数理解析研究所講究録
591巻 1986年 111-120

メタデータ制御を考慮した重複分散データベースシステムの並行処理

九大·工　仲　興国（Xingguo Zhong）
九大·工　上林 弥彦 (Yahiko Kambayashi)

1. まえがき

分散データベースシステムでは、集中データベースに対する諸要求の他にデータの位置不可視と重複不可視という要求がある[4]。位置不可視とはデータベースの利用者や応用プログラムなどが局を意識することなく、データベースに記憶するすべてのデータは自分の利用する局にあると考えて良いことをいう。重複不可視とは同一の論理データが異なる局に複数個格納されるが、利用者から見た場合、これが唯一になることである。

位置不可視と重複不可視を実現するために、データベースシステムの各局においてその局から利用(操作)可能なすべてのデータに対する記述が必要である。即ち、その各コピーがどこにあるか、どのような利用状況になっているかなどの記述がなければならない。このようなデータを記述するためのデータをメタデータという。同一のデータまたはデータコピーに対する記述(メタデータ)が複数の局に記憶されているため、メタデータは高度に重複したデータと考えられる。図1は重複したデータxとそれを記述するメタデータとの関連を記述している。

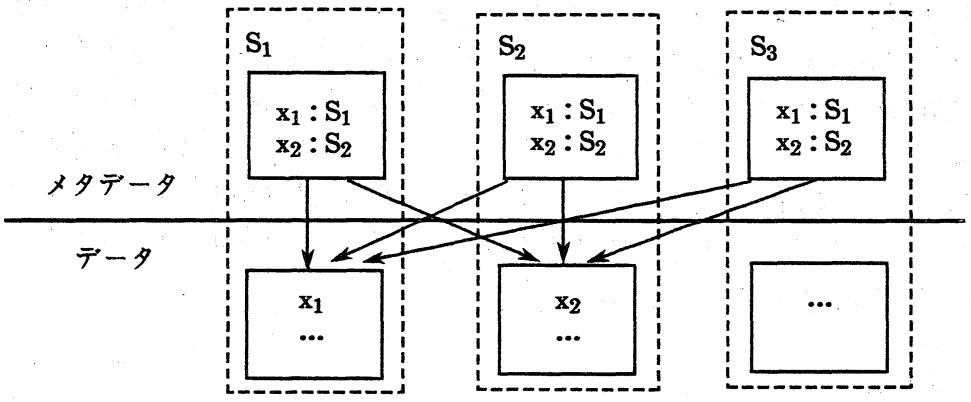

図1. データとメタデータとの関連

分散データベースでデータを重複させる主な目的はシステムの信頼性を高めることにある。重複した同一のデータ中の一部分のコピーを故障によって失ったり或いは局の保守などで一時的に操作できなくなったりするときに、システムの機能に影響がないことである。従って、重複した分散データベースシステムでは、システムにおける各データ或いはそのコピーが操作できないときがありうると考える必要がある。それを表現するために、メタデータにはデータおよびそのコピーだけでなく、データの利用状況に対する記述も必要となる。しかも、そのような記述はシ

ステム内で動的に変更されるものである。メタデータが動的に変更されると考えると、並行処理はデータのみならず、メタデータに対しても行わなければならないことになる。

本稿ではこのようにデータとメタデータが存在する場合の並行処理について基本的な問題点を整理し、分散データベースにおいて1つの局を停止させたり再実行させたりした場合の処理方式について考察した。メタデータを考えた場合の並行処理については、従来ほとんど扱われていない。この場合次のような問題点がある。

(1) メタデータとデータの内容が矛盾してはならない。

(2) メタデータ参照が多く、このための効率低下を避けなければならない。

(3) データとメタデータの両方を利用するトランザクションの扱いを考えなければならない。

(4) 処理コストをできるだけ下げるための方法を開発しなければならない。

上記の問題は一般的な立場から考えると、メタデータを含めた直列可能性の保障にいかにしてメタデータ上のボトルネック問題を解決することになる。文献[1、2]では、二相ロック方式を用いて理想化したデータ故障およびその回復を扱う並行処理アルゴリズムを提案している。本稿では、上記に示す問題点により一般的な立場の特殊例として、時刻印方式による局の停止およびその回復を扱った。上記の問題点を解決するには、主に以下の3つの手法を使用している。

a) 従属トランザクションの導入

b) ユーザトランザクションの実行に伴うメタデータの変更

c) データ操作状態の設定

本稿で述べるアルゴリズムは文献[1]に仮定したの場合を含めて、システムの管理、保守によるデータの一時的な利用不可能の扱いなどに容易に拡張可能である。以下2節では、基本的事項として時刻印方式の紹介、本稿に関連するシステムの仮定および上記の問題点について記述する。3節ではメタデータを考慮した並行処理アルゴリズムの記述およびその正当性の討論を行なう。4節は検討として3節で記述するアルゴリズムにより上記の問題点をどのように解決しているかおよびそのアルグリズムの拡張可能な機能について述べる。5節はむすびであり、メタデータとデータの干渉に関する課題をも与える。

2. 基本的事項

2.1 並行処理

ここでは、時刻印方式を簡単に紹介する[3]。データベースを操作する論理的な処理単位をトランザクションという。トランザクションはデータに対する読み出し操作と書き込み操作からなるとする。時刻印方式では基本的に、各トランザクションT_iは到着順に時刻印t_iを割り当てられる。データxの読み込み時刻印$t_r(x)$はxを読んだトランザクションの時刻印のうち最大のもの、xの書き込み時刻印$t_w(x)$はxを書き換えたトランザクションの時刻印のうち最大のものとする。

$t_w(x) < t_i$ならば、T_iはxを読み出すことができ、$\max(t_r(x), t_w(x)) < t_i$ならば、$T_i$はxを書き換えることができる。$t_w(x) > t_i$または$t_r(x) > t_i$ならば、$T_i$はロールバックされる。

時刻印方式には多重版法がある。各データxは過去の値も多重版として持つ。書き込みはデータの新しい版を生成する。T_iによる読み出しは時刻印t_iに有効なデータ値を読む。時刻印t_iが最大の$t_r(x)$および$t_w(x)$と比較して$\max(t_w(x), t_r(x)) < t_i$ならば書き込むことができる。

更に多重版法の場合、操作の矛盾が起こったとき上のような時刻印の小さいトランザクションではなく、時刻印の大きいトランザクションをロールバック対象として扱う手法がある[5]。その方法により、書き込み操作を持つトランザクションが優先して並行性を高めると考えられる。

重複分散データベースシステムの場合に、トランザクションが実行する場合データの読み出し操作は重複するデータの任意の1つのコピーに対して行い、書き込み操作は重複するデータのすべてのコピー上に書き込むことになる。また、データコピーの利用可能性が動的に変更されるとき、読み出す対象は読み出し可能なコピーでなければならない。書き込み操作は書き込み可能なすべてのコピーに対して行わなければならないことになる。

2.2 システムに対する仮定

分散データベースシステムは、論理的に通信回線で接続された複数の局(site)からなる。任意の2つの局間に通信により互いにメッセージおよびデータの転送がなされる(図2を参照)。

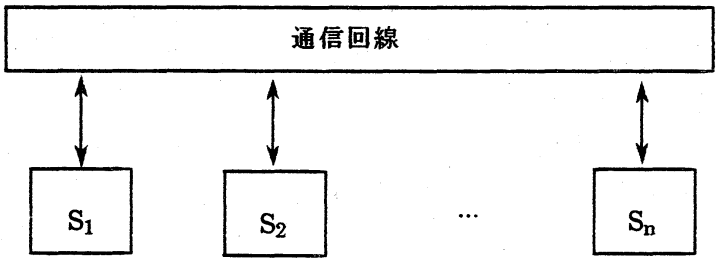

図2. 分散システムの構成

データベースは論理的なデータ項目x、y、... の集合からなる。同一のデータxは異なる局において、複数のコピーx_i、x_j、... が存在することがある。システムにおける任意の局S_iが一時的に停止して再実行することは許容される。実行局から停止局に通信できず停止局に記憶するデータ或いはデータコピーを操作できないことになる。局の停止は他の実行局間の通信に影響なく、実行局のみからなる部分システムは分割されないとする。局S_iが停止している間に、S_iで記憶しているデータxのコピーが他の実行局にも存在すればデータxが利用でき、データxを操作するトランザクションが実行可能である。データxのすべてのコピーが操作できないときデータxが利用不可能となる。特にxに対し唯一のコピーしか存在しないとき、xを記憶する局が停止すると、xが利用不可能になる。このような現象はtotal failureという。

2.3 問題点

以上で仮定したシステム機能では、データの利用可能性、局の状態などが動的に変化していく。従って、各局におけるメタデータが動的に変更される。このような場合に並行処理を行うと、次の問題を考えなければならない。

(1) メタデータとデータの一致性

並行処理の正しいアルゴリズムはメタデータとデータの一致性を保障しなければならない。即ち、任意のユーザトランザクションに対してそれが参照するメタデータと操作するデータが矛盾しではならない。特に分散の場合関連するメタデータとデータが異なる局に置かれる。従って、メタデータを考慮した並行処理は全域的に分散しているメタデータの変更および参照を含めた直列可能性を保障する制御となる。

(2) メタデータ上のボトルネック

データに対するすべての操作はメタデータの参照を経て行われる。従って、ユーザトランザクションの実行にあたって、メタデータに対する参照は頻繁に行われる。メタデータに対する変更操作が存在するとき、メタデータ変更による並行処理に関する矛盾する操作が非常に起こりやすくボトルネックとなる。従って、メタデータの操作上の矛盾をいかに減少させるか、或いはボトルネック問題をいかに緩めるかということは重要である。例えば、1つの局を回復するときそこに記憶しているデータコピーが他の局に記憶する同一のデータコピーを変更したため無効になる。そのときデータの回復を行わなければならない。またデータの回復に伴ってすべての局に格納しているメタデータの変更も必要となる。メタデータの変更のうち1つが矛盾する操作となると大きいコストがかかったデータ回復も無効となる。

(3) データ伝送コストの減少

分散データベースシステムでは、局間のデータ転送はできる限り減少させることは非常に重要である。重複分散データベースシステムの場合、データの回復に要するコストは大きい。従って各局の回復にあたって、やむを得ない場合のみに限り、データを転送するようにすべきである。上記と同じ例で1つの局を回復するとき、他の局に記憶する同一のデータコピーが変更されていなければ、その局のコピーを回復しなくて良い。従って、メタデータをうまく扱えばこの検査は可能である。更にtotal failureの場合 どのように回復するのもできなければならない。

3. 並行処理アルゴリズム

本節では、メタデータを考慮した時刻印方式で上述のシステム機能を持つ並行処理制御を記述するとともに、以上で提出した問題点に対する解決を指摘する。メタデータの変更および参照を含めた直列可能性を保障した上、メタデータ上のボトルネックの緩めおよび転送コストの減少を求めた並行処理制御を記述する。ここでは、従属トランザクションの概念を導入する。直列に実行する2つのトトランザクションT_1とT_2があり、T_2の実行はT_1のコミットを前提とするとき、T_2がT_1の従属トランザクションであるという。データを回復するとき、データの変更とメタデータの変更の2つに分割し、それぞれ従属トランザクションT_1とT_2にする。よってメタデータを変更するトランザクションを簡単化し、メタデータ上の並行性を向上させる。またメタデータ

にデータを変更したときの時刻印を付けることにより、回復する必要のないデータを知り不必要なデータ転送を済ませる。

局の再実行には局が停止していた間に行った変更操作を記憶し回復時にその局上に行なう方法と、各論理データをコピーしてくる方法がある。記録した操作を行なう方法には多くの問題がある。例えばそれらの操作を行なう間にユーザトランザクションが同一のデータに対して操作するとき、そのような操作を追加していかなければならない。本稿ではデータコピー全体をコピーしてくる方法を用いている。

以下に記述するアルゴリズムにおいて、特定の時刻印方式および並行処理単位にはよらず、メタデータとデータともに対して時刻印を付け、各種トランザクションに時刻印を割り付けられることだけを考える。データに対する処理単位はメタデータで記述する論理データ以下であっても良いことは容易に考えられる。アルゴリズムの構成は、a)局を停止するときのトランザクション、b)局を再実行するときのトランザクションの集まり、c)ユーザトランザクションの3つからなる。

3.1 システムおよびデータの状態 ── メタデータの設定

システムの状態はシステムを構成するすべての局の状態で決める。局の状態には、実行状態(on-state)と停止状態(off-state)がある。それを表現するには各局においてテーブルSys-Stateを設定する(図3を参照)。テーブルSys-Stateはすべての局にコピーのある重複メタデータとなる。

Sys-State

site	state
S_1	on
S_2	off
...	...

Data-Dir

data	site	op-state	stamp
x	S_1	A	534
x	S_2	N	
y	S_1	W	357
...	...	...	...

図3. 各局におけるメタデータ

データコピーの配置およびその操作状況を表現するために、各局においてその局から操作可能なすべてのデータをテーブルData-Dirに記入する(図3を参照)。データベースにおけるデータコピーの状態には、操作不可能(N)、書き込み可能(W)、書き込みかつ読み出し可能(A)の3つがあるとする。

テーブルSys-Stateにおける一組は1つの局の現在の状態を表す。テーブルData-Dirにおける一組は1つのデータコピーを表す。関係データベースの場合データが分割されたフラグメント(fragment)と考えて良い。テーブルData-Dir中の項目op-stateは対応するデータコピーの状態を表す。項目stampはその局に格納するデータのみに対して意味を持つ。そこに対応するデータコ

ピーを最後に書き換えたトランザクションの時刻印を記入する。Data-Dir中の組は、同一の局に格納しているデータを記述するとき(即ち局SiにおいてData-Dirのsite＝S_iを満たす組は)局所メタデータという。他の組は他の局に格納しているコピーを記述しているので全域メタデータということにする。テーブルData-Dirは局所および全域メタデータを1つに集中したものである。データベースに他のメタデータが多くあるが、本稿では考慮しない。

本稿において、簡単のため、テーブルSys-StateとテーブルData-Dirを関係データベースの関係と見なし、それに対する操作は関係操作で表現することにする。関係Sys-stateに属性site、関係Data-Dirに属性dataとsiteがキーとなることが分かる。

3.2 局の停止(site-stop)

システムに局S_1を停止するときに、すべての実行局がそれを知らなければならない。それを実現するために次のトランザクションを実行する。

テーブルSys-Stateによりすべての実行局を知る。

すべての実行局に、"私は停止する"というメッセージを伝える。局の故障による局停止の場合、故障の見付かったトランザクションを発行した局で同様なトランザクションを生成すれば良い。このメッセージにより、各局においてそれぞれ次のサブトランザクションを生成する。

```
Update    Sys-State
  Set     state='off'
  Where   site=S1
Update    Data-Dir
  Set     op-state='N'
  Where   site=S1
```

このトランザクションの実行により、他の局からその局に格納しているデータを操作しないことになる。すると書き込み操作を行うとき、その局に格納しているコピーに対して行わないことになる。

3.3 局の再実行

局S_1の再実行は、局の回復、メタデータの回復およびデータの回復の3つの処理に分割される。局の回復は1つのトランザクションで実現する。それにより、他の局からはその局を実行局と見なすことになる。メタデータの回復はトランザクションの集合である。各実行局S_iごとに対して、1つのトランザクションを実行し、必要なメタデータをコピーしてくる。この回復により、回復局からは他の実行局におけるデータが利用可能となる。データの回復はトランザクションの集合で、回復局に格納している各データx_1に対する回復を逐次に行なう。

<1>　Site-Recovery (局の回復)

テーブルSys-Stateを回復する。

回復したテーブルSys-Stateにより、すべての実行局に(自分も含めて)"私は実行する"というメッ

セージを伝える。このメッセージにより各局においてそれぞれ次のサブトランザクションを生成する。

```
Update    Sys-State
  Set     state='on'
  Where   site=S1
Update    Data-Dir
  Set     op-state='W'
  Where   site=S1
```

<2>Meta-Recovery (S_1)(メタデータの回復)

実行局S_iへメッセージを送り、それにより局S_iに格納しているデータコピーに対するメタデータを転送してくる。

```
Select    op-state
From      Si.Data-Dir
Where     site=Si
```

それを用いて局S1での対応するメタデータを置き換える(それに結合操作が含まれる)。

```
Select    op-state
From      S1.Data-Dir
Where     site=Si
```

<3>Data-Recovery(x_1)(データの回復)

回復局におけるすべてのデータコピーx_1に対して、二組のトランザクションT_1、T_2を生成する。T_1はデータコピーx_1の回復および対応する局所メタデータの変更を行う。T_2は回復したデータコピーx_1に対応する各局の全域メタデータを変更する。ここでT_2はT_1の従属トランザクションである。

T_1 : 上述の<2>により、データxの読み出し可能なコピーx_iを選択する。選択できると、時刻印の比較を行なう。

$$\text{S1.Data-Dir.stamp \{where data = x , site = S1\}}$$
$$\text{=Si.Data-Dir.stamp \{where data = x , site =Si\}}$$

満たすならば、コピーする必要がない。そうでなければ、データxに対して局Siから局S_1へのコピー($x_1 \leftarrow x_i$)を行なう。即ちx_1の値をx_iの値に置き換える。そのとき対応する時刻印をもコピーすることにする。コピーをするか或いはしないかのいずれも、回復局においてData-Dir上の対応する組のop-stateをAにする。次いでトランザクションT_2を実行させる。

選択できないときデータxがtotal failureとなる。そのとき次のtotal recoveryを行なう。

データxを格納するすべての局を実行中かどうかをSys-Stateを参照することにより検査する。そうであれば各局からその局におけるデータxのコピーに対するstampを参照する。stampが最大となるデータコピーが有効とすることにし、無効なデータコピーに対する回復を行なう。xを格納する局に停止局があればデータxの回復ができず、T_1を中止する。そのときT_1に属するT_2も実行させないことにする。

T_2 ： データx_1を格納している局S_1から生成するトランザクションであり、すべての実行局に対して次のサブトランザクションを行なう。

Update　Data-Dir
　Set　op-state＝A
Where　site＝S_1　and　data＝x

3.4 ユーザトランザクション

ユーザトランザクションの実行は、従来のデータに対する読み出し、書き込みからなる以上、メタデータに対する操作を考慮する。データに対する読み出しまたは書き込み操作は、そのデータの対応する操作可能性を前提とする。また、1つのデータコピーを書き換えたとき、その局のテーブルData-Dir上の対応する組上にop-state＝Aならば、そのstampを時刻印方式の変更基準と同様にそのトランザクションの時刻印に変更する。それは対応するデータコピーがその時点で変更されたことを示す。op-state＝Wのときは、stampを変更しないことになる。

3.5 本アルゴリズムの正当性

以上で述べたアルゴリズムは局の停止および再実行を扱う。それにtotal failureの扱いも含めている。その正当性は次の2段階の解釈により考えられる。

まず従属トランザクションを除いて、即ち従属トランザクションT_1とT_2を合わせて1つのトランザクションとして実行するときメタデータを含めた時刻印方式となる。従って、そのときのトランザクションの直列可能性を保障することが容易に分かる。

次いで、T_1とT_2の分割によって、変更されたデータとメタデータの対応関係はユーザトランザクションと干渉する。その解決方法はデータの変更とその局所メタデータの変更はT_1に含まれ、他の局での全域メタデータをT_2に含めることにより実現される。局の回復時に回復局のデータコピーx_1は書き込み可能となっている。データxを書き換えるトランザクションTがT_1の前に実行したならば、コピーx_1が無効であってもT_1の実行により回復される。トランザクションTがT_1とT_2の間に実行したときすべての局においてコピーx_1が書き込み可となっているので、データx_1に対する書き込みは失わない。よって、上によりT_1とT_2が分割されて実行しても、その効果はT_1とT_2を1つのトランザクションとして実行するときの効果は同様であることが分かる。

並行処理では一般に1つのデータに対して書き込みできないときに読み出し可能な場合がある。それに対してデータの回復には読み出しできなくても以上によれば書かなければならないことがある。書き込まれるデータが既に有効のときその書き込みによりデータの有効性を保持する。無効なデータを書き換えても矛盾が生じない。これによって、異なる局における同一のデータコピーに対する状態記述(操作状態)が一時的に異なっても良いことがになる。

4. 検討

4.1 特色

従属トランザクションの導入により、データの回復にデータの変更とメタデータの変更を

2つのトランザクションに分けて扱う。メタデータを変更するトランザクションはごく簡単となる。メタデータに対するボトルネック問題はある程度緩和される。システム内でメタデータは頻繁に参照されるが、メタデータの変更は非常に少ない。そのため、メタデータを変更するトランザクションを優先する必要があると考えられる。即ち、メタデータ上で矛盾が生じたときその時点で書き込み操作を行なうトランザクションではなく、それと矛盾するトランザクションをロールバックすることが良いと考えられる。そうすれば、メタデータ上のボトルネック問題はシステム全体の性能の立場から考えたとき解決可能である。

局の再実行は、局の回復、メタデータの回復およびデータの回復の3つに分割されている。局の回復はすぐに完了するので、局の停止期間が短縮される。従って、他の局でのデータの変更による回復データコピー無効となる可能性は減少される。これは特に局の停止期間が短く、局の再実行に要する時間とほぼ同程度の場合に意味がある。また局が長時間停止して再実行する場合、データの回復に要する時間が長くなる。そのとき、その局が早く利用できるようにすることとその局から他の局に格納しているデータが利用可能にすることは重要である。以上のアルゴリズムにおいて、局の回復およびメタデータの回復を完了するとそのような利用は可能となる。

4.2 機能の拡張

分散データベースシステムでは、システムの故障やシステムに対する管理、保守などにより、1つの局が一時的に利用できない原因が多くある。それに基本的な事象は局の停止および再実行である。このアルゴリズムにおいて局の停止処理は他の局からトランザクションを生成すれば文献[1]に述べた環境に適用できる。他の原因による一時的なデータ利用不能やシステムの管理、保守に必要とするデータベース専有等の扱いも容易に扱えると考えられる。例えば、局S_1に格納しているデータxを局S_2にデータコピーを追加するとき、以下の次々に従属する4つのトランザクションの実行により実現可能である。

T_1 : 局S_2においてx_2に対する局所メタ情報(論理構造、物理構造など)や、データ領域の用意を行なう。

T_2 : xが利用可能な各局に対してコピーx_2を追加するためのメタデータを生成する。それにop-stateを書き込みのみ可能(W)にする。

T_3 : データ回復と同様に有効なデータコピーx_1を用いてコピーx_2にコピーする。

T_4 : xを利用可能な各局にメタデータの変更を行なう。即ちコピーx_2の状態op-stateを書き込みかつ読み出し可能に変更する。

5. むすび

重複分散データベースシステムの並行処理にメタデータ制御を考慮する必要がある。そのときメタデータ上のボトルネック問題が生じる。本稿ではそれらの問題を整理し、従属トランザクションの導入により1つの解決方法を提案した。分散データベースシステムに並行処理制御の可制御性[6]が重要である。そのうち異なる優先順位を持つトランザクションの処理機能が実現時

に必要である。制御可能な並行処理には、メタデータとデータの干渉はより複雑となる。またメタデータを変更するトランザクションのロールバックはデータベースの機能に大きな影響があたる。従って、並行処理を考えた上でメタデータを考慮したロールバックの扱い方法も考えなければならない。更にメタデータに対する並行処理単位とデータに対する処理単位の関係も1つの課題となり、メタデータに対する操作対象集合の表現は重要である。

参考文献 :

[1] Bernstein,P.A. and Goodman,N., "An Algorithm for Concurrency Control and REcovery in Replicated Distributed Databases", ACM Trans. on Database Systems, Vol. 9, No. 4, pp. 596-615, Dec. 1984.

[2] Goodman,N., Skeen,D. and Chen, A., "A Recovery Algorithm for a Distributed Database System", ACM SIGMOD, pp. 8-15, 1983.

[3] Bernstein,P.A. and Goodman,N. "Timestamp-Based Algorithms for Concurrency Control in Distributed Database Systems", Proc. 6th Int. Conf. VLDB, Oct. 1980.

[4] Traiger,I.L., et al., "Transactions and Consistency in Distributed Database Systems", ACM Trans. on Database Systems, Vol. 7, No. 3, pp. 323-342, Sep. 1982.

[5] 仲興国, 上林弥彦, 長時間トランザクションが存在する場合の並行処理制御, 情報処理学会第32回全国大会講演論文集 2C-4, 1986年3月.

[6] 上林弥彦, 近藤誠一, 並行処理制御方式の統合, 情報処理学会 第31回全国大会講演論文集 2B-4, 1985年9月.