データベースシステム　104－36
（１９９５．　７．　２０）

並行処理制御における２相施錠方式の改良

春原　典彦　　芝井　豊*　　宮崎　収兄

{sunohara,miyazaki}@mz.cs.it-chiba.ac.jp

千葉工業大学　情報工学科

〒275　千葉県習志野市津田沼　2-17-1

本稿では、データベースにおける並行処理制御について述べる。一般的によく使われている２相施錠方式は、条件によってデッドロックが生じ性能が低下する問題がある。保守的スケジューラを導入した２相施錠方式や、楽観的なスケジューラなどの他の方式の方がいくつかの応用には適切だという議論がある。それらはデッドロック問題はないが、トランザクションのread setとwrite setをあらかじめ宣言する必要があり柔軟さに欠ける。本稿ではどのような状態でデッドロックになるかを検討し、２相施錠方式の簡単な改良を検討する。提案方式は格上げ制限２相施錠方式(ＮＵ２ＰＬ)で、２相施錠方式におけるデッドロックを低減し、またあらかじめデータセットの宣言を行う必要がないため保守的スケジューラより適用範囲が広い。

Improving Two-Phase Locking for Database Concurrency Control

Norihiko Sunohara , Yutaka Shibai * , Nobuyosi Miyazaki

Department of Computer Science, Chiba Institute of Technology
2-17-1 Tsudanuma Narashino Chiba 275 Japan

This paper discusses concurrency control mechanisms for database.The most widely used method, two-phase locking, may have performance problem due to the deadlock under certain conditions.Other methods including cautious two-phase locking and optimistic schedulers are discussed more adequate for some applications.They do not have deadlock problems, but they are not flexible because they require predeclarations of read sets and write sets of transactions.We discuss conditions under whic two-phase locking may result in deadlock, and propose its simpl improvement.The proposed method, non-upgrading two-phase locking, reduces the probability of deadlock, and it is more flexible than the cautious schedulers because it does not require predeclarations.

---

*現在　株式会社ラック

1　はじめに

異なるトランザクションからの同時的な要求に対して、データベースシステムにはトランザクションの実行順序を制御するための、並行処理制御(concurrency control)が必要不可欠である。この制御には、データの首尾一貫性を保証することや、システム全体の利用効率を高めることが要求されている。現在までに並行処理制御方式として、施錠をかける方式やスケジューラで制御する方式など、いくつかの方式が提案されているが、現在主として２相施錠方式(two-phase locking)が使われている[EGLT76,BHG87,西尾91]。

ここ数年、多様化されたデータの格納を目的としたオブジェクト指向データベースシステム(Object-Oriented Database system)などが注目されている。これは、平坦なデータ構造によるモデル化では足りず、より高度なモデル化が必要になってきたのである。この様な新しい応用では、２相施錠方式よりも競合保存直列可能スケジューラ(Conflict Preserving Serializable)の方が有効であるという議論があり[NTI89,谷口93]、また、オンライントランザクションに関しては２相施錠方式は適していないという指摘もある[NTI89]。しかし、どちらの方式に於いても高負荷状態で参照された後、更新されるオブジェクト集合の割合が高い場合ではトランザクション完了率が低くなってしまうため、実用的なレベルには問題がある。この問題には保守的スケジューラ(Cautious scheduler)の導入が有効であると言われている[NTI89]。しかし２相施錠方式の保守的スケジューラでは、トランザクション中でアクセスするすべての対象を最初にすべて施錠する必要があり、実際の応用に適用するのは困難な場合が多い。

そこで本研究では、従来もっとも多く使われている２相施錠方式に従来の保守的スケジューラより弱い制限を導入し改良した、新たな施錠方式を提案した。さらに、それを拡張することによって多粒度施錠にも適用できるようにした。

本稿では、２節で２相施錠方式の問題点を述べるとともに新たな施錠方式の提案を行う。そして、３節では多粒度施錠への適用を考え、４節では他方式との比較・検討を行った。

2　２相施錠方式の問題点とその改良案

現在のデータベースシステムでは、ほとんどが２相施錠方式（以下２ＰＬとする）を採用している。２ＰＬとは以下のような方式である[EGLT76]。

（1）トランザクションはデータにアクセスする前にデータに対し施錠をしなければならない。
（2）トランザクションは、施錠部分とそれに続く解錠部分の２相に分かれ、解錠操作が１つでも行われるとそれ以後の施錠操作は一切できない。

２ＰＬでは、並列処理されるトランザクションの各々がこの方式に従えば直列可能性を保証できるという特徴を持っている。この方式には、以下のような問題点が指摘されている。

高負荷状態ではデッドロックになる確率が高くなり、トランザクション完了率が低くなってしまう。つまり、実行途中のトランザクションを棄却しなければならないという問題が起こるのである。この問題はオンライントランザクションやオブジェクト指向データベースなどの新しいシステムでより重大になると言われている[NTI89,谷口94]。

この問題点を解決するために、２ＰＬの改良版であるＣ２ＰＬ（Conservative two-phase locking）が提案されている[BHG87]。Ｃ２ＰＬとは以下のような方式である。

２ＰＬにおいて全データに対する施錠操作を、最初に施錠するときにすべてまとめて行う。つまり、デッドロックが起こらないよう２ＰＬよりロックの制限を強くする。

この方式ではデッドロックは起こらないが、次のような問題がある。

データを施錠している時間が２ＰＬと比べ、長くなるため並行性が低くなる。また、事前に操作するデータ項目がすべてわからない場合があるが、Ｃ２ＰＬでは途中で施錠することができないので、多くの応用において実用的ではない。

一方、実際の応用では２ＰＬのデッドロックが大きな問題になっているという報告はあまりない。これは、次のような理由が考えられる。

（１）実際の応用では、デッドロックが問題になるほど高負荷ではない。
（２）デッドロックにならないようにユーザが工夫して使っている。

（１）の場合、応用の多様化により将来は大きな問題になる可能性がある。（２）の場合、例えばデッドロックの危険性の高いトランザクションのみはＣ２ＰＬと同様に必要な対象を最初にすべて施錠する方法などが考えられる。しかし、その都度適した方法を考えるのではなくどのような方法が最適なのか明確化させることが重要である。いずれにしろ、２ＰＬを改良しデッドロックを軽減する、より柔軟な方式を研究することが重要である。

まず、２ＰＬでデッドロックが起きるのはどのような場合かを考える。ここで簡単化のために２つのトランザクションによるデッドロックを考える。各トランザクションは１つの対象をｒｅａｄし、その後１つの対象をｗｒｉｔｅする２段階の処理を行う。その時、以下のような２通りの場合にデッドロックが起こる。

タイプ（１）Ｔ１：ｒ１［ｘ］→　ｗ１［ｘ］
　　Ｔ２：ｒ２［ｘ］→　ｗ２［ｘ］

タイプ（２）Ｔ１：ｒ１［ｘ］→　ｗ１［ｙ］
　　Ｔ２：ｒ２［ｙ］→　ｗ２［ｘ］

この中でも（１）またはその変形は実際のトランザクション処理でも多く現れると言われている。この様なデッドロックは、多く存在するデータの中から目的のデータを選び出しその後にアップデートするような場合に起こる[BHG87,KM94]。トランザクションのｒｅａｄ対象とｗｒｉｔｅ対象は、何らかの関連性を持っているため、両者に重複のある可能性は高い。そこで、タイプ（１）のデッドロックを防ぐにはどのようにすればよいかを考える。この様なデッドロックはあるデータをｒｅａｄした後ｗｒｉｔｅする時、最初にｒｅａｄ施錠をかけ後からｗｒｉｔｅ施錠に変更するために起きてしまう。従ってデッドロックを防ぐには最初からｗｒｉｔｅ施錠をかければよい。この考えははＣ２ＰＬと似ているが、Ｃ２ＰＬのようにアクセスするデータをあらかじめ宣言するのではなく、最初にアクセスする時点で将来のアクセスを考慮して施錠を行う点が異なる。一般にあるデータをｒｅａｄする時それをｗｒｉｔｅするかどうかの判断を行うのは、トランザクションの開始時にどのデータにアクセスするかを判断するよりは容易である。この考えを具体化するために以下の格上げ制限（non-upgrading）を導入する。

同一の対象について共有施錠を行った場合は、後で排他施錠を行えないように制限する。

格上げ制限を導入した２ＰＬをＮＵ２ＰＬ（non-upgrading two-phase locking）と呼ぶ。

この方式においてトランザクションは、後でｗｒｉｔｅするデータをｒｅａｄする時には共有施錠でなく排他施錠をする必要がある。以下に例を示す。

例１）
・トランザクションが
　ｒ［Ａ、Ｂ、Ｃ］　→　ｗ［Ｄ］
　　のような場合の施錠方法は、
　Ｓ［Ａ、Ｂ、Ｃ］　→　Ｘ［Ｄ］
　　のように施錠する。（２ＰＬと同じ）

・トランザクションが
　ｒ［Ａ、Ｂ、Ｃ］　→　ｗ［Ｃ］
　　のような場合の施錠方法は、
　Ｓ［Ａ、Ｂ］Ｘ［Ｃ］
のように参照する前にまとめて施錠してしまう。

この方式をとることにより、先ほどのタイプ（１）のようなデッドロックを防ぐことが可能になる。これは、高負荷状態でのトランザクション完了率と並行性の両者を向上させるために、２ＰＬのよいところをそのままに、Ｃ２ＰＬよりもロックの制限を柔軟にした物である。

3　多粒度施錠への適用

データベースではデータがリレーション、タプルのように階層型構造になっている。この様に階層化されたデータに対しデータ項目毎の施錠方式をとると、処理範囲が広範囲にわたるとき施錠を細かくかける必要があり、処理に時間がかかってしまう。その問題を解決するために、２相施錠方式の多粒度化として警告施錠（Intention lock）の概念を用いた方法が使われている[GLP75]。警告施錠とは、例えばあるオブジェクトに対して処理を行おうとした場合それに対して排他施錠をし、その上の階層については排他モードの警告施錠をかける仕組みである。つまり、あるオブジェクトに施錠をかける時その下の階層には同じモードの錠が暗黙にかかり、上にあるすべての階層に対しては同じモードの警告施錠をかけるのである。上位レベルの警告施錠はアプリケーションが陽に要求するようにもできるが、ＤＢＭＳが自動的に行うことも可能である。後者の場合はアプリケーションの負担が軽減できる。表１は、警告施錠を用いた方式において、２つのトランザクションが並行して実行できるかどうかの多粒度施錠の両立表を示し、表２には、同一対象に対して２度以上施錠が要求されると施錠モードがどのように変化するかをまとめた変換表を示す。また表２から施錠タイプの強さは図１のようになる。

多粒度施錠に我々の提案した方式を拡張するにはどうすべきかについて考える。まず、ＮＵ２ＰＬの概念を導入しても施錠のタイプは従来と同じ物を用いる。従って、多粒度施錠の両立表は表１と同じ物となる。施錠の格上げ制限は変換表で施錠タイプの変更が行えないようにすることによって導入する。つまり、ＮＵ２ＰＬの導入は変換表の再設計によって行う。これを再設計するに当たり、施錠タイプが増えないため、基本方式の場合と異なりいくつかの選択肢がある。どの選択肢をとるかにより格上げ制限の強さが変化するが、本稿ではできるだけ制限を少なくする方針をとった。まず以下に基本的な原則を述べる。

＜原則＞

（１）同じ対象について、r→w へ陽の変更は行わない。（２節で提案した、ＮＵ２ＰＬの基本原則）

（２）（１）以外の制限は、できるだけ行わない。具体的には、階層構造をできるだけユーザが意識しないでもよいようにする。（例えば、施錠した対象の上位の対象へ施錠を後からかけることは可能とする。）

表２　２ＰＬ変換表
2PL Conversion Table

Requested lock type \ Old lock type	r	w	ir	iw	riw
r	r	w	r	riw	riw
w	w	w	w	w	w
ir	r	w	ir	iw	riw
iw	riw	w	iw	iw	riw
riw	riw	w	riw	riw	riw

表１　多粒度施錠の両立表
A Compatibility Matrix for Multigranularity Locking

	r	w	ir	iw	riw
r	y	n	y	n	n
w	n	n	n	n	n
ir	y	n	y	y	y
iw	n	n	y	y	n
riw	n	n	y	n	n

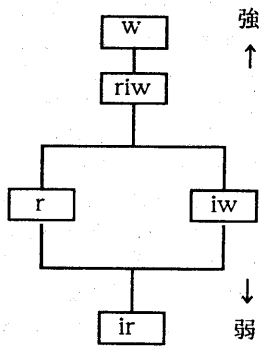


図１　施錠の強さ

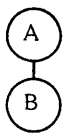
ＮＵ２ＰＬ Conversion Tableについて、以下に検討結果を述べる。ただし、同じロック同士は当然のことながらロックタイプは変化しない。また、Old lock Typeの方が強い場合も変化しないので省略する。

（１）＜Old lock Typeが　ｒの場合＞

・ｒ→ｗ

これは原則（１）にも書いてあるように、ＮＵ２ＰＬの基本原則なので実行できないようにする。

・ｒ→ｉｗ



左の図において、Ａにｒ施錠した後ｉｗ施錠をかけるというのは、Ａの下にある対象Ｂをｗをする事である。これは、上記原則だけでは決定できないが２節で述べたようにデッドロックが起きやすい形態なので不可とする。

・ｒ→ｒｉｗ

上記と同様に実行できないようにする。

（２）＜Old lock Typeが　ｗの場合＞

ｗは、もっとも強いロックタイプなのですべてｗである。

（３）＜Old lock Typeが　ｉｒの場合＞

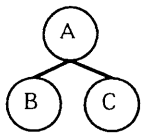
・ｉｒ→ｒ

原則（２）に従いｒとする。

・ｉｒ→ｗ

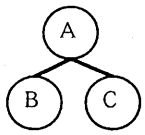
原則（２）から実行できるようにする。但しこれによりデッドロックが起きやすいのなら不可とする案もある。

・ｉｒ→ｉｗ



これは左の図でＢをｒしＣをｗする時のＡの施錠の変化で、もしこれを制限した場合あらかじめｒとｗの階層構造の中で把握しておかなければならない。つまり、どこから先でおなじ階層となっているかを知らなければならない。従って、これは原則（２）から　ｉｗにする。

・ｉｒ→ｒｉｗ



左の図において、Ｂをｒした後にＡをｒしてその下のＣにｗする事である。ｉｒ→ｗに準じて可能とする。

（４）＜Old lock Typeが　ｉｗの場合＞

・ｉｗ→ｒ

原則（２）からｒｉｗにする。

・ｉｗ→ｗ

原則（２）からｗにする。

・ｉｗ→ｒｉｗ

原則（２）からｒｉｗにする。

（５）＜Old lock Typeが　riwの場合＞

・ｒｉｗ→ｗ

原則（１）からは不可となる。しかし、最初の段階で１部を書く予定だった物が実際には広範囲に変更する場合もあるので今回はｗとする。

上記の結果得られたのが下の表３にある、ＮＵ２ＰＬ Convertion Tableである。

表３　ＮＵ２ＰＬ変換表
ＮＵ２ＰＬConversion Table

Requested lock type \ Old lock type	r	w	ir	iw	riw
r	r	w	r	riw	riw
w	--	w	w	w	w
ir	r	w	ir	iw	riw
iw	--	w	iw	iw	riw
riw	--	w	riw	riw	riw

結果としてｒからの格上げのみを制限する方式になった。もっと強い制限を導入することも可能であるが、頻繁に起こらないような変化はデッドロックを起こす可能性も少ないので可としても影響は小さい。

多粒度ＮＵ２ＰＬでは、ｒｅａｄする段階で上位にｒｉｗ施錠をし、下位にｗ施錠をするのが原則である。しかし、実際の応用ではｒｅａｄ結果によりどの対象をｗｒｉｔｅするかを決定することも多い。このような場合は、最初は上位にｒｉｗ施錠をし、ｗｒｉｔｅする時に下位の対象にｗ施錠をすることも可能である。このようにしてｗｒｉｔｅ対象が前もって決められないような場合にも、並行性を大きく損なうことなくデッドロックを減少させることができる。

4　他方式との比較

本稿で提案したＮＵ２ＰＬと従来の方式を高負荷でアクセス競合する可能性が高い条件において以下の項目を比較する。

（1）トランザクションの並行性（待ち時間）
（2）トランザクション完了率（デッドロック問題を含む）

以下で示すOVL（オーバーラップ率）とは、簡単な２段階のトランザクションで考えるとｒｅａｄした後にｗｒｉｔｅする対象が前のｒｅａｄとオーバーラップする率で、０～１で表される。０の時ＮＵ２ＰＬではデッドロックを防ぐことはできないが１の時は完全にデッドロックを防ぐことができる。

（1）トランザクションの並行性
・２ＰＬでは、デッドロックにならないケースの待ち時間については比較的少なくてすむ。
・Ｃ２ＰＬでは、待ち時間が長くなり並行性が低くなる。
・ＣＰＳでは、直列可能性が保証されていれば待ち時間はない。
・ＮＵ２ＰＬでは、若干２ＰＬより待ち時間が増えるがＣ２ＰＬほどでない。

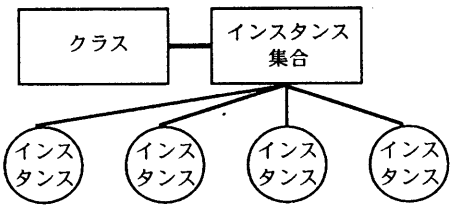
（2）トランザクション完了率
・２ＰＬでは、デッドロックになる可能性が高く完了率が低くなる。
・Ｃ２ＰＬでは、デッドロックがないため完了率は100％である。
・ＣＰＳでは、直列可能性が保証されない可能性が高く完了率が低くなる。
・ＮＵ２ＰＬでは、デッドロックが起こる可能性が０ではないものの２ＰＬに比べトランザクション完了率は高い。特にOVL（オーバーラップ率）が高くなるとデッドロックが発生しやすくなると言われているが[谷口93]、OVLが高いほど本方式ではデッドロックを防止する効果が大きい。

システムスループット（単位時間に完了する平均トランザクション数）は、並行性と完了率が高いほど大きくなるが、この両者を両立させるのは困難である。しかし、[谷口93]などによれば高負荷状態では２ＰＬとＣＰＳは完了率が低くなるに従ってスループットも低下する。従って、提案方式のデッドロック防止によりスループットが向上すると考えられる。Ｃ２ＰＬはデッドロックがなくなるので高負荷状態で適している可能性があるが、最初にすべて施錠する方式なので適用範囲が限定される。

オブジェクト指向データベース対しＣＰＳを改良したＤＣＰＳ（遅延式ＣＰＳ）が提案されている[谷口93]。オブジェクト指向データベースではIT（インスタンス操作トランザクション）及びCT（クラス操作トランザクション）の両者が混在した場合、ITとITまたはCTとITの競合が多く起こることが考えられる。ＤＣＰＳの制御は参照操作段階の競合から将来の競合を予期して実行を制御しトランザクション完了率を向上させている。この方式では、CTに着目して制御を行っているためITとCTの競合に対しては効果があるが、ITとITの競合に対して効果がない。それに対しＮＵ２ＰＬはITとCTの競合に対してＤＣＰＳと同様な効果があるとともに、ITとITの競合に対しても効果がある。図４のようにクラスとそのクラスのインスタンスがあり、インスタンス集合がインスタンスの上位になるように階層化する。インスタンスがタプルに、インスタンス集合が関係に対応すると考えれば、この階層は関係データベースでの階層と同様なので自然な階層化である。クラスのｒｅａｄはインスタンスに影響しないのでインスタンスに施錠する必要はない。これに対し、クラスのｗｒｉｔｅはすべてのインスタンスに影響するのでインスタンス集合にもｗｒｉｔｅ

施錠する必要がある。従って、CTにＮＵ２ＰＬを適用するとクラスをｒｅａｄする前にクラスとそれに対するインスタンス集合にｗｒｉｔｅ施錠をすることになる。この様にすれば、CTとITのデッドロックが防止できＤＣＰＳと同様な効果がある。ITとITのデッドロックは多粒度施錠を行わなくても低減できる。また、ＤＣＰＳの制御はクラスのｒｅａｄの段階で競合の可能性を判定するため、ｗｒｉｔｅしないトランザクションも遅延させてしまい、性能低下を起こす場合があるがＮＵ２ＰＬにはその心配がない。さらに、拡張性に関してもＤＣＰＳの方は特定のトランザクションに対して特殊な処理を施しているためクラス階層が複雑な場合などへの拡張が困難である。それに対しＮＵ２ＰＬは多粒度２ＰＬを改良した物であるため、これからの新しい応用に対しても適応性が高いと考えられる。

図４　クラス-インスタンスモデル



5　むすび

本研究では、２相施錠方式を基に新たな施錠方式として施錠モードの制限を導入したＮＵ２ＰＬを提案した。

まず、ＮＵ２ＰＬによって、あるデータを読み出してからアップデートしたり、多数のデータの中から目的のデータを選び出しその後にアップデートするような場合に起きるデッドロックを解消できるようになった。この様な形態は文献[BHG87,KM94]にもあるように、実際のトランザクションで多く使われている。また、Ｃ２ＰＬではデッドロックが起こらないが、途中で施錠ができないので事前に操作するデータ項目がわかっていないと使えないという問題があり実用的ではない。ＮＵ２ＰＬはトランザクション完了率をあげながらロックの制限をＣ２ＰＬより柔軟にしたため適用範囲が広い。

次に、ＮＵ２ＰＬを多粒度施錠に対して従来の多粒度２ＰＬと同様に警告施錠の概念を元に拡張した。多粒度ＮＵ２ＰＬでは、楽観的スケジューラを改良した遅延式ＣＰＳ方式と同様にクラス操作トランザクションとインスタンス操作トランザクションの競合に対して効果があるとともに、インスタンス操作トランザクション同士の競合にも有効な効果がある。また、この方式は元々ある多粒度２ＰＬを改良してあるため、これからの新しい応用に対しても適応性が高いと考えられる。

今後、制御性能の定量的評価を行うとともに、クラス階層や複合オブジェクトへの適用について研究を進めていく予定である。

「参考文献」

[BHG87] Bernstein, P.A., Hadzilacos, V., and Goodman, N., Concurrency Control
and Recovery in Database Systems, Adison-Wesley 1987

[EGLT76] Eswaran, K.P., Gray, J., Lorie, R.A., and Traiger, I.L., The Notions of Consistency and Predicate Locks in a Database System, Commun. ACM, Vol.19, No.11, pp.624-633 (1976)

[Gray79] Gray, J.N., Notes on Data Base Operating Systems, Lecture Note in Computer Science 60 (Operating Systems), pp.393-481, Springer-Verlag, 1979

[GLP75] Gray, J.N., Lorie, R.A., and Putzolu, G.R., Granularity of Locks in a Shared Database, Proc. VLDB, pp.428-451 (1975)

[KM94] Kemper, A. and Moerkotte, G., Object-Oriented Database Management: Applications in Engineering and Computer Science, Prentice Hall (1994)

[NTI89] Nishio, S., Taniguchi, S., and Ibaraki, T., On the Efficinecy of Cautious Schedulers for Database Concurrency Control - Why Insist on Two-Phase Locking?, J. Real-Time Systems, Vol.1, pp177-195, Kluwer

[NTI89] Nishio, S., Taniguchi, S., and Ibaraki, T., On the Efficinecy of Cautious Schedulers for Database Concurrency Control - Why Insist on Two-Phase Locking?, J. Real-Time Systems, Vol.1, pp177-195, Kluwer Academic Publishers, 1989

[西尾91] 西尾章次郎、"オブジェクト指向データベースシステムにおける並行処理制御", 情報処理, Vol32, No.5, pp.540-549 (1991)

[谷口93]谷口伸一　西尾章次郎　久保信也、"オブジェクト指向データベースにおける競合保存直列可能スケジューラの有効性",電子情報通信学会論文誌, Vol.J77-D-I, No.7, pp514-524 (1993)