データベース・システム 62－1
（1987・11・16）

分散データベースのデッドロック解決方式の考察

小林哲二
NTT情報通信処理研究所

データベースの同時実行制御として，ロック方式が広く用いられている．この場合，デッドロックを解決する必要がある．本稿では，分散データベースの同時実行制御にロック方式を用いる場合について，時刻印を利用したデッドロック解決方式を提案する．この方式の特長は，(1)時刻印をデッドロック解決のためにのみ用いているので，既存のシステムへの導入が比較的容易であること，(2)トランザクションのロックの実行順序は，時刻印の順でなくてもよいので，ロックの設定後に時刻印の小さなトランザクションが後から到着した時でも，先に更新を行っているトランザクションのロールバックは不要であること，及び(3)時刻印は，トランザクションにのみ付与し，データ実体には付与しないので，二次記憶のオーバヘッドがないことである．

A DEADLOCK RESOLUTION SCHEME FOR DISTRIBUTED DATABASE SYSTEMS

Tetsuji KOBAYASHI
NTT Communications and Information Processing Laboratories
1-2356, Take, Yokosuka-shi, 238-03 Japan

Locking schemes are widely used for cocurrency control in database systems, and one of the important problems is to resolve deadlocks. In this paper, a deadlock resolution scheme which uses timestamps is proposed for distributed databases. The main merits are as follows: (1) since timestamps are only used for resolving deadlocks, the scheme can be easily introduced to systems which use locking schemes, (2) since the timestamp orders do not mean the execution orders of transactions, no rollback is necessary, and (3) since timestamps are only attached to transactions, there is no storage overhead.

1. まえがき

データベースの同時実行制御の手段としては，一般にロック方式が広く用いられている[4],[6]．この場合，デッドロック[4],[5]を解決することが，必要である．従来，デッドロックを解決する方式[3],[5]として，第一に，デッドロックの検出を行う方式が検討されており，具体的には，システム状態グラフ方式，及び時間監視方式などが，主要な方式である．システム状態グラフ方式は，システム状態グラフの作成により，デッドロックを検出し，デッドロックサイクルに含まれるトランザクションの内の一つを処理中断してから再開することにより，デッドロックを解決する方式であり，システム状態グラフの更新のためのオーバヘッドが大きい．時間監視方式は，タイムアウトによりデッドロックを検出し，タイムアウトとなったトランザクションを処理中断してから再開することにより，デッドロックを解決する方式であり，デッドロック検出までの時間がタイマの設定時間となり，その値は最大の正常処理の時間よりも大きくなければならないので，デッドロック検出までの時間が大きく，且つ再開後もデッドロックが再発することがある．また，これらの方式では，デッドロック検出時には，処理の再開始が必要である．第二の方式として，デッドロックの防止方式は，デッドロックが発生しないように制約を設ける方式であり，性能が低下する．第三の方式として，デッドロックの回避方式は，事前情報を利用して，トランザクションの実行時にデッドロックを避ける方式であり，事前情報の取得が必要である．また，時刻印により同時実行制御を行う方式[2],[7]が提案されており，時刻印による同時実行制御では，デッドロックは発生しないが，ロック方式とは異なる同時実行制御であるので，既にロック方式を使用しているシステムには適用できない．

本稿では，分散データベースのデータアクセスにおいて，同時実行制御にロック方式を用いる場合において，時刻印の利用により，比較的容易に実現できるデッドロック解決方式を提案する．

2. 分散データベースのデッドロックの定式化

2.1 分散データベースのモデル

図1に分散データベースのモデル例を示す．データベースに対する処理の単位をトランザクション(ＴＲ)と呼ぶ．センタ又は端末をノードと呼ぶ．単一ノードの場合は，分散システムでノードが一つの場合と見なせる．分散システムでは，一つのトランザクションは，一つ以上のサブトランザクション(Ｓ)から構成されており，サブトランザクションの間のコネクションにより，データベースへのアクセスを実現する．サブトランザクションは，一つの主サブトランザクションと一つ以上の従サブトランザクションから構成する．ロックの単位(例えば，一つ以上のレコード，ファイルなど)をエンティティと呼ぶ．一つのトランザクションに属するサブトランザクションは，一つのノード内には一つのみ存在する．すなわち，

ＴＲi＝｛Si1，Si2，･･･，Sin｝

ここで，Sij，(j=1,2,･･･,n)は，ＴＲiに属し，且つノードjに存在するサブトランザクションである．分散データベースを，ＤＢk＝｛Dkj｝と表す．Dkjは，ノードjに属するエンティティkの構成要素を表す．Dkjの値をVkjとすると，重複データのときは，

Vk1＝Vk2＝･･･＝Vkn

である．各ノードに存在して，ロック制御を行うプロセスを，ロック制御プロセスと呼ぶ．

2.2 ロック制御

ロックモードには，リードロックとライトロックがある．リードロック又はライトロックを，ロックと呼ぶ．エンティティのロック状態には，アンロック状態，リードロック状態，及びライトロック状態がある．リードロック状態またはライトロック状態を，ロック状態と呼ぶ．アンロック状態は，リードロックとライトロックのいずれも設定されていない状態である．リードロック状態は，リードロックが設定されている状態であり，リードロックを設定したトランザクションは，そのエンティティに参照を行える．ライトロック状態は，ライトロックが設定されている状態であり，ライトロックを設定したトランザクションは，そのエンティティに，参照および更新を行える．

同一エンティティに対するロック要求の競合制御は，次のとおりである．エンティティがアンロック状態であれば，リードロック又はライトロックを設定できる．エンティティがリードロック状態のときは，別のリードロックを設定することは可能であるが，ライトロックを設定することはできない．エンティティがライトロック状態のときは，別のリードロックとライトロックのいずれも設定することはできない．

2.3 デッドロックの定式化

(1) デッドロック

システム状態グラフG(V, E)において，V={P, D}，P={Pi；i=1,2,･･･,p}，及びD={Di；i=1, 2, ･･･,q}とする．ここで，Pはトランザクション（又はサブトランザクション）の集合，Dはエンティティの集合である．アークEは，次の規則で設定する．

① PiがDjをロック中であれば，Dj→Pi，

② PiがPjでロック中のDkをロック待ちしているならば，Pi→Dk→Pj，

③ PiがDjをロック中であり，且つ，Pm，Pn，･･･，Prが，PiのDjに対するアンロックを，FIFO（First In First Out）方式のキューでロック待ちしているのであれば，

Pm→Pn→･･･→Pr→Dj→Pi と表す．

また，Piが複数のエンティティを同時にロック待ちする場合には，ロック待ちしているすべての資源をPiがロックするまで，Piは動作できないとする．トランザクション又はサブトランザクションが，エンティティをロックできないときには，必ず待ち状態になると仮定する．図2は，トランザクションP1，P2とエンティティD1，D2により発生しているデッドロックの例である．

［定理1］： システム状態グラフG(V, E)で表される集中システムまたは分散システムにデッドロックが存在するための必要十分条件は，G(V, E)中に有向サイクルが存在することである[4],[5]．

(2) 重複データ更新におけるデッドロック

すべてのノードが同一のエンティティを有する場合が重複データである．重複データ更新のときに発生するデッドロックも，有向サイクルにより検出できる．

3. 時刻印を利用したデッドロック解決方式

3.1 デッドロック解決方式の概要

時刻印Tを，次のように定義する．

T＝｛α，β，γ｝

ここで，αは，ロック要求発生時刻，βは，トランザクション番号，γは要求元ノード番号である．任意の二つの時刻印T1＝｛α1，β1，γ1｝とT2＝｛α2，β2，γ2｝で，T1＜T2 となるのは，

α1＜α2 ，又は

α1＝α2 且つ β1＜β2 ， 又は

α1＝α2 且つ β1＝β2 且つ γ1＜γ2

のときである．αは物理時刻であるが，時刻印のαは，ロック要求の順序制御のための論理的な値にのみ使用するので，ノード間の時計の厳密な同期は不要である．本稿で提案する時刻印デッドロック解決方式の考え方は，次のとおりである（詳細は，次節で述べる）．

① 一つのトランザクションについて，必要なロックは一度に要求する静的ロック方式を用いる．トランザクションは，そのすべてのロックが取得できてから，処理を開始できる．この時に，一つのトランザクションが，各ロック要求に付加する時刻印は，同一である．

② ロック制御プロセスは，時刻印の時刻部分(α)の値がtのロック要求にロックの仮許可を設定時には，t＋kの時刻まで，待ち合わせる．ここで，kは，ロック要求の最大送信時間である．このことにより，tよりも小さい時刻印のロック要求が到着することを十分大きな確率で保証する（命題1参照）．

③ 一つのエンティティのデータ更新において，時刻印は，デッドロックを防ぐことにのみ使用するので，時刻印の順序は，データの更新順序を意味しない．このことにより，ロック要求が，ノードの処理時間，回線遅延，又は伝送誤りによる再送などが，異常に大きいときに，エンティティの存在ノードに時刻印の順には到着しないときにも，任意の一つのトランザクションは，それが要求を行ったロック要求についてのロック許可をすべて取得時には，以後の処理を開始でき，且つそのトランザクションが行ったロック要求の時刻印よりも小さい時刻印のロック要求が後から到着したとしても，そのトランザクションは処理を続行でき，処理の打ち切りなどは不要である．

④ トランザクションは，ロック要求の付加情報に，ただ1つのエンティティにのみ，ロック要求を行っていることを表示できる．ロック要求の付加情報に，この表示がなされているときには，ロック制御プロセスは，一定時間kの待ち合わせを行わずに，ロックの仮許可を直ちにロック許可にしてよい（命題3参照）．

以下の命題が成立する．

［命題1］時刻印のロック要求の時刻部分(α)がtのトランザクションよりも小さい（すなわち，早い）トランザクションは，メッセージの紛失がないときには，kをメッセージの最大送信時間とすると，時刻t＋kまでには，すべて到着する（図3参照）．

［命題2］この方式は，時刻印をデッドロックの解決

に用いるのみであるので，各トランザクションが，強二相ロック方式[1],[4]に従って，ロック要求およびアンロック要求を行えば，直列可能性，及びロールバックの非連鎖は保持できる．従って，正当性の保証がなされているロック制御に，このデッドロック解決方式を追加しても，正当性は保証される．

［命題３］一つのトランザクションは，ただ一つのエンティティにのみロック要求を行うときは，デッドロックサイクルを構成する要素にならないので，そのトランザクションは，デッドロックの要因にはならない．

３．２　デッドロック解決方式と通信処理の手順

トランザクションの処理に必要なエンティティのロックは，処理の開始時に全部要求することとする(静的ロック方式)．非重複データまたは重複データに対する時刻印デッドロック解決方式では，デッドロックを，次の処理で解決する．

ロック制御プロセスは，各トランザクションからの各エンティティへのロック要求の探索時（例えば，エンティティがアンロックされ，且つロック要求がそのエンティティに存在する時，又はアンロック状態のエンティティに，新たなロック要求が到着時）に，一つのエンティティへのロック要求の内，最小の時刻印を有するロック要求にロックの仮許可を与え，且つ時刻印Ｔの時刻部分（α）が，例えばｔの値のロック要求に仮許可を設定時には，ｔ＋ｋの時刻まで待ち合わせて，仮許可を与えたロック要求の時刻印よりも小さい時刻印を有する新たなロック要求の到着がないときには，ロックの仮許可を与えたロック要求に，ロックの許可を与え，又は仮許可を与えたロック要求の時刻印よりも小さい時刻印を有する新たなロック要求の到着があるときには，その新たな一つ以上のロック要求の内の最小の時刻印を有するロック要求に，ロックの許可を与える．ここで，一定時間ｋは，それぞれのノードであらかじめ設定しておく値であり，例えば，ロックの仮許可を与えたロック要求の備える時刻印よりも小さい時刻印を持つロック要求を含むメッセージが，そのロック制御プロセスに到着するまでの最大通信時間に，若干の余裕を加えた値とすることができる．ｔ＋ｋの時刻まで待ち合わせることにより，仮許可を与えたロック要求の時刻印よりも小さい時刻印を有するロック要求の到着を，十分大きな確率で保証できる．また，ノードまたは通信回線の異常を監視するための監視時間は，例えば，トランザクションの正常処理時間（すなわち，トランザクションのノード内処理時間および通信時間）の最大値に余裕値を加算した値に，ノード又はトランザクションごとに設定できる．一つのエンティティについてのロックの仮許可は，例えば，そのエンティティのロックの許可が一つのトランザクションに与えられた時，又はそのエンティティのロック要求が設定できなかった時（ロック要求が設定待ちになった時を含む）に無効となる．

トランザクションは，ロック要求の付加情報に，そのトランザクションが，ただ一つのエンティティにのみロック要求を行っていることを表示できる．ロック要求の付加情報に，この表示がなされているときには，ロック制御プロセスは，一定時間ｋの待ち合わせを行わずに，ロックの仮許可を直ちにロック許可にできる．

デッドロック解決のための通信処理の手順の例を以下に示す．

［通信処理の手順］

トランザクションの処理が，二つ以上のノードの分散データベースに及ぶ場合の処理手順の例を述べる．一つのトランザクションは，一つ以上のサブトランザクションから構成する．一つのトランザクションは，一つ以上のエンティティにそれぞれ，ロックを要求することができる．ここで，ロック制御プロセスは，トランザクションとサブトランザクションの区別はしない．各ノードの従サブトランザクションは，主サブトランザクションに，ロック要求の結果を通知する．

Ｓtep　１：　各トランザクションは，各エンティティへのロック要求に，要求元の主サブトランザクションで時刻印を付与して，　各エンティティの存在ノード（自ノードのエンティティは，自ノード）に転送する．

Ｓtep　２：　各エンティティの存在ノードでは，主サブトランザクションまたは従サブトランザクションは，ロック制御プロセスにロック要求を行う．ロック制御プロセスは，次の動作を行う．一つのエンティティへのロック要求の選択時に，最小の時刻印を有するロック要求にロックの仮許可を与え，更に，その時刻印の時刻部分(α)の値に一定時間ｋ（例えば，他のノードからのロック要求のメッセージの最大送信時間に余裕を加算した時間）を加えた時刻まで，新規のロック要求の到着を待ち合わせてから，目的のエンティティへのロック要求群の内の最小の時刻印を有するロック要

求を選択し，次の処理を行う．

(1)　選択したロック要求がライトロックのとき

エンティティがアンロック状態であれば、そのライトロックを設定し，そのライトロック要求元のトランザクションにロック許可を通知する．エンティティがリードロック状態またはライトロック状態であれば，次のロック要求の選択時点まで待ち合わせる．

(2)　選択したロック要求がリードロックのとき

エンティティがアンロック状態またはリードロック状態であれば，そのリードロックを設定し，リードロック要求元のトランザクションにロック許可を通知する．又は，エンティティがライトロック状態であれば，次のロック要求の選択時点まで待ち合わせる．新規のロックを設定後に，より小さな時刻印を有するロック要求が到着するときでも，そのままとする。

Ｓtep　３：　ロック要求元の主サブトランザクションでは，その主サブトランザクションが要求した全部のロック要求についてのロック許可が，自ノードの分は，自ノードのロック制御プロセスから通知され，他ノードの分は，従サブトランザクションから通知された時点で，その主サブトランザクションの処理を実行し，その結果として，従サブトランザクションの処理も実行される．一定の監視時間待ち合わせて，すべてのエンティティのロック許可が得られない場合は，取得済みのロックはアンロックし，トランザクションの処理を，最初から再開始する．この時，時刻印も再設定する．　　（手順終り）

４．考察

４．１　代替方式との比較

次の方式を比較する．

［方式１］：　静的ロック方式で，デッドロック対策に本稿のデッドロック解決方式を適用する場合．

［方式２］：　静的ロック方式で，デッドロック対策は，①時間監視方式，又は②システム状態グラフを用いた周期検出方式の場合．

［方式３］：　トランザクションの処理中に，必要に応じてロック要求を行う動的ロック方式で，デッドロック対策は，①時間監視方式，又は②システム状態グラフを用いた周期検出方式の場合．

［方式４］：　同時実行制御に，時刻印を用いる場合．

表１に，各方式の比較例を示す．

４．２　動作モデルによる考察

本稿のデッドロック解決方式の欠点は，各トランザクションは，必要なロックを処理の開始時にすべて要求する必要があることである(静的ロック方式)．そこで，各ノードが同一エンティティを有する重複データのデータ更新について，静的ロック方式と，エンティティへのアクセスが必要時にロック要求を逐次行う動的ロック方式の比較を，デッドロック対策を考慮せずに行う．重複データは，非重複データの特別な場合である．ノードは，ノード１，ノード２，・・・，ノードｎ，のｎ個のノードとする．各ノードのトランザクションは，自ノードおよび他のすべてのノードの同一エンティティに，ライトロック要求を行うこととする．評価基準として，延べロック時間を，一つのエンティティがロック状態の時間と，ロックしているエンティティ数との積で定義する．

Ｌ＝一つのエンティティのライトロック要求を全部のノードに送信してから全部のロック許可が得られるまでの平均時間
（複数のエンティティにも，並列して処理可能とする．）

ａ＝一つのエンティティについて，全部のノードに対する一つのトランザクションのロック設定後の平均処理時間

ｍ＝一つのトランザクションでロック対象とする一つのノードの総エンティティ数

Ｗｓ＝静的ロック方式の延べロック時間

Ｗｄ＝動的ロック方式の延べロック時間

とすると，

$Ws=(m\cdot a)\cdot m\cdot n=m^2\cdot a\cdot n$

$Wd=\{1\cdot(a+L)+2\cdot(a+L)+\cdots+m\cdot(a+L)-mL\}\cdot n$

$=mn\{m(a+L)+a-L\}/2$

これから，

ｍ＝１のとき，Ｗｓ＝Ｗｄ

ｍ＞１のとき，ａ＜ＬならばＷｓ＜Ｗｄ

ａ＝ＬならばＷｓ＝Ｗｄ

ａ＞ＬならばＷｓ＞Ｗｄ

である．

［考察］　上述の結果から分かるように，分散データベースでは，ロック要求の送信に通信時間が必要であることを考慮すると，動的ロック方式は，静的ロック方式と比べて必ずしも性能がよいとは限らない．

5. むすび

分散データベースにおける同時実行制御にロック方式を用いる場合のデッドロック解決方式として，時刻印を用いたデッドロック解決方式を提案した．この方式の特長は，①時刻印をデッドロック解決のためにのみ用いており，同時実行制御は既存のロック方式であるので，既存のシステムへの導入が比較的容易であること，②複数のトランザクションのロックの実行順序は，時刻印の順でなくてもよいので，ロックの設定後に時刻印の小さなトランザクションが後から到着した時でも，先に更新を行っているトランザクションのロールバックは不要であること，及び③時刻印は，トランザクションにのみ付与し，データ実体には付与しないので，二次記憶量のオーバヘッドがないことである．他の方式との性能の詳細比較などが今後の課題である．

［謝辞］有益なコメントを頂いた，ＮＴＴ情報通信処理研究所の栗原定見・主幹研究員に感謝します．

参 考 文 献

(1) Bayer,R., et al.: "Parallelism and Recovery in Database Systems", ACM Trans. on Database Systems, 5, 2, pp.139-156(1980).

(2) Bernstein,P.A, et al.: "Timestamp-Based Algorithms for Concurrency Control in Distributed Database Systems", Proc.VLDB, pp.285-300(1980).

(3) Elmagarmid,A.K.: "A Survey of Distributed Deadlock Detection Algorithms", ACM SIGMOD RECORD, 15, 3, pp.37-45(1986).

(4) Gray,J.N.: "Notes on Database Operating Systems", Operating Systems: An Advanced Course", Springer-Verlag(1978).

(5) Isloor,S.S.: "The Deadlock Problem: An Overview", Computer, 13, 9, pp.58-78(1980).

(6) 小林哲二："データベースのロック方式における並列性向上の一方式"，情処学論，27，2，pp.236-242(1986).

(7) Rosenkrantz, D.J., et al.: "System Level Concurrency Control for Distributed Database Systems", ACM Trans. on Database Systems, 3, 2, pp.178-198(1978).

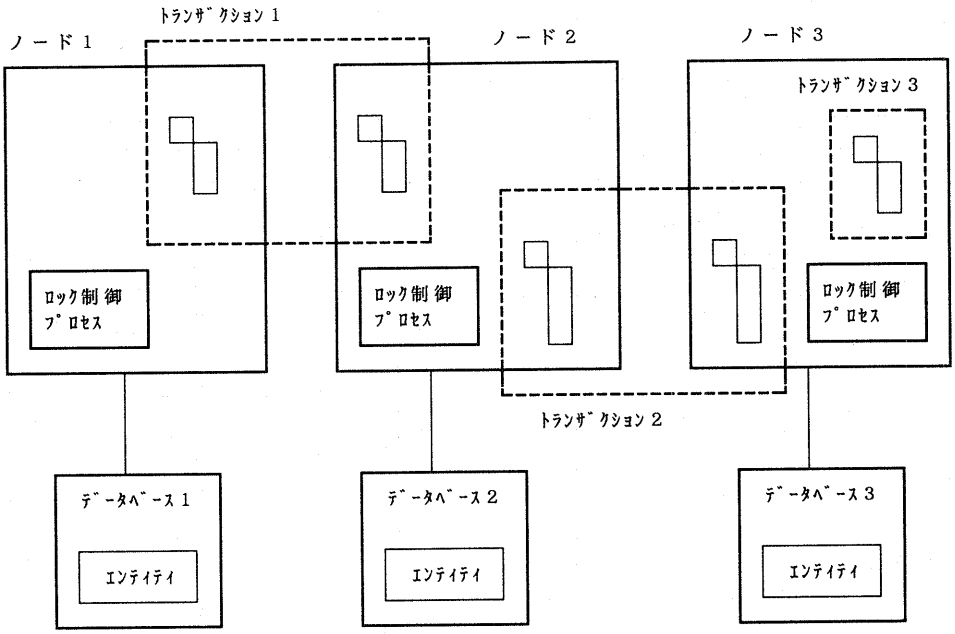

図１　分散データベースのモデル例

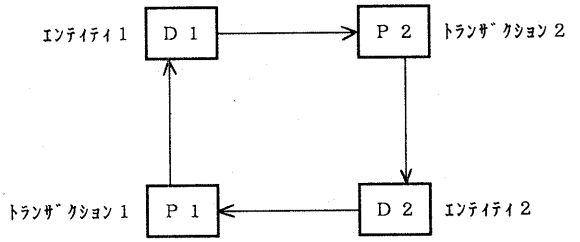

図２　デッドロックの例

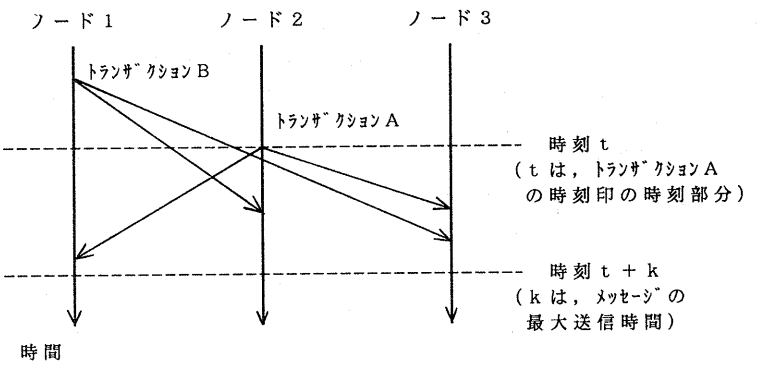

図３　トランザクションと時刻印の時刻部分の関係の例

表1　本稿のデッドロック解決方式と他の方式との比較例

項目＼方式	静的ロック方式に本稿のデッドロック解決方式を適用	静的ロック方式にデッドロック検出方式を適用	動的ロック方式にデッドロック検出方式を適用	時刻印方式
アクセス方法	最初に，必要な全部のデータにロック要求を行う	最初に，必要な全部のデータにロック要求を行う	必要に応じてロック要求を行える	必要に応じてアクセス要求を行える
ロック要求への時刻印付加	あり	なし	同左	あり
データ実体への時刻印付加	なし	なし	同左	あり
同時実行制御	強二相ロック方式	同左	同左	時刻印の順序でデータ更新
デッドロック対策	本稿の方式で防ぐことが可能	時間監視方式 システム状態グラフ方式（周期検出）		デッドロックは発生しない
デッドロック検出時間の主要な要素	メッセージの最大送信時間 （本方式の長所）	（時間監視方式のとき） メッセージの最大送信時間 ＋　最大正常処理時間 （システム状態グラフ方式のとき） システム状態グラフの転送時間 ＋　検出周期時間		デッドロックは発生しない
デッドロック回復処理	メッセージの紛失，長遅延などの異常時以外は不要 （本方式の長所）	処理の再開始が必要	処理の再開始が必要	時刻印の矛盾が発生時は，トランザクションの再開始が必要
ロック方式との整合性	良い （本方式の長所）	良い	良い	悪い