並行制御機能のないデータベースシステムによる分散データベース†

上林弥彦††*　近藤誠一††**

最近のパーソナルコンピュータの機能の向上にともない，それらに対するデータベースシステムが多く市販され始めている．本論文では，このような独立システムをローカルネットワーク上の分散データベースシステムとして運用する場合，比較的簡単に実現できかつ実用的と考えられる並行制御を考慮した質問処理方式について示した．個々のシステムは，単体として利用する場合，利用者の要求を順に処理することにより，並行制御の必要がないが，このようなシステムを統合すると，更新命令の反映の仕方により，意味的な矛盾が生じ得る．統合の問題を扱う場合，通信手続き，データモデル，質問の変換等を考慮することが必要であるが，これらは，従来，種々考えられてきているので，本論文では，このような大域的並行制御という従来考慮されていなかった問題について考察し，その処理方式を示した．ここでは，(1)個々のシステムでは並行処理を行わない．(2)放送機能を持つ通信線を利用する．(3)大域的な質問は参照に限るという場合について考察したが，拡張も可能である．この方式は，各システムに固定順を付けパイプライン式に処理することにより，質問処理で生じ得る矛盾を除いている．この方式は，(1)各システムへの入力は質問言語のみであるのでそれらの変更の必要がない，(2)制御が単純であるので追加機能が少ない，(3)集中制御が不要である，(4)木プロトコルを基本にしているため，コストの高いロールバックが不要であるといった特長を持つ．

1. まえがき

最近のパーソナルコンピュータの機能の向上につれ，それらに対するデータベースシステムが多く市販され始めている．本論文では，このような独立システムを統合して，オフィス環境で利用できるようなローカルネットワーク上の分散データベースシステムを構成する場合の，並行制御を考慮した質問処理アルゴリズムについて考察する．単一のシステムでは考える必要のない並行制御が，分散システムでは不可欠となるが，もとのシステムでは並行制御の機能が実現されていないので，質問処理機能を用いて等価的に実現するための方法を示した．したがって，ファイルのロック等の機能を仮定しない並行制御方式となっている．

分散データベース構成のための実際的な方法は，トップダウン的方法ではなく，ボトムアップ的方法と考えられる．このようなボトムアップ的方法を採用した場合，性質の異なるデータベースシステムの統合が必要となる．そのために以下のような問題が生じる．

(1)　システム間の通信手続きの統合．

(2)　利用者の参照命令や更新命令の表現法の統合．

(3)　大域的な並行制御方式の実現．

(1)は，標準的な通信の問題である．(2)は，システムによって異なるデータモデル，データベース言語が用いられているので，その相互の変換を行うためのものである．たとえば，全システムに共通のデータモデルとして関係モデルを採用し，関係データベース言語を共通言語とすれば，(2)の問題を扱うことができる．従来，(2)の問題については，種々の論文が発表されてきている．しかしながら，(3)の問題については，著者らの知る限り研究がなされていない．このため，本稿では，このようなボトムアップ的方法によって，異種型分散データベースシステムを構成する場合の大域的な並行制後方式について基本的な考察を行っている．3章で示すように，大域的な処理（複数の局を使用する処理）にはデータの更新を許さなくても，局内でデータの更新が許されていれば，分散データベースシステム全体に対する大域的な並行制御がなければ正しい結果が得られない．そのため，このような大域的並行制御の実現は非常に重要といえる．

ここでは，次のような仮定でこの問題を扱う．

(a)　個々のシステム間の通信手続きは同一である．

(b)　各システムは関係モデルに基づくデータベースであり，システム相互間の通信には共通の関係データベース言語が用いられる．

(c)　各システムは並行処理機能を持っていないため，質問は入った順に蓄えられ，直列的に処理さ

---

† A Distributed Database System Consisting of Subsystems without Concurrency Control Capability by YAHIKO KAMBAYASHI and SEI-ICHI KONDOH (Department of Information Science, Faculty of Engineering, Kyoto University).

†† 京都大学工学部情報工学科
* 現在　九州大学工学部情報工学科
** 現在　三菱電機(株)

れる．

（d） バス型の通信線でシステム同士が結ばれているため，すべての局に対して放送することができる．

一般的には複数のシステムを対象とする更新操作を考えなければならないが，実用的である次のような制限を追加する．

（e） 更新操作は一つの局の中のデータを対象とする局所的なものに限り，複数の局を対象とする操作は参照操作に限るものとする．

このような問題を扱うため，本稿では次のような方法を採用した．

（i） 複数の局にまたがる参照のみの質問は，データ伝送量を減少させることのできる準結合を用いた処理[2]を基本とする．

（ii） 大域的な並行制御に対しては，木プロトコル[11]に類似した方式を用いる．木プロトコルでは，データ項目をロックの単位としているが，ここでは局を単位とする．

（iii） 大域的な質問処理や並行処理の実現に際し，既に存在するシステムには変更を行わず，システムの処理できる質問言語でこれらの制御情報も表現する．すなわち，質問の変形という形で問題を取り扱う．

（iv） バス線は，データの放送による通信回数の削減，システム全体の同期，バス線に流れる情報の順序による正当性の制御といった目的に使われる．

本稿の結果は，個々のシステムが処理効率を上げるための並行制御機構を持っている場合や，ネットワークがバス線による方式でない場合にも拡張することが可能である．

2章では，本稿で提案する方式の基礎となる質問処理と並行制御に関する基本的定義をまとめる．3章では，大域的な正当性を扱うための問題点，およびそれに対応する基本的方法と，本稿で提案する方式の方針について述べる．4章では，ネットワークの放送機能を利用して，各局を訪れる順序を任意に選択することができる質問処理方式について示す．この方式は，準結合を利用して，処理効率の向上を図っている．5章では，局に付けられた固定順を利用した大域的並行制御方式を併合した質問処理方式について述べる．この方式はロックを基本とした並行制御方式として知られている木プロトコルに類似したもので，パイプライン式に処理を進めていくことにより，各局の処理順序が矛盾しないように制御を行う．あらかじめ決められた固定順に従って，4章で示した質問処理方式を適用する．

2. 基本的事項

本章では本稿で提案する方式の基礎となっている質問処理と並行制御に関する基本的定義をまとめる．

2.1 質問処理

関係 $R(X)$ は属性集合 X 上の関係で，組 u の集合である．Xが不要の場合は，Rで表すものとする．また，$\underline{R}=X$ としてこれを関係Rの**関係スキーマ**という．関係データベースにおける質問処理で最もコストがかかるのは結合操作であり，特に分散データベースシステムでは異なる局間に蓄えられた関係の結合をいかに経済的に実現するかが重要である．結合操作の中では自然結合が特に重要なので，以下では自然結合に絞ってこの処理方式を示す．不等号結合への拡張は容易である．

基本的な関係演算を以下に示す．

射影：$R[Y]=\{u[Y]\mid u\in R\}$

自然結合：$R_i\bowtie R_j$

$$=\{u\mid u\in R,\ u[\underline{R}_i]\in R_i,\ u[\underline{R}_j]\in R_j,\ \underline{R}=\underline{R}_i\cup\underline{R}_j\}$$

自然結合質問はその構造をグラフ表現を用いて，分かりやすく表すことができる．自然結合質問qに対する**質問グラフ** $G_q=(V,E,L)$ は，ラベル付無向グラフである．Vは節点の集合で，Vに属する v_i は，qに現れる関係 R_i に対応する．R_i と R_j に対応する二つの節点 v_i と v_j は，$R_i\bowtie R_j$ が存在するとき，およびそのときに限り，枝によって連結される．枝のラベルは $\underline{R}_i\cap\underline{R}_j$ となる．枝のラベルとして $\underline{R}_i\cap\underline{R}_j$ の部分集合を許してもよく，その場合，そのような質問グラフで表現される質問のクラスを半自然結合質問と呼ぶ．このクラスの質問は自然結合質問の等価変換を行う場合に中間的に心要となるが，属性名の変更で必ず自然結合質問に変換できるのでここでは扱わない[1]．Eは枝の集合であり，LはEのためのラベルの集合である．与えられた質問に等価である質問に対応し，かつ，閉路のない質問グラフが存在するとき，その質問は，**木型質問**であるという．そうでないとき，**巡回型質問**であるという[4),6),13)]．

分散データベースシステムにおいて，異なる局に蓄えられた関係を結合する場合，いずれかの局に別の局

の関係全体を送信すると，通信コストが非常に大となる．そこで，通信コストを軽減するために，関係の結合属性値集合のみを送信する**準結合**と呼ばれる方式が使用される．関係 R_i と R_j の準結合は $R_i \ltimes R_j$ で示し，以下のように定義する．

$$R_i \ltimes R_j = (R_i \bowtie R_j)[\underline{R}_i]$$
$$= R_i \bowtie R_j[\underline{R}_i \cap \underline{R}_j]$$

木型質問に対しては，準結合のみを用いた，すべての関係の結合の部分結果を効率良く計算するための手続きがある．ここでいう R_i の部分結果とは，結合の結果を $\underline{R}_i$ に射影したものである．巡回型質問を木型質問に変換して扱う方法[7),8)]が存在するので，木型質問についてのみ考察してよい．しかし，本稿の方法はどちらのクラスの質問にも適用可能である．

各局には，複数の関係が存在し得るが，簡単のため，局 S_i には一つの関係 R_i があるものと仮定する．この関係は，質問に含まれる局 S_i のすべての関係を前処理（局所的な射影，選択，結合）することによって得られる．通常，この仮定が使用され，結果は容易に一般の場合に拡張することができる．

2.2 並行制御

複数の処理を少しずつ分けて実行することによって処理効率を上げかつ応答時間を均一化できる．このような処理を**並行処理**という．この場合，意味的に正しい結果を得ることができるようにするために，**並行制御**が行われる．意味的にまとまった利用者の要求を**トランザクション**と呼ぶ．各トランザクションを順に並べ，並行処理を行わないスケジュールを**直列**であるという．あるスケジュールが，等価な直列なスケジュールを持つとき，およびそのときに限り，**直列可能** (serializable) であるという．直列可能なスケジュールを意味的に正しいものと仮定する[9),10),12)]．

直列可能性を保証する手法としてよく知られているものに二相ロック方式[5)]がある．この方式では，デッドロックが生じ得るが，次に述べる木プロトコル[11)]では，直列可能性とデッドロックの排除を保証する．

木プロトコル：項目間の関係は，木によって表現されていると仮定する．考慮すべき基本命令は，ロック命令 (LOCK) とロック解除命令 (UNLOCK) である．各トランザクションの参照命令あるいは，更新命令は，項目 x にロックがかかっているときに限り，実行することができ，各項目には一度に一つのトランザクションのみが，ロックをかけることが許される．次のような条件を満足するとき，トランザクションは木プロトコルに従うという．

（1） 最初は任意の項目にロックをかけることができる．

（2） 最初にロックをかけた項目以外がロックをかけることができるのは，現在ロックをかけている項目に対応する節点の子節点に対してのみである．

（3） ロックの解除はいつでも可能である．

（4） 各トランザクションは，同じ項目に二度以上ロックをかけることはできない．

分散データベースにおいて，直列可能性を保証するためには，すべての局において実行されたスケジュールに矛盾しないような等価な直列スケジュールが存在しなければならない．このような場合に**大域的に正当**であると仮定する．

3. データベースシステム統合時の問題点と基本的方法

本章では，独立に生成されたデータベースシステムを Ether タイプのバスラインで統合した場合について考察する．第1章で示した仮定をまとめると以下のようになる．

（1） システム間の通信手続きの統合の問題は考えず，同一の通信手続きにより，物理的には，既に統合されている．

（2） データモデル，質問言語の統合の問題を避けるため，各システムは既に変換が行われ，関係モデルに基づくデータベースであり，システム間の通信には，共通の関係データベース言語が用いられる．

（3） 各システムは，並行制御機構を持たず，質問は順に直列に処理される．したがって，大域的な正当性のための同期の手段を持たないので，仮定(2)の関係データベース言語のみで制御を行う．

（4） バス型の通信線で，システム同士が結ばれ，すべての局に対して放送することができる．

（5） 独立に生成されたシステムの統合であるので，大域的質問の頻度はそれほど高くない．

（6） 大域的質問は参照のみであり，更新は局所的質問によってのみ可能とする．

（7） 個々のシステムは独立に生成されたものであるのでデータの重複はない．

まず，局所的質問には更新を許すが，複数の局にまたがる質問には参照のみしか許さないモデルにおいて

も大域的並行制御が必要であることを示す．

大域的な参照のみの質問は，仮定(2)により，各局における部分質問に変換される．このとき，ある一つの局に対して複数個の部分質問を許すとする．個々のシステムは，要求された質問を順に処理するのみであり，並行制御機構を持っていないので，これらの部分質問の間に，局所的な更新を含むような質問が行われることを妨げることができず，局所的な正当性を保証することができない．逆に言うと，複数個の質問を許すということは，まさにその局において並行処理を行っていることになり仮定に反する．したがって，各局に対して部分質問は一つのみ，すなわち，各局へのアクセスは最高一回までに限れば，少なくとも，局所的な正当性を保つことができる．しかし，そのような条件を満たす場合でも下の［例1］で示すように大域的正当性を保つことができない場合が生じる．

［例1］次のような四つの質問 Q_1, Q_2, Q_3, Q_4 について考える．Q_1 と Q_2 は，大域的な参照操作のみであり，Q_3 と Q_4 は，局所的な更新操作を含む質問である．

Q_1：局 S_1 で項目Aを参照した後，局 S_2 で項目Dを参照する．

Q_2：局 S_2 で項目Cを参照した後，局 S_1 で項目Bを参照する．

Q_3：局 S_1 で項目AとBを更新する．

Q_4：局 S_2 で項目CとDを更新する．

図1に示すように命令を実行すると，局 S_1 と S_2 では各々以下に示すような順序付けがなされる．

局 S_1：$Q_1 \rightarrow Q_3 \rightarrow Q_2$

局 S_2：$Q_2 \rightarrow Q_4 \rightarrow Q_1$

各局において，等価な直列な順は，上記の場合しか存在しない．しかし，大域的に見て，等価な直列な順はなく，正当性の条件である直列可能性に反する．したがって，Q_1 あるいは Q_2 を再実行することが要求される．すなわち，各局における局所的な更新操作が許されると複数の局にまたがる処理としては参照のみであっても大域的な並行制御を必要とする．

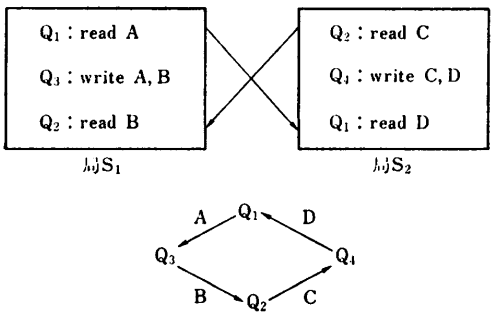

図 1 大域的並行制御機構の必要性

Fig. 1 Necessity of a global concurrency control mechanism even if there are no global write transactions.

この例では，Q_1, Q_2 とも参照のみであり，しかも，共通の項目がないにもかかわらず，Q_3, Q_4 という局所的質問による更新操作により，順序関係が生じる．仮定(2)で示したように，システム間の通信は，共通の関係データベース言語によってのみ，すなわち，質問変換によってのみであるので，このような更新命令を検知し，対応することは困難である．そこで，大域的質問は，各局において処理順に順序付けされるものと仮定することにより，問題を大域的質問に注目すればよいように簡単化する．すなわち，局をアクセスの単位とし，その順により局ごとに全順序が付けられ，その順序が全体の正当性の判定の指針となるようにする．これは，大域的正当性を保証する問題に関して十分条件である．この条件を用いて，**強大域的正当性**を定義する．また以下のような有向グラフ $G_{sg}(V, E)$ を定義する．

スケジュールSに対応する有向グラフ G_{sg} において，V は節点の集合で，大域的質問に対応する．Sの中から，大域的質問の命令を抜き出した部分スケジュール S' で，Q_i の命令→Q_j の命令の順になっているとき，およびそのときに限り有向枝 e_{ij} を付ける．このグラフがループ（自己ループも含む）を持たないとき，強大域的正当であるとする．明らかに，強大域的正当ならば，大域的正当である．

システム全体で常に大域的な質問がたかだか一つであればこのような問題を避けることができる．この考え方は次のように一般化できる．

基本的方法：質問Qに対して参照する局の集合を $S(Q)$ とする．現在システムが実行中の大域的質問の参照する局の集合の和を T とする．

- $T \cap S(Q) = \phi$ ならばQを実行する．そうでないならば，Qを待ち行列に入れる．

この方法は単純であるが，次のような問題がある．

（1） 大域的質問が多い場合，効率が悪い．

（2） 多くの局で処理を行う質問はいつまでも実行できないといういわゆるライブロックが生じやすい．

（3） 大域的制御のため中心となる局が必要となる．

本論文では，単純でありかつ上記の問題を考慮した方法を示す．

上記の基本的方法は，局をロックの単位とし，大域的質問にのみ注目すると，普通の並行制御のための方式である二相ロック方式[5]の変形であるとみることができる．すなわち，強大域的正当性を保証する問題は，以下のような見方をすると，ロック機能を用いた普通の並行制御の問題に置き換えることができる．

（a） 各局へのアクセスは最高1回とする．

（b） 局単位にロックをかける．このロックは，大域的質問のみを排除するものであり，局所的質問は関係しない．

通常，このようなロック機能を実現するため，ロックテーブル等を用意して制御を行うが，本論文の方式では，各局の直列処理と，バス型ネットワークの直列的な送信により，ロック機能を模倣する．以下の方針に従った方式を示す．

（1） ロックのための機能を必要としない．

（2） 決められた順に局を訪れる木プロトコルに類似した方式を用いる．そのため，訪れる局の順位は任意に決定できる質問処理方式を示す．

4. 放送機能を用いた質問処理

前章で示したように，局所的な更新命令を許すと，二段階に分けて処理を行う準結合を利用した基本的な質問処理手続きは，そのままでは，局所的正当性さえ満足することができない．そこで，この章では，大域的正当性を保証するための手法を示すための準備として，次のような性質を持つ質問処理手続きを示す．

（1） 各局には，最高一度のみ訪れる．

（2） 質問グラフにおける各節点は，任意の順序で処理できる．

前章で示した問題のため，(1)の性質が必要となる．(2)の性質は，質問処理手続きを，次章で述べる大域的並行制御方式に適用させるために必要となる．性質(1)，(2)を満足させるために，準結合を利用した基本的な手続きを修正する．

[手続き1] 放送能力を利用した質問処理手続き．

（1） 与えられた質問に対応する質問グラフを G_q とする．

（2） G_q の中から，ある一つの関係 R_i を選択する．目的関係の属性集合を X とする．すなわち，質問に含まれるすべての関係の結合を R とすると，局 S_i において質問の結果である $R[X]$ が要求されているものとする．次のような条件を満足する属性の論理和を Y とする．

$$Y=\underline{R}_i\cap(\bigcup_{j\neq i}\underline{R}_j\cup X)$$

$\underline{R}_i\cap(\bigcup_{j\neq i}\underline{R}_j)$ は，R_i のすべての結合属性を示す．$R_i[Y]$ をすべての局に向けて放送する．

（3） $\underline{R}_i\cap\underline{R}_j\neq\phi$ を満足するすべての関係を，$R_{j1}, R_{j2}, \cdots, R_{jm}$ とする．それらの中からある関係 R_k を選択する．R_k として任意の関係を選択できるが，例えば，R_j の中から R_i に最も近い R_k を選択する方法が考えられる．ここで，質問グラフにおける距離は，二つの節点間の枝の数によって決定される．

(3-1) R_k を除くすべての局 R_j ($j=j1, j2, \cdots, jm, j\neq k$) について，次のような準結合を行う．

$$R_j\ltimes R_i=R_j\bowtie R_i[\underline{R}_j\cap\underline{R}_i]$$

(3-2) R_k について，次のような結合を行う．

$$R_k\bowtie R_i[Y]$$

R_i が $\underline{R}_k\cap Y$ に属する属性を持つならば，局 S_k における関係の属性集合が，$\underline{R}_k$ から，$\underline{R}_k\cup Y$ にかわる．

（4） R_i を取り除くことによって質問グラフ G_q から導かれる新しい質問グラフを G_q' とする．G_q' は次のようにして得られる．

(4-1) R_i と R_j ($j=j1, j2, \cdots, jm$) を直接結ぶすべての枝を取り除く．関係 R_i を取り除く．

(4-2) R_j ($j=j1, j2, \cdots, jm$) を，(3-1)または(3-2)の準結合，または，結合によって得られた関係に置き換える．

(4-3) R_k と R_j ($j=j1, j2, \cdots, jm, j\neq k$) を結ぶ．(4-1)～(4-3)の変換を図2に示す．

G_q' を新しい G_q として，（2）へ進む．

（5） 目的とする局の関係のみが残るまで，上の処理を繰り返す．このとき，その局で結果の $R[X]$

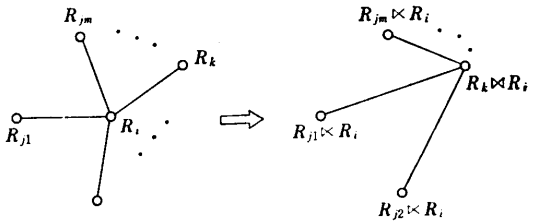

図2 質問グラフの変換
Fig. 2 Conversion of a query graph.

が得られる．

[定理 1] 手続き2は正しい．

(証明) 手続き2の(3)，(4)の変換が正しいことのみを証明すればよい．これは以下に示す式より明らかである．

$$(R_i \bowtie R_{j1} \bowtie R_{j2} \bowtie \cdots R_{jm})[X]$$
$$= \{\bowtie_{\substack{j \\ j \neq k}} (R_j \ltimes R_i) \bowtie (R_i \bowtie R_k)\}[X]$$
$$= \{\bowtie_{\substack{j \\ j \neq k}} (R_j \ltimes R_i) \bowtie (R_i \bowtie R_k)[Y]\}[X]$$

5. 大域的並行制御方式を併合した質問処理方式

前章で示した手法は，どの質問に対しても，同じ順序で処理することができるという性質を持つ．本章では，その性質を利用した大域的並行制御方式を示す．3章で触れたように，大域的正当性の保証の問題は，ロック機能を用いた並行制御の問題に非常に類似している．そこで，ロックを基本とした並行制御方式の木プロトコルの変形と，4章で示した質問処理方式を併合した方式を示す．この方式は，単に，質問言語のみで処理することができる．

まず，三つの局 S_1, S_2, S_3 とそれらすべてを利用する三つの質問 Q_1, Q_2, Q_3 があるものと仮定する．前章で示した質問処理方式では，並行処理は考えず，一度に一つの局でのみ処理を行い，その処理順序は任意に選択できるという特徴を持つ．そこで，**図3**に示すように局を訪れる順をあらかじめ規定し，各局では到着順に直列処理を行い，次の局へ進むというように，パイプライン式に処理を進めていくと，個々の局での処理順序はすべて一致する．これは，木プロトコルでは，項目間の関係が木であったものを鎖状にし，一度に一つの局のみ占有できるように特殊化したものと見ることもできる．(1)各局では到着順に直列に処理，(2)通信バスラインの直列処理の性質により，パイプライン処理を行う．この処理手続きを以下に示す．

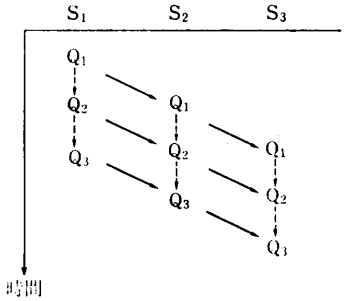

図 3 パイプライン式の処理
Fig. 3 Pipeline processing.

[手続き 2] 大域的並行制御方式を併合した質問処理手続き

(1) 局には，$S_1, S_2, \cdots, S_n$ の順に，固定された順序が付けられているとする．各局は，並行処理能力を持たないので，同時に複数の質問を処理することはできず，到着順に直列に処理を行う．

(2) 各質問は任意の順序で処理可能なので，各局に付けられた順 $S_1, S_2, \cdots, S_n$ に一致するようにその処理順を決定する．すなわち，手続き1の局の選択のために，順序 $S_1, S_2, \cdots, S_n$ を使用する．手続き1を適用するため，以下の二つの場合を考慮することが必要となる．

(2-1) 局 S_i における処理終了後，S_{i+1} に進むが，S_{i+1} にある関係 R_{i+1} が質問に含まれない場合がある．このような場合，質問に S_{i+1} での仮の処理を置く．

(2-2) 目的の局 S_t が，処理すべき最後の局 S_l と異なるかもしれない．このような場合，S_t が目的の局であるかのように手続き1を適用し，S_t で結果を得た後，S_t に送信する．

(3) 各局 S_i では，以下のように処理を行う．

(3-1) S_{i-1} での処理終了後，あるいは，S_i から処理を始めるとき，S_i において他の質問が処理中であるならば，その処理終了まで持つ．すなわち，各局では，要求順に待ち行列にならび，順に処理を進める．

(3-2) 質問を処理するために，局が割り当てられたならば，局所的な処理，S_j $(j<i)$ から送られた値との結合操作を行う．

(3-3) S_i が処理すべき最後の局 S_l ならば終了．そうでない場合，次の局 S_{i+1} に対して処理要求を出す．

[定理 2] 手続き2は，強大域的正当性を満足し，デッドロックが生じないことを保証する．

(略証) 直列可能性とデッドロックの生じないことを保証する木プロトコルの特殊ケースとなっていることを示す．まず局間の関係は，全順序関係であるので，木の特殊ケースとなっている．

(1) 最初は任意の局から始めることができる．

(2) S_i の処理終了後，S_{i+1} の待ち行列の最後に置かれる．各局では，到着順に直列に処理が行わ

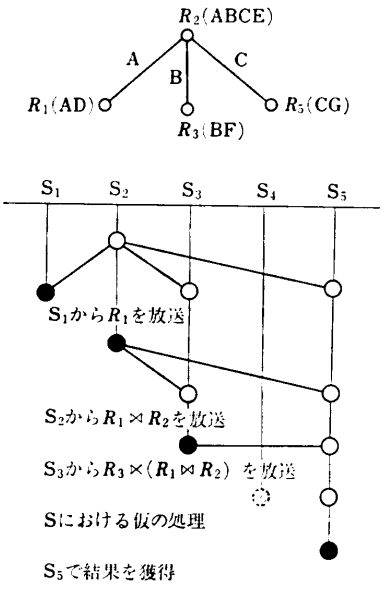

図 4 手続き2の例
Fig. 4 An example of Procedure 2.

れ，また，通信バスラインは直列処理されるので，他の質問に追い抜かれることはなく，木プロトコルの「親にロックがかけられているとき子にロック可能」を模倣している．

(3) 同じ局に二度訪れることはない．

(4) 各局では，一つの質問が占有して，直列に処理を行うので，ロックをかけている場合と等価である．

以上のことにより，手続き2は，強大域的正当性とデッドロックが生じないことを保証する．

[例 2] 図4で示した質問について考える．各関係の属性は，以下のとおりであるとする．

R_1(AD) R_2(ABCE) R_3(BF) R_5(CG)

各関係 R_i は，局 S_i (i=1, 2, 3, 5) で保持されており，すべての関係の結合は，R ($=R_1 \bowtie R_2 \bowtie R_3 \bowtie R_5$) であるとする．局には，次のような固定順が付いているものと仮定する．

$S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \rightarrow S_5$

以下に示すように，順に処理を進める．

S_1: R_1 を S_2 に送る．

S_2: R_2 と S_1 から送信された R_1 とを結合する．結合を行う関係 R_k として任意の関係が選択できるので，R_3 と R_5 のうち，いずれでもよい．ここでは，R_5 を選択したものとし，$R_1 \bowtie R_2$ を放送する．局 S_5 では，$R_1 \bowtie R_2$ をそのまま保持し，S_5 が処理を行う局になったとき，計算を始める．

S_3: R_3 を S_1 から送信された $R_1 \bowtie R_2$ とで，準結合を行う．その結果である $R_3 \ltimes (R_1 \bowtie R_2)$ を放送する．

S_4: R_4 は，質問に含まれないが，他の質問と同期を取るために使用される．待ち行列の最後に置かれ，前の処理がすべて終了した後，次の局へ進む．実際には，その局では，計算は行わない．

S_5: S_2 から送信された $R_1 \bowtie R_2$，S_3 から送信された，$R_3 \ltimes (R_1 \bowtie R_2)$，$S_5$ における R_5 を結合する．その結果は，

$$(R_1 \bowtie R_2) \bowtie \{R_3 \ltimes (R_1 \bowtie R_2)\} \bowtie R_5$$
$$= R_1 \bowtie R_2 \bowtie R_3 \bowtie R_5$$
$$= R$$

となる．Rは，局 S_2 で要求されているので，S_2 へ送信する．

なお，局間の直列的な順序付けは，頻度の高い質問集合をまとめて，それらの処理の効率が良くなるように考えればよく，これには準一連検索ファイルの構成法のような考え方が利用できる．

6. むすび

本稿では並行制御機能のないシステムを統合して分散データベースを作った場合，大域的な質問が参照のみであっても大域的な並行制御が必要なことを示し，ある種の単一データベースシステムでは実現されていないロック等の機能を仮定しない場合の大域的並行制御を考えに入れた質問処理方式を示した．

基本的方式をまず与え，その欠点である(1)大域的質問の多い場合の効率低下，(2)ライブロックの危険性，(3)中心となる局が必要という問題を解決するため，局に固定順を付け，パイプライン式に処理を行う方式を示した．この方式は，木プロトコルの(a)直列可能性の保証，(b)デッドロックの排除，(c)ロールバックが不要という性質を利用している．質問が局を訪れる順序を固定した処理方式は，処理効率の面から考えると問題があるが，本稿の目的は，本来独立に作られたデータベースシステムを統合した場合，簡単な方法で大域的な並行制御を行わせることにあるため，この目的には十分であると考えている．多数の局にわたる質問が非常に多い場合は，トップダウン的に設計されたシステムを用いるべきであろう．

謝辞 日頃，熱心にご討論いただく，京都大学工学

部矢島脩三教授，ならびに研究室の皆様に深謝いたします．

参 考 文 献

1) Bernstein, P. A. and Goodman, N.: The Theory of Semi-Joins, CCA Report, No. CCA-79-27, Nov. 15 (1979).
2) Bernstein, P. A. and Chiu, D. M.: Using Semi-Joins to Solve Relational Queries, *JACM*, Vol. 28, No. 1, pp. 25-40 (1981).
3) Bernstein, P. A. and Goodman, N.: Concurrency Control in Distributed Database Systems *ACM Comput. Surv.*, Vol. 13, No. 2, pp. 185-221 (June 1981).
4) Bernstein, P. A. and Goodman, N.: Power of Natural Semijoins, *SIAM J. Comput.*, Vol. 10, No. 4, pp. 751-771, (1981).
5) Eswaran, K. P., Gray, J. N., Lorie, R. A. and Traiger, I. L.: The Notions of Consistency and Predicate Locks in a Database System, *CACM*, Vol. 19, No. 11, pp. 624-633 (1976).
6) Goodman, N. and Shmueli, S.: The Tree Property is Fundamental for Query Processing (Extended Abstract), Proc. 1st ACM SIGACT-SIGMOD Symposium on PODS (Mar. 1982).
7) Kambayashi, Y., Yoshikawa, M. and Yajima, S.: Query Processing for Distributed Databases Using Generalized Semi-Joins, Proc. ACM SIGMOD (June 1982).
8) Kambayashi, Y. and Yoshikawa, M.: Query Processing Using Dependencies and Horizontal Decomposition, Proc. ACM SIGMOD (June 1983).
9) Papadimitriou, C. H.: The Serializability of Concurrent Database Updates, *JACM*, Vol. 26, No. 4, pp. 631-653 (1979).
10) Papadimitriou, C. H. and Kanellakis, P. C.: On Concurrency Control by Multiple Versions, Proc. VLDB (Sept. 1982).
11) Silberschatz, A. and Kedem, Z.: Consistency in Hierarchical Database Systems, *JACM*, Vol. 27, No. 1, pp. 72-80 (1980).
12) Ullman, J. D.: *Principles of Database Systems*, Second Edition, Computer Science Press, Maryland (1983).
13) Yu, C. T. and Ozsoyoglu, M. Z.: An Algorithm for Tree-Query Membership of a Distributed Query, Proc. of IEEE 3rd International Computer Software and Applications Conference (COMPSAC) (Nov. 1979).

(昭和59年2月6日受付)
(昭和60年4月25日採録)