分散データベースシステムのオブジェクト指向設計

二村祥一　河内一浩　神田健二　荻晋也

大分大学工学部 知能情報システム工学科

概要

分散データベースシステムをオブジェクト指向を応用して設計した．分散データベースを，分散配置された複数の関係データベースの統合体としてとらえ，その統合技術について考察した．分散システムモデルとしてサーバ・クライアントモデルを用い，オブジェクト指向設計により実現容易性とシステム効率の向上をめざした．また，応用機能とユーザインタフェースを分離して扱うＭＶＣが分散システムの開発に有効であることを示した．

１．はじめに

分散した計算機資源の活用を図るには分散した資源を，あたかも一つの計算機の中に集中しているかのように見せかけ処理するための機構が必要である．そのためのシステムの一つが分散データベースシステムである［9］．

分散データベースの構成法としてトップダウンとボトムアップの二つが考えられる．トップダウン構成法では最初に目的とする機能を設定し，それにそって分散システムを構築していく．これに対してボトムアップ構成法では最初にデータが分散配置されており，それらのデータを扱う分散システムを構築していく．ここでは後者の立場をとる．そこでは個々に作成されて分散配置しているデータベースを統合することが必要であるとともに，データベースの置かれている計算機の違いを吸収するための機構が必要になる．ここではオブジェクト指向［5,7］の考え方を応用することにより，分散システムの開発容易性とユーザからみたデータ操作性の向上を図る．

基底データベースとしては分散システムでのデータ統合性の良さから，構造が平坦な関係データベースを採用した．関係データベースは世界モデルの構築が容易であり，その上に多くの応用システムの稼働が可能であるためである［6,8］．

分散計算機環境としてはＬＡＮによって統合されたＵｎｉｘワークステーション群を想定している．

以下，まず分散データベースシステムのモデルについて第２章で述べる．第３章で基底データベースシステムの応用プログラムインタフェースについて，さらに第４章で要素データベースから分散データベースへの統合について述べる．最後に，第５章で応用機能とユーザインタフェースを分離して扱うＭＶＣ［5］の考え方を応用した分散システムの設計について，ハイパーメディアシステム［1,4］を例に示しながら論じていく．

２．分散データベースシステムのモデル

複数のＵｎｉｘワークステーションをＬＡＮによって接続した分散計算機環境のもとで，分散データベースを構築する．分散データベースシステムの機能階層を図１に示す．

Object-Oriented Design for Distributed Database Systems
by Shouichi FUTAMURA, Kazuhiro KAWAUCHI, Kenji KANDA, and Shin'ya OGI
(Department of Computer Science and Intelligent Systems, Oita University)

各ワークステーションに配置された要素データベースを統合して一つの仮想的なデータベースを構成する．ここではそのデータベースを仮想データベースと呼ぶ．仮想データベースの上にオブジェクト・データベースを構築し，これをユーザあるいは応用プログラムにみせる．要素データベースと仮想データベースは分散計算機環境でのデータ操作性を考えて関係データベースで管理する．オブジェクト・データベースはオブジェクト指向概念のオブジェクトの集まりとしてのデータベースで，これはユーザあるいは応用プログラムに，より自然で扱いやすいデータ処理のインタフェースを提供する．

分散した要素データベースを統合するためのシステムモデルとしてサーバ・クライアントモデル［10］を用いる（図２を参照のこと）．各ワークステーションにクライアントとサーバをそれぞれ配置する．クライアントはユーザからの質問を受け，処理結果をユーザに返す．サーバは対応する要素データベースの処理を担当する．クライアントはユーザからの質問を，複数のサーバへの処理に分解し，それぞれのサーバに処理依頼を行う．サーバにおける処理は並列に実行できる．クライアントは複数のサーバからの処理結果を待ち，処理結果の統合を行ったのちユーザに回答を返す．

サーバ・クライアントモデルは，ユーザ質問が発生するワークステーションを含めて全てのワークステーションを対等に扱うことを可能にする．サーバおよびクライアントをそれぞれデータと処理をカプセル化したオブジェクトとして設計することにより，その間のメッセージ通信量，すなわちネットワークにおけるデータ転送量を最小化できる．

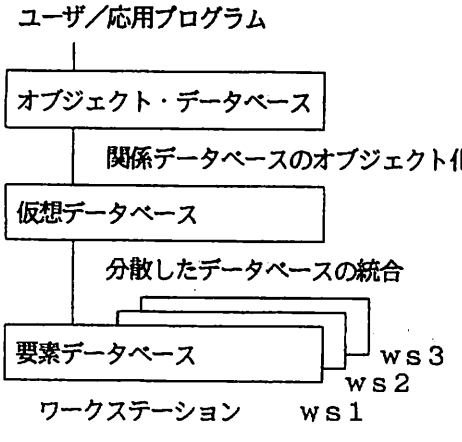

図１．分散データベースシステムの機能階層

３．基底データベースシステム

３．１　Ａｄｂｉｓ／ＤＭＰインタフェース

分散した要素データベースの管理のために関係型データベース管理機構Ａｄｂｉｓ／ＤＭＰ（以後，ＤＭＰと省略する）を使用する．ＤＭＰは関係データを扱うためのモジュール群である［3］．ＤＭＰは当初，九州大学大型計算機センターの汎用大型計算機のもとでプログラミング言語Ｆｏｒｔｒａｎを用いて記述していたが，それを大分大学知能情報システム工学科のＵｎｉｘワークステーションのもとで効率的に動作するように言語Ｃを用いて書換えを行った［2］．Ｕｎｉｘ版のＤＭＰは，データベース定義やデータベースのオープ

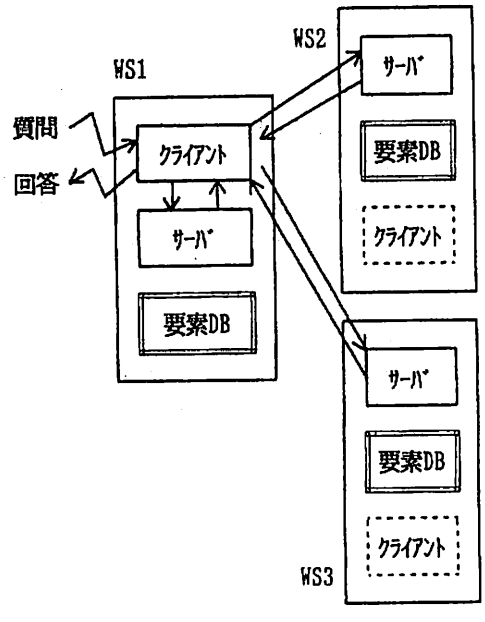

図２．サーバ・クライアントモデル

ン・クローズ，データベース定義情報の参照，データベースの構築や検索，結果表示，表の射影や結合などの機能をもつ．

DMPでは関係単位に表形式（tabular form）を作り，また表の属性値から組を高速に求めるための転置形式（inverted form）を作る．検索結果は値集合として保存して再利用できる．DMPはそれぞれの機能をCの関数として実現しており，それらを組み合わせることによりデータベースシステムや応用プログラムを記述することができる．ここでは要素データベースの管理のためにDMPを利用する．

DMPは異機種計算機のための共通データベースインタフェースとしても利用できる．学科の分散計算機環境にはデータベース処理を高速に実行できるデータベースマシンBL300がある．これはデータベースサーバ機能を提供する．データベースマシンは検索言語SQLをサポートしており，またそれをC言語のプログラムから関数の形で呼び出すことができるようになっている．そこで，それらの関数を利用してBL300をDMPと同じインタフェースで扱うための関数群を作成した．これにより，DMPのもとで構築したデータベースとデータベースマシン上のデータベースとを区別なく扱えるようになった．

DMPインタフェースへの変換は，データベースマシンのような関係型データベースの場合は容易である．また，階層型やネットワーク型のデータベースについても親子集合での親の主キーをうまく使えば比較的容易であると思われる．

3．2　関係データベース処理のためのクラス

DMPによるデータベース処理は通常の関数呼出しによるため手続き的な処理になる．そのため分散データベース等のマルチプロセス環境では非常に使い勝手が悪い．そこで，関係データベースにおける自然な操作対象に対してオブジェクトを用意することにより，分散オブジェクト環境での，手続き型（how）から非手続き型（what）への処理移行を図った．

関係データベース処理のために用意したオブジェクトのクラスを表1に示す．

DBsystemは特定のワークステーションにおける複数のデータベースを管理する．Databaseは一つのデータベースを管理するためのクラスである．データベースのオープン時にデータベース内の全ての関係をインスタンス化し，それらを扱えるようにした．Relationは関係に対応するクラスである．Relationは関係表からの検索機能を提供するselectなどのメソッドをもつ．関係表からの検索結果は，値集合として順次保存され，後の処理で再利用できる．検索結果に対応するクラスがSetである．関係や値集合からの組はクラスTupleのインスタンスとして扱う．Tupleクラスから複数の異なったサブクラスを導出することにより，それぞれのサブクラスのインスタンスに同一のメッセージを送り異なる動作を行わせることができる（ポリモルフィズム）．このほか，関係や値集合

表1．関係データベース処理のためのクラス

クラス名（格納データなど）	メソッド	機能説明
DBsystem（ﾃﾞｰﾀﾍﾞｰｽ名ﾘｽﾄ）	display	ﾃﾞｰﾀﾍﾞｰｽ一覧
	connect	ﾃﾞｰﾀﾍﾞｰｽのｵﾌﾞｼﾞｪｸﾄ化
	quit	ｼｽﾃﾑ終了
Database（ﾃﾞｰﾀﾍﾞｰｽ定義情報）	open	ﾃﾞｰﾀﾍﾞｰｽｵｰﾌﾟﾝ
	close	ﾃﾞｰﾀﾍﾞｰｽｸﾛｰｽﾞ
	display	ﾃﾞｰﾀﾍﾞｰｽ定義情報一覧
Relation（関係情報）	select	値集合検索
	display	関係情報一覧
	getno	組獲得
Set（値集合情報）	display	値集合情報一覧
	no	値集合要素数
	getno	組獲得
Tuple（組情報）	display	内容表示
J-operation（結合操作）	join	結合
B-operation（集合演算）	and	集合の積
	or	和
	not	差

の結合を行うためのクラスＪ－ｏｐｅｒａｔｉｏｎと，集合演算のためのＢ－ｏｐｅｒａｔｉｏｎを用意した．

ここではＤＭＰを使ってこれらのクラスを作成したが，それぞれのワークステーションのデータベース機能の上にこれらのクラスを作成することもできる．これら関係データベース処理のためのクラスを利用することにより，システム独立な分散データベースシステムが構築できる．また，クラス設計により分散環境でのメッセージ転送量は処理を非手続き的に指定できるため，最小のものとなる．

４．分散したデータベースの統合

４．１　分散データベースの管理情報

分散データベースをワークステーションごとに配置された要素データベースから構成し，ユーザに仮想データベースをみせる．ここでは要素データベースおよび仮想データベースは関係データベースを基礎にして考えていく．

データベース統合のために，外部スキーマと共通スキーマ，ローカルスキーマの３個のスキーマと，システム情報を用意した．これらの管理情報は次の形式の表によって管理する．

外部スキーマ（接続キー，外部表名，属性名，
　　属性の型，結合属性，制約条件）
共通スキーマ（接続キー，ホスト名，共通表名，
　　属性名１，属性の型，制約条件）
ローカルスキーマ（共通表名，属性名１，
　　制約条件，ローカル表名，
　　属性名２，属性の型）
システム情報表（ホスト名，ＲＰＣ情報）

３個のスキーマにおける各組は，外部表，共通表，ローカル表の属性を単位としている．それらスキーマの関連を図３に示す．外部スキーマは分散データベースのユーザのための仮想表である．共通スキーマは分散データベースの世界モデルを記述する．外部スキーマと共通スキーマの変換は接続キー属性を用いて行う．ローカルスキーマは要素データベースに対応したスキーマで，要素データベースの管理者が分散データベースのユーザに提供する仮想表を定義する．共通スキーマとローカルスキーマの変換は，共通表名と属性名１の組をキーにして行う．システム情報は各ホストにおけるデータベース通信プログラムを識別するためのもので，ＲＰＣの引数となるものである．

外部スキーマと共通スキーマ，さらにシステム情報表はクライアントが使用するもので，これらは分散データベース全体で一つあり，全てのクライアントが共通に利用する．ローカルスキーマはワークステーション単位にもつもので，これはサーバが使用する．

このような３個のスキーマを用意することにより，表の位置透明性や分割透明性，また表名，属性名，属性の型の変更を含むデータベーススキーマ透明性などの分散データベース透明性を達成できた．また３層のスキーマ構成にすることによって，分散データベースの応用プログラムにデータ独立性を持たせ，さらに分散した要素データベースの自立性を持たせることを可能にした．

管理情報の管理には，外部スキーマと共通スキーマを管理する分散データベース管理者と，それぞれのローカルスキーマを管理する要素データベース管理者が必要である．

４．２　管理情報によるデータベースの統合

分散データベースのユーザは外部スキーマに従ってクライアントに質問式を入力する．この質問式を，管理情報の外部スキーマと共通スキーマを

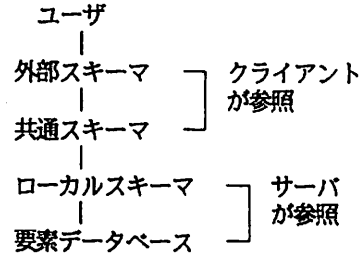

図３．分散データベーススキーマの関連

用いて複数のコマンドに展開する．各コマンドはシステム情報表に従って各サーバに送られる．サーバでは，送られてきたコマンドをローカルスキーマを用いて要素データベースのコマンドに変換し，そこでデータベース処理を実行する．実行結果はクライアントに送り返され，クライアントで処理の統合を行う．その結果がユーザに返される．

データベース統合処理の例を以下に示す．

表h1 (tid, title, file, media) がワークステーションws1に，表h2 (tid, kword) がワークステーションws2にあり，表h1と表h2を結合した仮想表H12 (TID, TITLE, MEDIA, KWORD) を共通スキーマおよび外部スキーマに定義する場合を考える．

各スキーマとシステム情報の定義を図４に示す．表名，属性名および属性の型はスキーマ間で変更してよい．ここでは同一の属性名を指示するために大文字と小文字の変化だけで異なる名前付けを行っている．

ここで，例えばクライアントに

```
select TITLE,KWORD from H12
        where KWORD='inf%' ;
```

のようなＳＱＬコマンドが入力された場合，ｗｓ１，ｗｓ2には次のようなＳＱＬコマンドが転送される．

外部スキーマ

接続キー	外部表名	属性名１	属性の型	結合属性	制約条件
2	H12	TID	int	-	-
4	H12	TITLE	char 40	TID	-
6	H12	MEDIA	char 6	TID	MEDIA='TEXT'
8	H12	KWORD	char 12	TID	-

共通スキーマ

接続キー	ホスト名	共通表名	属性名１	属性の型	制約条件
2	WS1	H1	TID	int	-
4	WS1	H1	TITLE	char 24	-
6	WS1	H1	MEDIA	char 6	-
2	WS2	H2	TID	int	-
8	WS2	H2	KWORD	char 12	-

システム情報

ホスト名	prog_no	version_no	proc_no
WS1	20000005	1	1
WS2	20000006	1	1

ローカルスキーマ（WS１）

共通表名	属性名１	制約条件	ローカル表名	属性名２	属性の型
H1	TID	-	h1	tid	int
H1	TITLE	-	h1	title	char 20
H1	FILE	-	h1	file	char 20
H1	MEDIA	-	h1	media	char 6

ローカルスキーマ（WS２）

共通表名	属性名１	制約条件	ローカル表名	属性名２	属性の型
H2	TID	tid>=20	h2	tid	int
H2	KWORD	-	h2	kword	char 12

図４．分散データベース管理情報の例

```
[ws1] select TID,TITLE from H1
                    where MEDIA='TEXT' ;
[ws2] select TID,KWORD from H2
                    where KWORD='inf%' ;
```

両コマンドの評価属性にＴＩＤが加わるのは，ＴＩＤが結合属性であるからで，またｗｓ１の方に検索条件 MEDIA='TEXT' が加わっているのは制約条件が科せられているためである．それぞれのサーバではローカルスキーマの定義に従って，次のＳＱＬコマンドが生成され実行される．

```
[ws1] select tid,title from h1
          where media='text';
[ws2] select tid,kword from h2
         where kword='inf%' and tid>=20 ;
```

ＷＳ２の方に検索条件 tid>=20 が加わっているのは，ローカルスキーマに制約条件が科せられているためである．

それぞれのワークステーションでの処理は並列に行われ，結果がクライアントに返される．クライアントでは，結合属性ＴＩＤは最終的には評価属性から削られる．

５．ＭＶＣによる分散システム設計

ここでは分散システム実現のためにＭＶＣ（Model-View-Controller）［5］の概念を応用する．ＭＶＣではシステムを，モデル（Model），ビュー（View），コントローラ（Controller）の三つの部分から構成する．モデルは応用機能そのものを扱う部分である．ビューはディスプレイを通しての情報の見え方を扱う部分であり，またコントローラはユーザからの入力を解釈して，モデルあるいはビューに適切な調整を施す部分である．ビューとコントローラは応用機能のユーザインタフェースを提供する．計算機依存部分をビュー・コントローラにまかせるため，モデルは応用機能そのものに集中することができる．

この章では，ハイパーメディアシステムの構築を例にして，分散計算機環境のもとでのＭＶＣ構成について考察する．

５．１　ハイパーメディアシステム

マルチメディア文書をハイパーメディアとして構造化し，その中での計算機支援によるより自然な文書航行や文書検索などの機能の実現をめざしている［1］．マルチメディア文書全体を文書要素に分け，文書要素をノードとし，要素間の関連付けのためにリンクを用意する．文書要素として，ここではまとまった内容を持つ文章や，表，図形，画像などを想定している．ノードとリンクの情報は，関係データベースを用いてノード表，リンク表として管理している．

ハイパーメディアシステムの現在のユーザインタフェースは，マルチウィンドウシステムＸ－Ｗｉｎｄｏｗを用いることにより，主にマウス操作により，各種の処理が行えるようにしている．システムのウィンドウは，注目文書ウィンドウと資料文書ウィンドウ，ブラウザウィンドウ，コンソールウィンドウから構成した．注目文書ウィンドウは注目文書を表示するためのウィンドウであり，資料文書ウィンドウは注目文書に関連した複数の文書を表示するためのウィンドウである．ブラウザウィンドウはハイパーメディアの文書構造の表示と文書の直接選択の機能を提供する．コンソールウィンドウはいくつかのボタンを持ち，それをクリックすることにより，対応した応用機能を起動・停止する．

５．２　ハイパーメディアシステムのモデル設計

ハイパーメディアシステムの機能をまとめると次のようになる．

(1) 注目文書を注目文書ウィンドウに表示する．このとき注目ノードから連動リンクで結ばれているノードは同時に資料文書ウィンドウに表示する．

(2) 注目ノードを移動する．この移動には通常リンクのリンク先への移動（順行）やリンクを逆にたどる移動（逆行）などがある．

(3) 注目ノードに関連するノードを資料文書ウィンドウに参照する．

(4) このほかに，ブラウザ機能やキーワード検索機能，ヒストリー機能などがある．

これらの機能から，そのユーザインタフェース部分を分離して，ハイパーメディアシステムのモデルを作る．ハイパーメディアシステムのモデルは，ノードとリンクの集合，システム内部状態をデータとし，ハイパーメディアの各機能に対応した手続きをもつオブジェクトである．

ハイパーメディアの構成要素であるノードとリンクもそれぞれオブジェクトとして扱う．ノードとリンクは，低レベルではノード表，リンク表として関係データベースで管理しており，各表から組の内容に従って組をオブジェクト化し，応用プログラムからは，検索集合がオブジェクトのリストとして見えるようにした．ノードおよびリンクには多種類のものがある．それらノードとリンクを，それぞれクラス継承機能を使って，分類整理して扱うようにした．例えば，ノードには文章や表，画像などの区別があるので，ノードクラスのもとに，サブクラスとして文章クラスや画像クラスを配置する．ノードリストには文章クラスと画像クラスのインスタンスが同時に並び，それぞれのインスランスが所属のサブクラスに応じたメソッドを選択できるようにした（仮想関数による多態性を利用）．

クラス継承を利用することにより，差分プログラミングによるシステム開発が可能となり，また多態性によって一様なプログラミングインタフェースの提供が可能となった．

5．3　ハイパーメディアシステムのビュー・コントローラ設計

ここでは，ビューとコントローラを一つのオブジェクトとしてまとめて扱う．ハイパーメディアシステムでは，注目文書ウィンドウや資料文書ウィンドウ，ブラウザウィンドウ，コンソールウィンドウなど，ウィンドウに対応するものがビュー・コントローラを形成する．これらのウィンドウはマウスなどのポインティングデバイスやキーボードからの入力を受け付け，その指示に従ってウィンドウの表示内容を変更する．

我々はX－Ｗｉｎｄｏｗシステムのウィンドウ操作ライブラリＸｌｉｂを用いて，ウィンドウ設計のためのクラスを表２のように作成した．

Ｂａｓｅ－ｗｉｎｄｏｗは，他のウィンドウオブジェクトを置く下地のウィンドウクラスである．Ｄａｔａ－ｗｉｎｄｏｗは文章や画像の内容表示を行うウィンドウクラスである．ハイパーメディアシステムのモデルへのメッセージ通信にはｅｖｅｎｔを用いる．Ｂｕｔｔｏｎはウィンドウ上に配置されるボタンのクラス，Ｔｉｔｌｅはタイトル表示のためのクラス，Ｉｎｐｕｔはユーザの入力文字列を得るためのクラスである．Ｖｓｃｒｏｌｌ，Ｈｓｃｒｏｌｌはそれぞれ垂直と水平方向のスクロールバーであり，これらはＤａｔａ－ｗｉｎｄｏｗの内容と連動する．

これらの基本クラスのインスタンスを組み合わせて，注目文書ウィンドウ以下のウィンドウを複合オブジェクトとして構成した．例えば，注目文書ウィンドウは，Ｂａｓｅ－ｗｉｎｄｏｗのインスタンスを下地にして，その上にＤａｔａ－ｗｉｎｄｏｗのインスタンスと，それに連動したＶｓ

表２．ウィンドウ設計のためのクラス

クラス名	メソッド	機能説明
Base_window	make	ﾍﾞｰｽｳｨﾝﾄﾞｳの作成
	label	ﾀｲﾄﾙの付加
	resize	ｳｨﾝﾄﾞｳのﾘｻｲｽﾞ
	map	ｳｨﾝﾄﾞｳのﾏｯﾌﾟ
Data_window	make	ﾃﾞｰﾀ表示ｳｨﾝﾄﾞｳの作成
	event	ｲﾍﾞﾝﾄ処理
	display	内容表示
	getline	行通知
	revline	行反転表示
	getpoint	座標通知
Button	make	ﾎﾞﾀﾝ作成
	event	ｲﾍﾞﾝﾄ処理
	reverse	（白黒）反転表示
Title	make	ﾀｲﾄﾙｳｨﾝﾄﾞｳ作成
	event	ｲﾍﾞﾝﾄ処理
Input	make	入力ｳｨﾝﾄﾞｳ作成
	event	入力文字列通知
Vscroll, Hscroll	make	ｽｸﾛｰﾙﾊﾞｰ作成
	event	ｲﾍﾞﾝﾄ処理

ｃｒｏｌｌのインスタンスを置き，注目文書ウィンドウの機能に対応したＢｕｔｔｏｎ群を配置したものである．

５．４　ＭＶＣによるシステム統合の有効性

モデル，ビュー，コントローラによるシステム設計は，システムの生産性を飛躍的に向上させる．分散計算機環境では異機種の計算機が，同一の応用のために使用されることがあり，モデルを機種から独立させて作成することが有効になる．

ハイパーメディアシステムの実現で用いたウィンドウ設計のためのクラスは，異なるウィンドウシステムのもとでも，同様のインタフェースでクラスとメソッドを設計できるため，分散環境での通信メッセージを共通にすることができる．この場合は，異なるウィンドウシステムのもとでのメソッドの実現だけが必要である．

また，ウィンドウシステムの動作しない端末などに対しても，モデル自身は有効である．この場合は，ビュー，コントローラのクラス設計と，それに付随したメソッドの実現を行えばよい．

６．おわりに

分散データベースをボトムアップ的に構成する場合においてのオブジェクト指向概念の応用について述べた．各計算機上に分散配置された要素データベースをもとにして仮想的な世界モデルを提供する仮想データベースを作った．仮想データベースの上に位置するオブジェクト・データベースは応用レベルのオブジェクト指向処理プログラムと相性が良いため，それらを結合することにより，より自然な分散データ処理が期待できる．

ここでは，現在の利用環境であるＵｎｉｘ分散計算機環境において，システムの設計と実現を行ったが，これらの技術の多くが，より広域のＷＡＮや，より規模の小さいパソコンＬＡＮ等へも適用できると考えている．

要素データベースの結合処理において，現在はサーバからクライアントにメッセージ通信により処理依頼を送り，処理結果をクライアントからサーバに返している．この部分において，クライアント間のメッセージ通信（分散協調処理に相当）を利用することにより，システム性能を向上することが課題の一つにあげられる．

参考文献

[1] 二村，井手，永田：ＵＮＩＸ環境におけるマルチメディア文書処理システムの設計と実現，情報処理学会研究報告，91-SE-77-16，1991.

[2] 二村ほか：ＵＮＩＸのためのデータベース操作言語の設計，電気関係学会九州支部連合大会論文集，p.537，1988.

[3] 松尾，二村，高木：推論関係型データベース管理システムＡｄｂｉｓ，情報処理学会論文誌，Vol.24，No.2，pp.249-255，1983.

[4] Conklin,E.J: Hypertext: An Introduction and Survey, IEEE Computer, Vol.20, No.9, pp.17-41, 1987.

[5] Cox,B.J: Object-Oriented Programming: An Evolutionary Approach,Addison-Wesley, Reading,Mass., 1986.

[6] Futamura,S., Fujita,Y. and Utsumiya,K.: A Hypermedia Document System Based on Relational Database, Proc 2'nd Far East Workshop on Future Database Systems, pp.240-243, 1992.

[7] Parsaye,K.,et.al.:Intelligent Databases, John Wiley & Sons Inc., 1989.

[8] Premerlani,W.J., et. al.: An Object-Oriented Relational Database, CACM, Vol.33,No.11, pp.99-109, 1990.

[9] Sheth,A.P., and Larson,J.A.: Federated Database System for Managing Distributed Heterogeneous and Autonomous Databases, ACM Computing surveys, Vol.22, No.3, pp.183-235, 1990.

[10] Sinha,A.: Client-Server Cumputing,CACM, Vol.35, No.7, pp.77-98, 1992.