

マルチデータベースシステムにおけるグローバルデッドロック回避による
一貫性制御方式

吉田 誠

(株) 沖テクノシステムズラボラトリ

異なった機種 of 異なったデータベース管理システムをネットワークで接続し、異機種分散型データベースシステムを実現するマルチデータベースについて述べる。マルチデータベースシステムを実現する上で重要なことは、分散アクセスが行えることに加えて、既存データベース管理システムの自律性を保つことである。

本論文では、ローカルデータベース管理システムの自律性を最大限生かし、既存のデータベース管理システム及び運用中のアプリケーションプログラムを全く変更することなく運用でき、かつグローバルデッドロックを回避することにより、分散データベースシステムとしての分散アクセスを一貫性を失うことなく実現できる方式について述べる。

Global Deadlock Avoidance Mechanism in Multidatabase Systems

Makoto Yoshida

OKI Technosystems Laboratory, Inc.

3-8-10, Uchiyama, Chikusa-ku, Nagoya 464, Japan

The global deadlock avoidance mechanism in Multidatabase systems is described. Multidatabase systems provide uniform access to the existing database systems, retaining local autonomy, without modifying the existing database systems and current applications.

However, such systems in contrast to their advantages, create complexities which necessitate various distribution controls such as global deadlock resolution and consistency maintenance which in turn boost up overheads.

This paper presents the global deadlock avoidance mechanism in multidatabase systems. Applying our mechanism, it can maintain the global consistency with minimum effort. Generalized schema for global deadlock avoidance mechanism is also presented.

1. はじめに

分散データベースシステムの世界で、マルチデータベースと言う言葉が流行している[4, 6, 8, 9, 10, 12, 19, 20]。各種ベンダーにより開発・運用されている既存のデータベースをネットワークで接続し、分散データベースとしても利用しようとする試みである。

しかしながら、マルチデータベースによる分散データベースシステムは、分散データベースシステムと言う機能面に着目すると同じであるが、同機種同種のホモジニアスな分散データベースシステムとは技術的にも非常に異なるものである。各システムにより、データモデル[15, 18]、データベース言語[9, 16, 17]、共有制御方式[3, 7, 13]、トランザクション処理方式[5, 6]、インテグリティ、セキュリティ、OS、ネットワーク[1, 2]等が異なり、ほとんどのシステム間で合い入れないのが現状である。このような既存システムをマルチデータベースとして機能させるためのアプローチとして、既存データベース管理システムをマルチデータベース用に改造し分散アクセスを可能にする方法と、既存データベース管理システムを全く変更することなく自律性を失うことなく、分散アクセスも可能なマルチデータベースを構築する2つのアプローチが考えられる[5, 6]。前者のアプローチだと、既存データベース管理システムの変更に加えて当該システム上で動作しているアプリケーションプログラムの変更が必要になるかも知れない。

著者らは、後者のアプローチにより、既存のデータベース管理システム及び既存のアプリケーションプログラムを全く変更することなく、一貫性のある分散アクセス可能なマルチデータベースシステムの提案をおこなう。

分散データベースシステムにおける更新処理のための問題としては、グローバルトランザクションからサブトランザクションの生成及び実行、グローバルトランザクション及びサブトランザクションのアトミック性の維持、ローカルサイトでのサブトランザクションの実行順序の保証、グローバルデッドロックに対する制御方法・等があるが[4]、著者らは、マルチデータベース実現のためにグローバルデッドロックを回避することにより、ローカルサイトでのサブトランザクションの実行順序を保証し、一貫性を保つ方式を提案する。第2章では、マルチデータベースのモデル及びシステム構成を示し、第3章で当該システムで問題となるグローバルデッドロックについて記述する。第4章では、マルチデータベースにおけるグローバルデッドロックの回避方式の提案を行い、第5章で当該方式に基づいた一貫性制御方式について述べる。第6章では、当該方式の一般化を行なう。

2. マルチデータベースモデル

分散データベースシステムと言うとき、一般に我々は2つのシステム—同機種同種システムから構成されるホモジニアスな分散データベースシステムと異機種異種システムから構成されるヘテロジニアスな分散データベースシステム—を想定する。ホモジニアスな分散データベースシステムは、トップダウン的に構築される分散データベースシステムで、その威力を処理コスト、応答性、信頼性・等の側面に発揮することができる。この背景としては、通信コストの削減、資源の効果的な有効利用・等がある。一方、ヘテロジニアスな分散データベースシステムの背景としては、既に作成され運用されている種類の異なった目的により作成されているデータベースを相互に接続し、複数のデータベースアクセスを行いたいと言う要求がある[10, 11]。このようなヘテロジニアスな分散データベースシステムで重要なことは、複数のデータベースに対して分散アクセスできることに加えて、既存のデータベース管理システム上のアプリケーションプログラムを変更することなくそのまま運用でき、自律性を損なわないことである。このようなヘテロジニアスな環境にお

いて当該目的を達成する分散データベースシステムをマルチデータベースと呼び、そのモデルを図-1に示す。又、マルチデータベースシステムのシステム構成を図-2に示す。図-1に示すように、既存のデータベース管理システムは、既存のアプローチに加えて、マルチデータベースを通してのアクセスに対しても一貫性を維持しなければならない。図-2は、マルチデータベース実現のうち特に本論文の提案の中心となるトランザクション処理に焦点を当てて図示化したものである。図中のTMは、トランザクション管理部を意味し、TCはトランザクション制御部を意味する。

3. マルチデータベースの問題点

図-3は、2つのサイトを想定し、各サイト上にローカルデータベース管理システムとマルチデータベースシステムを配置した場合の、トランザクションの実行順序を示す。同図では、データベースリソースとして、a、b、c、dを想定し、サイト1にデータベースリソースa、bが存在し、サイト2にデータベースリソースc、dが存在していることを示している。トランザクションとしては、L1、L2、G1、G2を想定し、L1はサイト1に存在しマルチデータベースシステムを通さずに直接当該サイトのローカルデータベース管理システムをアクセスするトランザクションで{r3(a) r3(b) w3(b) w3(a) : rはreadを意味し、wはwriteを意味し、a、bはそれぞれアクセスの対象となるリソースを示すものとする。又、r/wの後の数値は、各トランザクションを識別するための仮の添え字である}で示される一連の処理を行うものとする。L2は、サイト2に存在しマルチデータベースシステムを通さずに直接当該サイトのローカルデータベース管理システムをアクセスするトランザクションで{r4(c) r4(d) w4(d) w4(c)}で示される一連の処理を行うものとする。G1、G2は、各サイトに存在する分散データベースシステムを通してのグローバルなトランザクションでそれぞれ{r1(a) w1(d)}, {r2(c) w2(b)}で示される一連の処理を行うものとする。当該環境での上記トランザクションの同時実行時の各サイトでのローカルデータベース管理システムのスケジューラの実行結果を次のように仮定する。Scheduler S1はサイト1のDBMSの実行スケジュール、Scheduler S2はサイト2のDBMSの実行スケジュールとする。

Scheduler S1:r1(a)r3(a)r3(b)w3(b)w3(a)w2(b)

Scheduler S2:r2(c)r4(c)r4(d)w4(d)w4(c)w1(d)

各々のDBMSは、MRSWに基づくロッキングを基本としたデッドロック検出型のデータベース管理システムであるとする。当該スケジュールを実行すると、L1のw3(a)がG1のr1(a)と競合状態となりL1が待ち状態となる。同様に、サイト2でのスケジュールを実行すると、L2がG2と競合を起こし、待ち状態となる。本状態をサイトに閉じないグローバルな視点からとらえると、図-4に示すような、G1がL2にブロックされ、L2がG2にブロックされ、G2がL1にブロックされ、L1がG1にブロックされる巡回ブロック状態(グローバルデッドロック状態)となり各トランザクションは無限待ちの状態となる。

このようなグローバルデッドロック状態の回避方法としては、トランザクションの巡回有向グラフをなんらかの方法により構成し、当該グラフの巡回性診断を行うことによりデッドロックを回避する方法が一般に良く知られている[14]。著者らは、巡回有向グラフ作成のための通信オーバーヘッドの削減及び当該プロトコルのヘテロジニアスシステム間での統一性の問題をなくすために、デッドロックをマルチデータベースシステムで回避する制御方式を考案した。

4. グローバルデッドロック制御方式

本章では、既存のローカルデータベース管理システムを全く変更することなく、グローバルデッドロックを回避するための制御方式について記述する。

本方式は、マルチデータベースシステム中にトランザクションカウンタ (TC) を設定し、マルチデータベースシステムに入ってくるそれぞれのトランザクションを TC の値に応じて編集し、ローカルデータベース管理システムに対して、トランザクション処理を依頼することによりグローバルデッドロックを回避する方法である。

以下に、TC の制御方式と当該サイトに存在するローカルデータベース管理システムへのマルチデータベースによるトランザクション処理の依頼方式について記述する。

4. 1. TC (Transaction Counter) の制御方式

TC は、マルチデータベースシステム中で処理されているトランザクション数を示す数値であり、初期状態では 0 が設定されている。当該マルチデータベースシステムに対してトランザクション開始要求があると TC の値がプラス 1 され、トランザクション終了要求があると、TC の値はマイナス 1 される。

4. 2. ローカルデータベース管理システムへのトランザクション依頼方式

マルチデータベースに対するトランザクション開始要求時 (ST_i : マルチデータベースに対する要求トランザクションを ST_i とする)、TC の値が 0 ならば、新規トランザクション (GT : Global Transaction) を発生させ、TC をプラス 1 し、当該新規トランザクションとして、トランザクション処理の実行をローカルデータベース管理システムに要求する。もし、トランザクション開始要求時、TC の値が 0 以外ならば、ローカルデータベース管理システムに対する新規トランザクションを発生させないで、TC をプラス 1 し、当該進行中のトランザクション GT としてローカルデータベース管理システムにアクセス要求を行う。トランザクション終了要求時は、TC をマイナス 1 し、もし TC の値が 0 ならば当該進行中のトランザクション GT の終了要求をローカルデータベース管理システムに対して行う。もし、TC をマイナス 1 した時点で TC の値が 0 以外ならば、トランザクション終了要求があってもローカルデータベース管理システムへのトランザクション終了要求は行わない。またデータベース管理システムからデッドロック通知があると、TC の値を 0 に設定し、GT トランザクションとして処理している全てのトランザクション ($\sum ST_i$) に対して当該通知を行う。

5. トランザクション処理制御方式

5. 1. トランザクション管理方式

マルチデータベースシステムにおけるトランザクション管理は {ST_i : マルチデータベースシステムに対してトランザクション処理要求を行うトランザクションの集まり} と {GT : ST によるトランザクション要求を契機としてマルチデータベースシステム中で作成され、ローカルデータベース管理システムに対して発行される唯一つのトランザクション} と TC から構成される。トランザクション管理テーブルは一つの TC、一つの GT、及び複数の ST_i から構成されている。上述の TC に関する値は本テーブルの TC 部に格納され、作成された GT は本テーブルの GT 部に格納される。また、ST_i の集まり

は本テーブルのST部に格納される。

5. 2. トランザクション制御方式

上記方式の適用による、グローバルデッドロック回避による、一貫性制御方式を以下に記述する。

上記方式を図-3の環境に適用すると、図-5のようになる。同図は、トランザクションG1によりマルチデータベースシステム中にGT1が発生し、G1要求に基づきGT1トランザクションとして、サイト1に存在するデータベースリソースaが確保され、L1によるw3(a)で待ち状態となった後、G2要求によるGT1トランザクションとしてのw2(b)要求をローカルデータベース管理システムに要求した図である。同様に同図はサイト2でのトランザクション処理形態図を示す。当該環境でのスケジューラの実行シーケンスを以下に示す。

Transaction	GT1	L1	GT1	GT1/GT2/L1/L2はアクセスを行うTr。
Scheduler S1:	r1(a)r3(a)r3(b)w3(b)w3(a)w2(b)			
Scheduler S2:	r2(c)r4(c)r4(d)w4(d)w4(c)w1(d)			
Transaction	GT2	L2	GT2	

図-6のAにサイト1でのローカルデータベース管理システムのリソース競合グラフを示し、Bにサイト2でのローカルデータベース管理システムのリソース競合グラフを示す。各ローカルデータベース管理システムは、デッドロック検出機能を持っていると仮定し、図-6のリソース競合グラフから明らかなように、サイト1、2でデッドロックが検出されサイト1ではGTまたはL1のいずれか、サイト2でもGTまたはL2のいずれかがロールバックされる。ロールバックの方法によっては、以下に示す4つのスケジュールの可能性があるが、いずれのトランザクションが各サイトでロールバックされようとも、グローバルデッドロックは回避され一貫性が維持されていることがわかる。

ケース1：GT1とGT2がロールバックされた場合

Scheduler S1:r3(a)r3(b)w3(b)w3(a)

Scheduler S2:r4(c)r4(d)w4(d)w4(c)

ケース2：LT1とGT2がロールバックされた場合

Scheduler S1:トランザクションなし

Scheduler S2:r4(c)r4(d)w4(d)w4(c)

ケース3：GT1とLT2がロールバックされた場合

Scheduler S1:r3(a)r3(b)w3(b)w3(a)

Scheduler S2:トランザクションなし

ケース4：LT1とLT2がロールバックされた場合

Scheduler S1:r1(a)w2(b)

Scheduler S2:r2(c)w1(d)

ここでは、各トランザクションは明示的にStart-Tra/End-Traを発行し、Rollback通知はEnd-Traの処理を含むとした場合のプロトコルシーケンスを図-7に示す。また他サイトへのトランザクション処理要求は、当該サイトのマルチデータベースで作成されたGTiトランザクションの要求として行われるものとする。

図-7のケース1は、サイト1でGT1、サイト2でGT2に対してそれぞれデッドロック通知が行われた場合であり、サイト1のマルチデータベースシステムは当該通知を受け取ると、トランザクションテーブル中のTCを0にし、ローカルデータベース管理シス

テムに対してG T 1 トランザクションのRollbackを行うとともに、G 1 及びG T 2 に対してRollback通知をおこなう。サイト2のマルチデータベースシステムも同様にTCに0を設定し、G T 2、G 2 及びG T 2 下のG T 1 に対してRollback通知を行う。結果的に、サイト1ではL 1、サイト2ではL 2 が実行されたことになる。

図-7のケース2は、L 1 とG T 2 に対してデッドロック通知が行われた場合で、サイト2ではG T 2、G 2、G T 2 下のG T 1 がRollbackされる。サイト1のマルチデータベースシステムはG T 1 のRollback通知を受信すると当該トランザクションのRollbackを行い、結果的にサイト1では、G 1、L 1 の両トランザクションがRollbackされ何も実行されなく、サイト2ではL 2 が実行されたことになる。

6. 一般化

本章では、前章で提案したグローバルデッドロック回避方式の一般化を行う。

[定理]

グローバルデッドロックは、巡回有向グラフで表わされる。当該グラフを有限状態機械 (Finite State Automaton) とみなし、当該グラフと等価 (Equivalent) な有向グラフを考える。もし、当該等価グラフが複数のサイクルを含み、かつ各サイトに対応づけられる状態に閉じたサイクルが存在すれば、トランザクションを基本としたグローバルデッドロックは、(サイトに閉じた処理として) 回避することができる。

巡回有向グラフで示されるグローバルデッドロックの状態を図-8 (a) に示す。本グラフからG 1 とG 2 を同一ノードとする等価グラフを構成すると図-8 (b) のようになる。

図-8 (b) においては、2つのサイクル ($L 1 \leftrightarrow G 1 / G 2$ 、 $L 2 \leftrightarrow G 1 / G 2$) が存在し、かつそれぞれのサイクルはサイトに閉じたサイクルである。[L 1 とG 1 から構成されるサイクルは、サイト1に閉じ、L 2 とG 2 から構成されるサイクルはサイト2に閉じている。] よって、上記定理によりグローバルデッドロックは、G 1 とG 2 を対等とすることにより、回避することができる。

具体的に本定理をトランザクション処理に適用すると次のことを意味する。

G 1 とG 2 を対等に見るとは、G 2 トランザクションとしての要求をG 1 トランザクションとみなすことであり、G 1 トランザクションとしての要求をG 2 トランザクションとしてみることである。これにより、サイトローカルなデッドロックが発生するため、もし、各サイトにトランザクション処理に対しデッドロック検出機構があれば、グローバルデッドロックをさけることが可能となる。

7. おわりに

本論文では、既存のデータベース管理システム及び当該システム上のアプリケーションプログラムを全く変更することなく、マルチデータベースとしての分散アクセスができ、グローバルデッドロックを回避することにより分散トランザクション処理としての一貫性を維持できる制御方式について記述した。また、当該方式の一般化を行った。

本方式により、巡回有向グラフ作成のための通信オーバーヘッドが削減され、ローカルデータベース管理システムと共存した効果的なマルチデータベースシステムの構築が可能になる。

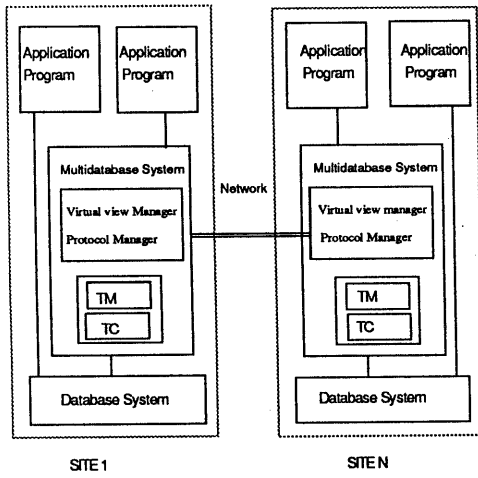


Figure 2: System Structure

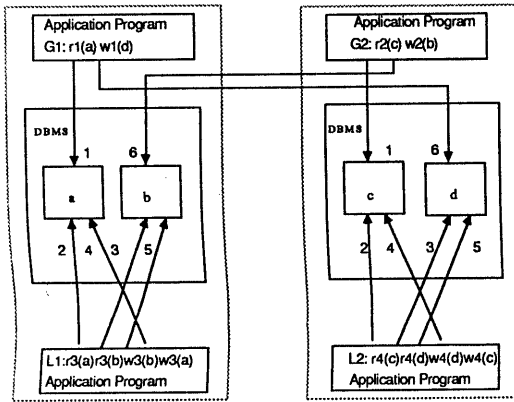


Figure 3: Access Sequence Example

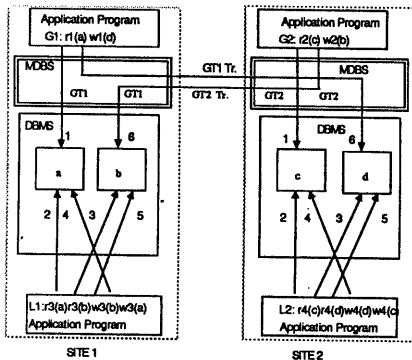


Figure 5: Access Sequence Example

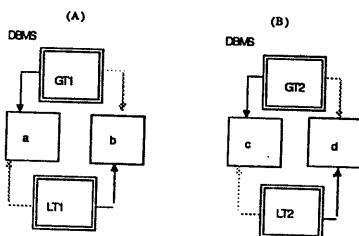


Figure 6: Directed Graph

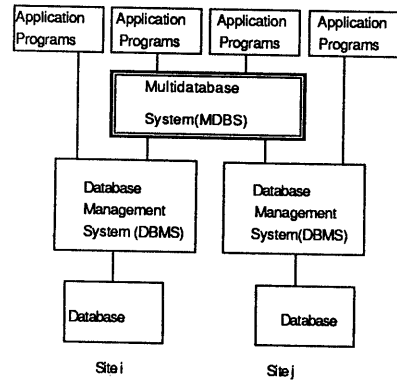


Figure 1: Multidatabase model

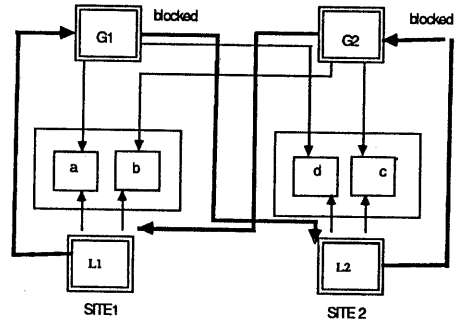


Figure 4: Global Deadlock Situation

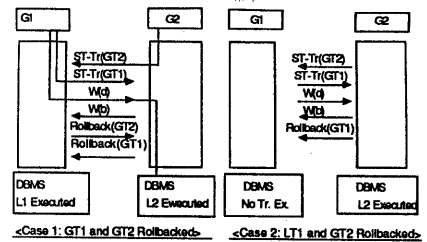


Figure 7: Protocol Sequence Example

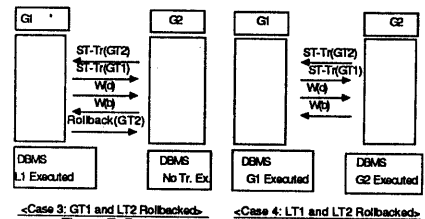


Figure 8: Global Deadlock Avoidance Graph

8. 参考文献

- [1] ISO/TC97/SC21 N1926:IPC-OSI-Remote Database Access(July 1987)
- [2] ISO/JTC 1/SC 21 WG3:Information Systems-Generic Remote Database Access Service and Protocol.Sep 1988.
- [3] Y. Kambayashi, Integration of Different Concurrency Control Mechanisms in Heterogeneous Distributed Databases, ISIIS, 1988. pp313-320.
- [4] Y. Breitbart, A. Silberschts, Multidatabase Update Issues, SIGMOD 88, pp135-142.
- [5] R. Alonso, H. Garcia-Molina, K. Salem, Concurrency Control and Recovery for Global Procedures in Federated Database Systems, IEEE Data Engineering, September, 1987.
- [6] Y. Breitbart, A. Silberschatz, G. Thompson, An Update Mechanism for Multidatabase Systems, IEEE Data Engineering, September, 1987, pp12-18.
- [7] A. K. Elmagarmid, Y. Leu, An Optimistic Concurrency Control Algorithm for Heterogeneous Distributed Database Systems, IEEE Data Engineering, September, 1987, pp26-32
- [8] W. Litwin, Implicit Joins in the Multidatabase System MRDSM, COMPSAC85, pp495-504.
- [9] W. Litwin, A. Abdellatif, Multidatabase Interoperability, IEEE Computer, 1986, pp10-18.
- [10] J. M. Smith, P. A. Bernstein, U. Daye, etc., Multibase-integrating heterogeneous distributed database systems, NCC81, pp487-499.
- [11] A. Ferrier, C. Stangret, Heterogeneity in the Distributed Database Management System SIRUS-DELTA, VLDB, 1982, pp45-53.
- [12] T. Landers, R. Rosenberg, An Overview of MULTIBASE, CCA Tech. Rep.
- [13] J. F. Pons, J. F. Vilarem, Mixed Concurrency Control: Dealing with heterogeneity in distributed database systems, Proc., 14th VLDB., 1988, pp455-456.
- [14] E. Knapp, Deadlock Detection in Distributed Databases, ACM Computing Surveys, Vol. 19, No. 4, Dec., 1987, pp303-328.
- [15] R. Hull, R. King, Semantic Database Modeling: Surveys, Applications, and Research Issues, ACM Computing Surveys, Vol. 19, No. 3, Sep., 1987, pp201-260.
- [16] L. Tucherman, A. L. Furtado, M. A. Casanova, A tool for modular database design, VLDB., 1985, pp436-447.
- [17] D. S. Batory, T. Y. Leung, T. E. Wise, Implementation Concepts for an Extensible Data Model and Data Language, ACM TODS., Vol. 13, No. 3, Sep., 1988, pp231-262.
- [18] C. Batini, M. Lenzerini, S. B. Navathe, A Comparative Analysis of Methodologies for Database Schema Integration, ACM Computing Surveys, Vol. 18, No. 4, Dec. 1986.
- [19] M. Rusinkiewicz, Panel on Multi-Database Systems, SIGMOD88.
- [20] A. P. Sheth, J. A. Larson, Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases, ACM Computing Surveys, Vol. 22, No. 3, Sep. 1990, pp183-236.