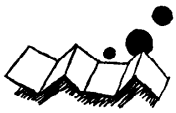
解　説

分散形データベース技術†



高　平　　叡††　鈴　木　健　司††

1. はじめに

近年の VLSI 技術を中心とするコンピュータ・ハードウェアの高性能化と低価格化は，プロセッサとメモリのコンピュータ・リソースを処理要求の発生場所で専有することを可能にしてきている．また，光ファイバ等を用いたローカル・ネットワーク技術は，大容量高速伝送による構内通信の高速化を可能にする方向にある．これらのハードウェア技術等の著しい進歩は，分散処理における処理ノードの細分化を可能にし，オフィス情報システムを基軸とした新たな分散処理が進められようとしている．現状の広域ネットワークを用いた地理的に離れた分散処理環境と，新たなローカル・ネットワークを用いた分散処理環境を結合し，総合的に処理できるような分散処理の必要性が今後増してくると考えられる．そのような分散処理環境において，共用性の高いリソースである分散形データベースの技術を確立することがますます重要になってきている．

本稿では，分散形データベースとは，複数のコンピュータ・システムが広域およびローカルな通信ネットワークを介して物理的に結合され，各コンピュータ・システムに存在するデータベースが論理的に結合され，データベースのアクセス利用者および応用プログラム（両者を含めアプリケーションと呼ぶことにする）からは，1個のデータベースとしてアクセスできるようなデータベースであると定義する．このような分散形データベースを構築し，アプリケーションからは分散を意識することなしにアクセスできる機能を提供するデータベース管理システム（DBMS）を分散形 DBMS（DDBMS）と呼ぶ．また，この DDBMS を用いて構成されたアプリケーション・システムを分散形データベース・システムという．本稿では，分散形データベースの構成方法，ネットワーク・アーキテクチャ，問合せ処理，および無矛盾性を保持するための制御の技術について，現状および将来の方向について概観する．特に，最近のローカル・ネットワーク技術の動向に注目して，データベース技術との関連を述べることとしたい．

2. 分散形データベース・スキーマ構成技術

分散形データベースの構成方法には，次の2つがある[1)~3)]．

（1）　トップ・ダウン構成

分散形データベースは最初に全体設計がなされ，ネットワーク上の各ノードに分割配置される構成法である．このとき必要に応じデータの重複が考慮される．本方式は，データベースの大規模化や権限の分散化に対応して，新規に分散形データベース・システムを構築する場合に有効であり，一般には，各ノードの DDBMS は同種である．この方式に分類されるシステムとして SDD-1[4)] が著名である．

（2）　ボトム・アップ構成

お互いに独立に存在するデータベース・システムを統合し，あたかもネットワーク上に新たなデータベースを持つシステムが存在するかのように構成する方法である．一般には，各ノードの DDBMS は異種である．本方式は今後のオフィス情報システムのように個人ベースで独立して存在するデータベースを統合していくような分野での必要性が高くなってくると考えられる．この方式に分類されるシステムとして POLYPHEME[5)] がある．

この2つの構成方法における DDBMS の技術課題には，分散形データベース・システム設計者あるいは管理者に対し，どのような統合的なデータベースの見方を与えるかという問題がある．すなわち，データ・モデルとそのスキーマ構成の問題である．

データ・モデルは現実世界の情報をデータベースからみてモデル化したものであり，階層モデル，ネット

† A Survey of Distributed Database Technology by Satoshi TAKAHIRA and Kenji SUZUKI (Yokosuka Electrical Communication Laboratory, N.T.T.).

†† 日本電信電話公社横須賀電気通信研究所

ワーク・モデル，リレーショナル・モデルなどがある．このデータ・モデルに基づくデータベースの記述情報をスキーマという．スキーマはもともと，ネットワーク・モデルを提案しているCODASYLデータベース仕様[6]で使われた用語であるが，ここではデータ・モデルの記述の一般的な用語として用いる．スキーマとしては，データ独立性の程度が高いという観点からANSI/SPARCの3階層スキーマが著名である[7]．3階層スキーマは，企業などの組織からみたデータベース全体レベルの記述を概念スキーマとして，アプリケーションからみたデータベースの部分的なレベルの記述を外部スキーマとして，コンピュータ上で実際に物理的に表現するレベルの記述を内部スキーマとして構成する．

これらのスキーマは集中形データベース・システムのもとで考えられたものであり，分散形データベース・システムにおいてはさらにどのような分散形スキーマ階層を設定するのか，そして設定した各スキーマはどのようなデータ・モデルを設定するかといった課題があり，種々の提案がなされている[1),8),9)]．基本的には分散形スキーマは，ローカルなレベルとグローバルなレベルの2つから構成される．ローカルなレベルは各ノードのデータベースの記述であり，グローバルなレベルでは，分散形データベース全体の論理的な記述とデータの配置情報の記述が必要になる．種々の提案はANSI/SPARCの3階層スキーマを拡張するものであり基本的には図-1のようになる．ただし，3階層スキーマのそれぞれに厳密に一致するものではなく，概念的に対応するものであることをことわっておく．図-1でローカル内部スキーマは集中形のスキーマ階層で構成される．ローカル概念スキーマは，ローカル内部スキーマの持つデータベースの論理的および物理的な異種性を排除する階層である．したがって，異種DDBMSによるボトム・アップ構成のシステムにおいて必要になるスキーマ階層である．グローバル概念スキーマは，分散形データベース全体を論理的に統合する階層であり，さらに，データとノードの対応などの物理的情報を管理する．グローバル外部スキーマは，基本的には，ローカルなレベルの外部スキーマと同じ位置づけである．ここで，分散形データベース・システムがただ1つのデータ・モデルによる同種DDBMSで構成される場合には，ローカル概念スキーマは不要になり，グローバル概念スキーマのデータ・モデルはローカル内部スキーマのものと通常同じものを採用することになる．なお，ローカルおよびグローバル概念スキーマの実現にあたっては，データ独立性を考慮して，分散形データベースの構造の論理的記述と，配置などの物理的情報とを分けて階層化する必要がある．

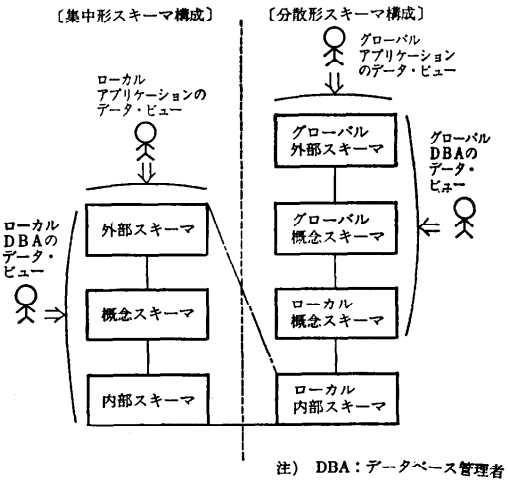


図-1 集中形スキーマ構成と分散形スキーマ構成

分散形スキーマ実現上の中心課題はローカル概念スキーマにあり，ネットワーク・アーキテクチャと関連し，いかに分散形データベース・システム全体で共通的なデータ・モデルを設定するかであり，これについては次章で述べる．今後の課題としては，オフィス情報システムなどでは，個人ベースの小規模データベースが中心となり，非専門家によるデータベースの結合が必要になる．したがって，グローバル概念スキーマを容易に設定できるような支援技術などが必要になる．

3. 分散形データベース・ネットワーク・アーキテクチャ技術

分散形データベース・システムにおいて，DDBMSは，ネットワークを介して相互にデータベースの結合処理を行う．このとき異種DDBMSを接続して分散形データベース・システムを構成するためには，標準的な分散形データベース・ネットワーク・アーキテクチャをいかに定めるかが課題になる．

ネットワーク・アーキテクチャは国内外で種々の発表がなされているが[10]，基本的には図-2に示すように，新技術の導入や機能の拡張に伴うネットワークの拡張性，柔軟性の面から，システム構造を階層化し，

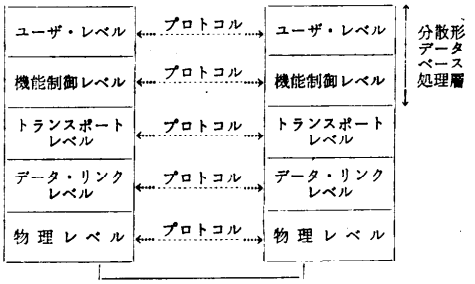


図-2 ネットワーク・アーキテクチャにおけるプロトコル階層の概念

同一階層レベル間であらかじめきめられた通信規約(プロトコル)に従って通信する構成からなる．最下位の物理レベルは装置間の物理的，電気的な接続制御を与え，データ・リンク・レベルはリンクを介した装置間のデータ転送制御を与え，トランスポート・レベルはリンクを介したエンド・ノード間のトランスペアレントな転送制御を与え，機能制御レベルはエンド・ノード上のプロセス間の通信処理を与える．そして，最上位のユーザ・レベルはアプリケーションの処理レベルである．これらの各レベルの接続方法の仮想化を通して，上位の階層レベルは下位の階層レベルから独立した構成をとることができる．異種 DDBMS 間の接続を実現するためのデータベース・アクセス・プロトコルは，機能制御レベル以上の階層において設定される．すなわち，共通に認識できるデータ・モデルを設定し，それに基づいて相互に通信可能な操作方法を設定する．このことにより，トランスポート・レベル以下のネットワークの物理的実現手段から独立した分散形データベースの論理的な実現をはかることができる．一方，各ノードの DDBMS は，設定された共通データ・モデルと自データ・モデルの相互のデータ表現の相違などに対し，変換などによる解決をはかる必要がある．この共通データ・モデルが図-1 で示したローカル概念スキーマのデータ・モデルの位置付けに相当する．

データベース・アクセス・プロトコル設定の技術的な課題としては，種々の提案されているデータ・モデルと整合性がよく，変換を最少とするようなデータ・モデルの設定，分散処理における通信回数，通信量を最少とするような高水準なアクセス機能の設定，および分散形データベースのアクセス競合や障害に対する効率的な制御機能を設定することである．データベース・アクセス・プロトコルの具体例として，DCNA の例を紹介する[11]．DCNA のデータベース・アクセス・プロトコルでは，ネットワーク内で各ノードのデータベース・システム間で共通的に認識できるデータベースを仮想データベース (VDB) として規定し，これに対する操作法を規定している．VDB のデー

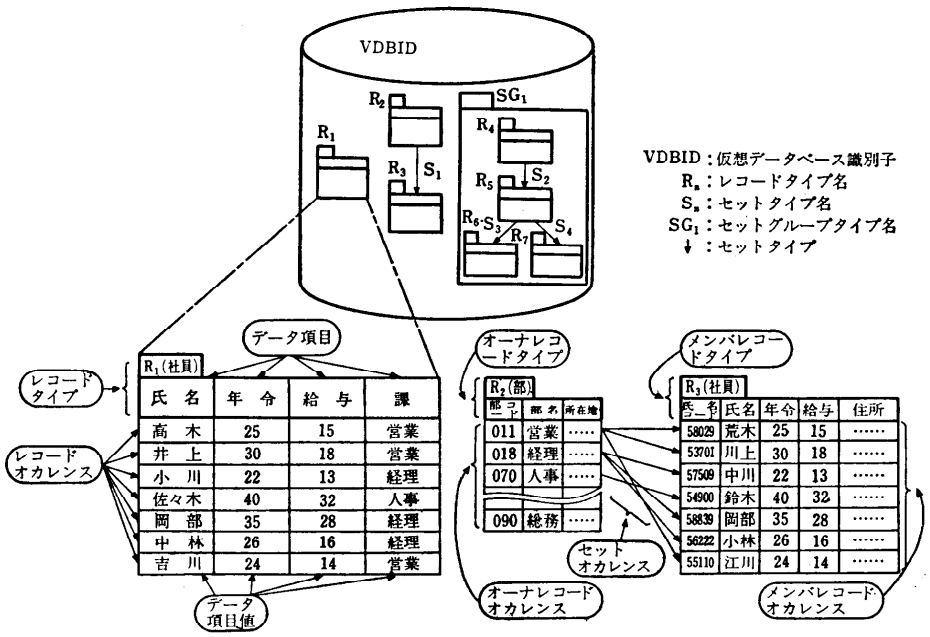


図-3 DCNA の仮想データベース (VDB) の例

タ・モデルとしては，階層モデル，ネットワーク・モデル，リレーショナル・モデルを対象とし，各モデルのデータ表現要素のすべてを包含し，統合したデータ・モデルを設定することにより，これらデータ・モデルとの高い整合性の実現をねらっている．VDB の構成要素とその例を図-3に示す．また VDB の操作機能としては，複数のレコード・オカレンスを一括して選択する高水準な機能を設定することにより，通信回数，通信量の最少化，効率化をねらっている．

ネットワーク・アーキテクチャは1対1通信の広域ネットワークから出発したものである．一方，最近のローカル・ネットワークにおけるプロトコルへの影響は，図-2 に示したデータ・リンク・レベルと物理レベルに現われ，データベース・アクセス・プロトコルには，データ・モデルの観点からは影響を与えない．しかし，ローカル・ネットワークの1対多の放送(broadcast)通信の特徴は，次章に述べるように分散形データベースのアクセス方法に大きな影響を与える．

4. 分散問合せ処理技術

分散形データベース・システムにおいて，アプリケーションはデータの分散や重複などの配置情報を知らずに，グローバル外部スキーマに基づいて問合せを行うことができる．この問合せQは，グローバル概念スキーマに基づく問合せに変換され，さらにローカル概念スキーマに基づく副問合せ $Q_1, Q_2, \cdots Q_n$ に分解される．このような分散問合せ処理が集中処理と異なるところは，副問合せ処理で必要なノード間の通信時間による遅れがあることと，複数ノードに対する並列処理がありうることである．ノード間の通信形態には，広域ネットワークとローカル・ネットワークがある．広域ネットワークとして，50 bit/s～48 Kbit/s の通信速度を持つディジタル・データ交換網を考えた場合，これを用いた分散形データベース・システムでは，ノード間の通信コストは，通信量または使用時間に依存し，複数ノードと通信する場合にはさらにノード数に依存することになる．またノード間の応答時間は，通信回数と通信速度に依存する．したがって，広域ネットワークを用いた分散形データベース・システムではネットワークがシステム・ボトルネックとなる可能性が大きいため，ネットワークの通信コストを最少にすることが課題になる．そこでこの広域ネットワークを用いる場合には，データの地域性が高く，最初に問合せが生じたノードでの完結性の高さが要求される．

それに対し，ローカル・ネットワークは同軸ケーブルなどの通信媒体を用いており，その通信速度は 0.1～数 10 Mbit/s であり，通信総距離は 1～15 km を対象範囲としている．広域ネットワークとの相違は短距離，高速伝送であり，伝送遅延が無視できるほど小さく，ローカル・ネットワークに接続される各ノードは通信路を共有することができることである．したがって，通信方式は，1つのノードから出されたメッセージが同時に複数のノードで受信できる放送通信を基本にすることができる．このローカル・ネットワークでは通信コストはノード間の距離やノード数から独立で，通信量のみに依存する[12]．

このような状況から，これまでの広域ネットワークでの分散問合せ処理方式は，ノード内でのローカルな処理コストより，ノード間のトータル通信コストが十分に大きいという仮定のもとに，通信コストを最少とすることで検討が進められてきた．たとえば，リレーショナル・モデルを扱う SDD-1 では，問合せの最適化として半結合 (semijoin) を用いている[13]．半結合は，異なるリレーションに対し，共通の属性により結合 (join) する場合の最適化方式である．たとえば，それぞれ別のノードにあるリレーション R_i, R_j を結合する場合，まず R_i について必要な属性だけをとり出す射影 $R_i(A)$ $(=R_i')$ を施し，この結果を R_j のノードに転送し，R_i' と R_j の結合を行う方式であり，これにより結合のみ行う場合に比べ通信コストを低減できるとするものである．

しかし，ローカル・ネットワークの場合には，ローカルな処理コストが応答時間に影響を与えてくる．したがって，ローカル・ネットワークでの課題は放送通信により複数ノードへの並列処理を最大とし，ローカル処理における応答時間を最少となるような分散問合せ処理方式を考慮することであり種々の試みがなされている[14),15]．

5. 分散アクセス制御技術

データベースのアクセス制御の主要課題は，同一データに対する複数のアプリケーションからの同時アクセスに対し，データの矛盾を生じないよう一貫性を保つことである．このために同時制御 (Concurrency control) 技術が必要になり，集中形データベースでは次に述べる2フェーズ・ロック方式により実現されている．それに対し，分散形データベースでは複数の

ノード間におよぶ制御を必要とし，新たな課題を生じ多くの提案がなされている[16]．種々の提案は，基本的な技術である2フェーズ・ロック方式と時刻印（time stamp）順方式をもとに発展させたものである．

2フェーズロック（2PL）方式[17]

アプリケーションの問合せをトランザクションと呼ぶ．2PL方式は，同一データに対するトランザクションからの同時アクセス要求の間の競合を検出し，同期をとる方式である．トランザクションはデータにアクセスする前にロックを行う必要がある．ロックの基本的な規則は次の通りである．（i）異なるトランザクションは競合を起すようなロックを同時に行うことはできない．（ii）トランザクションは一度ロックを解放すると次のロックは行えない．この規則をもとに，トランザクションは次の2フェーズでロックを行う．（i）ロックを行うことができる成長フェーズ，（ii）ロックを解放し，規則（ii）により次のロックを行うことが禁止される縮小フェーズ．

2PLの分散形データベースにおける実現方法には次のものがある．**基本2PL**はデータの存在するノードに2PLの規則に従ってロック処理するスケジューラを置く方法である．**主コピー2PL**は重複データに着目した方式であり[18]，コピーのうち1つが主コピーに指定され，任意のコピーのデータにアクセスする前に，この主コピーに対しロックをかける．**多数決2PL**も重複データに対する方式であり，THOMAS[19]の多数決方式（植村氏解説[20]参照のこと）から考え出された方式であり，トランザクションの更新要求はコピーを持つすべてのノードに送出され，各ノードからの返事のうちロックの応答数が過半数に達していたならば，すべてのロックが行われたかのように処理する．**集中2PL**はスケジューラを1ノードに集中させる[21]．

分散形データベースにおいて，あるデータ項目Xはコピー$x_1,\cdots x_m$をもち，それぞれ別のノードに存在すると仮定したとき，基本2PLでは，参照時（read）は実際に参照するコピーx_iのみのロックでよいが，Xを更新（write）するときは，すべてのコピーをロックし，更新する必要がある．主コピー2PLでは，参照時は主コピーx_1と参照するコピーx_iをロックし，更新時は主コピーx_1に対しロックを要求し，更新はすべてのコピーに対して行う必要があるなど，それぞれの方式は，ロックと更新のための通信に特徴がある．

これらの2PL方式では，トランザクションはロックできない場合待ちになるが，この待ちが制御されないとデッドロックが起る．このデッドロックの解決方法には防止と検出がある．デッドロック防止は，ロック要求が拒否されたとき，スケジューラは要求しているトランザクションT_iと，現状ロックしているトランザクションT_jについて，T_iがT_jを待ちあわせてもデッドロックが起りえないように制御することである．待ちあわせることができない場合には，2つのトランザクションの1つの処理は止めさせられ再実行される．この制御の簡単な方法は，トランザクションに優先度を与え，優先度が低ければ待ち合わせるようにすることである．しかし，この方法は優先度が低いと終ることなく再実行され続けるという問題がある．この問題を避けるには優先度に時刻印を使用する方法がある[22]．トランザクションの時刻印はその実行開始時の時刻であり，古いトランザクションは新しいトランザクションより高い優先度を持っている．時刻印はたとえば，上位ビットがノード内のローカル時計の時刻，下位ビットがシステム一意のノード識別子からなるシステムで一意な値である．

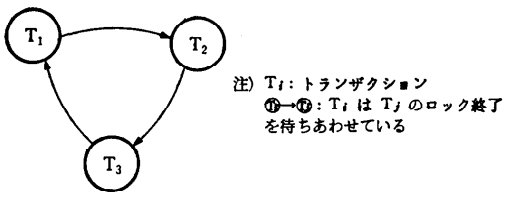


図-4　ウェイト・グラフとデッドロックの関係

デッドロック検出はトランザクションが制御されず待ち合わせたとき，デッドロックをウェイト・グラフを使用して見つけ出すことである．ウェイト・グラフはトランザクション同志の待ち合わせを示す有向グラフであり，図-4に示すようにグラフの節点はトランザクションを，枝は待ちの関係を表わす．デッドロック検出は，ウェイト・グラフがサイクルを形成したときなされ，サイクルが見つかると，1つのトランザクションが選ばれる処理を止めさせられ，デッドロックを解除する．分散形データベースにおいてデッドロック検出を実現するには，ウェイト・グラフを効率よく構成する必要がある．各ノードの2PLスケジューラは，そのローカルなウェイト・グラフを容易に構成できるが，問題はグローバルなウェイト・グラフをどのように構成するかである．このグローバルなウェイト・グラフの構成方法の1つとして集中化方式がある．この方式は，1つのノードを分散形データベース・シ

ステムにおけるデッドロックの検出者として指定する．この検出者は周期的に各ノードから送られてくるローカルなウェイト・グラフの和を構成することによりシステム全体のウェイト・グラフに結合する[18]．他の方法に2フェーズ・デッドロック検出方式がある[23]．この方式は，最初のフェーズで各ノードで管理する部分ウェイト・グラフを用いて，各ノードに閉じたデッドロック（ローカル・デッドロック）の検出を行う．そして，複数ノードにわたるデッドロック（グローバル・デッドロック）の検出の要否の判定を行い，その必要がある場合には，次のフェーズで部分ウェイト・グラフの集合を用いてグローバル・デッドロックの検出を行う．検出の要否の判定は，その部分ウェイト・グラフの節点からの枝が，あるいは節点への枝が，他のノードの部分ウェイト・グラフに存在するか否かによる．

時刻印順方式[24]

時刻印順方式は直列化の順序があらかじめ決められ，トランザクションはこの順序に従って実行される方式である．各々のトランザクションは，前述したように一意な時刻印を付与され，この時刻印順に従って実行されサイクルは形成されない．時刻印順方式の基本的な実現方法は，スケジューラを各ノードに配置し，各データ項目に対しスケジューラはデータ処理結果の中で最も大きな時刻印を記録しておく．そして，処理要求に対し記録されている時刻印を比較し，記録されたものより小さければその時刻印を持つトランザクションの処理は止めさせられ，大きければそのトランザクションの処理を行い記録を書き直す．処理を止めさせられたトランザクションを再開始するときは，新しくより大きな時刻印が与えられる．

以上述べた同時制御においては，同時アクセスに対するデータの一貫性を保つために，トランザクションは処理の途中で中止され，再開始できる必要がある．この場合，トランザクションによる部分的な更新がデータベースになされたまま中止されると，データの一貫性は保たれない．したがって，この一貫性を保つためには，トランザクションにおける更新要求がすべて処理されたか，あるいは1つも処理されないかのいずれかであることを制御する必要がある．この制御の標準的な方式として2フェーズ・コミットメントがある[25]．DCNA でもこの方式に基づくプロトコルが提案されている（図-5）．トランザクションが処理の終了コマンドを発行すると，DDBMS は2フェーズ・

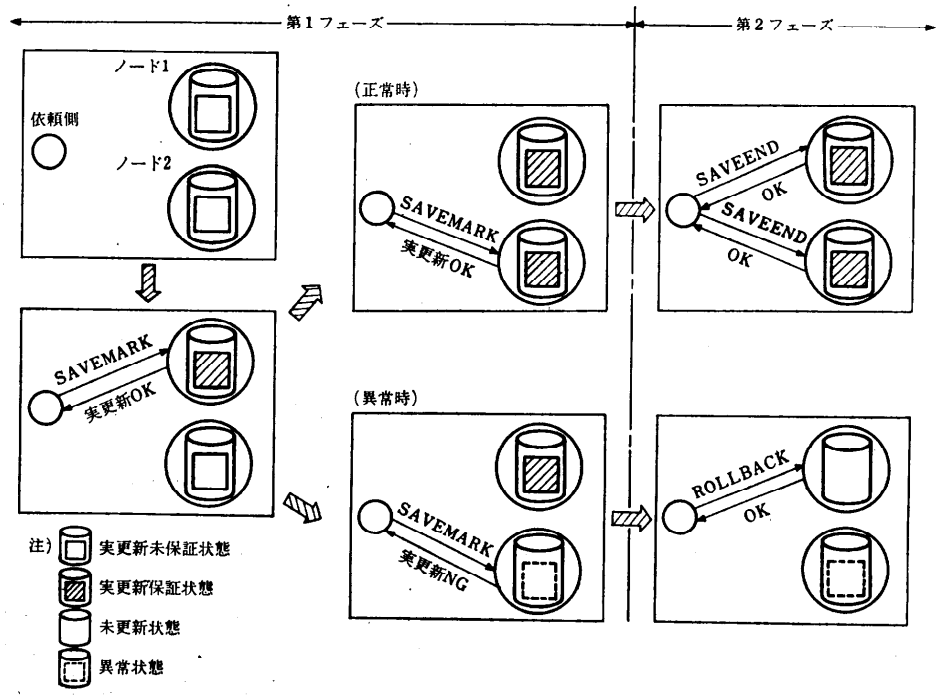


図-5 DCNA の2フェーズ・コミットメント・プロトコル

コミットメントの制御を行う．まず第1フェーズではデータベースの更新の必要なノードに対し，データベース更新の保証の指示（SAVEMARK コマンド）を出す．これにより更新を依頼されたノードはデータベースを更新前の状態に戻せる処理および実更新を正しく行える処理を行う．第2フェーズではすべてのノードから成功のむねの応答があれば実更新を行う指示（SAVEEND コマンド）を出す．もし1つのノードからでも不成功のむねの応答があれば，更新前の状態に戻す指示（ROLLBACK コマンド）を出す．第2フェーズで障害がある場合には，データベースはシステム全体からみたとき一貫性のない箇所が存在することになるが，これは各ノードの復旧時に正しく実更新され，一貫性は回復される．

同時制御は2つの基礎的な技術の発展形として種々提案されている．現状でも，適用環境とデータの特性により適当な方式を採用し実現することは可能であるが，今後は，どの方式が最もよいか評価を通して決めていくことが課題である[16]．

6. おわりに

分散形データベース技術として，スキーマ構成，ネットワーク・アーキテクチャ，問合せ処理，アクセス制御について課題とその実現方式の解説を試みた．本稿ではふれなかったがその他に，最適な分散形データベース・システム構築のための設計技術，異種データ・モデル間の変換技術，分散形スキーマの変更に対する同期管理技術，分散形データベース・システムのセキュリティ技術など重要な技術が多々ある．加えて，先にもふれたように，個人ベースの小規模データベース向きの分散形データベース構築支援技術等々，今後の研究開発に期待する問題も多い．

これまでの分散形データベースの研究開発はおもに理論的興味のもたれる問題について多くなされ，具体的なアプリケーションへの適用がはかられていない向きがあった．たとえば，同時制御技術についても多数の方式が提案されているが，スループットや応答時間の性能の観点からの定量的分析，評価を通し，アプリケーションへの適用性を明確にしていくことが必要である．すでに，大規模データベースを構築する手段として分散形データベースの適用が有効であることは具体的に示されており，今後はアプリケーションを直視した技術の選択によって実用に耐えうるものを早急に実現していくことが最も肝要であろう．

参考文献

1) Adiba, M. et al.: Issues in Distributed Data Base Management Systems: A Technical Overview, Proc. 4 th Int. Conf. on Very Large Data Bases, pp. 89-110 (1978).
2) Peebles, R. and Manning, E.: System Architecture for Distributed Data Management, Computer, No. 3, pp. 40-47 (1978).
3) 増永，野口：分散データベース統合問題について，信学会オートマトン研，AL 81-23, pp. 51-58 (1981).
4) Rothnie, J. et al.: Introduction to a System for Distributed Databases (SDD-1), ACM Trans. Database Syst., Vol. 5, No. 1, pp. 1-17 (1980).
5) Adiba, M., Caleca, J. Y. and Euzet, C.: A Distributed Data Base System using Logical Relational Machines, Proc. 4 th Int. Conf. on Very Large Data Bases, pp. 450-461 (1978).
6) CODASYL: Data Base Task Group Report to the CODASYL Programming Language Committee (1971).
7) ANSI/X 3/SPARC: Interim Report, Study Group on Data Base Management Systems, FDT Bulletin of the ACM-SIGMOD, Vol. 7, No. 2 (1975).
8) Takizawa, M. and Hamanaka, E.: The Four-Schema Concept as the Gross Architecture of Distributed Databases and Heterogeneity Problems, J. Inf. Proc., Vol. 2, No. 3, pp. 134-142 (1979).
9) Suzuki, K., Tanaka, T. and Hattori, F.: Implementation of a Distributed Database Management System for Very Large Real-time Applications, COMPCON Fall (1982), to appear.
10) 信国弘毅：データ通信網アーキテクチャの現状，情報処理，Vol. 20, No. 1, pp. 50-56 (1979).
11) 河津他：DCNA のデータベースアクセスプロトコル，通研実報，Vol. 30, No. 3, pp. 799-823 (1981).
12) 滝沢 誠：分散型データベースシステム JDDBS-Ⅱと通信処理，信学会オートマトン研，AL 81-22, pp. 43-50 (1981).
13) Bernstein, P. A. et al.: Query Processing in a System for Distributed Databases (SDD-1), ACM Trans. Database Syst., Vol. 6, No. 4, pp. 602-625 (1981).
14) Gouda, M. G. and Dayal, U.: Optimal Semijoin Schedules for Query Processing in Local Distributed Database Systems, Proc. of ACM-SIGMOD, pp. 164-175 (1981).
15) Toan, N. G.: Distributed Query Management

for a Local Network Database System, Proc. 2nd Int. Conf. on Distributed Computing Systems, pp. 188-196 (1981).
16) Bernstein, P. A. and Goodman, H.: Concurrency Control in Distributed Database Systems, Compt. Surv. Vol. 13, No. 2, pp. 185-221 (1981).
17) 松下, 吉田, 脇野: 分散データベースにおけるデッドロック回避方式による Concurrency Control の比較評価, 情処分散システム研, 11-9, pp. 67-74 (1981)
18) Stonebraker, M.: Concurrency Control and Consistency of Multiple Copies of Data in Distributed INGRES, IEEE Trans. Softw. Eng. Vol. SE-5, No. 3, pp. 188-194 (1979).
19) Thomas, R. H.: A Solution to the Concurrency Control Problem for Multiple Copy Databases, COMPCON Spring, pp. 56-62 (1978).
20) 植村俊亮: 分散型データベースシステム, 情報処理, Vol. 20, No. 4, pp. 295-300 (1979).
21) Garcia-Molina, H.: A Concurrency Control Mechanism for Distributed Databases Which Uses Centralized Locking Controllers, Proc. 4th Burkely Conf. on Distributed Data Management and Computer Networks, pp. 113-124 (1979).
22) Rosenkrantz, D. J., Stearns, R. E. and Lewis, P. M.: System Level Concurrency Control for Distributed Database Systems, ACM, Trans. Database Syst., Vol. 3, No. 2, pp. 178-198 (1978).
23) Kawazu, S. et al.: Two-phase Deadlock Detection Algorithm in Distributed Databases, Proc. 5th Int. Conf. on Very Large Data Bases, pp. 360-367 (1979).
24) Kaneko, A., Nishihara, Y., Tsuruoka, K. and Hattori, M.: Logical Clock Synchronization Method for Duplicated Database Control, Proc. 1st Int. Conf. Distributed Computing Systems, pp. 601-611 (1979).
25) Bernstein, P. A. and Goodman, N.: Timetamp-Based Algorithms for Concurrency Control in Distributed Database Systems, Proc. 6th Int. Conf. on Very Large Data Bases, pp. 285-300 (1980).

(昭和57年7月1日受付)