データベース・システム　68－6
（１９８８．１１．１７）

同報通信を指向した分散ＤＢ

坂本　明史、　斎藤　邦子、　疋田　定幸

沖電気工業（株）

同報通信は分散データベースシステムの処理効率化に大きな役割を果たす可能性を持っている。同報通信を用いて分散データベースシステムを構築するためには、コミットメント制御、問い合わせ処理など多くの観点から検討する必要がある。

本稿では、従来あまり行われていなかった問い合わせ処理の観点から、同報通信を用いた分散データベースシステムでの問題点を整理する。問題への対応と同報通信機能まで含めたシステムのアーキテクチャを述べ、さらに実現した同報通信システムに関して述べる。

Distributed Database System Architecture with a Broadcast Communication

Akifumi SAKAMOTO, Kuniko SAITO, Sadayuki HIKITA

OKI Electric Industry Co., Ltd.
16-8, Chuo 1-chome, Warabi-shi, Saitama,
335, JAPAN

A broadcast communication seems to be attractive for distributed database systems. Research efforts have been focused on a commitment control. However, in order to implement them as integrated systems, design efforts must be focused not only on commitment control, but also on query processings.

In this paper, authors discuss a broadcast communication scheme for distributed database system from the viewpoint of query processing. Then the architecture of distributed database system is presented. Facilities of its communication system and its implementation method are also described.

１．はじめに

分散データベースシステムは、物理的に分散した複数サイトの計算機を利用できるため、集中型システムに比べて負荷の分散や並列性などの点で有利である。反面、サイト間の情報交換による通信ボトルネックが問題となりやすく、この解決が重要な課題となる。

SDD-1[ROTH80]、分散INGRES[STON77]、R*[WILL81]といった分散データベースシステムプロジェクトでは基本的にはネットワークとして、１対１の通信形態を前提にしている。一方、Ethernet[METC76]や、衛星通信回線のように物理的な同報性を持つネットワークを前提にして分散データベースシステムを構築する方法も考えられる。同報通信では、データの転送コストが通信相手のサイト数に依存せず一定である。この性質を用いてコミットメント制御や、重複データの制御を効率的に行う検討が従来からなされている[CHAN83]。しかしながら、分散データベースを統合されたシステムとして実現するにはこれ以外にさまざまな側面から、同報通信の検討が必要である。

本稿では、従来研究が十分なされていない問い合わせ処理の観点から同報通信を整理する。すなわち、分散データベースの問い合わせ処理する上で、同報通信ネットワークまでを含めたシステムに必要な機能と構成を述べ、実現した同報通信システムについて述べる。

なお、本研究は、通商産業省工業技術院の大型プロジェクト「電子計算機相互運用データベースシステムの研究開発」の一環として研究開発を進めているものである。

２．システムモデルと用語

同報通信ネットワーク上での分散データベースシステムのシステムモデルを図１に示す。分散データベースシステムは複数のサイト上で分散データベース管理プロセスが動作することにより実現される。分散データベース管理プロセスには、問い合わせ発行サイトにあるクライアントプロセスとデータ格納サイトにあるサーバプロセスがある。以下これらを単にクライアントおよびサーバと呼ぶ。サーバは一つのクライアントに対するサービスを行う。クライアントは利用者が分散データベース管理システムを利用する時に起動され、サーバはクライアントから起動される。

クライアントとサーバは分散データベースアクセスのプロトコルを用いて通信する。通信には同報通信を用いることができる。この部分を本稿では同報通信システムと呼ぶことにする。

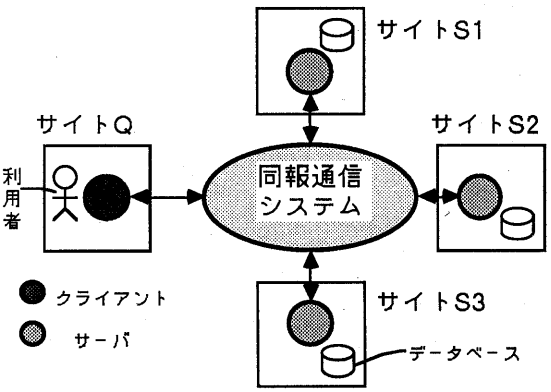


図１．分散データベースシステムモデル

３．問い合わせ処理

ここでは問い合わせ処理の観点からシステムの実現上の問題点を検討する。

図１のモデル上で二つの問い合わせ処理を考える。

例１：水平フラグメントされたリレーションＲからＲ．Ｂ＞２０を満たすタプルのＲ．Ａを検索する。利用者から与えられるＳＱＬ[SQL]での検索のステップは次のようになる。

```
・DECLARE C1 CURSOR FOR   ――検索カーソルC1
                            の設定
          SELECT R.A
          FROM   R
          WHERE  R.B > 20
・OPEN C1
・FETCH C1 INTO :RESULT     ――:RESULTは結果
                            を入れる変数
.....繰り返し
```

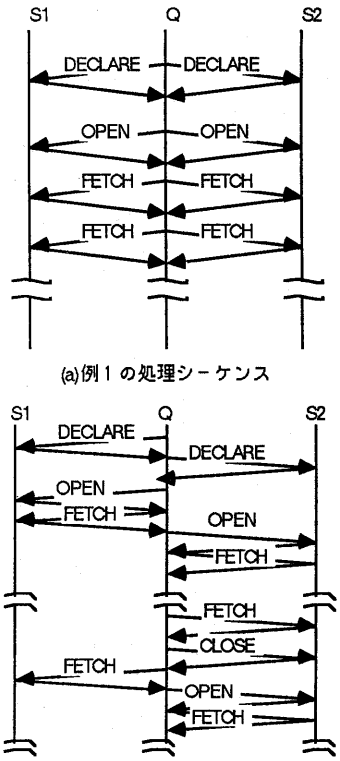
図１でＲの水平フラグメントＦ１、Ｆ２がそれぞれサイトＳ１、サイトＳ２にある場合この例１の検索に対するシーケンスは図２(a)になる。

クライアントは、DECLAREに対応してフラグメントのあるサイトＳ１とサイトＳ２に検索条件を同報しシステムの内部的なカーソルを生成する。

OPENに対しては両サイトの内部的なカーソルをオープンする。FETCHに対しても同様に両方のサイトから条件を満たすタプルをFETCHする。

例２：図１でリレーションＲ１、リレーションＲ２がそれぞれサイトＳ１、サイトＳ２にあるとしこれらに対するジョイン検索を行う。

ここでは単純にクライアントが問い合わせを分解してネステッドループによって解く方式を考える[WONG76]。この場合、クライアントはＲ１をアウタリレーション、Ｒ２をインナリレーションとして個々のリレーションに対する二つの問い合わせに分解し、これらを該当するサイトに送る。次にクライアントはサイトＳ１のアウタリレーションＲ１からジョイン属性をFETCHし、この値をインナリレーションＲ２の検索条件としてＲ２のFETCHを繰り返す。これがつきたらＲ１のジョイン属性の次の値をFETCHしてこの値を用いて再度Ｒ２の検索を行う。このシーケンスを図２(b)に示す。



(a)例１の処理シーケンス

(b)例２の処理シーケンス

図2. 問い合わせ処理の例

例１では、全ての通信で同報通信が利用できるため処理効率を向上できる。しかしながら、このような場合でも同報通信システムの実現には二つの問題点を解決しなければならない。第１は、サーバの必要なサイトは、DECLARE文が実行されるまでわからない点である。すなわち、問い合わせが実行されることにより初めて対象となるサイトが決定される。そのため、あらかじめトランザクションで使う全サーバを決めることができない。同報通信システムは、問い合わせ処理に対応して使われるサーバの増減に対応できるものでなければならない。第２はタプルのFETCHが繰り返された場合の通信回数オーバヘッドの問題である。ＳＱＬの検索は、タプルを順次アクセスする形態であり、これに対応して検索したタプルを順次アプリケーションに返すためには、転送単位はある程度以上には大きくできない。このため、FETCHは通信回数のオーバヘッドが発生し応答性が低下することになる。システムは、このような状況にも何らかの応答性の改善を行う必要がある。

第３の問題は同報通信の性質が見かけ上はまったく使えない場合があることである。事実、例２では同報通信を利用できる場合が見かけ上全くない。問い合わせの処理は逐次的に他のリレーションのタプルを照合する処理からなるため、同報通信システムを使用しても通信回数の削減が期待できないように思える。このような状況でも同報通信システムには、何らかの通信回数削減のための対策が望まれる。

以上３点が、同報通信を指向した場合、分散データベースの問い合わせ処理を効率よく実現する上で発生した問題点である。４章では、システムでの問題点に対する対応を、５章ではシステムの実現方式を述べる。

４．問題への対応

４．１　同報コネクションの動的対処

通信においては転送をリライアブルにするためにコネクションという考え方を用いる。コネクションは通信相手との論理的な通信路である。同報通信では、通信相手が複数であるため、ここではこれを同報コネクションとよぶ。

同報コネクションでは、コネクション生成時に複数の通信相手との論理的な通信路を設定し、こ

の通信路に送出されたデータがコネクション内の相手にとどけられる。BANET[KISH82]ではこの概念に基づいた通信グループと呼ばれる同報コネクションを生成する機能を持っている。たとえば、A，B，Cからなる同報コネクションを生成するには、グループ生成のプリミティブを用いる（図3）。

しかし、BANETの同報コネクションの機構には次のような問題がある。すなわち、問い合わせによって別のサイトが必要になった場合、それらのサイトへの同報コネクションが新たに生成される。コネクションは論理的な通信路であるため、異なるコネクションでアクセスされるサイト間では同報通信ができない。このためコミットメント制御など全サイトへの同報通信は、さらに別の同報コネクションを生成しなければ実現できない。

このようにコネクション生成時に、通信相手を指定して論理的な通信路を作ってしまう従来のコネクションの概念では、必要なサイトが動的に決まる場合には対応できない。

そこで、本システムでは、コネクション設定時に論理的な通信路を固定して相手を決めてしまうのではなく、必要に応じて論理的な通信路を変えることのできる同報コネクションの機構を開発した。この機構により、問い合わせによってサーバが増える場合でも、それらを同一の同報コネクションとすることができリライアブルな同報通信を実現することができる。

本システムでの同報コネクション機能は、自分の通信端点と相手の通信端点を指定する次の通信プリミティブで実現される。

CONNECT(自分の通信端点、相手通信端点の並び)

このプリミティブによって指定された自分の通信端点と相手通信端点の間に同報コネクションを生成する。もし、すでに自分の通信端点が同報コネクションを持っていたら、このプリミティブで指定された相手が既存の同報コネクションのなかに取り入れられる。

図4に例を示す。クライアントプロセスAが、サーバプロセスBとの間に同報コネクションを生成する場合、Aの通信端点をaと小文字で表わせばCONNECT(a,b)によって実現できる。ここでさらに、クライアントプロセスAが、サーバCを必要とすれば、すでにbとコネクションの張られている通信端点aを用いてCONNECT(a,c)とすることにより、プロセスA、B、Cからなる同報コネクションが生成される。1：Nの同報コネクションではAが通信端点aへ送り出したメッセージはBの通信端点bと、Cの通信端点cへ同報できることになる。

同報コネクションの扱いは下位のプロトコルレイヤにも影響する。下位のプロトコルレイヤが、コネクション型の同報プロトコルを持ち、コネクション設定時にバーチャルサーキットによって同報相手を固定してしまうものであれば、分散データベースシステムの実現に必要な同報通信プロトコルでのコネクション機能を実現できない。したがって下位の通信レイヤはコネクションレスのプロトコルが好ましい。本システムでは、1対1のアドレスまたはブロードキャストアドレスのどちらも選択的に指定して通信できるデータグラムのプロトコルが用いられている。

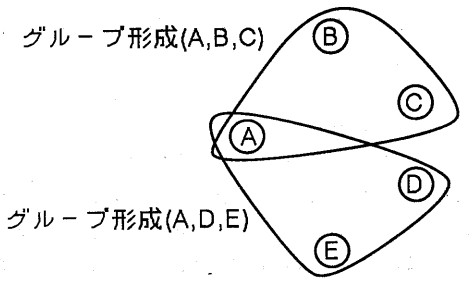


図3.BANETにおける同報コネクション

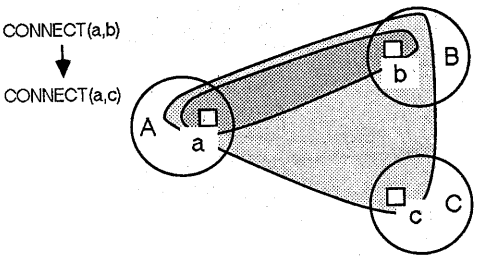


図4.本システムでの同報コネクション

４．２　応答性の向上

分散データベースプロトコルは、コマンドメッセージと応答メッセージによって実現される。これに対応した通信を行った場合、コマンドメッセージ送信側は、対応する応答メッセージを待つため、自サイトの処理、通信処理、および他サイトの処理が逐次的に実行される（図５(a)）。これにより通信回数が全体のコストに直接に反映されてしまう。先に挙げた第２の問題は、この点を解決しなければならない。

そのためには、コマンドメッセージと応答メッセージからなるプロトコルを用いた処理においても、処理をパイプライン的[LU85]に実行する必要がある。すなわち、応答メッセージが到着するまでコマンドメッセージ送信側の処理を待ち合わせるのではなく、コマンドメッセージを送信したら処理を続け、適当な時点で応答メッセージを受けとる非同期型の処理方式をとる必要がある。

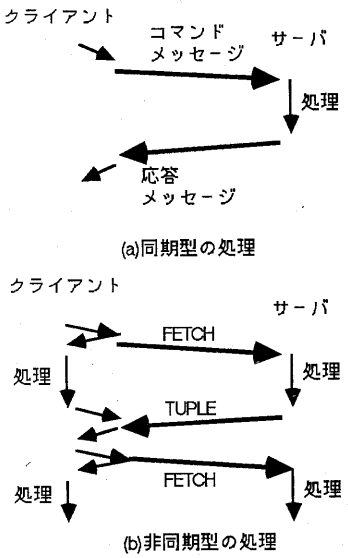


図５同期型/非同期型の処理

したがって、同報通信システムはこのための通信プリミティブを実現しなければならない。これは自サイトの処理、通信処理、および他サイトの処理をオーバラップさせるので、並列性による応答性向上の利点が生まれる。FETCHの処理に対して、この非同期型の通信プリミティブによりプリフェッチを行う例を図５(b)に示す。クライアントはサーバから結果タプルをFETCHしながらその結果を用いた処理、たとえばクライアントのサイトにあるリレーションとのジョインを行う。この場合、クライアントは常に一つ先のタプルのFETCH要求をサーバに送り、直前のFETCH要求の結果を使って処理を行う。これにより、クライアントとサーバの処理をパイプライン的に並列実行させることができ、応答性を向上させることができる。

４．３　重畳同報

先に第３の問題点として示したように、図２(b)の場合はこのままでは同報通信が使えない。しかし、インナリレーションＲ２のタプルがつきた場合、インナリレーションのカーソルのCLOSEとアウタリレーションのカーソルのFETCHは同じタイミングで発行できる。本システムではこのように異なるメッセージを異なるサイトに通信する場合にも、同報通信の利点をいかせるように重畳同報[KASH88]を導入した。すなわち、この場合、図６に示すように、CLOSEとFETCHを重畳して両方のサイトに同報する。受けとったサイトでそれぞれ必要な部分を抜き出すことにより、通信を達成できる。

この通信を行った場合、通信量はメッセージを個々に送った場合と変わらないが、通信回数を減らすことができる。分散データベースシステムでは、通信量を問題にすることが多いが、ネットワークが高速になると通信回数に依存する固定的なコストが相対的に大きくなるため、この方法によって通信のコストの削減が期待できる。

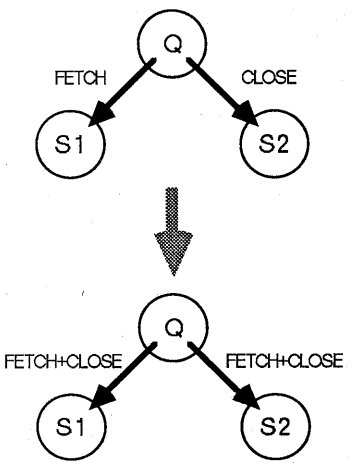


図6.重畳同報

重畳同報は次のプリミティブによって実現される。

Broadcast((サイトの並び１),メッセージ１,(サイトの並び２),メッセージ２...,(サイトの並びＮ),メッセージＮ)

このプリミティブは、サイトの並び１で指定された複数のサイトへメッセージ１、サイトの並び２で指定された複数のサイトへメッセージ２というように、１回の同報通信を用いて送る。メッセージ１からメッセージＮまでが重畳して同報される。サーバではメッセージを受け取ったら、重畳されたメッセージの中からそのサイトで必要なメッセージを取り出す。

重畳同報のプリミティブで

Broadcast((S1,S2,S3),メッセージ１)

のようにメッセージが一つの場合には、従来の同報通信であり、

Broadcast((S1),メッセージ１,(S1),メッセージ２..)

のようにすべてのサイトの並びが、同じ一つのサイトであれば、従来の重畳になる。したがって、重畳同報によって、従来の同報、従来の重畳、および重畳同報のすべてを統一的に扱うことができる。これは、本システムでの重要な機能である。

５．実現方式

５．１　モジュール構成

分散データベースシステムは以下に示す五つのモジュールから構成される。これを図７に示す。

（１）分散データベース管理モジュール

図１で示したクライアントやサーバに相当し実際にデータベースの処理を行う機能である。

（２）分散データベースプロトコルモジュール

分散データベース管理モジュール間で通信を行うのに必要なプロトコルを管理する。利用者から与えられたＳＱＬの文を分散環境で実行するために、他サイトの分散データベース管理モジュールへ処理を依頼し、結果を受けとるために用いる。コマンドメッセージと対応する応答メッセージの生成／解釈の機能を持つ。

（３）重畳同報プロトコルモジュール

重畳同報のためのプロトコルを管理する。重畳同報プロトコルヘッダの生成および解釈を行う。

クライアント側の重畳同報プロトコルモジュールはメッセージを重畳すると同時に、サーバで必要なメッセージを取り出すための重畳同報プロトコルヘッダを生成する。

（４）同報通信プロトコルモジュール

重畳同報プロトコルモジュールに対して複数サイトでのリライアブルな同報通信をサポートする。すなわち、あるプロセスから同報コネクション内に同報されたメッセージは、このプロトコルによって紛失することなく確実に相手に届けられる。

本モジュールは重畳同報プロトコルモジュールから渡されたメッセージデータを下位プロトコルで転送可能なデータサイズに分割し、データ転送およびメッセージが紛失した場合の再送を行う。

同報コネクション内でのメッセージ紛失などを管理するために、それぞれのサイトのプロトコルモジュールでは、自分の通信相手との間で行われた通信の状態遷移を管理している。また、同報コネクション内のサーバ数の動的な増減に対応する機能を持つ。

このモジュールの機能として、コマンドメッセージを送信し応答メッセージを受信するCALLREPL

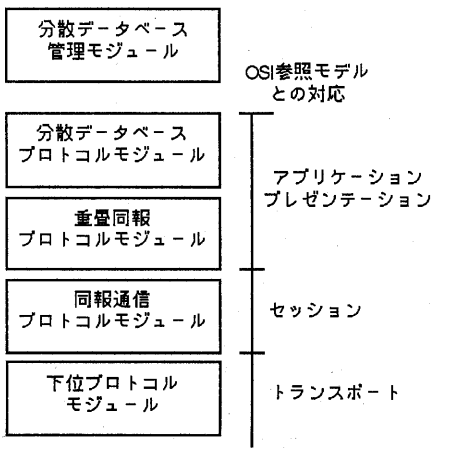


図7. 分散データベースシステムの
モジュール構成

Yプリミティブがある。このプリミティブでは、応答メッセージを待ち合わせなる必要がある場合と、パイプライン的な処理を行う場合のどちらにも対応できるように、処理の待ち合わせを制御することができる。CALLREPLYは次の形をしている。

CALLREPLY(自プロセスの通信端点、コマンドメッセージ、応答メッセージ用の領域、モード)

このプリミティブで、同報コネクションのある相手にコマンドメッセージを送り結果を受けとることができる。モードによって、応答メッセージを待ち合わせる送受信、応答メッセージを待ち合わせない送信、および応答メッセージの待ち合わせを区別する。

(5) 下位プロトコルモジュール

同報通信プロトコルの実現に必要なデータ転送を提供する。同報通信プロトコルに対して他サイトへ物理的な同報性を持つ通信の機能を提供する必要がある。

物理的な同報性は、重畳同報プロトコルを考えた場合必要である。重畳同報プロトコルは1対1通信路上で実現すると、各サイトに無駄なメッセージを通信するため通信量が増えてしまう。通信量に依存するコストが無視できない場合には逆に通信コストがかかることになる。

5.2 システム実現上の留意点

本節では、システム実現において解決した課題について述べる。

(1) 重畳同報とアドレシング

図2の例で示したように、問い合わせ処理では1対1通信も多い。

同報通信における1対1通信は不要なメッセージを破棄することによって実現される。例えば本システムでは、重畳同報プロトコルによって、各サイトが当該サイトで必要なメッセージを選択的に受信することによって実現できる。

しかし、メッセージの選択をこのように高いプロトコルレイヤで実現すると、1対1通信が多くなった場合問題となる。すなわち、不要なメッセージが高いプロトコルレイヤまで到達する確率が高くなり、これより下のプロトコルモジュールが、他サイト間で交換されるメッセージに対して多くの処理をすることになるため、システムの負荷が大きくなるという問題がある。

これに対応するためには、なるべく低いプロトコルレイヤで不要なメッセージを捨てることができる方式とする必要がある。

例えば、先に述べたCALLREPLYプリミティブでは同報コネクションのある通信端点を指定して通信を行うが、実際にはさらに、コネクションのなかから選択的に相手を指定できるようにする必要がある。したがって、CALLREPLYプリミティブを次のようすることによって、より低いプロトコルレイヤで不要なメッセージを捨てることができる。

CALLREPLY(自プロセスの通信端点、相手プロセスのリスト、コマンドメッセージ、応答メッセージ用の領域、モード)

これは、下位プロトコルレイヤへのマッピングの時も同様である。本システムでの下位プロトコルはブロードキャストアドレス以外に、1対1のアドレスも指定できるので相手が1プロセスの場合には、CALLREPLYのプロトコルを1対1のアドレシングを行う下位プロトコルにマッピングして実現する。これにより、さらに低いプロトコルレイヤで不要なメッセージを捨てることができる。

(2) リライアブルな同報通信の実現

リライアブルな同報通信を一般的に実現するには、メッセージの再送の制御に加えて、大量データの連続転送時にそれぞれのサイトと通信の同期をとるフロー制御を実現する必要があり、実現が複雑になる[AHAM85]。

しかし、分散データベースプロトコルは、ファイル転送のように不定な個数のPDUを連続転送する必要はない。たとえば、コマンドメッセージFETCHによりクライアントからサーバへ要求が出されると、サーバで処理が行われ、結果タプルが応答メッセージとしてクライアントへ送り返される。この場合FETCHの要求や結果タプルを送るメッセージの長さは、最大タプル長などシステムによって規定された長さで押えられることがあらかじめわかる。さらに、クライアントが次のコマンドメッセージを送信しなければサーバがメッセー

ジを送ることはない。

このように、フロー制御が分散データベースアクセスプロトコルによって行えるため同報通信プロトコルレベルで実現する必要はない。これによってプロトコルをより簡素化することができる。

（３）バッファコピー回数の削減

リライアブルな通信のためには、バッファが書き換えられた場合の再送に備えてメッセージを一旦プロトコルモジュール内部にコピーした後、通信を開始する必要がある。通信の処理においては、プロトコルモジュール間でメッセージがコピーされるのは、通信処理のオーバヘッドとなってしまう。

これに対して、図６(a)に示すような同期型のCALLREPLYプリミティブでは、呼び出したプロセスが応答を受けとるまで待ち合わされるため、同報通信プロトコルモジュールに渡したメッセージのバッファが書き換えられることはない。したがって、再送が必要な場合にもこのバッファをそのまま使うことが可能であり、バッファをコピーすることなく直接送信することができる。

応答メッセージが到着するまで次の処理が開始できない待ち合わせ型の処理において、バッファのコピー回数を減らせることは、待ちの時間を削減することになるため、実現上効果がある。

６．おわりに

本稿で述べた、同報通信システムは現在、OKIT AC8300ミニコンピュータ上で、同報通信機能を提供できるEthernet型のＬＡＮであるOKINET2000/20を用いて開発されている。

通常分散データベースシステムを構築する場合、ネットワークアーキテクチャは与えられたものとして受け入れられ使用されるのが一般的である。しかしながら、処理の特徴を考慮して通信システムの機能を決定し、構築することは効率よいシステムを実現する上で重要である。本システムはこのような観点で構築されている分散データベースシステムである。

本稿では、検索処理を前提として分散データベース制御方式について述べたが、今後は更新処理からの観点をも含めて同報通信を指向した分散データベースシステムの研究開発を行う予定である。

＜＜参考文献＞＞

[ROTO80]Rothnie,J.B.,Jr., Bernstein,P.A., Fox,S., Goodman,N., Hammer,M., Landers,T.A., Reeve,C., Shipman,D.W, Wong,E."Introduction to a System for Distributed Databases(SDD-1)",ACM TODS Vol.5 No. 1, 1980

[STON77]Stonebreaker, M., Neuhold,E. "A Distributed Database Version of Ingres", 2nd Berkly Workshop on Distributed Data Management and Computer Networks,Lqwrence Berkley Laboratory 1977

[WILL82]Williams,R., Daniels,D.,"R*:An Overview of the architecture", Improving Database Usability and Responsiveness,Academic Press, 1982

[METC76]Metcalf,R.M., Broggs,D.R. "Ethernet: Distributed Packet Switching for Local Computer Networks",CACM Vol.19 No.7,1976

[CHAN85]Chang,J.M., "LAMBDA:A Distributed Database Systems for Local Area Network" Database Engineering, IEEE vol.4 1985

[WONG76]Wong,E., Youssefi,K., "Decomposition - A Strategy for Query Processing", ACM TODS Vol.1 No.3, 19876

[KASH88]柏原幸一、坂本明史、川上英、疋田定幸、"分散データベースにおける同報通信に関する一考察"、情報処理学会第３６回全国大会3E-4,1988

[SQL]日本規格協会，日本工業規格データベース言語ＳＱＬ，JIS X3055-1987

[AHAM85]Ahamad M., Bernstein A.J.,"Multicast Communication in Unix 4.2BSD",Proc. 5th Distributed Computing Systems,1985

[KISH82]岸田　一、吉田　勇、山崎　晴明、"分散処理向きローカルネットワークについて"、情報処理学会分散システム研究会14-4,1982