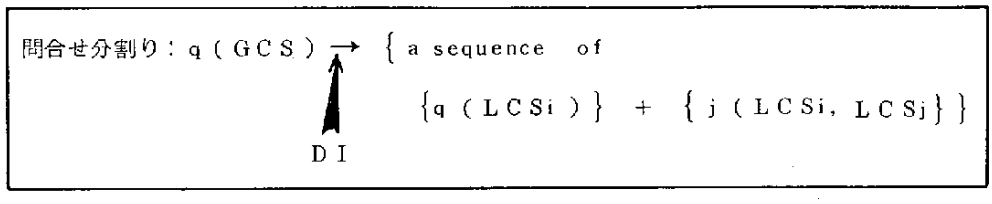
前述した３種のアクセスマッピングをまとめると図5.3のようになる。即ち，問合せ変換とは，２つの層のモデルが異なっているが，上位層の問合せの参照するデータは，下位層の１つのデータにユニークに対応している。

例えば，リレーショナル問合せから，ＤＢＴＧ ＤＭＬプログラムを生成することである。

これに対して，問合せ分割では，２つの層のモデルは互いに等しいが，上位層で参照されるデータ集合は，下位層の複数の（サイトの異なった）データ集合から成っている。

5.2　問合せ分割

問合せ分割とは，ＧＣＳ問合せを，各サイト単位のＬＣＳ問合せと，サイト間の結合処理とのシーケンスに変換することである。



前述したように，問合せ分割では，ＧＣＳとＬＣＳとのモデルは等しい。しかし，ＧＣＳ問合せの参照するデータ集合は，異なったＬＣＳデータ集合から成っている。このために，問合せ分割は，次の２つのプロセスから成り立つ。

1) ＧＣＳ問合せの意味を，対応するＬＣＳ要素によって表現する。例えば，ＧＣＳリレーションを，対応するＬＣＳリレーションによって置き換える。

2) サイト内／間での通信処理を行なう。例えば，異なったサイトに存在する２つのＬＣＳリレーションの結合を行なうために，一方を他方にネットワークを用いて転送する。

ここで，前者を表現変換，後者を通信処理と呼ぶことにする。

5.2.1 表現変換

表現変換とは，ＧＣＳ問合せの意味をＬＣＳ要素によって表わすことである。ＧＣＳ要素とＬＣＳ要素との対応情報は，分散情報（ＤＩ）内に格納されている。表現変換は一般的に図 5.4 のように表わせる。

表現変換：

$$GQ(g_1, \cdots, g_n) \longrightarrow GQ(GD_1(\ell_{11}^{S_{11}}, \cdots, \ell_{1k_1}^{S_{1k_1}}), \cdots, GD_n(\ell_{n1}^{S_{n1}}, \cdots, \ell_{nk_n}^{S_{nk_n}}))$$

$$\{ g_i = GD_i(\ell_{i1}^{S_{i1}}, \cdots, \ell_{ik_i}^{S_{ik_i}}) \mid i = 1, \cdots, n \} \subseteq DI$$

GCS definiton

where

GQ ≙ GCS問合せ（の意味）

g_i ≙ GCS object（$i = 1, \cdots, n$）

GD_i ≙ g_i の定義式（DI内に storeされている）i.e. ＧＣＳとＬＣＳとの対応情報

$\ell_{ij}^{S_{ij}}$ ≙ サイト S_{ij} にある LCS object

S_{ij} ≙ サイト $S_{ij} \in \{1, 2, \cdots, m\}$

よって，必ずしも $\ell_{ij}^{S_{ij}} \neq \ell_{k\ell}^{S_{k\ell}}$（ここで $i \neq k$，$j \neq \ell$）とは限らない。

図 5.4　表現変換

ＧＣＳとＬＣＳとが，ともにリレーショナルモデルに基づいた場合を考えてみる。ＧＣＳリレーションは，ＬＣＳリレーションの射影，制限，結合，和によって表わされる。もし，ＧＣＳリレーションがＬＣＳリレーションの射影，制限，結合とから成っていれば，問合せ変形（query modification ）手法を用いて表現の変換を行なえる。リレーショナル計算言語で，和（union ）をどう表わすかは今後の問題である。 又，∃，∀といった限定作用素を問合せが含む時，どのように表現変換するかも今後の課題である。

今まで，検索について考えてきたが，ＧＣＳ更新の意味を，どのようにＬＣＳ層に伝えるかは重要な課題である。ＧＣＳリレーションは，ＬＣＳリレーション上に定義された視野と構文的に等価であるので，いわゆる更新異常問題が生じる。これに対する１つの解は，ＧＣＳリレーションを抽象データ型として，これに対して意味的に許される更新演算を抽象演算として定義し，この演算を通してのみ更新を行なわせることである。このデータ抽象化手法は，統合的なスキーマ層定義に有効であると考えられる。

問合せが aggregate 関数をふくむ時の処理も今後の課題である。 aggregate 関数としては，独立して処理できるものと，出来ないものとがある。

5.2.2 通信処理

表現変換されたＧＣＳ問合せを，全体ＬＣＳ問合せと呼ぶ。全体ＬＣＳ問合せは，ネットワークで結合された複数サイトを，一般に参照している。このために，各サイト内／サイト間での処理が必要となる。これを通信処理と呼ぶ。

通信処理は，

1） 各サイトの処理プロセッサ（即ち，データベースシステム）と，

2） 各サイト間の通信ネットワーク

とを用いて行なわれる。現在のコンピュータシステム技術において，2）のネットワークは，その速度と容量の小ささから，システムの主要なボトルネックとなっている。一方，各サイトが処理能力を持つことから，各サイトでの並列処理が可能である。このことから，通信処理は，次の目標を達成する必要がある。

1） 通信コストの最少化

2） 応答時間の最少化（並列度の最大化）

更に，実際の運用面から，次の目標を達成する必要がある。

3） 通信処理に必要な分散情報（ＤＩ）の最少化と静的化。

通信処理を考えるうえで，ネットワークの役割は重要である。ネットワークには，大別して次の２種類がある。

1） パケット交換ネットワーク

2) 共有媒体ネットワーク

1)では，ネットワーク内に，同時に複数のデータ・パケットが存在し得るが，ポイント－ポイント形式の転送を行なう。2)では，放送転送が可能であるが，同時にただ１つのデータ・パケットが存在し得るだけである。2)の例としては，光ファイバー，無線通信があり，主にローカルネットワークとして用いられる。

現在まで，前者のネットワークについて，多くの通信処理アルゴリズムが検討されてきている。１つは，Yao，S. B. やWong，E. 等による手法である。これらは，結合選択度（join selectivity）に基づいて，生成される中間結果の見積りを行ない，通信コスト又は応答時間を最少とするような通信処理スケジュールをヒューリスティクスに見つけるものである。これを静的決定アルゴリズムと呼ぶ。この問題点として，属性の値が均一分散しており，属性間には値分散の依存関係がないという前提に基づいた結合選択度（特に$<$，$\leq$，$\geq$，$>$といった不等価結合の選択度）の確かさがある。もう１つは，ネットワークの負荷状態によって動的に生じる待ち行列遅延を，どのように静的決定のなかに取り組めるかという問題である。

この問題への１つの解は，システムの実行状況をモニタしながら，次の通信処理スケジュールを決める動的決定アルゴリズム（Takizawa，M）がある。この手法では，結合選択度，カーディナリティといった動的性質を持ったパフォーマンス情報を不要とするという利点がある。しかし，これは最適解を保障しない。

より正確な決定を行なうためには，より多くの統計情報を必要とすることになる。決定の正確さと。必要情報（即ちＤＩ）の量と管理とのトレードオフが主要問題となってくる。さらに，通信ネットワーク上に生成される待ち行列による動的な要素が，重要な要因となる。

上述のようにして決定された通信処理は，次の２つの処理から成る。

1) サイト内処理 ── そのサイトで独立して行なえる処理。e.g. そのサイト内のリレーションに対する結合，制限，射影。サイトのＤＢＳのモデルが異なれば，問合せ変換を行なう。

2) サイト間処理 ── サイト間で，互いに協同して行なう処理。サイト間結合では一方を他方に転送する必要がある。各サイトは，ＬＣＳレベルの処理（e.g. relational ＤＢＳ）を出来るサイト間処理のためのプロセッサを持たねばならない。

通信処理は，まず可能なサイト内処理を全て行ない，転送量を出来るだけ減少させた後に，サイト間処理を行なう。これは，図5.4を用いて，図5.5のように示せる。サイト間処理ＪＪは，図のように，結合（$\bowtie$）演算をノードとする２進木によって表わせる。

前述した通信処理アルブリズムは，この木をどのように決定するかに関している。

通信処理：

$$CQ(GD_1(\ell_{11}^{s11} \cdots \ell_{1k_1}^{sk_1}), \longrightarrow JJ(L_{11}(\ell_{11}^{1} \cdots \ell_{11}^{m_{11}}), \cdots,$$

$$\cdots, GD_n(\ell_{n1}^{sn1}, \cdots, \ell_{m_{kn}}^{smkn})) \quad , Lsps(\ell_{sps}^{1} \cdots \ell_{sps}^{m_{sps}}))$$

ここで

$L_{ij} \triangleq$ サイトi での処理

$\ell_{ij}^{1}, \cdots, \ell_{ij}^{m_{ij}}$ は互いにJoin ℓink とによって連結な LCSリレーション。 他 $\ell_{ik}^{k'}$ (where $j \neq k$)とは非連結

$JJ \triangleq$ サイト間処理

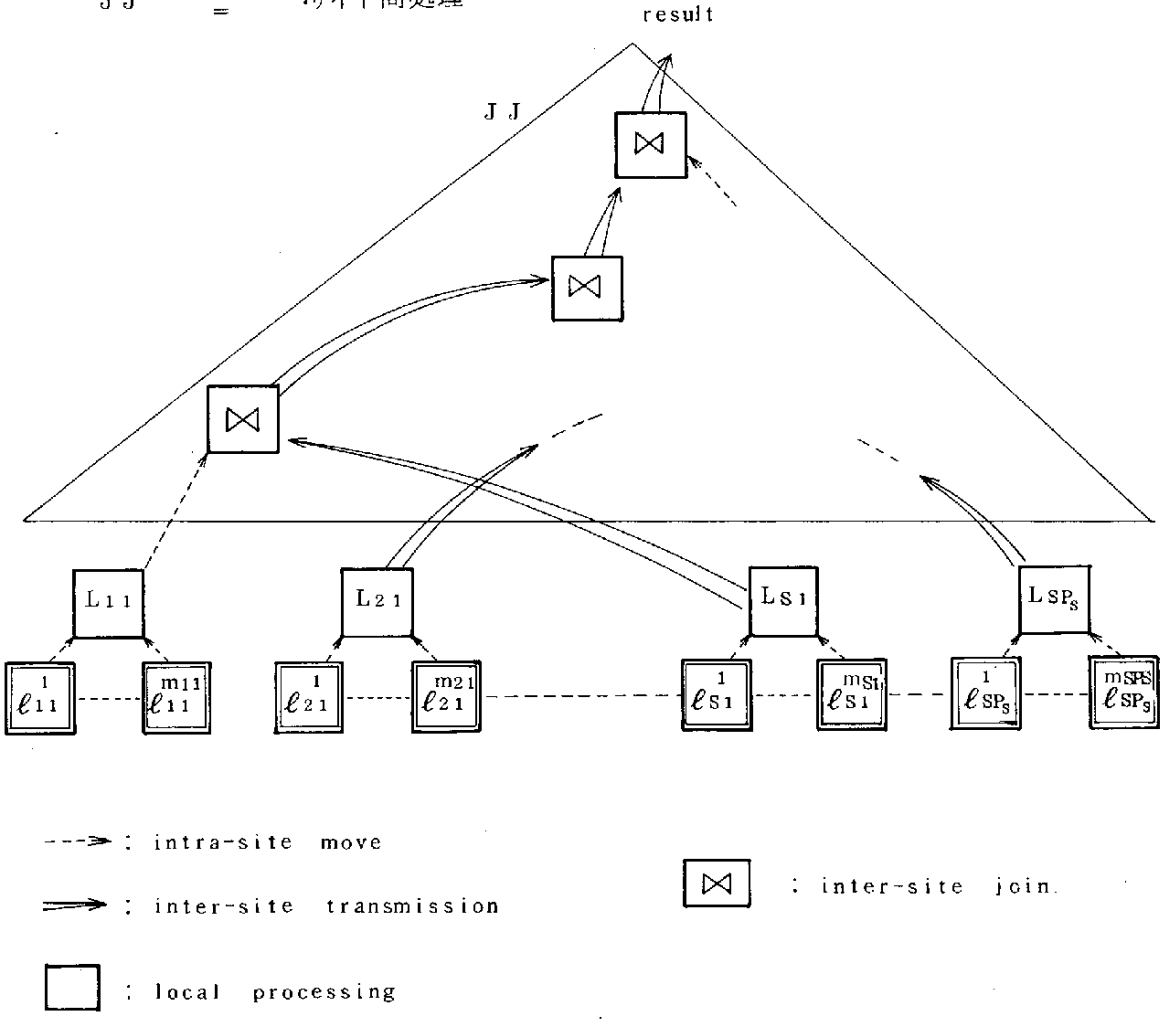
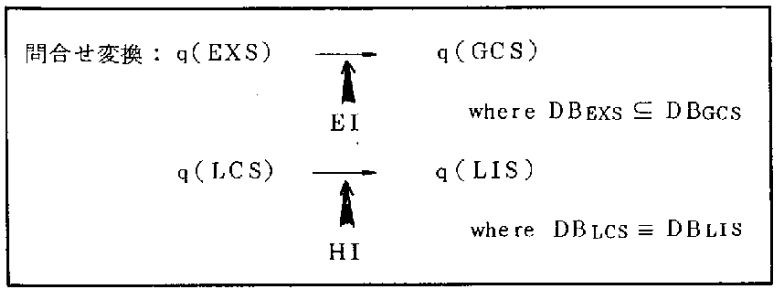


図 5.5 通信処理

今後の課題として，共有媒体ネットワーク上での通信処理アルブリズムの開発がある。ここでは，1のリレーションを複数サイトに同時に転送できる。結合リンク数の次数が多い複雑な問合せには有効になる。又，データベースマシンとの関連でも重要である。

5.3 問合せ変換

問合せ変換は，あるモデルに基づいた問合せを，他のモデルに基づいた問合せに変換することである。



問合せ変換としては，上記するようにＥＸＳ問合せをＧＣＳ問合せへの変換と，ＬＣＳ問合せをＬＩＳ問合せへの変換との２種がある。前者では，ＥＸＳはＧＣＳの部分集合（i.e. $DB_{EXS} \subseteq DB_{GCS}$）であり，後者ではＬＣＳとＬＩＳとは等価（i.e. $DB_{LCS} \equiv DB_{LIS}$）である。

問合せ変換では，上述したモデルの相違とともに，問合せ言語が非手続的か，手続き的（procedural or navigational）かが問題となる。よって，次のような変換パターンがある。

i) procedural ⟶ procedural

ii) procedural ⟶ non-procedural

iii) non-procedural ⟶ procedural

iv) non-procedural ⟶ non-procedural

このように，問合せ変換を行なうためには，次のようなプロセスが必要となる。

1) 問合せが基づいているモデル構造の変換。隣接する２つの層の対応情報（ＥＩ，ＨＩ）を用いて変換する。例えば，リレーショナル問合せの参照するリレーションをＤＢＴＧのレコード型で置き換える。

2) 必要な procedural 又は non-procedural な問合せ記述を生成する。

1)は，スキーマ層の設計に依存している。2)では，下位層レベルでのアクセスの最適化が必要となる。

ＬＣＳ（リレーショナルモデル）問合せからＬＩＳ（CODASYL DBTGモデル）

プログラムの生成について考えてみよう。

1) モデル構造変換

リレーショナル問合せをグラフ表現して，問合せの参照するリレーションを対応するレコード型又はセット型で置き換える。これによって，ＤＢＴＧモデルによる問合せの非手続き的表現が生成されたことになる。

2) アクセスパスの生成 （ non-procedure → procedure ）

グラフをサーチしてアクセスパスを表わす木を生成する。この木は

i) 中間結果数を最少とするように，かつ

ii) アクセスされるオカーランス数を最少とするようにつくられる。

このようにして，ＱＵＥＬ問合せからＣＯＢＯＬ ＤＢＴＧプログラムを生成できる〔 Takizawa，M. 〕。

問題点として，更新要求をどのように下位層に伝えるかがある。この問題は，問合せ分割における場合に加えて，更にモデルが相違している点である。各データモデルの保有する（インテグリティ）構造のどこまでが，他のモデルによって表わせるかが問題になる。リレーショナル，階層，ネットワークといったモデルの一般構造を，より概念的な概念モデル（e.g Ｅ－Ｒモデル）（これを設計モデルと呼ぶ）によって表わすことが必要である。逆に言えば，各モデルの構造のなかで，この設計モデルによって表わし得る部分について考えることが必要である。

問合せ分割と同様に，限定作用素（i.e.∃，∀），aggregate 関数の変換が問題となる。

5.4 今後の課題

問合せ分割では，次の点が今後の問題としてある。

1) 式の変換

－限定作用素（∃，∀）の処理

∀を含む問合せを，各サイトに分割することは困難である。

－ aggregate 関数の処理

限定作用素の処理と同様の問題がある。

－更新の意味を，ＬＣＳ層で正しく表わせるか。データ抽象化技法が１つの解となる。

2) 通信処理

－通信処理スケジュールの決定の正確さと，決定のための情報の量と性質との間のトレードオフ。統計情報の正しさの検証。

－通信ネットワークの速度の向上。放送機能の実現。

3) トランザクションの処理

トランザクションを，問合せ（検索，更新）から成る手続とする。この時，１つのトランザクション内で１つのデータが複数回通信処理されることになる。こうした手続をどのように通信処理するか（最適化するか）は今後の課題である。

4) 対話手法

ユーザが対話的に処理を行なう時，3)と同様な問題が生じる。

問合せ変換では，次のような問題点がある。

1) モデル変換手法の確立

問合せが基づくモデル構造を，他のモデルに変換するためには，この２つのデータモデルの等価変換が必要となる。又，種々のデータモデルに対して，変換手法が確立されねばならない。

2) 更新変換

1)の手法の確立が必要である。

3) 非手続き問合せから手続の生成（最適化），下位層（ＬＩＳ）での最適化の目標の設定が必要である。

4) 手続から非手続の生成

ＥＸＳからＧＣＳの変換には必要である。

6.ディクショナリ/ディレクトリ

6. ディクショナリ／ディレクトリ

6.1 ディクショナリ／ディレクトリ概要

データベースはその構造を形式化した記述を伴っている。この記述はデータベースの操作（生成，検索，更新，削除等）における処理過程で利用される。

データベースにはデータベース利用の検討，開発，構築，運用と保守のライフサイクルがあり，システムの管理を効率よく行うためには単にデータベース内部の形式的記述のみでは不充分である。ライフサイクルの全般に渡ってのシステム管理の効率化にディクショナリ／ディレクトリは大きな役目を果す。

データディクショナリはデータエンティティの意味に関する情報を与え，データディレクトリはデータエンティティや関係づけの物理的属性や格納場所に関する情報を与える。データディクショナリ／ディレクトリシステム（ＤＤ／ＤＳ）はディクショナリ／ディレクトリ情報を一部文章表現も含め形式的に記述したデータを管理する。ＤＤ／ＤＳをデータベースライフサイクルにどの程度密着させるか，分散データベースでのＤＤ／ＤＳの役目は何かを検討する必要がある。後者を検討するため，

(a) 分散データベースシステム環境の分類

homogeneous ／ heterogeneous

directory 制御方式（集中／分散）等

(b) システム関係者（関係機能）の分類

エンドユーザ

分散システムのシステム管理者

分散データベースのデータベース管理者

(c) システム関係者への提供サービスの分類

定型／非定型，問合せ／更新，visible ／ guided ／ invisible

…… エンドユーザ

資源，運用管理 …… システム管理者

開発，構築，検査，保守支援 …… データベース管理者

に従って考察することが必要と思われる。

6.2 ディクショナリ／ディレクトリ情報

ディクショナリ／ディレクトリ情報は以下３つの異なるタイプの情報からなると考えられる。

(a) 外部情報（External Information）

応用システム自体の意味的，論理的情報及び応用システムと分散データベース全体との意味的，論理的結合を示す情報。

(b) 分散情報（Distribution Information）

分散データベース自体の意味的，論理的情報及び分散したローカルなサブシステムとの意味的，論理的結合を示す情報。

(c) 異種性情報（Heterogeneity Information）

ローカルな概念スキーマと特定データベース（異種なものの中の１つ）の論理サブ構造との結合に関する意味的，論理的情報及び論理サブ構造と特定物理構造を結合するディレクトリ情報。

以上は，heterogeneous な分散データベースを前提としている。

外部情報は分散データベースのデータベース管理者と応用システム管理者によって作成され，エンドユーザによって利用される。

分散情報はデータベース管理者の要求のもとに分散データベースのシステム管理者によって作成され，運用時の管理を行う。利用はエンドユーザによって直接的又は間接的に行なわれる。

異種性情報はデータベース管理者によって作成され，以後の運用時の管理が行なわれる。利用はエンドユーザによって直接的又は間接的に行われる。

6.3 分散データベースにおけるＤＤ／ＤＳの支援

(a) 意味情報の有効利用と高度な処理支援

分散データベースシステムでは特に顕著になると思われることは，ＶＬＤＢにも見られる分散した巨大な量のデータ，それも多種，錯綜するものを有効に活用することは計算機利用に精通しているだけでは行えなくなることであろう。情報の内容と情報処理サービスに対する要求が高度化，多様化，複雑化することによると思われる。この対策としてディクショナリに含まれる意味情報の充実が必要となろう。問題は正確さ，完全さ等において意味はどの程度まで形式化できるか，又はすべきなのか，人間との関わりをどの程度のレベルのサービスで実現するのかを充分検討せねばならない。

(b) 分散データベースでのエンドユーザ支援

分散データベースの利点はデータの可用性を増すことにあるとされるが，そこへ容易にしかも異常な性能低下なしに至る必要がある。特に，非定型なデータベース利用や計算機利用やデータの詳細な内容に精通していないユーザによる支援が大切である。

(c) 分散データベース運用でのシステム管理支援

分散データベースシステムにおける資源の有効利用をディレクトリ情報を活用して行うことも大切である。ローカルなサブシステム同志を有機的に結合させることもこれに含まれる。

(d) 分散データベースの開発，改訂支援

複雑，大規模なシステムの開発，改訂には段階的な開発を行う必要があり，ソフトウェア工学的にも開発の初期からの文書化の必要が伴う。ディクショナリ情報は開発初期から文書としての役目だけでなく，データ相互間の関係等意味に関する有効な情報を与える。分散データベース，特に heterogeneous な場合はこれらの情報の価値は高くなる。

6.4 ＤＤ／ＤＳの課題

以上，述べてきたＤＤ／ＤＳの機能を充分実現するためには，以下の課題の検討が行われる必要がある。

(a) 概念スキーマ，外部スキーマの記述法

（完備性，無矛盾性等を満足すること）

(b) 異種性情報と相互変換則の記述法

（完備性，等価性等を満足すること）

(c) より一般的なデータモデルとそれによる意味記述法

(d) 意味情報の人間とのインタフェイス

（人間による判定の度合決定）

(e) 上記記述法による記述の正しさの検証

(f) キーワード検索のためのマシン開発

7.DDBSの制御問題

7. ＤＤＢＳの制御問題

7.1 目　的

分散データベースにおいて，トランザクションの制御を行う目的には次の２つがある。

ひとつは，どのような状況下においても，常にデータベース内データのコンシステントなビューを与えるための制御であり，他のひとつは，オーソライズされないユーザのイリーガルなアクセスに対しデータを保護するための制御である。コンシステントなビューを与えるための制御はデータの更新と深くかかわるものであり，データ保護のための制御は，アクセス権チェック，暗号化／復号化といった問題にかかわるものとなる。

7.2 データベースの更新と制御問題

分散データベースにおいて，データの更新を行なう場合，その適応業務により２つのタイプの更新形態が存在する。ひとつは，バンキング，座席予約等のシステムにみられるような，実時間性の厳しい更新であり，データのコンシステンシィ維持のために，システムによる何らかのトランザクション制御機構が必要となる。これに対し，各種文献検索システム，情報案内システムにみられるような，比較的実時間性のゆるい更新においては，コンシステンシィ維持を運用上の問題に帰することが可能となる。

7.3 データベースにおける制御

前述のように，即時的な更新を行なうシステムでは，システム内に，トランザクション制御のための機構が必要となる。この制御機構には，分散配置されたひとつの仕事を完結させるために必要なコミットメント制御と，複数のプロセスが共有リソースをアクセスする時必要となるコンカレンシィ制御とがある。この２つの制御は，データベースの分散／集中にかかわらず必要なものである。

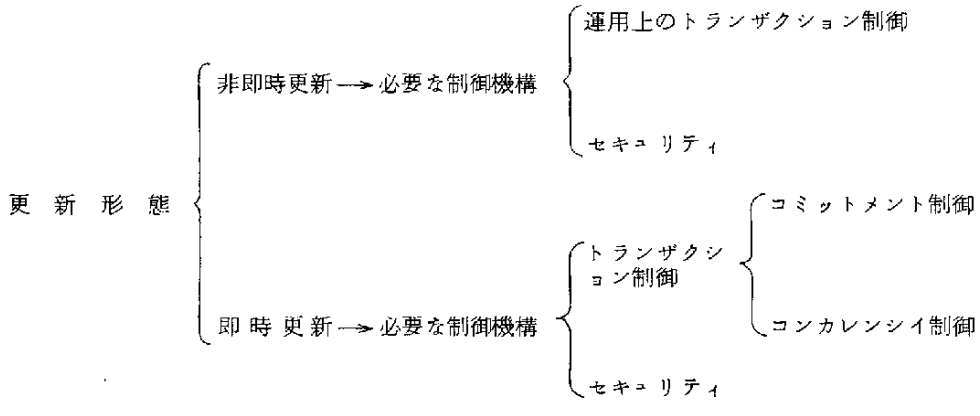


図 7.1　分散データベースの更新形態と制御

7.3.1　コミットメント制御方式

現在，INGRES，SDD－1をはじめとするほとんどすべての汎用分散データベースシステムでは，このコミットメント制御に，2フェーズコミット法と呼ばれる方式を採用している。これは，トランザクション処理を，セキュアフェーズとコミットフェーズの2つに分け，クリティカルな時間帯での異常事態発生のチャンスをできるだけ少くしたものである。2フェーズコミット法に基づくプロトコルのシーケンスを図7.2に例示する。

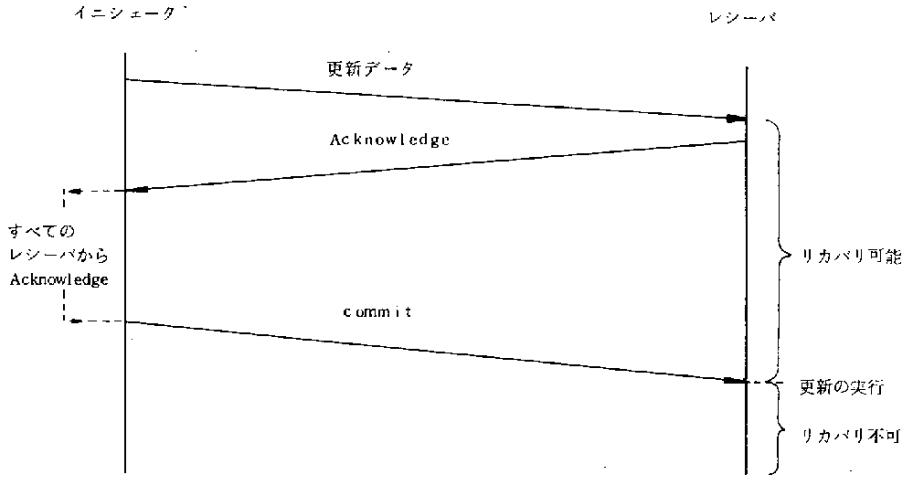


図7.2　2フェーズコミットシーケンス

なお，プロトコル実行中に障害が生起した場合，種々の対処の方式が現在までに提案されているが高信頼，高能率なアルゴリズムの研究については未だオープンな分野として残されている。

将来の分散データベースシステムの実現に当り，コミットメント制御の方式を，2フェースコミット法に基づくものとして問題はないが，障害処理については，以下のような諸機能の効率的な実現を計っておく必要がある。

① リライアブルブロードカスト機能

② 障害サイトのアイソレーション機能

③ 障害サイトの再開機能（ギャランティードデリバリ機能）

④ ネットワークパーティションの検出とアイソレーション機能

⑤ ネットワークパーティションからの再開機能

なお，このようなコミットメント制御方式を可能とするため，次のような技術課題を解決することが強く望まれる。

ひとつは，Ethernet あるいは，衛星通信のような手段を用いて可能となる高性能ブロードカストネットワークの実現技術である。このようなネットワークでは，コミッ

トメント制御におけるクリティカルな時間帯を著しく短縮でき，また制御に要する通信コストを大幅に軽減することも可能である。

次に必要な技術は，コミットメント制御のベースとなっている基本機能が通信機能として実現されているようなネットワーク・アーキテクチャの開発である。この基本機能には，ギャランティードデリバリ機能，リライアブルブロードカスト，障害サイトおよびパーティションネットワークのアイソレーションと再開機能，クロックの同期といったものが含まれることになる。

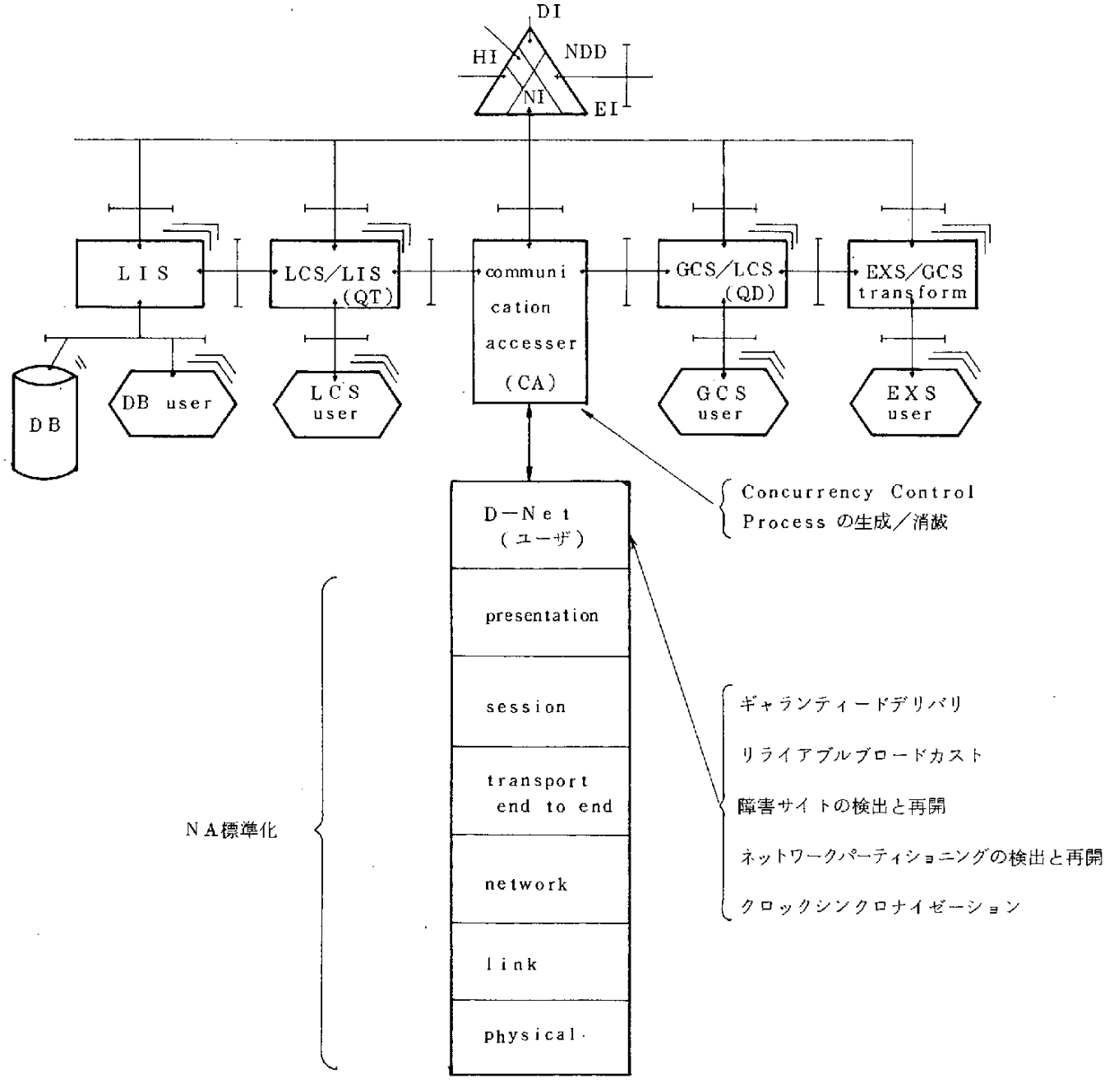


図 7.3 通信機能とＤＢＭＳ側の機能分担

また，さらにクリティカルな時間での障害を最小におさえるための信頼性技術の開発が強く望まれる。

7.3.2 コンカレンシイ制御

コンカレンシイ制御については，現状ではシステムにより様々な方式が採用されている。特に顕著な差異として，制御を単一のサイトで実行（集中型）するか，複数サイトにその役割を受持たせる（分散型）か，さらには，分散型制御を行う場合，従来の集中型データベースで行なわれているロック機能を拡張させるか，あるいはタイムスタンプと呼ばれるトランザクションの順序付け機構により制御を行なうのか，といったものがある。現状での，コンカレンシイ制御方式の類別を図7.4に示す。なお，スタティックなコンカレンシイ制御とは，あらかじめ入力されるトランザクションのコンフリクト関係を解析しておき，それに基づいてトランザクション実行のシーケンシングを行なうも

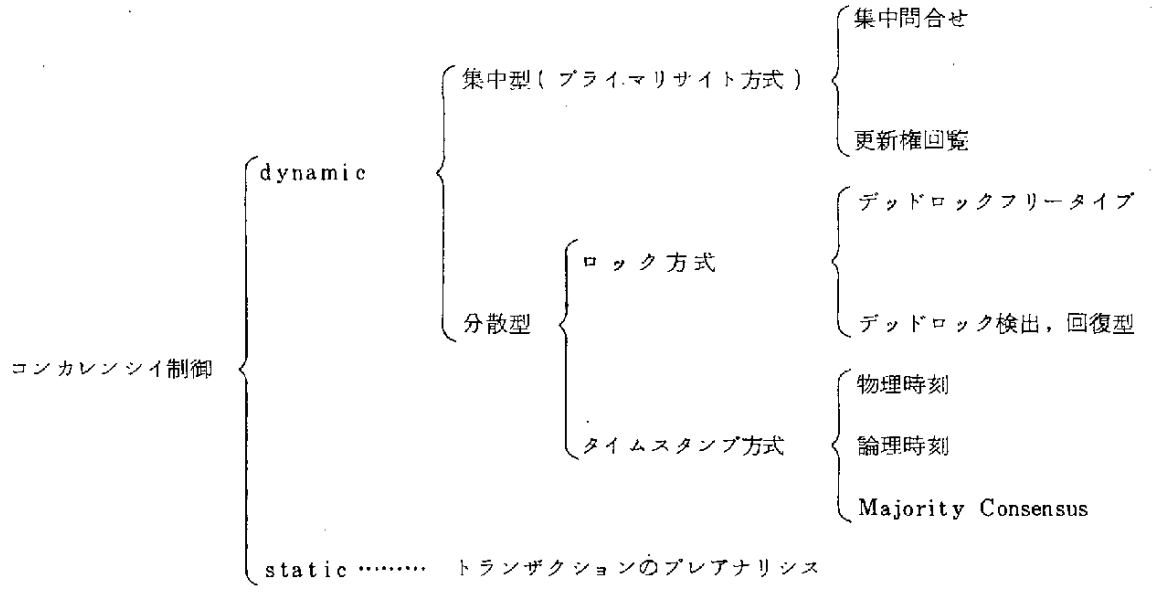


図7.4 コンカレンシイ制御方式の類別

ので，定型的な業務が大きな比重を占める応用分野においては，処理の並列性が増す非常に有効な方法となる。現状では，このスタティックな方式は，ＳＤＤ－１のみが（ダイナミック方式と併用の上で）採用している。

今後の汎用分散データベースシステムにおいては，**dynamic**，**static** 両方式の併用が，トランザクション処理の並列性を増すためにも，特に必要になってくると思われる。定型的な業務の多いアプリケーション分野では，トランザクション制御のほとんどをこのプレアナリシス方式によって解決でき，効率の良い処理が可能となる。一方，非定型

的な業務に関しては，その適用目的に応じて，ロック方式あるいはタイムスタンプ方式を採用するのが望ましい。集中型制御は，比較的小規模なネットワークでは望ましいものと思われるが，高信頼度，広域分散ネットワークにおいては，難点が有ると思われる。

なお，このような信頼性の高く，効率の良いコンカレンシイ制御を達成するためには次のような技術課題の解決が強く望まれる。

ひとつは，効率的なプレアナリシス手法の開発である。これにより分散システムの特徴である並列性を充分に生かした効率の良い制御が可能となる。また，ロック方式を採用した場合には，ロックのグラニュラリティ（ロックする単位）の大きさとパフォーマンスとの関係に関する解析を充分に行っておく必要がある。

7.3.3 セキュリティ

現状におけるセキュリティ機構は，そのほとんどのものがパスワードあるいはＩＤ，等の組合せによって実現されている。

また，入出力端末装置側での物理的なキー，チェックカード等によって，イリーガルなアクセスからの保護を行っているものもある。さらには，一部のデータベースマネジメントシステム（e.g. ＡＤＡＢＡＳ）では，暗号化／復号化方式を採用しているものもある。

分散データベースのセキュリティにおいては，これらの従来システムにおけるセキュリティ保護機構の他に，分散環境下でのデータの送受信時のデータの漏洩（盗聴），データのイリーガルな改ざん等を防止する有効な方法を備えていなくてはならない。特に分散データベースをオフィスオートメーション等に使用する場合には，署名等のオーセンティケーション機構は，必ず必要になってくるものと思われる。

もちろん，従来のパスワード方式，入力装置側での物理的なセキュリティ保護機構も，あわせて採用すべきものであり，記憶域中にストアされたデータの保護およびネットワーク中を伝送されるデータの保護の両面を備えていることが分散データベースでは不可欠となる。

このようなセキュリティ機構の実現を計るために，ネットワークアーキテクチャの標準化等ですすめられている暗号化方式との整合を計ること，暗号化を採用するとすれば，それをエンドツウエンドで行うのか，あるいはリンク毎に行なうのか，それらの併用とすべきなのかといった問題，オーセンティケーションが可能な高性能暗号／復号化チップの開発といった課題が残ることとなる。

8.通信とDDBS処理のモデル

8. 通信とＤＤＢＳ処理のモデル

8.1 ＤＤＢＳにおける通信処理の基本概念

コンピュータネットワーク技術において，その基本的な機能は異なるホスト（あるいはターミナル）内に存在する一対のプロセス間における情報交換の機能（プロセス間通信）である（図8.1）。この機能によってプロセスは互いにメッセージを交換する。送り出されたメッセージはそのまま相手に届けられるか，あるいは（両端のプロセスが既知の）定

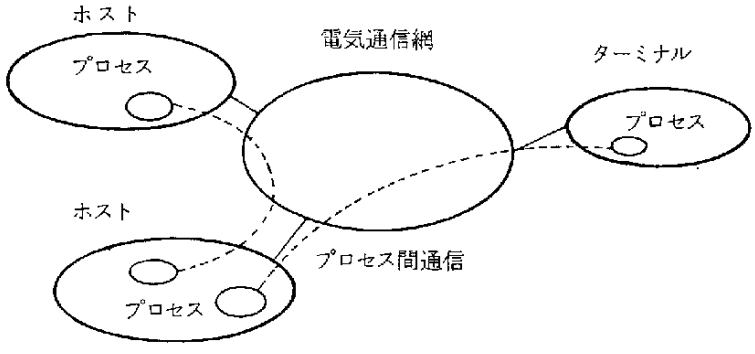


図8.1　コンピュータネットワークにおけるプロセス間通信

形的な変形が途中で加えられて届けられる。定形的な変形とは例えば文字コード変換などである。

このように，コンピュータネットワーク技術における通信の基本概念は，一対の通信実体間におけるメッセージ通信であり，この概念は必ずしも分散形データベースにむいたものとはいいがたい。分散形データベースでは，単なる"一対の実体間における通信"ではなく，"いくつものデータ実体にまたがる通信と処理"である。すなわち，単にデータの移動ではなく，与えられた条件を満たすようにデータ間で処理を行い，その結果得られたものを提供する。

ＤＤＢＳにおける通信処理の基本概念を次のように考える（図8.2）。

複数のデータモジュールに対し，アクセスモジュール（ホスト内プロセスあるいは端末ユーザ）からの要求に応じて，それらの中のデータ相互を通信・処理し，アクセスモジュールに提供する（あるいはデータモジュール中のデータの更新をする）。この場合，通信と処理は別々の概念というより，むしろ通信しながら処理を行うというような，より高次な概念と考える。大量のデータを対象として処理するためには，データを移動させながら処理を行うという考え方が有効と考えられるがらである。

図8.1のコンピュータネットワークの基本概念には，広域ネットワークからローカル／

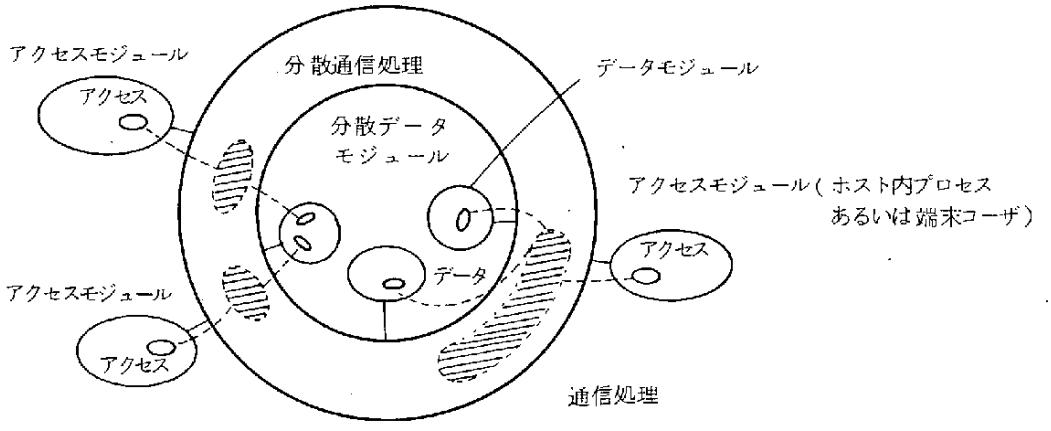


図 8.2　DDBSにおける通信処理

インハウスネットワークまで含まれるのと同様，図8.2のDDBSの基本概念には，広域のものから，Back - End Storage Network* （図8.3）やデータベースマシンまで含まれる。

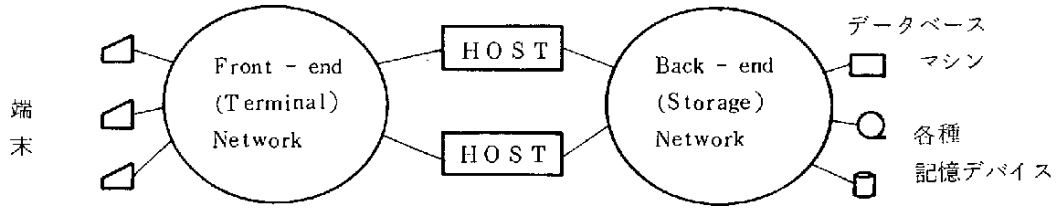


図 8.3　Front-end Network と Back-end Network

コンピュータネットワーク技術は，図8.3のFront-end Networkの観点から発展してきたものといえる。それに対して，Back-end Storage Networkは今後コンピュータにおける通信処理の重要課題となる。Back-end Networkの技術的基盤は主としてローカルネットワーク技術である。したがって，バス結合や線状結合，ループ／リング結合などの技術を主とするが，マイクロウェーブ，衛星通信などによる遠距離リンク結合も含まれる。

Back-end Networkの概念は，しかし，通信と処理をいまだ別の機能として取扱っているようである。それに対して，我々のDDBSの概念では，通信しながら処理を行うとい

---

*　Back - End Storage Networks，　IEEE　COMPUTER．　　Feb，1980．

うより高次な機能を取り入れるものとする。

8.2 ＤＤＢＳにおける通信処理の参照モデル

図 8.2 に示した，ＤＤＢＳにおける通信処理の基本概念を，より詳細なレイヤ構造で表わすと，図 8.4 となる。これを参照モデルと呼ぶことにする。

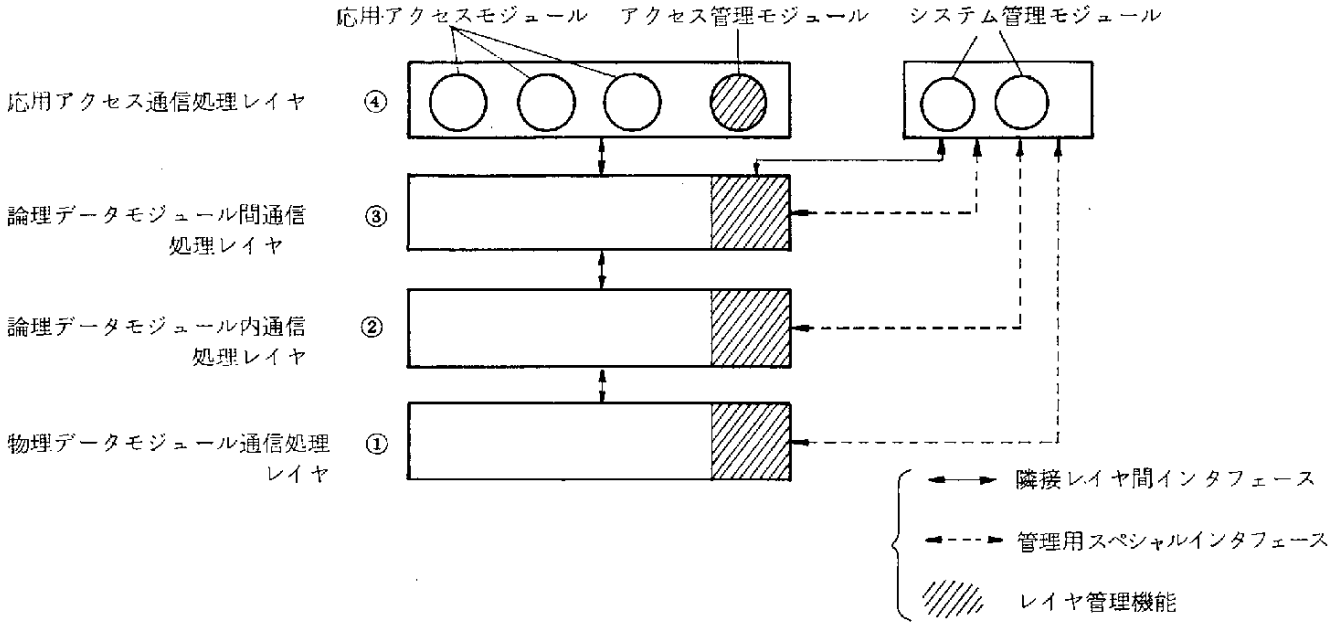


図 8.4 ＤＤＢＳ通信処理の参照モデル

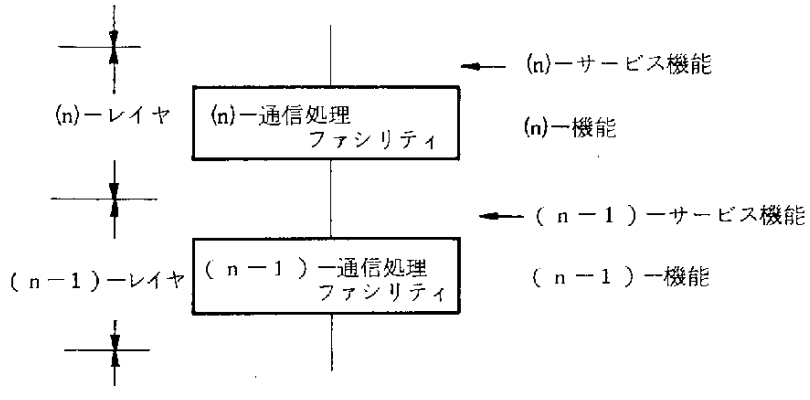


図 8.5 レイヤ構造

レイヤ構造の表現の基本的な考え方は，図 8.5 に示されている。あるレイヤは，　レイヤ内の通信処理（通信および処理）ファシリティを利用して種々機能を実現する。そしてこれらのレイヤ内機能と直下のレイヤから提供されるサービス機能を用いて，直上のレイヤにサービス機能を提供する。

さて，図 8.4 の参照モデルにおいて，レイヤ④には各種応用アクセスモジュールがあり，それぞれのユーザ応用に対応する際にいくつかの互いに独立した別個の分散形データベースアクセスを行う。それらアクセスの各々は，一般に複数データモジュールにまたがるアクセスである。これらのアクセスは，レイヤ③のサービス機能を利用して行われる。

レイヤ③は，論理データモジュール間の通信処理レイヤである。ここに論理データモジュールとは，論理的に一つのまとまったデータモジュールを指している。これに対し，物理データモジュールは，一つの物理記憶媒体を指している。一つの論理データモジュールは，必ずしも一つの物理データモジュール内に格納されているとは限らず，いくつもの物理データモジュールに分散して格納されていることもある。

レイヤ③は，レイヤ④から依頼されたアクセスがいかなる論理データモジュールに関係するかを判断し，また，それらの論理データモジュール内で処理を受けたデータを相互にどのように関連（通信処理）させるかを判断し，それを実行する。各論理データモジュール内でのデータ処理は，レイヤ②のサービス機能を利用して行う。（いかなる論理データモジュールが関係するかを判断するための参照情報（ディレクトリ）は，特別の論理データモジュールに格納してあると考えてよい。）

レイヤ②では，レイヤ③から依頼された論理データモジュール内処理要求が，いかなる物理データモジュールに関係するかを判断し，また，それらの物理データモジュールで処理を受けたデータを相互にどのように関連（通信処理）させるかを判断し，実行する。各物理データモジュール　でのデータ処理はレイヤ①のサービス機能を利用する。

レイヤ①では，レイヤ②から依頼された物理データモジュール内処理を行う。

8.3　参照モデルにおけるアクセス処理の例

この参照モデルにおけるアクセスの処理の様子を例で示す。今，レイヤ④におけるある応用アクセスモジュールでユーザ応用を処理中に，分散形データベースアクセス f(x, y, z) を行う必要性が生じたとする。ここに，x, y, z はこの分散型データベース内の論理データで，f はそれらに対するオペレーションを示す。f(x, y, z) はレイヤ③のサービス機能を用いて処理される。（図 8.6，図 8.7）

レイヤ③，すなわち論理データモジュール間通信処理レイヤでは，まずデータ x, y, z が，それぞれどの論理データモジュールに属するものか判断される。ここでは，x, y, z は

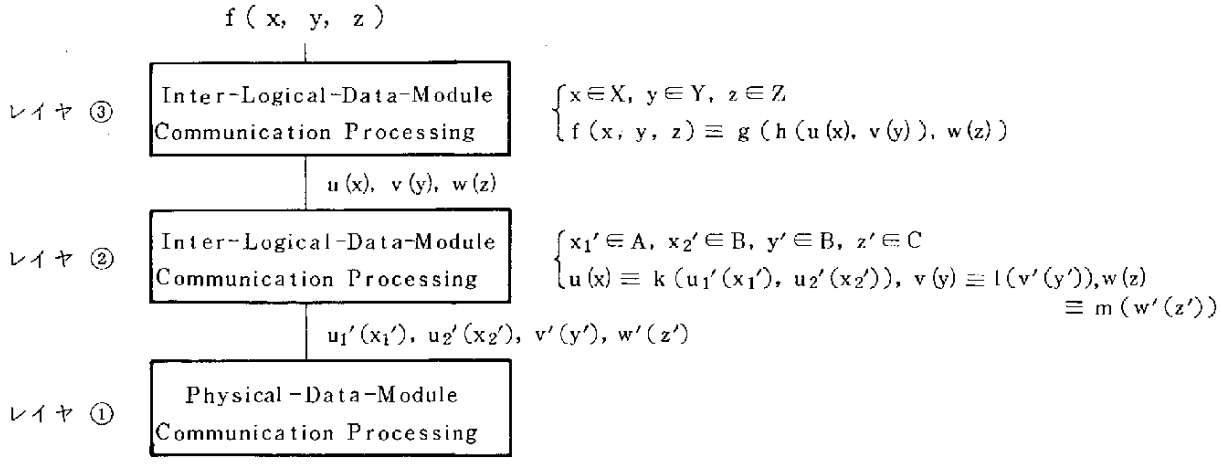


図 8.6　参照モデルにおけるアクセス f (x, y, z) の処理

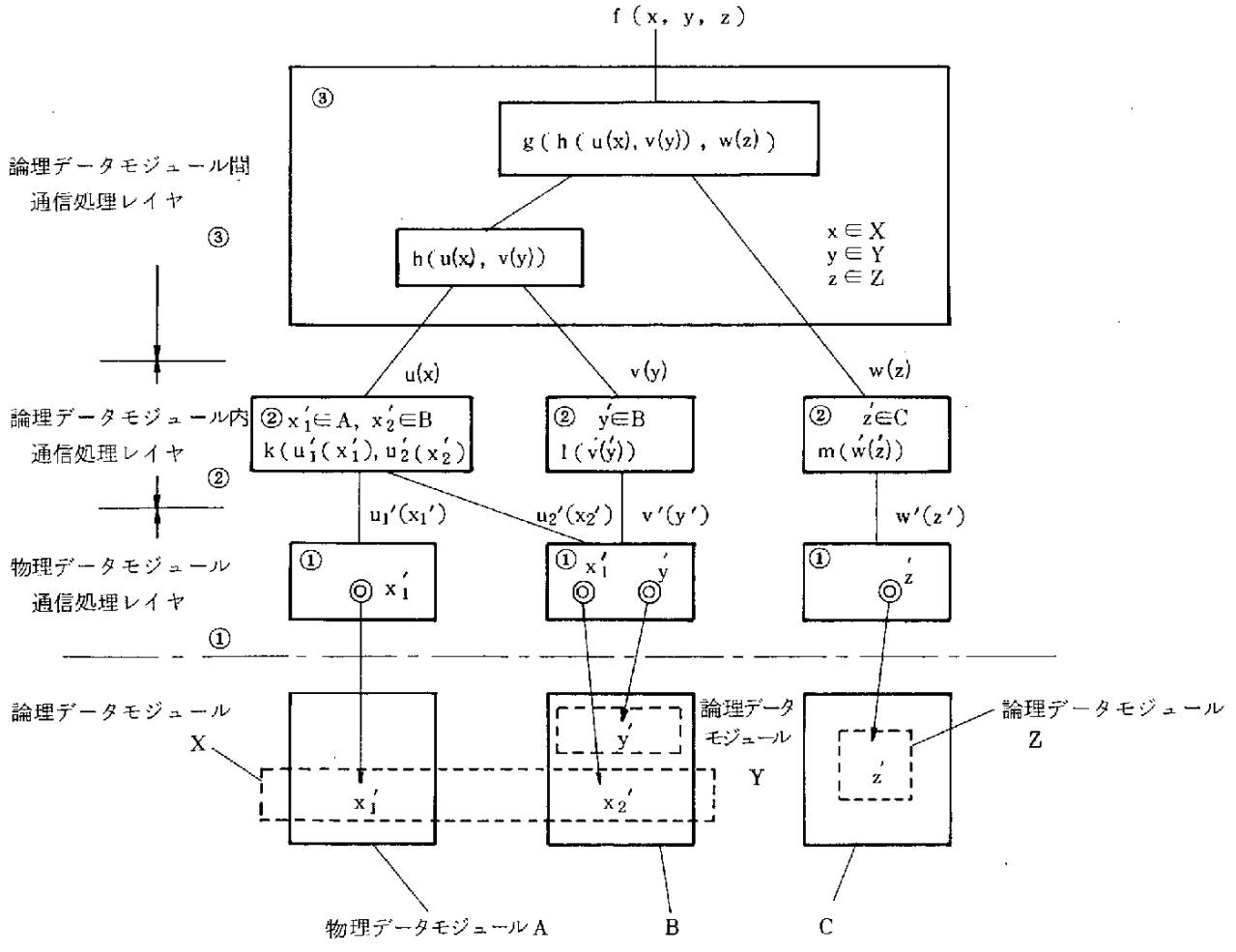


図 8.7　f (x, y, z) の処理の詳細 (図 8.6 参照)

それぞれ論理データモジュールX，Y，Zに属しているものとしている。そして，f(x，y，z)を，レイヤ②のサービス機能を利用して得ることができるu(x)，v(y)，w(z)に対する論理モジュール間オペレーションg(h(u x)，v(y))，w(z))の形に分析して処理する。

レイヤ②，すなわち論理データモジュール内通信処理レイヤでは，u(x)，v(y)，あるいはw(z)をそれぞれX，Y，あるいはZの論理データモジュール内のオペレーションと理解して処理を行う。論理データxは，物理データモジュールAおよびB内にそれぞれx_1'，x_2'の形で存在し，y，zはそれぞれB，C内にy'，z'の形で存在するとする。u(x)は，レイヤ①のサービス機能を利用して得ることができる。$u_1'(x_1')$，$u_2'(x_2')$に対する論理モジュールx内オペレーションk($u_1'(x_1')$，$u_2'(x_2')$)の形に分析して処理する。v(y)，w(z)も，それぞれ論理モジュールY，Z内オペレーションl(v'(y'))，m(v'(z'))で処理される。

レイヤ①，すなわち物理データモジュール通信処理レイヤでは，物理データモジュールAに対応してx_1'，Bに対応してx_2'，y'，Cに対応してz'の物理データ処理を行い，$u'(x_1')$，$u_2'(x_2')$，v'(y')，w'(z')の要求に答える。

8.4 参照モデルとDDBSの構造との対応

この参照モデルとDDBSの構造例と対応を図8.8に示す。

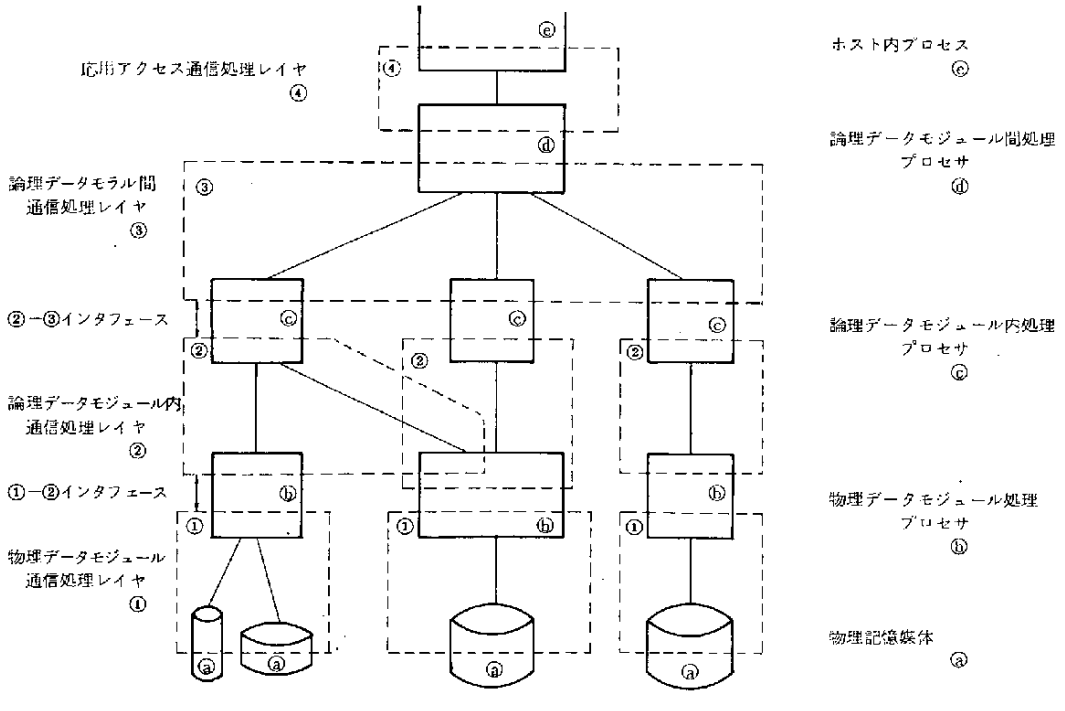


図8.8 参照モデルとDDBS構造例との対応

この参照モデルとDDBSの構造例と対応を図８に示す。

ホスト内プロセス，論理データモジュール間処理プロセサ，論理データモジュール内処理プロセサ，物理データモジュール処理プロセサおよび物理記憶媒体の間に，点線で囲まれた通信処理レイヤ④～①が存在する。

例えば通信処理レイヤ③では，斜線で示した４つの部分がそれらを結ぶ通信線を介し，互いに協同処理を行うことによって，このレイヤで受け持つべき通信処理を遂行する。

表 8.1　参照モデルと既存概念／システムの対応

レイヤ①	RAP（回転記憶＋logic-per-track），EDC（バブル），Staging DBC（構造記憶＋大容量），DIRECT（クロススイッチ）
レイヤ②	EDC（バンド），DIRECT（クロススイッチ）
レイヤ③	Back-end-(Storage)-Network （単に通信で通信処理ではない）
レイヤ④	Front-end-(terminal)-Network

8.5　まとめ

本稿で提案した分散型データベースシステムの参照モデルは，物理的データモジュール，論理的データモジュールの地理的存在位置に依存していない。すなわち，遠距離通信によるものまでを対象としている。

データベースマシンは，大量のデータを複数の物理データモジュール内に分割格納しており，これら分割格納されたデータの（アクセス要求内容に対応した）結合処理を行うことによって，データベースアクセスに応答する。したがってデータベースマシンについてもここで示した分散形データベースシステムの参照モデルで表現することができる。

表 8.1は，広域網からデータベースマシンにおける既存の概念／システムを参照モデルのレイヤとの対応で示したものである。現状ではデータベースマシンのほとんどはレイヤ①の段階を対象としており，レイヤ②の段階については，ようやく研究対象となり始めたというところにある。論理データモジュール間を対象とするレイヤ③については，それら論理データモジュールの存在位置が互いに遠い場合，近い場合いずれにおいても，本格的な研究はこれからである。

これらのレイヤにおいてどのような処理機構が望ましいかを考えるにあたっては，通信と処理を別個のものと考えず，それらを有機的に統合した通信処理の概念でアプローチするのも一つの行き方であろう。

そして，このアプローチには光通信や衛星通信技術による高速通信とVLSI技術による高速処理を組み合わせた新しい技術展望が期待されている。

9．アプリケーションモデルのシステム化とDDBS

9. アプリケーションモデルのシステム化とDDBS

DDBSはその性格上，コンピュータ技術の多岐にわたる分野を対象として包含せざるをえないので，必要技術についての議論を一般論的に進めていたのでは成果を集約が難かしく，ひいては実現技術，設計製造技術としての確立が遅れる。昨今，DDBS必要論が宣伝され技術論が盛んな割になかなか実現が進まぬのはこの要因が大きいのではないかと思われる。

この隘路を抜けでる一つの方策は，DDBSに対するアプリケーションモデルの設定とシステム化にあると考える。これはDDBSが実際にどのような環境で使われ，どのような目的を充足させるべきであるかを明らかにすることであり，換言すればDDBSの市場ニーズを明らかにすることである。

現在分散データベース論に関して最も不足しているのは市場ニーズの具体化，明確化である。議論の対象となっている，個々の技術項目が現実どのように応用され，利用者を満足させうるのかといった観点からの検討努力の不足である。

つまり諸々の技術の成果を

－誰が購入してくれるのか。

－何の目的で使われるのか。

－市場の大きさはどの位か。

という観点のもとで検証し選択する必要がある。

分散データベースシステムの対象範囲は広く，特殊な目的であるが故に特殊な技術と高い実現コストを要し市場も小さなものから，一般性の高い利用目的で市場も広く，ゆえに普遍の高い実現技術とそこそこのコストにおさえることを求められるものに分けられる。この両者についてのアプローチを同一の議論でまかなおうとすることは無駄が多すぎる。

例えば，DDBSが論ぜられるとき必らず議論になる重複データに対する更新や同時アクセス制御の問題がある。これらについては一般的な考察から多彩な方式や理論が導びかれるが，データベースの更新権は運用管理者が握り一括して処置するというような運営を前提にすれば，多くの問題は一挙に解決し技術的には単純明解なもので可となってしまうだろう。しかもそのような運用条件は決して特異なものではなく，充分普遍的で応用範囲の広いものですらある。

似たような傾向として，分散データベースシステムの主要な技術リソースである計算機網技術についても過去に市場ニーズと隔離された包括的で理論的なアプローチに偏する傾向があって実用化に向けての進歩が停滞しているように見うけられた。

しかし，最近になってようやく実用条件や範囲を設定し（換言すれば市場ニーズを設定し）必要技術と開発コストのトレードオフを試みる開発事例が現われはじめている。

ゼロックス，ＤＥＣ，ＩＮＴＥＬの３社が共同して商品化と米国における方式の標準化の働きかけを進めているＥＴＨＥＲＮＥＴの例はその一つで，このシステムの開発目標，もしくは市場設定の動機は次のような要件にもとづくものであるといわれる。

① 米国内におけるオンラインシステムについてはその８０％がたかだか半径６０マイルの範囲をカバーするものであること。

② さらにその内６０％は同一構内で使用されていること。

すなわち，同一構内に限った条件で安価で高信頼，高能率の通信網設備を商品化できれば単純に考えて，オンライン需要の約５０％を対象市場にしうるということである。

分散システムの重要性がコンピュータ・サイエンスの世界でとりあげられるようになって久しいが，研究者の興味にひきずられ，個々の技術テーマのより一般論的な議論の展開が盛んである。そうそう実用化技術の確立に多くの努力が振り向けられてもよい時期に到っていると考えられる。

実用化技術への明確なアプローチの最初の段階はその市場を選択，もしくは限定することであろう。その際重要なことはより共通的，普遍的で大きな市場を目標として設定することである。そういった市場で要求される製品仕様にとって有効な技術は重要でありそうでない技術はたとえ研究者の興味をひくものであっても比較的重要ではない。

そういった観点からの分散システム技術に対するリクワイアメントの一般化，集約化が急がれる。もうそろそろ集約化への方向転換が意識的にはかられないと，１０年後（１９９０年代）に向けての充分な実用化は難かしいのではないか。

最も共通的で大きな市場については企業的な見地から見ても種々の考え方があろうが，その一つの考え方の例は次のようである。

(1) 製造業を中心とする民間営利企業

－ＤＤＢＳに限らずＥＤＰに対する設備投資の最も大きな部分はこれらが占めるものと思われる。

－（たとえ今後の１０年が現状程度の低い経済成長の時代であったとしても）

(2) 企業内利用システムが大半を占めること

－おそらくＤＤＢＳ需要の９０％以上は企業内システムではなかろうか。

－利害が一致しない複数企業もしくは組織にまたがる分散システム化は，例外的需要であり話題性は高くともほとんどの設備投資対象ではなかろう。

－この企業内システムであるという市場設定は重要であり，ここから導びかれるリクワイアメントはおそらく現在ＤＤＢＳ論として華やかな技術論の大半をそう重要でないものにするだろう。たとえば同時更新，機密保護，システム構成の概念化。

(3) 企業内システムであれば，企業のより根幹的な活動に沿ったＥＤＰＳ化ニーズが主と

なり，それ以外のニーズはあっても規模の小さなものであること。

－一次，二次産業であれば，生産管理，物流，資材管理，営業情報，経理情報管理などが中核的部分を占めるだろう。

－例えば，そのような企業における広報，宣伝活動などはＥＤＰ化の重要な部分ではないだろう。

以上のような市場設定を行なった上で，現在時点でのＤＤＢＳ技術に対するリクワイアメントを明確化すれば，製品化への過程はかなり短縮されよう。

前述のようなＤＤＢＳ需要のカテゴリに入らないものでしばしば話題になるのは次のような分野についてである。

① 図書館システム，学術情報システムのような公共システム

－システム規模はより大であるが，システムとしての普遍性は低い。実現技術に対するリクワイアメントはより特殊になり，重要ではあるが実現技術としての採算性は低下せざるをえない。

② バンキングシステム

－ＤＤＢＳの初期の実際化事例として欧米におけるバンキングシステムがよく話題にのぼった。

－バンキングシステムは処理の局所性も高く分散システム化に有利という判断もあり，ＥＤＰ化への投資意欲も投資額も巨大ではあるが……。

－要求性能からくるシステムリクワイアメントが特殊で，技術仕様として一般的なものにはならないだろう。

－現在と同じく特殊な技術対応をせまられよう。これに対する必要技術の一般化はやはり採算が合わないだろう。（１０年後であっても。）

－例えば，求められる応答性，スループット，リカバリ機能の完ぺきさ，ファイルの大きさ，ファイルのインテグリティ保証の程度など。

③ 複数企業にまたがる分散システム

－従来のＤＤＢＳ論は暗にこのような利害の一致しない組織にまたがるシステム形態をイメージとして持っていたようである。

－いきおい利用環境は無秩序，勝手なものであり，システム機能はそれらを正しく制御でき，システムの完全性がそこなわれないものとして要請されるようになる。特にデータベースの内容保証のための技術要請が困難であり過ぎる。

－しかし現実にはこのようなシステム形態はほとんど存在することを要求されないのではないか。もしあったとしても，極めて例外的でありそれ故に技術条件もかなり緩和されたものとなろう。たとえば，データベースは決まりきった手順で参照するだけと

いうように……。

④ 複数官公庁にまたがるシステム

－前項と類似した環境といえる。

－この環境に分散システムが存在するようになることを否定したり疑問をさしはさむことは許されない雰囲気である。

－しかしこのシステム環境が分散データベースシステムを利用する目的やイメージは，はなはだ観念的であって具体性に欠ける。これに要する技術を検討する前に利用環境の具体化検討を試みる必要があるのではないか。

１９８０年代に入る直前からコンピュータを含むシステム化技術は高度で複雑なものを組合せて大規模なものにまとめあげることを非とし，できるだけ単純明解なものより単純な組合せを是とするようになった。

分散システムの動機は正にこのことであったはずだし，最近のオフィスオートメーションへのＥＤＰ技術の傾斜もこの発想が大きな位置を占めていると考える。一時期までのオートメーション（合理化）は複雑高度なコンピュータがオフィスを支配するような発想にもとずいていたが現在はそうでなく単能的なワードプロセッサやコピーマシン群が主役でコンピュータは補助的，裏方的な存在として扱われはじめている。

このことはコンピュータによる高度で複雑な管理，制御のからくりが主としてソフトウェアの支えるところと期待されて来たのに，オペレーティングシステムをはじめとする巨大システムソフトウェアの高コスト化，発展の停滞などからシステムの巨大化への反省，疑念が生まれてきたことによる。

分散システム技術もこの流れに沿って複雑な部分はより単純な部分の組合せに，大きな部分はより小さな部分の組合せに導くことが正しい方向であると考える。

この意味において，分散システムの全体像をスマートな概念に統一表現するような努力は不要とは考えぬまでもあまり重要なテーマとは思えぬし，ましてや分散システムとその運用の正当性を包括的に管理制御するシステム機能の具体化技術も必須なものとは思われない。分散データベースシステムであれば，昨今データベース技術そのものがモンスター化しはじめていることを考えればなおさらである。たとえば，データベースの同時更新の問題は分散データベースシステムの宿命的で難解なテーマであるとされ技術者の興味や努力が集まる。しかし，この問題のなり立ちはデータベースに不特定多数の使用者が随意に，ときには悪意をもって接触し，それに更新を行なってもシステムはそれを即時に解決させ矛盾を発生させないことという，いわば無理難題的な命題にもとずいている。現実には更新権は特定少数者に限るとか，更新は非即時で可とするというような運用条件緩和が許されるのが普通であり要求される技術もそれなりに簡易なものですむ。複雑難解な問題解決は高コストを要し信頼

度も低いが単純化された問題の解決は低コストで実現でき信頼性も高い。複雑な問題を高度なレベルで解決するのも技術であるが，命題を簡明なものにし低いコストで実現することも技術であろう。特に分散システム技術においてはこの側面は本質的に重要である。

このような技術アプローチを可能にするためには，アプリケーションモデルのシステム化，すなわち誰がなんの目的でＤＤＢＳを必要とし，どんな使い方で充足し，どの位のコストをかける動機をもつか，を具体的にすることが急務であると思われる。

10. 知識ベースとDDBS

10. 知識ベースとDDBS

10.1 知識とデータ

通常データベースに貯えられているデータは，単なる事実としてのデータであり，個々の項目の意味とか，一般的概念，等は収められていない。ここではそれらを知識（狭い意味での）と呼ぶことにする。データベースはデータとスキーマから成るとした時，スキーマに記述される事項は，

① データの名前とその形式

② Relation 内のキー項目

③ Relation の関数従属性，多値従属性

等であり，データの意味自身の記述は殆んど含められていない。②のキーが何故キーなのかという事は人間が外から指定しているし，③に関しては意味の一部を与えてはいるが不完全である。意味の記述を与えようという試みは，Entity - Relationship Model，Role Model 等に見られるが，未だ初歩的段階にある。データディクショナリに意味記述がされることはあっても，それはあくまでも人間を対象にしておりマシンが読んで使えるという形のものではない。

このように現在のデータベースに欠けている意味記述等の知識は今後技術の発達に伴ないシステム内に組み込まれてゆくものと思われるが，そのような知識ベースと分散データベースシステムとの関わり合いについて考える。

10.2 知識ベース・システム

(1) 知識ベースシステムへの期待

知識とは，通常のデータベースに貯えられるような単なる個別的事実の集合の他に，概念のような一般的知識，それらの知識の利用法としての知識（メタ知識）等が考えられる。それを集めた知識ベースを備えたシステムに期待されることは以下のようである。

- 人間－機械インタフェースの改善
- データ独立，プログラムのマシン独立のようなシステムの柔軟性
- プログラムの検証．合成．変更等が容易になること
- プログラム自身が知識となり得ることによるプログラムの蓄積性向上

即ち，従来データと呼ばれていたものもプログラムと言われたものも，述語論理のような形で表現することにより同一に見ることが出来るので，知識ベースに次々と追加してゆくことによって大規模ソフトウェアを漸進的に開発してゆく可能性がある。

(2) 実現上の問題

次のような問題がある。

a. 知識表現

b. 推論機構

c. 新知識の獲得と知識体系への組込み

aに関しては，従来よりSemantic Network，述語論理，Production System，KRL等が提案されているが，あらゆる知識を1つの形式で表現するのは難かしい。また，知識ベースと従来からのデータベースを一体のものとして扱うか，別ものとして扱うかにも依る。処理能率という点から見ると，分離して扱った方が少くとも近未来用としては実際的と思われる。cはシステムが成長するオープンエンデッドなものであるための必要項目で，新知識を既存のモデル（知識体系）に如何に組込むか，若し矛盾を生じればモデルを修正してより高位のモデルを適確に構築するにはどうするか等が問題となる。

(3) 知識ベースの3段階

a. データベース

論理構造は固定で変らず，貯えられているデータに関する知識が完全な場合である（閉じた世界）。但し，データ量は一般に膨大となる。この場合，推論は関係代数演算に帰着するので，それ用の高速プロセッサが望まれる。

b. 論理構造固定の知識ベース

知識のモデルは不変であるが，知識が不完全にしか存在しない場合（開いた世界）である。述語論理等で知識を表わし，分解原理を用いて演繹を行なう。メタ知識の組み込みが処理の能率上問題となる。

c. 学習機能を取入れた知識ベース

論理構造自体が新知識の導入により変化し発展する。知識体系は常に不完全である。このような場合，推論は仮説を立ててそれを検証するという形を取るが，その仮説が誤っていた場合の能率良い後戻り制御方式が必要になる。更に，新データより法則性を見出す帰能的推論が必要となる。

1.0.3 データベースに於ける知識ベースの役割

データベース一般に対し，知識ベースが導入されることによる意味を考えると以下のようになる。

(1) データ自身の意味を与える。

(2) データ相互間の関係を補充する。

関係には，データベース作成者が与えた関係と，データの意味から推論される関係が

あり，この内後者を与える。例えば，同一ドメイン名で結ばれていないRelation （直接，関接に）間の関係とか，異なった名前を持つRelation相互間の関係等。

(3) 複数のRelationsのセットに対し，それらの上位概念（抽象化）を与える。

(4) データの完全性条件を与える。

(5) 問合せの不完全性を補う。

例えば，一々細かく指定しなくても，その時点のView（自動設定）から推論するとか，問合せに含まれる条件から推論して，実際に存在するドメインに対する条件にマップする等（cf. 妊娠している２１才の人→２１才の女性）。

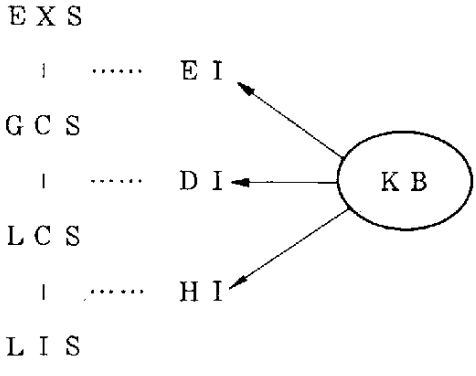
(6) データベース蓄積の効率向上

データベースの蓄積は，新データを追加することにより加法的に量が増すが，知識ベースでは，新知識の追加により全体としての知識量が組合せ的に増大する。即ち，データベースでは，あらゆるRelation 間の関連を付けることは困難である。

１０.４ 分散データベースに対する知識ベースの役割

(1) スキーマ間マッピング援助

分散データベースでは，一般に４つのスキーマ間のマッピング



が問題となり，それぞれに対しディクショナリ／ディレクトリとしてのマッピング情報が必要である。これらの情報は知識ベースの存在によってかなり補うことが出来るため，データベースの設計者自身が与える固有情報をかなり減らすことが出来る。特にＧＣＳ－ＬＣＳ間のマッピングでは，各ＬＣＳに対し十分な意味付けが与えられることにより，対応付けを推論することが出来る。その機構を描けば図１０.１のようになろう。

人間がこれらのマップ情報を記述すれば，その変換能率は確かに良いであろう。しかし，分散データベースの場合，要素となるデータベースの数はどんどん増すことが考えられ，成長性のネックとならないためにもこの機構は必要であろう。また，推論機構によって様々な対応付けを考えてみることが可能で，それを仮説として設定し，それによる影響を推論してからその対応付けを実際のマッピング情報として生成することが出来

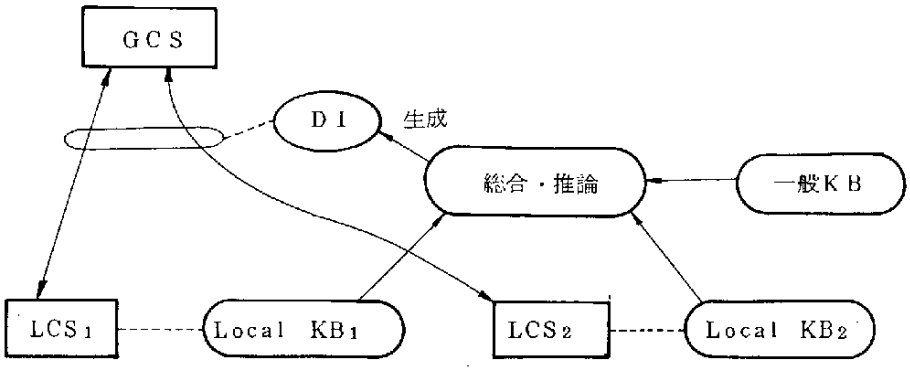


図１０.１　自動的対応付け

よう。即ち，スキーマ間変換情報作成の，コンピュータによる援助である。

(2)　データベースの正当性維持

ＬＣＳ相互間の関係を推論することによりデータの正当性維持をより確実化できる。データの更新に対し，データの依存関係を推論してその更新の正当性判断や，更新範囲の拡大や縮少（関数従属性や削除の範囲限定）を判断することが出来る。これは，定型的プログラムを作成する際に，その推論機構を用いてもよいし，非定型的な処理に対して即時的に利用することも考えられる。

(3)　本格的な知識ベースへの相互依存関係

将来の高度な知識ベースとしては，ユーザインタフェースを一様な知識ベースへのアクセスインタフェースとし，その内部構造ではデータベースと知識ベースが明確に分離されているようなものも考えられよう。学習機能を持ち，仮説設定と検定の機構を備えた知識ベースである。そのようなシステムの実現には，分散データベースが不可欠であろう。即ち，分散知識ベースとして多くの知識ベースを結合する分散機能が備わっていて始めて，その成長性の良さ，データ蓄積性の良さが生かされる。データベースは，分散データベースにより量的に機能が拡大し，知識ベースによって質的な機能向上が計られる。

１０.５　将来への構図

(1)　データベースシステムの発展

ファイルから出発して，データベースは次のような段階を踏みつつ発展することが考えられる。

1.　ファイル

2.　スキーマ＋データ

3. スキーマ＋意味記述＋データ

4. スキーマ＋特殊知識ベース＋推論機構＋データ

5. 知識ベース＋推論機構＋データ

6. 一般化知識ベース＋学習機能付推論機構

これらの各時点において，分散化は同時進行するものと思われる。

(2) 知識ベースの利用

まとめれば，次のようになろう。

1. スキーマの機能増強

2. データ正当性チェック

3. データ自身とスキーマとの統合概念として

4. 複雑度の増大に対する解決策として

4.は，特に分散データベースにおいて問題になることであり，量的／質的な複雑度の増大に対しては，データ蓄積性が良く問題解決機能を備えた知識ベースが不可欠である。

(3) プログラミング言語との結合

将来のプログラミング言語としては，蓄積性，検証容易性の良い述語論理形や関数形が有望だと言われている。それが実現すれば，データ自身とプログラムとの同一視が非常に自然な形で可能となる。知識ベースの基礎構造はこの概念があって始めて強靱なものとなる。知識の追加とはデータの追加でもあるし，またプログラムの追加でもある。

(4) 知識ベース実現への必要機能

次のような機能が一般に必要と考えられる。

1. 汎用的知識表現法

2. 推論機構：

仮説設定／検定機能，能率良い後戻り制御方式

3. 学習機能：

新知識の取得と自己再修正

4. データベース／知識ベースの統一：

能率の良さと高機能性を両立させる

5. 高速マシン：

連想，高度記号処理，非決定的演算，構造メモリなど

(5) 分散知識ベース実現の問題点

分散データベースが知識ベースの援助により発展し，分散知識ベースとなり得るための問題点としては，以下のような事項が考えられる。

1. データベース，知識ベースの特徴明確化と組合せ方式

2. 知識ベースのアクセス言語とプログラミング言語間の関係
3. 能率良い推論機構
4. 知識表現の差異対策：

   データベースの統合よりも容易となり得るか。
5. 複数知識ベースの融合問題：

   表現が同一としても，複数集めれば知識相互間の矛盾が生じうる。その解決策は，知識モデルの高位化により達成し得るか。

10.6 おわりに

現代のデータベースシステムには意味の記述が欠けており，そこから様々な困難が生じている。従って，意味の導入が様々な形で導入されてゆく状況にある。知識ベースはそのトップランナーであり，分散データベースが直面しつつある困難を解決するための抜本的な手段として期待されるものである。しかし，知識ベース自体，未だ研究段階であり，学習機能に至っては全く未知の分野であると言っても過言ではないであろう。今後，各方面に於けるその基礎的研究が進展することに期待したい。

昭 和 56 年 2 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
T E L　(434) 8211 (大代表)

印刷所　株式会社　正　　文　　社
東京都文京区本郷 3 — 38 — 14
T E L　(815) 7271