# ネットワークシステム上での経済統計の更新

姫宮利融　　丸山不二夫

## 1，緒言：分散型データベース・システム採用の利点とその問題

### 1－1　データベースシステムをなぜ採用するのか？

まず，「データベース・システム」という用語の定義から始める。情報処理の用語としてのデータベースシステムは「コンピュータ化された記録保持システム」を意味する。例えば，ある商店が顧客名簿を顧客毎のカード式で管理し，得意客の転出，新規の顧客の開拓に伴う変更をチエックし，常に最新の名簿を整理して揃えているとする。たしかに，これも，データのベース（基礎）システムであるが，これは，単なるカード式システムであり，データベース・システムとは呼ばない。既述した意味でのデータベース・システムの意義はカード式システムをコンピュータ処理することを考えるとわかりやすい。その利点は次の点にある。

①コンパクトであること。…… 100人，200人分の顧客名簿ならばカード方式の方法でも机の引出し一つを占有するだけであろう。しかし，これが，1000人，2000人分の顧客データ，金融期間のように万の単位の顧客を管理するとなると，カード式ファイルでは建物の何階分もの床面積を占有してしまう。コンピュータの補助記憶装置（ディスク）は比べものにならないほどコンパクトである。

②検索・更新が迅速であること。……データの検索・更新はファイル検索・書き変えの通常のコンピュータ処理の手続きの延長上で考えてよい。（後で分散処理の場合に起きる問題を述べる。）

③骨折り仕事が少ない。……前項と関係するが，計算機に処理させるのであるから，使用者はコマンドを打ち込むだけである。

④最新のデータを保持できること。……カード式ファイルの場合，カードを二重に作ってどちらが現在の状態のデータかわからなくなったり，データやカードの行方不明が生じやすい。データベースシステムでは（アルゴリズムが正しければ），最も新しいデータをつねに保持できるはずである。

ここで，データベース・システムの構成要素について簡単に述べる[1]。データ，②ハードウェア，③ソフトウェア，④ユーザである。まずデータは，統一された全体としての側面と，分割共有される側面がある。統一された全体としては，データはダブリ（あるいは冗語）が除去され統一されたファイルとして存在するということが大事である。分割共有される点からは，データベースの独立な部分が複数のユーザによって共有される。ハードウェアで重要なのは補助記憶装置であり，通常ディスクが使われる。ソフトウェアは，データベース管理システム（ＤＢＭＳ）の設計が重要であり，エンド・ユーザがハードウェアレベルの詳細を知ることなしにデータベースを使えるような橋渡しとシールドの役割を担う。ユーザは，エンド・ユーザ，アプリケーション，プログラマ・データベース管理者からなる。

### １－２　ネットワーク上への分散の目的

データベースのネットワークへの分散とは，各サイトにＣＰＵ（中央処理装置）と記憶装置，入出力装置を置き，それらの間を通信回路で結んでいるものであり，各サイトが自律的でありながら，相互に総合することができるものである。一つのある特定の地理的な場所にあるメインフレーム上にデータベースを構築し遠隔地のものは端末から接近からするだけの完全集中型データベースに比べ，次の点が有利であると言われる[1] [2]。

①局地的自律性……先の例で銀行が各地に支店をもっているとする。支店の顧客は地元の人が90％を占めるから，顧客のデータベースの場合，各支店でそれぞれの顧客を管理したい。同時に，客が取引支店とは離れた支店で取引を要求してもそれに応じなければならない。

②システム全体の信頼性の向上……完全集中型データベースの場合，メインフレームに障害が発生するとその間機能が停止するが，分散型ではいくつかのサイトが故障してもシステム全体は（能力は低下するが）働き続けることが出来る。

③処理能力を経済的に適切に増大させることが出来る。……メインフレーム一台のみで処理するとするならば，能力の向上は，最終的にメインフレームそのものを上級機と取り換える以外にない。分散型データベースではサイトを増殖させればよい。

④コンピュータの処理機能の分散，過負荷をさける。

以上，述べてきたが，簡単に言えば，経済的問題（データベース・システム全体にかかる経費）と処理速度の問題が大きい。即ち，計算機の演算素子・記録装置は10年間にその性能の向上に比べると格段に値下がりしているが，通信にかかる費用はそれほど安くなっていないこと，演算速度に比べ通信速度は1000～ 100分の１であることが，分散処理の大きな動機である。

### １－３　分散型データベースの問題

分散型データベースは，各サイトにデータが置かれているわけだが，利用者にとってはあたかもひとつのシステムであるかのように振舞わなければならない。これは，一応，３つの点にまとめられる。①利用者は，どのサイトに自分の望むデータが貯えられているか知る必要がないこと，②データの分割を自動的に処理し，利用者はサイト間の関係がどうであろうとも統一したシステムを扱うように利用できること。③多くの写しがその存在と異同について透明であること，である。このことに，前述した経費・処理速度の点から自律性という一見相反する要求が課せられて，分散型データベース管理システム（ＤＢＭＳ）の理論的課題（困難）を生じさせる。本報ではそのうち，「更新」の問題を取り上げる。

## 2，　「更新」に伴う問題

### 2－1　一貫性の保証－同時並行処理の制御

（一貫性の２つの相……相互一貫性と内部一貫性）

Thomasら [3]によると「更新」メカニズム基本的目的はデータベース・コピーに適用される「更新」が，コピー【複数】の集まりの間の「相互一貫性」と各のコピー【単数】の「内部一貫性」を保つように保証することにあるとされる。ここで相互一貫性とは全てのコピーが同じ状態を保持し，更新活動が終わったならば同一のものであることを保障することを意味する。内部一貫性とはデータベース内の項目の間にある不変な関係が保持されていることを意味する。（このような例は多い。例えば簿記で（借方合計）＝（貸方合計）が常に成立をしていなければならぬ。）

具体的な例としてＡ地とＢ地という２つのサイトにデータ項目ｘ，ｙ，ｚを含むデータベースを考えよう。（同一のデータが二重になっている）。ｘ，ｙ，ｚは始め両方のコピーですべて１であり，ｘ＋ｙ＋ｚ＝３という不変な関係が常に成立しなければならないとする。Ｒ１とＲ２という２つの更新を考える。

R1:y:=-1, z:=3

R2:y:=-1, z:=3

ここで“ :＝”は，置き換えるという操作を意味する。

①Ｒ１をＡ，Ｂにおよぼした場合つぎのようになる。

A; x=-1, y=3, z=1

関係　x+y+z=3

B; x=-1, y=3, z=1

関係　x+y+z=3

相互一貫性と内部一貫性は両方とも保たれている。Ｒ２をＡ，Ｂにおよぼした場合もｘ，ｙ，ｚの値は異なるが同様である。

②Ｒ１をＡにおよぼし，Ｒ２をＢにおよぼした場合，次のようになる。

A; x=-1, y=3, z=1
関係　x+y+z=3
B; x=1, y=-1, z=3
関係　x+y+z=3

ｘ＋ｙ＋ｚ＝３という内部的規約は守られているが，Ａ，Ｂで同じ項目に対して異なる値のコピーが作られ，相互一貫性が破られる。

③Ｒ１，Ｒ２の順番にＡ（とＢ）を更新する場合，まづＲ１によって次のようになる。

A; x=-1, y=3, Z=1
関係　x+y+z=3

つぎにＲ２によって次の値に更新される。

A; x=-1, y=-1, z=3
関係　x+y+z=1≠3

Ｂについても全く同じであり，最終的に次のようになる。

B; x=-1, y=-1, z=3
関係　x+y+z≠3

従って，A地とB地の２つのサイトのコピーが各項目で同じ値を持つという意味で相互一貫性は保たれているが（①で見たようにＲ１，Ｒ２は単独ではｘ＋ｙ＋ｚ＝３が守られるという意味での内部一貫性を犯さないにもかかわらず，連続して行うと），それを破壊してしまう。Ｒ２，Ｒ１の順番でもｘ＝－１，ｙ＝３，ｚ＝３となり内部一貫性が破壊される。

以上のように，一貫性(consistency）の保障は分散処理を行ううえで，避けて通れない問題であり，それを解決するため様々な方法が考えられてきた。同時に，「更新」の問題は，他にもデットロック（deadlock）の問題，故障の発生と回復の問題があって相互に関連している。と言うより，例えば，首尾一貫性を保障しつつ故障に強いシステムをいかに構築するかというような問題となる。そこで，デッドロックの問題，故障の発生と回復の問題について以下に述べる。

## ２－２　デッドロックの問題

デッドロックとは，一貫性を保護するためにデータにロックをほどこす（データをおさえてしまう）ことに伴い，必ず生ずる問題である[1]。その概念を理解するために，図の例を見てみよう[5]。この例では，２つの独立したトランザクションが同時に動作しており，トランザクションＡは更新の準備としてファイル$\alpha$をロックしており，（読みだし，書き込みを禁止しており），トランザクションＢは同じく更新の準備としてファイル$\beta$ロックしている。トランザルションＡを完了するため必ず$\beta$のフアイルを読む事が必要であり，トランザクションＢを完了するためには必ず$\alpha$のファイルを読むことが必要であったとすると，すでにロックされたフイルにアクセスできず，どちらも永久に「持ち」の状態を続けるとこになってしまう。

デットロックは，分散システムに限らずらず古くから多くのシステムを停止させてきたはずでるが，１９６５年に最初にDijkstraによって発見された[6]。次の４つ条件はデッドロックが発生するのにいずれも必要な条件である。

①同時に並行プロセス

②相互に排他的アクセス

③先取権を認めないスケジューリング

④部分的な資源の割当て

デッドロックは上記の必要条件のどれか１つの条件を除外することで回避することが出来る。ところが，この４条件のそれぞれは正常の動作の上では大変重要な条件であり，②と③は首尾一貫性を保障する上で不可欠である。ここに，デッドロックの問題が存在するといえる。

### ２－３　信頼生と回復

データベースの重要性が高ければ高いほど信頼性の要求は厳しくなる。集中データベース・システムおよび分割データベース・システムでは障害からの回復は，ビフォールック／アフタールックをとること，即ち，常に更新前の状態と更新後の状態をファイルとして保持しておくこと，また，バックアップ・コピーをもつことによって行なう。本稿で扱う複写データベース・システムでは，通信リンクやサイトが故障しても故障したモードで発生したトランザクションが処理できなくなるのみで他の部分は正常に動き続ける。問題は回復の際で，故障したサイトが復旧したときに他のサイトから更新情報を入手するが，この際首尾一貫性をどう保障するかが課題となる。複写データサービスシステムは，同期がとりにくいことから生ずる問題である。

実際には，両方とも許容できる水準に達する方法の開発が必要である。

## ３，　首尾一貫性を管理する幾つかの方法

### ３－１　用語の定義とモデルの制限

#### ａ．　用語の定義

「更新」で使われる用語の定義をまとめる。まず，トランザクション（Tran

saction)とは，ISSUEがデータベース・システムに対して読みだし・書き込みを行うプログラムである。[4]。サイト（site）とは，各に演算装置・制御装置・記憶装置・入出力装置を備えたデータベース・システムの単位で，チャンネル・リンクを通じてネットワークを構成している。必ずしも地理的に離れている必要はなく、独立した単位ならば，それでひとつのサイトである。項目の値（数字のみとは限らない）の読みだし作業（read operation）と書き込み作業（write operation)は，データベース・システムの提供する基本的作業でその間に「更新」のための新しい値の計算作業が入ることが多い。コピー（写しcopy）は，同じデータの各項目の写しを各サイトに分散させて同じデータのコピーが複数存在するのが普通でありその各をコピーを言う。項目（item）は，例えば，銀行の顧客名簿なら，その１人についての口座番号，氏名，住所，預金額，貸付額，資産評価などそれぞれ指し変数として取り扱う。

b．　検討の対象とするモデルの制限……仮定

次に本稿で扱う方法に共通する仮定について述べる。まず，すべてのサイトにコピーが存在していること（full replication）があげられる。特定の幾つかのサイトにのみコピーを分散させるデータベース・システムも存在し，それはそれで検討を要する問題である本稿では扱わない。また，各サイトやコピーへのアクセスに一般的制限がないことを仮定される。特定のサイトやコピーに特権的地位を与えデータベース・システム管理者のみしかアクセスできないようのデザインが考えられるがそのよう制限はもうけない。

## ３－２　首尾一貫性の保障＝連続性（serializability)の保持

首尾一貫性をを保障することは，トランザクションの連続性を保持することである[7),8),9),10),11)]。換言すれば，トランザクションがサイクル連関を形成しないことである。この考え方は多くの方法に採用されているものである。

## ３－３　首尾一貫性を保障する＝同期をとるための幾つかの方法。

首尾一貫性の保障の方法には２つの考え方がある。①集中管理，②分散管理である。集中管理とは全ての更新の要請が単一の中心点を通るような管理である。更新の要請はその点で批准され（be validicated）様々なサイトに送られる。分散管理では，更新の要請を批准しそれをデータベース・コピーに及ぼす責任がデータベース・サイトの集まりの間に分散されている [3]。集中管理は当然のことながら、本来分散管理のメリットであったはずの各プロセスの自律性が損なわれ，中心点にアクセスできないような障害が障害が生じたときには必ず停滞が生じるという問題がある。　又，分散管理は，障害の発生には強いが，首尾一貫性とデットロックの発生の回避を保障するにはいろいろな工夫を必要とする。

a．　一次コピー法（primary copy method）

Alsberg ら [2] によって提案されたものであり，更新をマスターコピー（一次コピー）で管理する。ノードで更新のリクエストが発生するとリクエストは一次ノードに送られる。一次ノードでは自分の値をまず更新し他のノードに送る。一次ノードの故障の場合を考えて必ずバックアップが必要である。

b．一次サイト法（primary site method）

Stonebraker ら [13] によって提案されたものであり，Alesbergらの方法の改良である。データベースの各部分にして一次サイトが決められており，その部分に関する更新はその部分の一次サイトが責任を負う。あるノードでトランザクションが発生すると、そのノードが「マスター」になり，他のノードに命令を出す。マスターは一次サイトを経由して２つのことを行うことができる。

①各ノード（スレーブ）でトランザクションを実行させること。

②各ノード（スレーブ）にデータベースの部分のコピーを送ること。

融通性のあるシステムであるが，スレーブでの動作の順序があらかじめ決まっているわけではないので，デッドロックをおこす可能性があり，その回避のための方法が問題である。

c．アベイラベル・コピー法（available copies method）

Alsberg ら [12] のアイディアをサイトが故障する状況に拡張し，その下でも首尾一貫性を確保しょうとするものでる。Bernstein ら [14] によって提案されたものである。「使用できる」コピーをリストするディレクトリーを各のデータ項目に対してそれぞれ保持し，読みだし，書き込みともそのディレクトリーに相談して行う点が特徴である。データ項目の利用ができるコピーのリストと，ディレクトリーのレストという二重のリストを保持し，特別なトランザクションを使用する。各のトランザクションはそれぞれ単一のサイト ― トランザクション，マネジャー ―によって監督をうける。この点がこの方法が集中型に属する所似である。トランザクション・マネジャーがトランザクションがロックされた地点に到着する前に故障したら，データベースシステムがトランザクションを御破算にしてしまう。

d．多数決法（majoity consensus method）

Bernstein ら [4]，Daniels ら [15]，Gifford ら [16]，Thomasら [8]，によって提案されそれぞれ特徴があるが，いずれも，更新リクエストを受け入れるか否かについてローカルなデータベース・マネージメント・プログラム間で投票を行い，多数決で決定するものである。この投票規則の決め方，多数決でも単純多数なのか定足数関しての多数なのかによって様々なバリエーションを生じる。

ここでは，Thomasら [8] の方法と，Bernstein ら [4] の方法について紹介しよう。

Thomasらの方法は，多数決法とタイムススタンプ法の結合である。データの各の「要素はデータとしての変数とタイムスタンプを持っている。タイムスタンプは現在の値（＝変数）を受け取った日付，時刻が刻印されている。「更新」は以下のステップでおこなわれる。

① データベースへの質問（query）：アプリケーションプロセス（ＡＰ）がデータベースに対してデータ要素の値を得るためにキューする。データベース運営プロミス（ＤＢＭＰ）はキユーに応えて，変数とタイムスタンプを提供する。

② 更新計算： ＡＰは新しい値－更新変数－を計算する。

③ 要請具申（submit request）： ＡＰは更新要請，即ち，新しい値の変数とタイムスタンプからなるベース変数を構成し，それをＤＢＭＡに具申する。

④ 更新同期： ＤＢＭＡは各のベース変数タイムスタンプをデータベース内のタイムスタンプと照合し，その結果に基づいて要請を受け入れるか否か決定する。これによって同時並行処理の管理を行う。

⑤ 更新の適用： 更新が受け入れられたら各のＤＢＭＰはコピーを更新する。

⑥ ＡＰへの通知： ＡＰに要請の結果を知らせる。

前述したように，集中管理型に対比して信頼性があり，通信コストもそれほど大きくない。しかし，①～⑥のステップの間短期間であるが記憶容量をとるという問題がある。

Burnstein らの論文[4]は，実は，ほとんどが同時並行処理の制御の正しさを解析するための理論的検討に向けられており，アルゴリズムそのものは簡単に述べられている。彼らが述べているのは，マルチバージョン・タイムスタンプ法とマルチバージョン・ロッキング法の混合法の一般化である。タイムスタンプ法ではトランザクション$T_i$と$T_j$の時間関係は，$T_i$のタイムスタンプをＴＳ（ｉ）として，

$$T_i << T_j, \quad \text{if} \quad TS(i) < TS(j)$$

となっている。また，ロッキング法では値Ｘｉを認職しているイベントを$c_i$と表記して，

$T_i << T_j$ , if $c_i < c_j$

である。ここで，”<<”は$T_i$と$T_j$が連続である（serial）ことをあらわす。従って，タイムスタンプ法とロッキング法をどのように割り振るかが問題となる。マルチバージョン・タイムスタンピングを読み出し専用のトランザクションに，マルチバージョン・ロッキングを一般的なトランザクションに割当てる。

① $T_i$と$T_i$を更新者とする。$c_i < c_j$ならばTS（i）<TS（j）となる。

② $T_q$をキューとし$T_i$を更新者とする。
$r_q[x_k] < w_i[x_i]$ ならばTS（q）<TS（i）となる。

一貫したタイムスタンプ発生器が，これらの性質を満足するタイムスタンプを割り当てるという方法である。

e，ミッシング・ライト法（missing write method）

Eagerらにとって提唱されたものであり，多数決法の改良である。これは，特に，サイトやリンクの故障に対して更新活動を続行できるという意味で，また回復の過程で同時並行処理の制御が行われているという意味で，「強壮な」システムを目指したものである。その基本的なアイディアは，サイトが以下の４つのファイルを保護することにある。

①ミッシング・更新・ポスティング・ファイル……行方不明の更新情報を含む。

②ミッシング・更新・変数・ファイル……行方不明の更新情報を含む。

③トランザクション・現状・ファイル……トランザクションが再度始まる

か，御破算か，付託かについて示す。

④ミッシング・更新・適用・ファイル……サイトに適用されたところの行方不明のレコードを提出する。

彼らは，このアイディアを分散ロッキング法およびタイムスタンプ順序付け法に応用しており，その方法を述べているが，詳細はここでは割愛する。

### 3．4　小括

以上，首尾一貫性を保持する考えと，その具体的なアルゴリズムの幾つかを簡単に述べたが，すべての場合に万能な方法というものは理論的にも存在し得ないし，それぞれの長所，弱点を考慮して採用の際の判断基準としていく以外にない。また，より性能のよい方法を開発していくことが望まれる。

## 4．経済統計の区分とその性質

経済統計という言葉は様々に解釈され得るだろう。ある経済学者は標本から母集団について推定し国民経済（あるいは国際経済）についての一定の像を描くことを目的に「統計」という語を使う。また，企業家にとっては企業活動を行う上でのデータの集積を「統計」なる語で想像するであろう。前者の代表的なものに例えば国民所得統計などがあり，「更新」は「白書」の形で出版される。後者の代表は，商品市況，株式市況である。リアルタイムで「更新」されねばならず，当然，データベースに依ることになる。また，システムダウンは許されない。その中間的なものに「春闘速報」などがあり，労働側にとっても経営側にとっても，リアルタイムの更新までは費用との関係で必要としないが，新聞や労働団体のニュースでは速報性に欠けるところがある。また，同一業種，同一規模，同一地域の会社について網羅的であることが望ましい。

同一規模，同一地域の会社について網羅的であることが望ましい。

こうして，経済データの更新の問題は，分散処理システム上での更新の一般的問題に帰着する。この点はさらに詳細な検討を必要とする。

## 5．中間的なまとめ

以上，ネットワークシステム上で経営データの更新について，その一般的問題を整理した。分散処理システム上で同時並行処理の制御を行なう問題は，理論的に興味深く，実践的にも重要な問題である。

著者らは，ひとつは，商品市況（魚市場など）を例にとって，それにふさわしい「更新」のアルゴリズムを本学のＳＵＮをホストにしたインサーネット・ネットワーク上で検討する予定である。また，もうひとつは，アルゴリズム自身を検討し，理論的解析を深めたいとかんがえている。

以上の結果は次報で報告する。

# 参考文献

1) G.J.Date "An Introduction to Database Systems" forth edition.
Addision-Wesley Publishing Company

2) Vijay Ahuja "Design and Analysis of Computer Communiaction Networks" Mc-Grow Hill Book Company

3) Robert H.Thomas "A majority consesus approach to concurrency control for multiple copy database."
ACM Trans. Database Syst.Vol.4,No.2(June 1979),180-209

4) Philip A.Bernstein and Nathan Goodman "Multiversion concurrency control-theory and algorithms"
ACM Trans. Database Syst.Vol.8,No.4(Dec.1983),465-483

5) George A.Champine "Distributed Computer Systems"
North-Holland Publishing Company, Amsterdam 1980

6) E.W.Dijkstna "Cooperating Sequential Processes"
Technological University of Eindhoven, The Neterlands, 1965

7) P.A.Bernstein,D.W.Shipman, and W.S.Wong
"Formal aspect of serializability in database concurrency control"
IEEE Trans. Software. Eng. SE-5,3(May 1979),203-215

8) K.P.Eswaran, J.N.Gray, R.A.Lorie, and I.L.Traiger
"The notions of consistency and predicate locks

in a database systems."
Commun. ACM 19,11(Nov.1976),624-633

9) C.H.Papadimitriou
"The serializability of concurrent database updates"
J.ACM 26,4(Oct.1979),631-653

10) C.H.Papadimitrou, P.A.Bernstein, and J.B.Rothnie, Jr.
"Some computational problems ralated to database confurrency control" In Proc.Conf. Theoretiacl Computer Science, (Materloo, Ontario)Aug,1977

11) R.E.Stearns, P.M.Lewis, Ⅱ.,and D.J.Rosenkranz
"Concurrency controls for database systems." In Proc. 17th Symp. Foundations of Computer Science, IEEE, New York,1976,19-32

12) P.A.Alsberg, and J.D.Day
"A principle for resilient sharing of distributed resources"
In Procedings 2nd International Conference on Software Engineering (Oct.1976),562-570

13) M.Stonebraker "Concurrency control and consistency of multiple copies of Data in INGERS" 3rd Berkley Wrkshp on Dist. Data Mgmt & Computer Net, 1978,235-238

14) Philip A.Bernstein and Nathan Goodman "An algorithm for concurrency control and recovery in replicated distributed databases"

ACM Trans. Database Syst. Vol.9,No.4(Dec.1984),596−615

15) D.Daniels and A.Z.Spector "An algorithm for replicated directories." In Proceeding ACM SIGACT−SIGOPS Symposium on Priciples of Computing(Aug.1983),104−113

16) D.K.Gifford "Weighted voting for replicated data." In Proceedings 7th Symposium on Operating Systems Principles (Dec.1979),150−159

17) Derek L.Eager and Kenneth C.Sevcik "Achieving robustness in distributed database systems" ACM Trans.Database Syst.Vol.8.No.3(Sep.1983),354−381

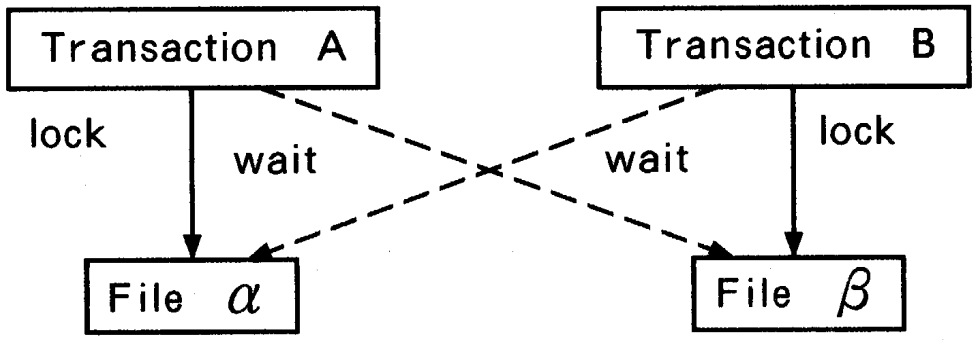


図　デッドロック