情報学基礎11－3
（1988. 11. 21）

知的分散データベースシステム

原嶋秀次　永瀬恵子　岡宅泰邦
（株）東芝　システム・ソフトウェア技術研究所

現在データベースシステムにおいて用いられている各種のデータモデルは，データ間の関連をユーザが把握していなければならない，あるいはデータアクセス時の動的な関連定義ができないなどの問題がある．

本稿では，これらの問題に対処するため現在開発中のオブジェクトモデルを用いた分散型関係データベースシステム，知的分散データベースシステムについて紹介する．

本システムでは，各リレーションをオブジェクトとして表現したオブジェクトモデルを採用することにより上述の問題を解決する．また，オブジェクトであるリレーションの自律性を利用して高度なデータ独立性の確保，分散スケジューラによる並行処理制御などが実現できる．

DISTRIBUTED DATABASE SYSTEM on IDPS

Shuuzi HARASHIMA, Keiko NAGASE, Yasukuni OKATAKU

TOSHIBA CORPORATION, Systems & Software Engineering Laboratory
70, Yanagi-cho, Saiwai-ku, Kawasaki, 210 JAPAN

Data models currently used such as a relational model are often difficult to deal with relationships among data.

Here, a distributed relational database system on IDPS(Intellectual Distributed Processing System) is described.

The proposed system is based on object data model. Each relation is implemented as an autonomous object. Therefore, users can access the database without the knowledge about relationships among individual relations.

Relations expressed as autonomous objects also provide high data independence and distributed concurrency control mechanisms based on cautious schedulers.

1．はじめに

リレーショナルデータベース（ＲＤＢ）は，その高いデータ独立性，平明性，および操作性のため，現在商用データベースの主流の１つとなっている．ユーザはデータベース管理システム（ＤＢＭＳ）に，リレーション名，領域の定義および一貫性の制約を宣言的に入力するだけでデータベースを構築することができる．

しかし，ＲＤＢではリレーション間は領域定義名である属性名を通してのみ関係づけられているので，ユーザはこれらの関係を把握してリレーションを操作しなければならない．従って，データベースが大きくなるとユーザの手に負えなくなるという欠点がある．

データベースが使いやすくなるためには，対象データに対する操作実行時に，必要な他のデータとの関連が得られ，かつそれら関連データへの操作が自動的になされるようなシステムが望ましい．

一方，安価で高性能なワークステーションやＬＡＮの出現で分散システムが実用化され，分散データベースも実用化の段階に入り，処理速度の向上やデータ一貫性の保証などに対する有効な解決法が必要になっている．

これらの問題に対して，分散処理技術の研究においていくつかの有用な成果が得られている．分散オペレーティングシステム（ＤＯＳ）は分散処理における基本技術の１つで，「どこに，どんな資源がどんな状態で存在するか？」をユーザーに意識させることのない分散システムの構築を可能とした．しかし，これらＤＯＳを用いたシステムが拡張性や適応性，信頼性の高いものであるためには，システムの管理機構を分散し（管理機構の分散化），要素自身が自律的に自身の管理を行なうことが必要である．これによりシステム要素の追加や変更などに対する柔軟な対応が可能となる．

このような要求を満たすシステム構築技法のひとつに，オブジェクトモデルによるアプローチがある．本稿では，オブジェクトモデルに基づくＯＳ，知的分散ＯＳ[1]上に構築中の分散型データベースシステムについて紹介する．

第２章で，知的分散システムおよび知的分散システムにおけるオブジェクトの概要を述べる．

第３章では，本データベースシステムの構成，およびその知的分散システム上での実現法について述べる．

第４章では，本システムで採用したいくつかの特徴的な機能について述べる．

本稿で提案する分散型データベースシステムは，リレーションをオブジェクトとし，自律機能を持たせ，適応性や拡張性，信頼性の高い分散型データベースシステムの実現を目指すものである．つまり，本データベースシステムではユーザの操作に対して，リレーション自身がシステム中での自身の位置づけを判断し，必要な処理を行なう．これにより，ユーザはリレーション間関係を意識することなく関係操作を施すことが可能となる．

2．知的分散システムの概要[2]

知的分散システムは，集中管理機構を排した分散システムのアーキテクチャである．自律的で互いに協力・協調するシステム要素群によって構成され，適応性，拡張性，信頼性の高い分散システムの実現を可能とする．

知的分散システムは現在，ＬＡＮによって接続されたワークステーション（サイト）上に構築されている．各サイトには知的分散ＯＳ核が存在し，オブジェクト間通信やそれに伴うサイト間通信，マルチタスキング機能などを提供する．

知的分散ＯＳでは，システム要素をデータとそれに対する操作の集合であるメソッド（手続き）群から成るオブジェクトとして実現する．オブジェクト群を管理する集中機構を排しているので，システム全体の状況の保持，管理を行なっているようなサイトやシステム要素は存在しない．オブジェクト間の同期や排他制御，負荷分散や機能配分などシステム全体に渡る管理はすべてオブジェクト間の協力・協調によって行なわれる．

このようにオブジェクトが自身の動作に必要な機構だけでなく，他オブジェクト群と協力・協調する機構を持つので，オブジェクトの追加や変更，削除，サイト間移動などが局所的な操作で実行でき，拡張性，適応性，信頼性の高いシステムの実現が可能となる．

ここで，「協力」とは与えられたジョブの処理をオブジェクト群が互いに処理結果をやりとりすることにより，遂行することである．また，「協調」とはオブジェクト間での処理の競合を調整し，システム資源の有効かつ正しい利用（排他管理，負荷及び，機能の配分管理）をはかることを意味する．

図１に知的分散システムにおけるオブジェクトの構成を示す．オブジェクトは自律的かつデータ独立である．オブジェクトが自律的であるとは，自身の行動規範を自身が有し，かつその行動方法（すなわち手続き）を自身内に有していることを，またデータ独立であるとは，その行動規範が外部から隠ぺいされており，外部からは決められたインタフェイスを通してしかアクセスできないことを意味する．これらに必要な知識は，共通知識部と固有知識部とに分けることができる．固有知識はオブジェクトに固有な性質を表わすデータと手続き群から，共通知識は排他制御，故障管理，負荷分散などのシステム全体の管理に必要なデータと手続き群から成る．前述したオブジェクト間の協力と協調は，各オブジェクトの持つ共通知識の利用によってなされる．

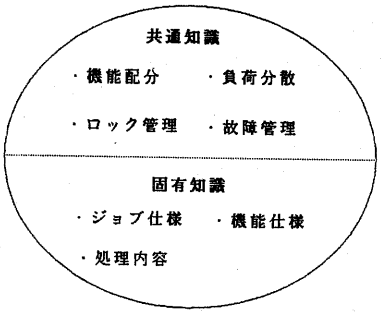

図１．システム要素（オブジェクト）の構成

システムに与えられたジョブは，前述したようにオブジェクト間の協力・協調によって処理される．この間の協力・協調に際しては，「どこに，どんなオブジェクトがどれだけあるか」といった情報が必要となるが，知的分散ＯＳでは必要になった時点で個々のオブジェクトが放送通信を用いて自律的に他のオブジェクトからこれらの情報を得る．その際，制御の簡略化と通信オーバーヘッドの削減をはかるため，直接通信方式を採用している．

３．システム構成

ここでは，本データベースシステムの構成およびその知的分散システム上での実現法について述べる．

3.1　リレーショナルモデルの実現

本データベースシステムでは，データモデルとしてリレーショナルモデルを採用する．データは第３正規形リレーションとして保持する．ただし，後述するように属性値としてリレーション名の記述を可能としている．

リレーショナルモデルを基本としたオブジェクト指向データベースにおいては，何をオブジェクトとするかについて，タプルをオブジェクトとする，複数リレーションをオブジェクトとするなどいろいろと議論されているが，知的分散データベースでは，次に述べる理由により１リレーションを１つのオブジェクトとすることとした．

タプルは，実世界での１つの対象が持つ性質，すなわち属性の集合に値を代入したものと考えられる．従って，タプルをオブジェクトと考えるのがもっとも自然である．

しかし，リレーショナルモデルでは同種のタプルの集合であるリレーションに対する操作が多用され，これらタプルの集合に対する操作をオブジェクトの自律機能で実現するにはタプルよりもリレーションがオブジェクトである方が都合がよい．また，１リレーションを１オブジェクトとすることによりデータ更新時の効率の低下，ビューの違いによる不都合の発生などを避けることが可能となる．

さて，１リレーションを１オブジェクトと考えると，各オブジェクトが自リレーションの管理を行なうので，これまでのいわゆるＤＢＭＳ(DataBase Management System)と呼ばれるような集中管理機構は存在せず，その機能は各オブジェクトに分散されることになる．すなわち，各オブジェクトは自身に関する情報（局所情報）のみを管理し，必要に応じて関連オブジェクト群と情報交換することによりシステム全体の管理を実現する．

3.2　３層スキーマの実現

本データベースシステムは，上述のリレーションに対応するオブジェクトを基本に構成されるが，データ独立性やビューの変化に対する適応性などを満足させるため，いわゆる３層スキーマにおける各層の機能を有するオブジェクトを設ける．

先のリレーションに対応するオブジェクトをデータオブジェクトと呼び，概念スキーマを提供する．したがって本システムは概念スキーマとして，リレーショナルモデルを提供する．

データオブジェクトの実装においては，メモリ効率を考えて，ひな型オブジェクトと呼ばれるオブジェクトのリエントラントな手続き群を複数のデータオブジェクトで共用する構造とした．従って，個々のデータオブジェクトは，手続き本体を持たず，それら本体の存在場所情報と自リレーション名，構成属性名などのローカルデータのみを持つ．

外部スキーマを提供するのは質問処理オブジェクトである．質問処理オブジェクトは，ユーザに応じたビューの提供を行なう．複数の外部スキーマ（ユーザインターフェイスなど）が必要な場合には，各々の外部スキーマ提供に必要な機能を有する質問処理オブジェクトを用意する．

内部スキーマは，ファイル処理オブジェクトが提供する．ファイル処理オブジェクトは，データオブジェクトに対してデータ記憶の内部構造に依存しないデータの記憶，読み出しなどの機能を提供する．ファイル処理オブジェクトの存在により，高度なデータ独立性を得る．現在，文字，数字からなるタプルを対象とする，図２に示すようなデータ構造のファイル処理オブジェクトを実装している．

以上述べた，本データベースシステムの各オブジェクトを図３に示す．また，これらのオブジェクトと，これらの機能を実現するメソッド群の対応を表１に示す．

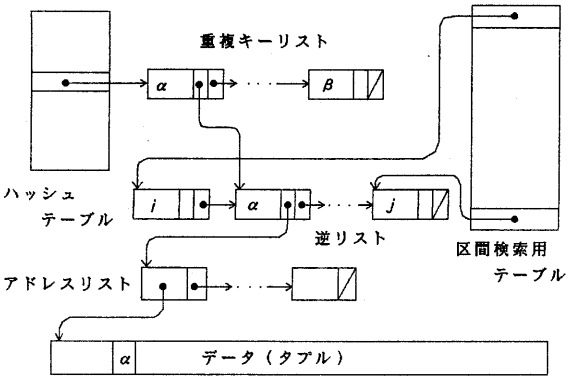

図２．データストレッジ構造

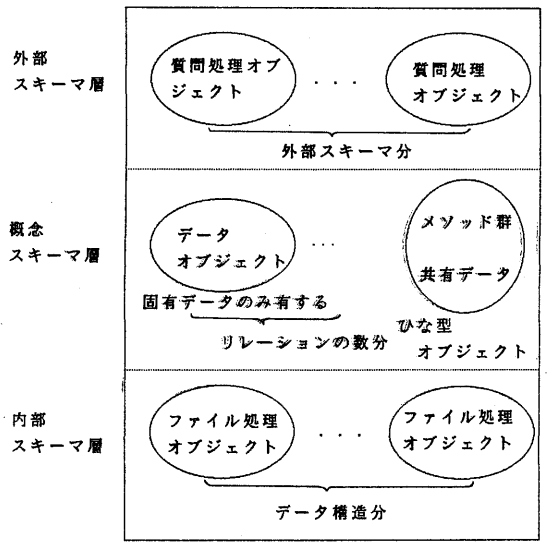

図３．　各ワークステーションのオブジェクト構成

表１　各オブジェクトの機能を実現するメソッド

質問処理 オブジェクト	質問文処理メソッド 最適化メソッド 自動結合メソッド　など
データ オブジェクト	関係操作メソッド　など
ファイル処理 オブジェクト	データ検索メソッド データ整列メソッド　など

3.3　オブジェクトの自律機能

本節ではオブジェクトであるリレーションの自律動作により可能となる機能について，複合オブジェクトによるデータ間の関連の表現および自動検索を例に述べる．分散管理による関係演算の並列処理もオブジェクト化による効果と見なすことができるが，これについては次章で述べる．

3.3.1　複合オブジェクトによるデータ間の関連の表現

多くの文献[3]で指摘されているように，複合オブジェクトの概念は重要である．複合オブジェクトの扱いの提供は，実世界のより自然な表現およびデータ操作の簡単化を可能とする．

例えば，ＥＲモデルなどではリレーション間の関連を記述するためのリレーションが必要となる．しかもリレーション間の関連を表わすには，各々のリレーションの構成属性名など自身以外のリレーションの内部構造に関する情報が必要である．

オブジェクト指向アプローチでは，オブジェクトの自律性を利用することにより，単にオブジェクト中に他のオブジェクトの名前を記述すだけで同様の関連を記述できる．

たとえば，「くるま」を考える．車を構成する部品として，「エンジン」，「サスペンションユニット」などが，「エンジン」には「キャブレータ」，「ピストン」，「コンロッド」，「サスペンション ユニット」には「スプリング」，「キャブレータ」には「スロットルバルブ」などがある．

この関係をリレーショナルモデルを用いて，第３正規形の リレーションの集合で 表現しようとすると，例えば図４のようになる．ここでは，リレーションV_Pがリレーション VEHICLEとPARTSの 間の 関連を 示している．従って，リレーションV_Pの作成にあたっては，VEHICLE，PARTSの 属性nameの定義域などを考慮しなければならない．また，リレーション V_Pの属性名p_nameと リレーションPARTSの属性名nameが，

V_P

name	・・・	p_name	・・・
くるま		エンジン	
くるま		サスペンション	

PARTS

name	p_name	・・・
エンジン	キャブレータ	
エンジン	ピストン	
・・・	・・・	
キャブレータ	スロットルバルブ	

VEHICLE

name	・・・
くるま	
くるま	
・・・	・・・
飛行機	

図４．リレーショナルモデルによる「くるま」の構成の表現例

意味的に等しいものであることを知っている必要がある．しかし，上述したように関係が込み入ってくるとこのような必要な関係の把握およびこの関連を考慮しながらのデータ操作は困難となる．

本システムでは，属性値としてのリレーション名の記述を許し，この問題に対処している．図５に表現例を示す．VEHICLE，RECIPROCAL_ENGINE，SUSPENTION_UNIT，CARBURETER，などの対象世界の実体毎に対応したリレーションを構成する．また，上記の実体間の関係を表現するために，リレーション VEHICLE の属性 name の属性値が「くるま」であるタプルの属性 p_name の属性値として，リレーション名 RECIPROCAL_ENGINE，SUSPENTION_UNIT を与える．さらに，リレーションRECIPROCAL_ENGINE の属性 p_name の属性値としてリレーション名 CARBURETERを与える・・・などによりユーザは異なるリレーションの属性間の関係を知らなくとも必要なデータを得ることが可能となる．

リレーションをオブジェクトとし，属性値として関係名の記述を許すことによって，以上のようなリレーション間関係を他のデータから独立で簡潔に記述することが可能となる．

VEHICLE

name	・・・	p_name	・・・
くるま		RECIPROCAL_ENGINE	
くるま		SUSPENSION_UNIT	
・・・		・・・	
飛行機		WING	

RECIPROCAL_ENGINE

・・・	p_name	・・・
	CARBURETER	

SUSPENSION_UNIT

・・・	p_name	・・・
	SPRING	

CARBURETER

・・・	p_name	・・・
	THROTTLE_VALVE	

図５．知的分散データベースにおける「くるま」の構成の表現例

3.3.2 自動検索

リレーショナルモデルにおいて，リレーションの数が多くなるとリレーション間の関係が複雑になり，ユーザの手に負えなくなることは上述した通りである．　知的分散データベースでは，複合オブジェクトの導入によりこの問題に対処しているが，ほしい属性がどのリレーションに存在するかがわからないなど，不完全な質問文の入力は避けられないと考えるべきである．

例えば，図６に示すデータベースにおいて，ユーザが以下のような質問を入力したとする．

Q；ＳＥＬＥＣＴ　ｇ　ＡＮＤ　ｉ
　　ＦＲＯＭ　　？
　　ＷＨＥＲＥ　　ａ＝”α”ＯＲ　ａ＝”β”

この質問は，ＦＲＯＭ句が？となっており，ユーザは属性ａ，ｇ，ｉがどのリレーションに属しているか知らない．

一般に，このような質問はリレーション間関係の陽な記述がなされておらず，処理不可能である．知的分散データベースでは，オブジェクトであるリレーションの自律機能を利用し，以下のようなリレーション間の自動結合によるデータの自動検索を可能としている．

SELECT g, i
FROM ?
WHERE a="α" OR a="β"

R1　　Load = 1

t_#	・・・	i	・・・	g	・・・
1		4		x	
2		5		y	
3		6		z	

R2　　Load = 3

a	b	c	t_#	・・・
α	A	Φ	1	
β	B	Ψ	2	
γ	Γ	Ω	3	

R3　　Load = 2

b	・・・	t_#	・・・
Γ		3	
Δ		7	
Θ		9	

図６．リレーション構成例

1）　質問を受け付けた質問処理オブジェクトは，全データオブジェクトに制約条件中の属性（制約属性）ａと出力対象属性（目的属性）ｇ，ｉを放送する．

2）　目的属性ｇ，ｉを含むデータオブジェクト（図６では，Ｒ１．このように自身を構成する属性のうち上記メッセージ中に含まれる属性をマッチング属性と呼ぶ）のみが以下を実行する．

もし，それらデータオブジェクトが制約属性ａも含むならば，図７に示すフォーマットのメッセージのパスオブジェクトリスト部にマッチング属性ｇ，ｉと自オブジェクト名を記入して質問処理オブジェクトに返送することにより，１つのアクセスパス

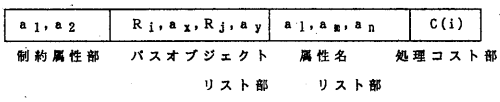

図7. 自動結合時メッセージフォーマット

が形成される.

ここでメッセージ構成要素各部の意味は次の通りである. 制約属性部は質問文の制約条件中の属性名のリストを有する. パスオブジェクトリスト部は, メッセージに対して処理を行なったオブジェクトの名前とマッチング属性の系列, 属性名リスト部は, 次に処理を行なうオブジェクトが有しているべき属性の名前のリスト, 処理コスト部は, パスオブジェクトリストの系列にしたがって処理を行なった場合の処理にかかるであろうコスト*)をそれぞれ有している.

制約属性aを含まず目的属性g, iのみを含む場合は, 上述の定義にしたがってメッセージを作成し, データオブジェクトを対象として放送する. その際, 属性名リスト部は自身の有する属性からマッチング属性を除いたものとなる.

3) 放送されたメッセージ中の, 属性名リスト部中の属性名に一致する属性名(すなわちマッチング属性)を持つデータオブジェクトのみが, 以下を実行する.

そのデータオブジェクトが制約属性aを含むならば, 2)の場合と同様, 送信されてきたパスオブジェクトリストにマッチング属性名と自オブジェクト名を付加し, 自身での処理コストを加算して質問処理オブジェクトに返送することにより, アクセスパスが形成される.

制約属性aを含まなければ, 2)と同様にマッチング属性名の系列と自オブジェクト名を付加したパスオブジェクトリスト, 自オブジェクトの属性よりマッチング属性を除いた属性名リスト, 自身での処理コストを加算した処理コストから構成されるメッセージをデータオブジェクトに放送する.

4) 新たなアクセスパスが形成されなくなるか, または必要なアクセスパスが発見されるまで, 3)を繰り返す.

上記の方法で質問処理オブジェクトに返信されたメッセージのパスオブジェクトリストに現われるオブジェクトを, オブジェクト名間のマッチング属性名で結合したものが, 求めるアクセスパスである. ここでは, 目的属性, 制約条件中の属性をデータオブジェクトがすべて含む場合について述べたが, それらの一部を含む場合についても同様な方法でパスを探索することが可能である.

上記により, アクセスパスの分散探索が可能であるが, 次のような問題がある.

- パスオブジェクトリストが, 無限長になる可能性がある. リレーションの数, 属性の数を有限とすれば, これはパスの無限ループを意味する.
- 複数パスが発見される可能性がある.

前者については, パスオブジェクトリスト中に自身の名前が存在しかつ, その名前の前に記されているマッチング属性名が, 現在のマッチング属性名と等しいならば, 無限ループに陥ると判断して処理を中止することにより, 無限経路のパスの生成を防ぐことができる.

後者については, ユーザビューによって定められる属性間の関連度をもとに, 関連度のより大きなパスを選択する[4]など 多くの議論がなされているが, 有効な解決法を見い出すに至っていない. 我々のシステムにおいてはユーザによる選択としているが, 今後, 関連度や処理コストなど各種方法の比較を行う予定である.

*) 例えば,
処理コスト

$$C(i)=C(i-1)+Natt*CpuLoad$$

とする. ここで,

Natt : 一致した属性名数,
CpuLoad : そのオブジェクトが存在するサイトの負荷.
C(i) : パスオブジェクトリスト上で, i番目のオブジェクトまでの処理に必要となるコスト. ここでは, 自身までのオブジェクトでの処理で必要となるコスト. C(i-1)は, 送られてきたコストになる. また, C(1)=Cuploadである.

表2に上記質問文に対する処理コスト付きのパスの候補群を示す. ユーザは示された候補群から任意のパスを選択することができ, 質問処理オブジェクトはそのパスに沿って処理をする. 上の探索手順では, 問い合わせの都度放送を行っているが, オブジェクトの登録時に予め自属性名から到達可能なオブジェクトと属性名リストを探索しておけば, 問い合わせ時には放送する手間を省くことが可能である[2].

表2 自動結合におけるアクセスパスの検索結果

	アクセスパス	コスト
パス1	R1 —t_#— R2	4
パス2	R1 —t_#— R3 —b,t_#— R2	9

4．分散管理の特徴，その他

本章では，関係演算の並列処理とデータ一貫性の保証処理における分散管理機能を紹介する．これらの機能は，個々のオブジェクトの自律機能によって集中管理機構を介することなく実現され，拡張性，適応性，信頼性などに優れたシステムの実現を可能とする．

4.1 関係演算の並列処理

資源の多重化は，信頼性の向上などを目的として分散型システムにおいてよく使用される手段の１つである．知的分散ＯＳにおいても，資源の多重化（オブジェクトのコピー）を許しているが，その際同種の資源利用に際しては利用可能な資源群間で負荷分散をおこなうことによって，信頼性の向上とともに効率の向上を可能としている．

以下，関係の多重化を利用した並列処理による高速化について，論理和の実行を例に説明する．

Ａ，Ｂ２つのリレーションの論理和を求め，結果をあるサイトＸに送る場合を考える．

リレーションＡとリレーションＢは異なるサイトに存在すると仮定する．このとき，次の２つの方法を考えることができる．

1) リレーションＡ，Ｂを有するサイトから，各リレーションをサイトＸに送り，Ｘにおいて論理和を求める．この方法では，並列実行は行なわれない．
2) 多重化されているリレーション（Ｂとする）を有するサイト間で，各サイトの相手のリレーション（Ａとする）に関する演算担当範囲を負荷などを考慮して決める．その後，これらのサイト群に，リレーションＡを送る．これらのサイトで，先に決めたＡに対する自身の担当範囲とＢの全部との間の論理和を求め，それをサイトＸに送る．サイトＸは，送られてきたこれら部分結果を併合することで解を得る．Ｂを有するサイトにおける部分結果算出時に，並列処理がなされることになる．

図８に，論理和処理の並列実行の例を示す．並列実行が可能な場合には，次に示すコストを考慮して，並列実行を行なうか否かを判定する．いま，

ＮＲ：リレーションＲのコピーの数
ＣＲ：リレーションＲのタプル数

とし，各サイトの担当範囲は，サイト間で均等割りをすると仮定する．また，処理コストとして

転送コスト＋リレーション処理時間

を考える．ここで，転送コストは転送すべきタプル数に比例し，リレーション処理時間はタプル比較数に比例するとする．このとき，処理コストは

1)の場合
$Cost1 = CA + CB + \alpha \times (CA \times CB)$
2)の場合
$Cost2 = CA + \alpha \times \{CB \times (CA / NB)\} + (CA + CB)$　（最大）

となる（放送通信利用を仮定）．

$Cost1 > Cost2$が成立するには，

$NB > CB / (CB - 1 / \alpha)$

となればよい．リレーション中のタプル数が十分に大きければ，

$CB / (CB - 1 / \alpha) \fallingdotseq 1$

と考えてよいから，Ｂのコピーが存在する限り2)の方法を利用したほうがよいことがわかる．

□

Ａ，Ｂともにコピーが存在する場合は，Ａを，Ｂを有するサイトに送る場合のコスト

$CostA = CA + \alpha \times \{CB \times (CA / NB)\} + (CA + CB)$　（最大）

と，ＢをＡを有するサイトに送る場合のコスト

$CostB = CB + \alpha \times \{CA \times (CB / NA)\} + (CA + CB)$

を比較して，どちらを送るかを決める．

□

上述のコスト計算は，演算の種類や負荷状態によって多少異なるが，基本的な考え方は上述の通りである．知的分散データベースでは，選

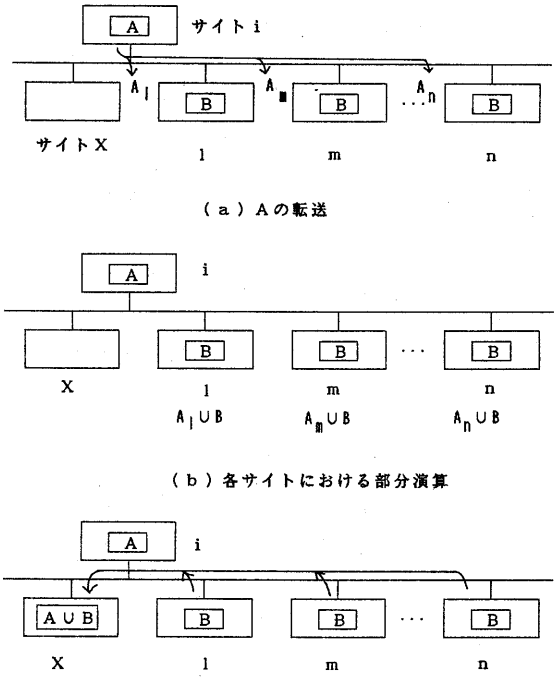

図８．知的分散データベースにおける論理和の並行実行

択，射影など単一リレーション演算についても，必要に応じて並列処理を行なう．

4.3 トランザクションの並行処理制御

トランザクションの並行処理制御は，データベースシステムにおける重要な問題の１つである[5][6]．

知的分散データベースでは，トランザクションのモデルとして，ユーザがreadまたはwrite操作であるサブトランザクションをシステムに実行要求として送り，システムがこの操作を実行後，結果をユーザに返し，ユーザはその結果をもとに次のサブトランザクションをシステムに送るということの繰り返しにより処理を行なうタイプを対象とする．

現在，このタイプのトランザクションを対象とした 並行処理制御の方法には 大きく分けて，ロック方式とスケジューラ方式の２つがある．ロック方式については，様々な 研究がなされており[7] 適用事例も 多い．スケジューラ方式としては，先読みスケジューラ（Cautious Scheduler）を用いた方法などが提案されているが，処理の実行に先立って実行するサブトランザクションを宣言する必要があるなどの制約のため普及するに至っていない．

しかし，先読みスケジューラを用いた方式では，宣言をしておけば必ず実行要求は実行される，デッドロックに陥ることはない，などの望ましい性質が知られている[8]．さらに最近，事前の宣言がなくとも楽観的スケジューラ[7]として動作する，スケジュールに要する時間の複雑さは，毎回デッドロック検出をするロック方式に比べ少ないことが明らかにされた[9]．

我々は，各オブジェクトが自身のリレーションに対する操作を制御する先読みスケジューラをもつ並行処理制御方式を採用する．トランザクションの実行に先立って正しい宣言がなされていれば，分散環境下においても上述の性質を満たして動作し[10]，正しい宣言がなされなかった場合にも，デッドロックの検出が可能である（証明略）．

5．おわりに

本稿では，現在構築中の知的分散データベースシステムについて，その概要を紹介した．

リレーションをオブジェクトとし，複合オブジェクトの扱いを許すことで，ユーザは容易にリレーション間の関連を得られるだけでなく，対象データに対する操作を施すのみで関連データの処理がなされるシステムを実現できる．さらに，並列処理や並行処理制御の分散管理が容易となった．

また，３層スキーマの各層に対応するオブジェクトを実装することで，高度な拡張性，信頼性，データ独立性を備えたシステムとすることが可能となった．

今後は，

・並行処理制御に関して，多版データベースへの対応
・分割障害などに対する対応

などが，必要と考える．さらに，

・ファイル自動結合実行時における，実用的なアクセスパス選択アルゴリズムの提案
・複合オブジェクトを前提とした，データモデルの提案

などを行なってゆく予定である．

参考文献

〔1〕 関 他:"知的分散OS"，情処学マルチメディア研資 (1988-9).

〔2〕 田村 他:"知的分散システムのアーキテクチャ",電学誌 108-C〔6〕(1988-6).

〔3〕 情報処理学会誌:"マルチメディアデータベース総論", 28,6 (1987-6).

〔4〕 田淵 他:"距離空間に基づいたデータモデルの提案について MeSOD"，情処学DB研資 (1987-9).

〔5〕 P.A.Bernstein and N.Goodman: "Concurrency control in distributed database systems", ACM Computing Surveys, 13,2 (1981).

〔6〕 M.Takizawa : "Transaction Management by Prolog ", Proc. of Logic Programming Conference, 9,2 (1987).

〔7〕 P.A.Bernstein, V.Hadzilacos, N.Goodman : "Concurrency Control and Recovery in Database Systems", ADDISON-WESLEY (1987).

〔8〕 茨木:"不完全な宣言のもとでの先読みスケジューラ", 信学DE研資 (1987-11).

〔9〕 N.Katoh, T.Kameda, T.Ibaraki : "Efficient Implementation of Cautious Schedulers". LCCR Tech.Rep. Simon Fraser Univ. 1988.

〔10〕 原嶋, 茨木:"先読みスケジューラによる分散型データベースシステムの並行処理制御", 信学論 J70-D, 6 (1987).