数理解析研究所講究録
522巻 1984年 280-294

異なる並行制御方式を持つデータベースシステムの統合

京大工学部　近藤誠一 (Sei-ichi Kondoh)
京大工学部　上林弥彦 (Yahiko Kambayashi)

1. まえがき

分散データベースシステムを構成するためには、全体を統一的に設計するトップダウン的方法と、既に存在するシステムを統合して作られるボトムアップ的方法がある。後者の構成法は、実用的でかつ重要であると考えられる。本稿は、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御方式について考察したものである。

個々に生成したデータベースシステムを統合して、分散処理を行う場合、(1) 通信 (2) 質問変換 (3) 大域的な並行制御等の問題が生じる。(1),(2) に関しては、従来、種々の研究がなされている。すべてのトランザクションが参照のみである場合を除いて (3)を扱うことが必要である。大域的並行制御について考察する場合、構成システムの機能によって以下のような類別が可能となる。

(a) システムが並行制御機構を持っていない場合。このようなシステムでは、すべてのトランザクションは、直列に順に

処理され、大域的制御のための機構（ロック機構等）を持たない。このため、システム間の同期をとるための手段として質問言語のみが許される。

(b) 並行制御機構を持っていても、大域的な制御のために利用できない場合。他のシステムと統合することを仮定していない集中型システムでは、自分自身の中では並行制御を行っているが、これを外部より制御できるように作られていない。このため並行制御のための同期の手段として質問言語のみが許される。

(c) すべてのシステムが、大域的な並行制御機構を持っていてもすべてが同一とは限らない場合。たとえば、あるシステムは、時刻印を基本にした機構を用いているが、別のシステムでは二相ロック機構を用いている場合が考えられる。この場合は並行制御のための同期の手段として、質問言語だけでなく各々の並行制御に応じた制御信号を用いることができる。

(d) すべてのシステムが全く同一の大域的並行制御機構を持っている場合。この場合は比較的容易に統合できる。

本稿では、(c) の場合を中心に、大域的並行制御について考察する。この問題は非常に重要であるにもかかわらず、従来ほとんど考察されていなかったと思われる。並行制御機構として、広く知られている二相ロック方式あるいは時刻印順方式を採用しているデータベースシステムを統合する場合の

大域的並行制御方式を示す。この方式は、時刻印順方式を主体とし、大域的な時刻印を与え、それに矛盾しないように制御を行う。

２．　基本的事項

複数の処理を少しづ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。複数のトランザクションを並行処理する場合、基本操作（参照、更新）をシャフルした並行処理スケジュールが作られる。各トランザクションを順にならべ、並行処理を行わず、順に処理を行うスケジュールを直列であるという。与えられたスケジュールを実行した結果がある直列なスケジュールを実行した結果と一致するとき、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当であるとする。直列可能性を保証するため、ロックを用いる方式の代表として二相ロック方式が、ロックを用いない方式として時刻印順方式が広く知られている。ロックを用いて制御を行う方式では、ロックによるオーバヘッドや、デッドロックの問題も考慮することが必要である。ロックを用いない方式では、ロールバックの頻度が高い可能性があるが、分散処理に適しているという利点もある。

２．１．　二相ロック方式　（2PL）

ロックの獲得に関して制限をもうけることにより、直列可能性を保証する方式がある。この中で一般に知られている方式に、二相ロック方式がある。二相ロックとは、ひとつでもロックを解除すると、その後、ロックを獲得することができないという規約である。すなわち、最初のフェーズですべてのロックを獲得し、後のフェーズですべてのロックを解除する。たとえば、

LOCK A　LOCK B　UNLOCK B　UNLOCK A

は、二相ロック方式に従っているが、

LOCK A　UNLOCK A　LOCK B　UNLOCK B

は、二相ロック方式に従っていない。すべてのトランザクションが、この条件を満足しているとき、それらによって構成されるスケジュールは、直列可能性を保証する。このとき、ロックフェーズからロック解除フェーズにうつる順序が等価な直列な順であることが示されている。

２．２．　時刻印順方式　（T/O）

ロックを用いないで、参照、更新の操作を行っていき、問題が生じたとき、ロールバックを行う。各トランザクションは、開始時にユニークな時刻印が与えられ、その順序が等価な直列なスケジュールになるように制御を行う。以下のよう

に制御する。

各データに対して参照時刻印trと更新時刻印twを割り当てる。これらは、このデータの参照または更新を行ったトランザクションの時刻印のうち最大のものとなっている。時刻印tを持つトランザクションが参照時刻印trと更新時刻印twを持つデータに参照命令または更新命令を行おうとしているものとする。

(a) 参照命令でありt≧twならば、命令を行い、t＞trのときはtを新しい参照時刻印とする。

(b) 更新命令でありt≧trかつt≧twならば、命令を行い、t＞twのときはtを新しい更新時刻印とする。

(c) 更新命令でありtr≦t＜twならば、何も行わない。

(d) 参照命令でありt＜tw、または、更新命令でありt＜trならば、トランザクションを無効とし、新しい時刻印を割り当て再実行する。

２．３．　二相コミットプロトコル

分散データベースシステムでは、個々のサイトやネットワークの故障により、部分的にしか処理が行われていないトランザクションが生じることがある。このようなトランザクションは、なかったものとして扱うか、完了するかのいずれかにすることが望ましい。そのための手法として二相コミットプロトコル、及びその変形が知られている。　二相ロック方

式、時刻印方式に、二相コミットプロトコルを適用すると以下のようになる。

ｉ）２ＰＬ

更新命令は一旦ローカル変数に書き込むようにして、すべてのロックの獲得、すべての処理を行い、中心となるサイトがコミット信号を送信した後、ローカル変数の値をデータベース内に書き込み、ロック解除フェーズに入る。したがってあるトランザクション集合について、コミット信号を送信した順序が等価な直列な順のひとつとなる。

ii）Ｔ／Ｏ

更新命令は一旦ローカル変数に書き込み、すべての処理が終了し、コミット送信後、ローカル変数の値をデータベースに書き込む。そのとき矛盾が生じた場合、ロールバックが要求される。

３．　同じ並行制御方式を用いている場合の統合

３．１．　二相ロック方式を採用しているシステムの統合

すべてのシステムが二相ロック方式を採用している場合はその統合は比較的容易である。単に、コミット信号の同期を取るだけでよい。ただし、各システムにより、コミット信号の形式が異なる可能性があるので複数のシステムで処理を行うトランザクション（大域的トランザクション）については

コミット信号の変換を行うことが必要となる。また、このとき、大域的なデッドロックが発生しうるので、それに対する処理も行わなければならない。

３．２．　時刻印順方式を採用しているシステムの統合

個々のシステムにおいて、同じ時刻印順方式を用いている場合でも、時刻印の形式は一般に各々のシステムによって異なると考えられる。こなようなシステムを統合する場合、原理的にはすべての大域的トランザクションに対して、矛盾が生じないように各々のシステムのための局所的な時刻印を与えればよい。ここでの方式は、個々のシステムには何の変更も行わず、制御装置を加える形で実現する。したがって局所的トランザクションは、従来の局所的なシステムのみで処理を行い、大域的制御に影響を与えない。まず基本的方法を与える。

［方法１］

(1) 与えられた大域的トランザクションに対して、大域的時刻印、および使用する可能性のあるシステムすべてのための時刻印を一斉に割り当てる。

(2) 処理中、矛盾が生じロールバックが必要となった場合はすべてのシステムの状態を開始前の状態にもどし、すべての時刻印を割り当て再実行する。

時刻印順方式では、時刻印の小さなトランザクションの方

が、ロールバックされる傾向が強い。したがって、方法１では、すべてのシステムに対する時刻印を開始時に与えるため実際の処理がずっと後になる場合には、再実行の可能性が高くなる。そこで、それを改善するため、時刻印の割り当てをできるだけ後に遅らせる方法を示す。この方法では、時刻印に関する表を用意し、局所的時刻印を与えるまでその属性には空値⊥を置き、まだ情報が与えられていないことを示す。

［方法２］

各システムＳｉに大域的時刻印Ｇと局所的時刻印Ｌを属性とする関係ＴＳｉを置く。ＧＴＳ（Ｔｉ）、ＬＴＳ（Ｔｉ,Ｓｊ）を各々トランザクションＴｉの大域的時刻印および、システムＳｊにおける局所的時刻印とする。

(1) トランザクションＴｉが開始された場合、大域的時刻印ＧＴＳ（Ｔｉ）を割り当て、Ｔｉの使用するすべてのシステムにおける時刻印に関する関係に〈ＧＴＳ（Ｔｉ），　⊥〉を挿入する。

(2) システムＳｊにおけるトランザクションＴｉの処理が開始されたとき　まだ局所的時刻印が与えられていないならば、それを割り当てる。ＴＳｉにＧＴＳ（Ｔｉ）より小さい大域的時刻印を持ちそのＬ値が⊥であるトランザクションＴｋが存在するならば、Ｔｉの局所的時刻印を割り当てる前にＴｋの局所的時刻印を割り当てる。

(3) Ｔｉが終了した場合、Ｔｉに関する組をすべて削除する。

［例１］独立システムＳ１、Ｓ２、Ｓ３を統合した場合を考える。以下の３つのトランザクションＴ１、Ｔ２、Ｔ３がシステムに加えられるものとする。Ｔ１はＳ１とＳ２を、Ｔ２はＳ２とＳ３を、Ｔ３はＳ１とＳ２とＳ３を使用するものとする。

例えばこれらのトランザクションは以下のように処理される。

Ｔ１：開始。ＧＴＳ（Ｔ１）＝１

Ｓ１で処理。ＬＴＳ（Ｔ１，Ｓ１）＝１０

ＴＳ１

G	L
1	10

ＴＳ２

G	L
1	⊥

ＴＳ３

G	L

Ｔ２：開始。ＧＴＳ（Ｔ２）＝２

Ｓ３で処理。ＬＴＳ（Ｔ２，Ｓ３）＝５

G	L
1	10

G	L
1	⊥
2	⊥

G	L
2	5

Ｔ３：開始。ＧＴＳ（Ｔ３）＝３

Ｓ２で処理。大域的時刻印に矛盾しないように局所的時刻印を割り当てる。

G	L
1	10
3	⊥

G	L
1	100
2	150
3	200

G	L
2	5
3	⊥

4．　異なる並行制御方式の統合

本節では、二相ロック方式を採用しているシステムと、時刻印順方式を採用しているシステムを統合する場合の大域的並行制御方式について述べる。二相ロック方式を採用しているシステムのみを用いるトランザクションあるいは時刻印順方式を採用しているシステムのみを用いるトランザクションについては、前節の方法がそのまま適用でき、しかも、制御は分散できるような方法を示す。

二相ロック方式では、コミットの順序（もっと一般的にいうとすべてのロックを確保しているインターバルの順序）が等価な直列な順序のひとつであるので、　この性質を利用する。すなわち、最後のロックをかけた順序、あるいは、コミットの順序と、大域的時刻印の順序を一致させれば全体として正当性を保つことができる。以下に示す方法３は、コミットの順を時刻印の順に合わせるものである。すなわち、時刻印の順に処理を完了する。

［方法３］

(1) 各システムのみで処理できるトランザクションは統合前と同様に各々のシステムのみで制御する。

(2) ２ＰＬを採用しているシステムとＴ／０を採用しているシステムの両者を利用するトランザクションは、以下のようにして処理を行う。

(2-1) 処理開始時に中心となる局(たとえば処理を始めた局)において大域的時刻印を割り当てる。T／Oを採用しているシステムは前節で示した方法2を利用するが、そのとき、この大域的時刻印を使用する。

(2-2) 二相ロックを採用しているシステムに下図に示すような関係を置く。将来使用する可能性のあるシステムで保持している関係に (2-1)で与えられた時刻印を登録する。このときSTATUSは"active"にする。 以後は各々のシステムにおいて、各々の並行制御方式に従った処理を行う。

TS (timestamp)	ST (status)

(2-3) 各々のシステムにおいて処理が終了した後、中心となる局へ ready信号を送信する。このとき、2PLを採用しているサイトでは、表の対応する組のSTATUSを "end"にする。このトランザクションの時刻印が表中で最小ならば、表から削除する。その結果、表中で最小の時刻印を持つ組のSTATUSが "end"となったら、この組も削除する。表から組がなくなるか、あるいは最小の時刻印を持つ組のSTATUSが"active"になるまでこの操作を繰り返す。

(2-4) 中心となるサイトがすべてのサイトから ready信号を受信した後、コミット信号をまずT／Oを採用しているサイ

トへ送信し、ローカル領域の値をデーターベース内に書き込む。

(2-5) (2-4)の操作が矛盾なく完了したならば、二相ロック方式を採用しているシステムにコミット信号を送信する。各システムの表において、対応する組が削除されていたら、データベースへの書き込みを行い、操作を完了する。削除されていない場合は、削除されるまで待つ。

(2-6) (2-4)の操作において矛盾が生じた場合、新たな時刻印を割り当てて再実行する。

[例２] ２ＰＬを採用しているあるサイトにおいて、ＧＴＳ(Ｔ1)＝10, ＧＴＳ(Ｔ2)＝20, ＧＴＳ(Ｔ3)＝30であるトランザクションの状態が現在下図に示すようであるとする。

(1) Ｔ1が終了した場合、表から〈10,active〉を削除し、中心となるサイトへ ready信号を送る。

(2) その後Ｔ3が終了した場合、Ｔ2はまだ実行中であるので〈30,active〉を〈30,end〉に変更して ready信号を送る。

ＴＳ	ＳＴ
10	active
20	active
30	active

(a)

ＴＳ	ＳＴ
20	active
30	active

(b)

ＴＳ	ＳＴ
20	active
30	end

(c)

Ｔ３はコミット信号を受信しても Ｔ２が終了するまで、処理を完了することはできない。

(3) その後 Ｔ２が終了した場合〈20,active〉,〈30,end〉を削除し、ready信号を送信する。その結果、Ｔ３は完了することができる。

方法３では矛盾が生じたときロールバックされる。通常、ロールバックが要求されたときは、初期状態にもどし再実行されるが、システム間の依存関係により、波及するシステムを少なくすることが可能となる。しかし、他の大域的トランザクションとの関係により、それができない場合がある。方法３のＴ／０を採用しているシステムに適用すると以下に示すようになる。

［方法４］

(1) ロールバックが必要となったシステムにおける時刻印に関する関係を参照する。もし、そのトランザクションの持つ時刻印よりも大きな時刻印を持つものが、まだ登録されていなかったり、あるいは、登録されていてもＬ値が⊥ならば、全体を元に戻す必要はなく、局所的な時刻印を割り当ててそのシステムにおける処理を再実行する。

(2) (1) の条件を満足しない場合は、そのトランザクションが使用しているすべてのシステムについて、元の状態に戻し

新しい大域的時刻印を割り当てて、再実行する。

［例３］Ｔ／Ｏを採用しているシステムとして S1,S2,S3 の３つがあるものとする。今、各々の時刻印表は下図に示すようになっているものとする。ＧＴＳ（T1）＝1、ＧＴＳ（T2）＝2、ＧＴＳ（T3）＝3であるとする。

ＴＳ１

G	L
1	10
3	⊥

ＴＳ２

G	L
1	100
2	150

ＴＳ３

G	L
2	5
3	6

(1) T2が、システムS2で矛盾が生じ、ロールバックが要求されたものとする。S2において、T2は最も大きな大域的時刻印を持つので、S2だけロールバックし、新たな局所的時刻印を割り当てて再実行する。

ＴＳ１

G	L
1	10
3	⊥

ＴＳ２

G	L
1	100
2	200

ＴＳ３

G	L
2	5
3	6

(2) その後、T1がシステムS2で矛盾が生じ、ロールバックが要求されたものとする。S2においてT1は条件を満足しないので、全体をロールバックすることが要求される。すなわち、新しい大域的的時刻印が割り当てられ、再実行される。

TS1

G	L
3	⊥
4	⊥

TS2

G	L
1	100
2	⊥

TS3

G	L
2	5
3	6

5. あとがき

本稿では、使用できる制御情報として、２ＰＬを採用しているシステムではコミット信号を、Ｔ／Ｏを採用しているシステムでは局所的時刻印の生成を仮定した。これらが使用できない場合は、いわゆる質問変換(query modificaion)だけで処理することが必要となる。

謝辞　日頃熱心に御討論頂く矢島脩三教授に深謝致します。また、有益な御助言頂く矢島研究室の皆様に感謝します。

参考文献

[BERNG8010] Bernstein,P.A. and Goodman,N. "Timestamp-Based Algorithms for Concurrency Control in Distributed Database System" VLDB, pp.285-300, Oct.1980

[ESWAG7612] 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 19,11 Nov.1976

[FUSEK8105] Fusell, D., Kedem,A.M., and Silberschatz,A. "Deadlock Removal Using Partial Rollback in Database Systems" ACM SIGMOD, pp.65-73 May 1981

[ULLM83] Ullman,J.D. "Principles of Database Systems" Second Edition, Computer Sience Press 1983