データベース・システム 48－4
（1985・7・15）

Cautious Scheduler（先読みスケジューラ）を用いた分散型データベースシステムの並行処理制御

原嶋 秀次，茨木 俊秀（豊橋技科大）

1 はじめに

各種計算機をネットワーク状に結合し，分散処理を行うことは，時代の趨勢となっており，そのさまざまな可能性が各分野で追求されている．このうちの1つに，分散型データベースシステムがある．分散型データベースシステムは，通信コストを低減化できる，ハードウエアや通信回線などの故障に対して信頼性が高い等の特徴を有し，今後次第に普及するものと思われる［BeG 81］［Koh 81］．

本論文では分散型データベースシステムにおけるトランザクションの並行処理制御を考察する．現在のところ，主に2相ロック(2-phase lock)や時刻印(time stamp)による方法が提案されているが，これらを用いた場合，デッドロックを避けるためや直列可能性を保証するためトランザクションの再実行(restart)が要求されることがあること，および永久に拒否（reject）され続けるトランザクションが出来てしまうことがある等の問題があり，これらを防止又は発見・処置する機構が必要となる．

本論文では，各サイトにおいて先読みスケジューラ(cautious scheduler)［IKK 85］によるデータベース管理システムを用いることを前提にし，時刻印を併用する並行処理制御のアルゴリズムを提案しその正当性を証明する．この方式ではデッドロックが生じることはなく，全てのトランザクションは一度システムに申し込むと，有限時間内に必ず実行されることが証明できる．従って，並行処理の目的には上述の様な機構を必要とせず，分散型データーベースシステムの管理が容易になり，並行処理の効果を高め得ると思われる．

2 データベースシステムモデル

データベースシステムはデータ項目の集合Dとトランザクションの集合$\mathbf{T}=\{T_0, T_1, T_2, \ldots, T_n, T_f\}$から成る．トランザクション$T_i$の読取り操作(read operation) $R_i[X]$は，データ項目Xの値をT_iに渡し，書込み操作(write operation) $W_i[X]$は，Xの値を更新する．$S \subseteq D$に対し，$R_i[S]=\{R_i[X] : X \in S\}$, $W_i[S]=\{W_i[X] : X \in S\}$をそれぞれ読取りステップおよび書込みステップという．$T_0=W_0[D]$および$T_f=R_f[D]$は各々初期トランザクション，最終トランザクションと呼ばれDの初期値を書き，最終値を読む架空のトランザクションである．T_0とT_f以外の各トランザクションT_iは，一列に並べられた読取りステップと書込みステップから成る系列として与えられる．

Tの全てのステップの集合をSTEP(T)とする．T上のスケジュールとは，(STEP(T), <)なる対である．ここで<は，各トランザクション内のステップの順序と矛盾しない全順序である．スケジュールは，その各ステップを<の順に左から右に書くことで表現され，例えば

$$s'=W_0[X,Y]\ R_1[X,Y]\ W_1[X]\ R_2[Y]\ W_2[X,Y]\ R_3[Y]\ R_f[X,Y]$$

となる．スケジュールs'において，$R_i[x]$に先行し（$R_i[X]$より左にあり）一番最後にXを更新した操作を$W_j[X]$とする時，T_iはT_jからXを読むといい，Ls'

$(R_i[X])=W_j[X]$ と記す．上例の $W_i[X]$ の様に他のどのトランザクションにも書込んだ値を読まれない書込みステップを不用書込み(useless write)と呼ぶ．またこのs'の様に，全てのステップが他のトランザクションの割り込みなしで，トランザクション毎にくによって順序付けられているスケジュールを直列(serial)であるという．

s と s'を同じ**T**上の２つのスケジュールとする．$L_s=L_{s'}$ の時 $s \equiv s'$ と書き，sとs'は等価(equivalent)であるという．適当な直列スケジュールs'と等価なスケジュールsを直列可能(serializable)であるという．全ての直列可能なスケジュールの集合をSRで表す．

定義１［IKM 83］　sを**T**上のスケジュールとする．sのTIO (transaction input-output)グラフTIO(s)は，節点集合 $\mathbf{T} \cup \mathbf{T'}$ と枝の集合Aから成るラベル付き多重グラフである．T_i が T_j からXを読む時，Xでラベル付けされた枝 $(T_i, T_j) \in A$ を付す．この枝を読取り枝(reads-from arc)と呼ぶ．T_i がYに関する不用書込みを有する時，追加節点 (dummy node) $T_i' \in \mathbf{T'}$ とラベルYの追加枝(dummy arc) $(T_i, T_i') \in A$ を導入する．TIO(s)中に上記以外の枝，節点は存在しない．□

sとs'が等価となる必要十分条件は，TIO(s)=TIO(s')である．

定義２［IKM　83］　sをスケジュールとする．TIO(s)の節点の集合上の全順序≪は次の条件を満たすとき，DITS(disjoint-interval topological sort)であるという．

(1) TIO(s)中で $T_i \ll T_j$ の時，T_j から T_i への経路がない．

(2) (T_g, T_i) と (T_j, T_k) を同じラベルXをもつTIO(s)中の２つの枝とする．但し，$g \neq j$ とする．この時 $T_g \ll T_k$ なら，$T_i \ll T_j$ である（排除規則；図１を参照）．□

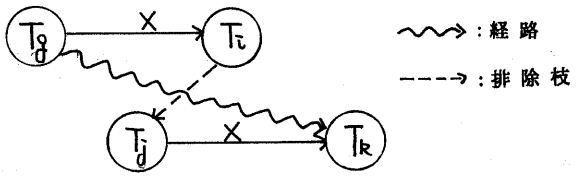


図１　　排除規則

定理１［IKM 83］ スケジュール　sが　⇔　TIO(s)が T_0 を最初として 直列可能である．　T_f を最後とするDITSを持つ．

3　先読みスケジューラ

定理１の結果を用いたデータベースシステム・スケジューラ（database sys-

tem scheduler；以下，単にスケジューラと呼ぶ）に先読みスケジューラCS(C)がある［IKK 85］．ただし，C⊆SRは与えられたスケジュール集合である．CS(C)では新しいトランザクションTがデータベースシステムに到着した時，スケジューラにそのトランザクションの読取り集合（read set；そのトランザクションが読取るデータ項目の集合）と書込み集合（write set；そのトランザクションが書込むデータ項目の集合）をあらかじめ宣言し登録するものとする．Tの登録が済むと，Tはそのステップの実行要求を次々とスケジューラに発するが，この時，前のステップの実行が完了した後初めて次のステップの要求を出すことができる．

ある時点のCS(C)において今までにすでに出力された部分スケジュール(partial schedule)をP，現在実行を要求されているステップをq，読取り，書込み集合で宣言されてはいるがまだ実行されていないステップの集合をPENDとする．CS(C)は$PqQR_f[D] \in C$なるPEND中のステップの系列Qが存在する時に限りqを受理しその実行を指示する．qが受理された場合には，PENDにあるステップの中で，すでに要求を出したがまだ実行されていないステップに対して，同様のテストを加える．$PqQR_f[D] \in C$なるQが存在するか否かのテストを完成テストと呼び，次の活性 TIO グラフを用いる．テストの実行方法については［IKK 85］参照のこと．

定義３［IKK 85］　ステップの集合PENDと部分スケジュールPqに対する活性TIO グラフATIO(Pq,PEND)は，PqあるいはPEND中に含まれるステップを持つトランザクションに対応する節点といくつかの追加節点を持つ．その枝の集合はAと A' なる２つの部分集合から成る．Aは部分スケジュールPqに対しTIO グラフに対するのと同様に定義する．A'はPEND中のステップに対応する追加枝の集合である．すなわち，$R_i[X] \in$ PENDはラベルXの追加枝(T_i'', T_i)，$W_i[X] \in$ PENDはラベルXの追加枝(T_i, T_i')で示される．簡単のためA内の読取り枝は太い線で，その他の追加枝は細い線で書く．□

単一のサイトに用いられる集中型先読みスケジューラは次の性質をもつ．

(1)　デッドロックを生じない．
(2)　ステップの再実行の必要がない．
(3)　要求されたステップは必ず有限時間内で実行する．

4　分散型データベースにおけるスケジューラの役割

地理的に分散したn個のサイトそれぞれに上述の様なデータベースシステムが構築されているモデルを考え，それらは通信制御機能を通じて結合されているとする．各サイトは，独立したデータベースシステムを構成しているが必要に応じて他のサイトのデータに通信機能を通してアクセスできるものとする．サイトiに含まれるデータ項目の集合をD_iとする．システム内で発せられる全トランザクションの集合を**T**,全データ項目の集合をD，つまり

$$D = \bigcup_{i=1}^{n} D_i$$

とする．システム全体でのスケジュールは前と同様(STEP(**T**),<)によって表わされる．あるトランザクションTがアクセスするデータ項目の集合を　D(T)とする．あるiに対し$D(T) \subseteq D_i$の時，Tを局所的なトランザクション，そうでなければ全域的

なトランザクションと呼ぶ．また，サイトiでの先読みスケジューラを $CS_i(C)$，そのTIOグラフ，ATIOグラフをそれぞれTIO_i，$ATIO_i$と書く．TIO_i（$ATIO_i$）はD_i内のデータ項目に関する操作をともなうトランザクションの集合に関するもので，サイトiの局所的トランザクションおよびD_iに関係した全域的トランザクションに対応する節点から成る．システム全体としてのTIOグラフ（ATIOグラフ）は，各TIO_iグラフ（$ATIO_i$グラフ）中の同じトランザクションの節点を同一の節点として重ねて全体を一つにまとめたものであり，TIO（ATIO）と記す．各サイトのスケジューラ$CS_i(C)$の動作を協調させることによって，すべてのサイトでの実行順序が常にシステム全体での直列可能スケジュールを構成するようにすることが，我々の最大の目的である．

5　分散型データベースシステムのモデル

ここで，本稿で対象とする分散型データベースシステムのモデルに関する仮定を列挙する．

(1) サイト数＝n．各サイトには1からnまでのサイト番号が割り当てられている．

(2) 各サイトは同一の機能を持つ．

(3) 各サイトはクロックを持ち（全クロックが正確に一致している必要はない）それに基づいて時刻印を発行することが出来る．（ただし，時刻印の末尾にはサイト番号を付し，全システムで同一の時刻印を生成することはないようにする．）

(4) サイトiで受付けたトランザクションの各ステップは，それが実行されれば有限時間で終了する．全てのステップが完了すると，サイトiにその旨伝える．

(5) サイト間でデータの重複はない．すなわち，全てのi≠jに対し

$$D_i \cap D_j = \phi .$$

(6) 各サイトiの構成は，図２の通りである．各構成要素は以下の役割をもつ．

トランザクション処理装置：　トランザクションの受付け，及び実行に先立つ処理．

通信装置：　他のサイトとの通信制御．

スケジューラ：　サイトiのデータ項目D_iに関するスケジュールを行う先読みスケジューラ．

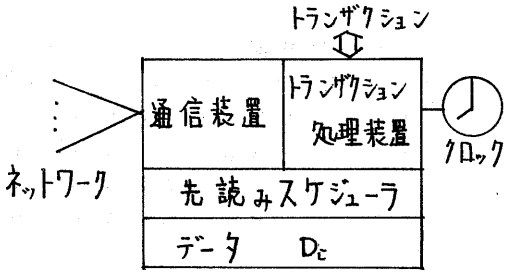


図２　　各サイトiの構成

6　分散型データベースシステムの並行処理制御アルゴリズム

各サイトiのトランザクション処理装置および先読みスケジューラは次に示す動作をすることによって並行処理制御を実行する．

A トランザクション処理装置の動作

(1) 局所的なトランザクションTを新しく受け付けたとき：D(T)⊆D_jをみたすサイトjの先読みスケジューラに直接登録し，登録が済んだ旨Tに知らせる．

(2) 全域的なトランザクションTを新しく受け付けたとき：

(2-1) Tにサイトiの時刻印t_iを与え，それと共に関係するサイトへその読取り集合，書込み集合およびt_iを送り，登録可能かどうかを問い合わせる．

(2-2) Tに関係する全サイトからの返答がそろったら，それらに付された最大の時刻印を見つける．それをt_{max}とする．（後述の様に各サイトは受け取った時刻印で実行可能かどうか，もし不可能なら処理可能時刻を示した時刻印を返答する．）t_{max}をTの登録時刻印と定め，登録の済んだ旨Tに知らせる．また，このときサイトiのクロック値がt_{max}より小さければ，時刻をt_{max}まで進める．

(3) 自己のサイトで登録済のトランザクションTのあるステップの実行要求を受取ったとき：

(3-1) Tが局所的トランザクションならば，そのまま関係サイトの先読みスケジューラへ送る．Tが全域的トランザクションならば，その登録時刻印tと共に，要求されたサイトへ送る．

(3-2) ステップの実行が済めば，結果をトランザクションTへ返す．

(4) 他のサイトjから全域的なトランザクションTの登録可能性を問われた時：

(4-1) 自己のサイトのクロックの値 t_iとTに付与された時刻印 t'を比べる．

$t' > t_i$の時：　自己のクロックをt'まで進め，実行可能であると返答すると共に先読みスケジューラに(T, t')で仮登録する．

$t' < t_i$の時：　自己の先読みスケジューラに時刻印t'のままで実行可能かどうか問い合わせる．可能ならその旨返答し先読みスケジューラにそのトランザクションを(T, t')で仮登録する．不可能ならt_i以上の時刻印であれば実行可能であると返答し(T, t_i)で先読みスケジューラに仮登録する．

(4-2) 仮登録が済んだ旨サイトjへ伝える．

(5) 他のサイトjから登録済のトランザクションTのあるステップの実行要求を受取ったとき：

(5-1) Tが全域的トランザクションであり，(T, t')の形で仮登録されているならば，Tの登録時刻印tを用いて，(T, t)の形で本登録する．Tが局所的トランザクションであるかあるいはすでに本登録されている全域的トランザクションである場合には直ちに次へ進む．

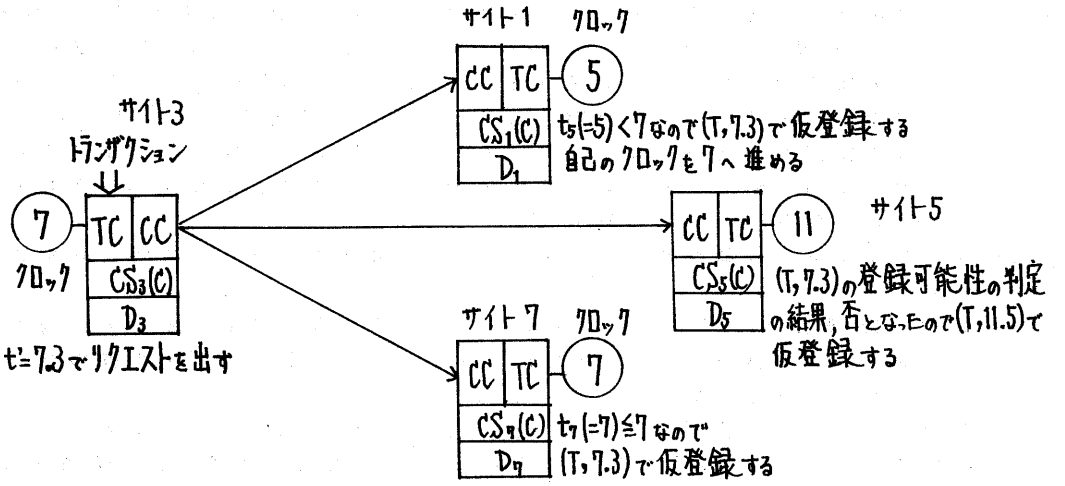
(5-2) 自己の先読みスケジューラへ要求されたステップを送る．

(5-3) ステップの実行が済めば，結果をサイトjへ返す．

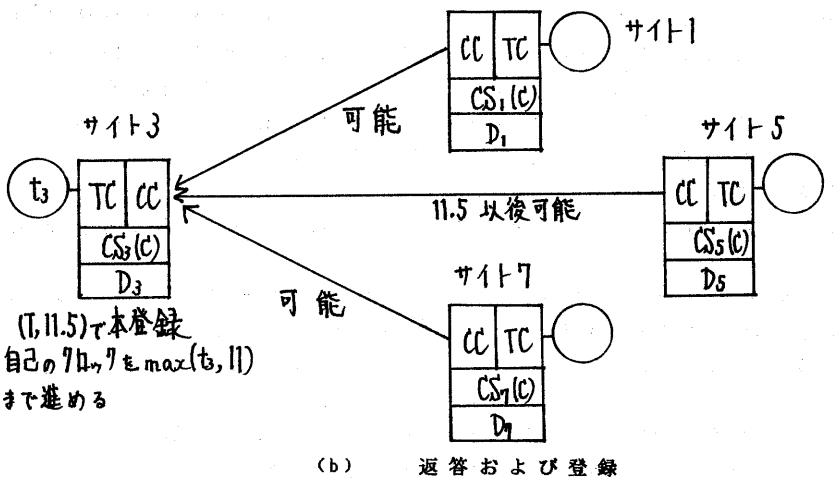
B 先読みスケジューラの動作

サイトiの$CS_i(C)$は集中型データベースの場合［IKK 85］と本質的に同一であるが，全域的トランザクションの扱いのみが若干ことなる．$ATIO_i$を構成するとき，時刻印の付いた全域的トランザクション(T_a, t_a)と(T_b, t_b)の各対に対し，$t_a < t_b$ならばT_aからT_bへ制約枝を付し，T_aがT_bの先に直列化されるとの制約を置く（$t_b > t_a$の場合は逆の制約を置く）．すなわち，$CS_i(C)$はこの制約の下で完成テストが成功するときに限り，要求のあったステップの実行を許可する．

図３にこのアルゴリズムによる全域的なトランザクションの実行手順の例を図示する．



(a)　リクエスト送信



(b)　返答および登録

（CC：通信装置，
TC:トランザクション処理装置）

図３　全域的なトランザクションの登録手順

7　アルゴリズムの正当性

ここでは，６で述べたアルゴリズムは決してデッドロックを生じることはなく，またすべてのトランザクションは申し込み後，有限時間で必ず実行されることを証明する．

補題1 このアルゴリズムが制御している分散型データベースシステムは，デッドロックを起こさない．

証明 各サイト内のスケジュールはCS(C)によって行われているので問題はない．よって，システム全体にかかわるデッドロックのみを考える．

全域的なトランザクションによってデッドロックが生じたとする．すなわち，あるサイトにおいて，ある全域的なトランザクションT_aのあるステップが，別の全域的なトランザクションT_bのあるステップの実行を待っている（局所的トランザクションのステップを介する場合もある）．同様に，そのT_bが他のサイトにおいてまた別のトランザクションT_cを待ち，この様にして得られるトランザクションの系列の最後のトランザクションT_dが最初のトランザクションT_aを待ってTIOグラフに閉路ができ，デッドロックを生じているのである．ところで，全域的なトランザクション間には前述したように，時刻印による制約枝が付いており，先読みスケジューラは，この制約下で直列可能スケジュールを出力する．従って，上述の様に，T_jがT_iを待つという状態が生じているととT_iの時刻印t_iとT_jの時刻印t_jは$t_i < t_j$を満たす（つまり，$t_i > t_j$ならば，T_jのステップはいつもT_iのステップより先に実行して矛盾を生じないから，T_iのステップの実行を待つことはない）．その結果，デッドロックの閉路に沿って，トランザクションの時刻印は単調に増加して行くが，これは閉路を構成することに明らかに矛盾する．□

次に，全てのトランザクションが有限時間内で実行できることを示すには，サイトiにおいて自己のクロックt_iを用いた(T, t_i)の形のトランザクションは常に$CS_i(C)$に受理可能であること（つまり，時刻印にともなう制約枝を加えても，先読みスケジューラの機能が保持されること），および(T, t')の形で仮登録の済んだトランザクションTに対し，$t \geq t'$をみたす(T, t)ならば常に受理できることを言わねばならない．とくに後者の性質は任意の先読みスケジューラ$CS_i(C)$において成立するとは言えず，ある程度限定する必要がある．[IKK 85]では，実用の可能性の高いCS(C)として，取り消し異常を持たない2種の先読みスケジューラCS(WW)と$\overline{CS}$(WRW)を導入したが，これらに対してはこの性質を示すことができる．

補題2 サイトi（ローカルクロックt_i）において，t_iなる時刻印を持ったトランザクションは必ず受理される（時刻印にともなう制約枝を付しても，$CS_i(C)$の機能が停止することはない）．

補題3 サイトiの$CS_i(C)$を前述のCS(WW)あるいは$\overline{CS}$(WRW)によるスケジューラとする．このとき，t'なる時刻印を持つ全域的なトランザクションT_iをすでに受理している$CS_i(C)$は，$t(>t')$でTの再登録を要求されたとき必ず受理することがいえる．

証明 最初にTが受け付けられたのは，Tを受理してもATIO$_i$がDITSを持つから

である．T以外のトランザクションのステップでTの受付後に実行されたものがあったとしても，それによって生じる制約枝を付け加えたATIO$_i$は依然としてDITSを持っている（完成テストはこの性質を保証する）．CS(WW)と$\overline{\text{CS}}$(WRW)においては，DITSの存在はATIO$_i$から得られるATIO'$_i$［IKK 85］に閉路が存在しないことと同値である．そこで，ATIO$_i$において時刻印がt'であるとして仮登録してあったTを時刻印t（＞t'）に変更することを考える．ATIO$_i$において，Tから出る枝は，Tのどのステップも未実行であることを考えると，時刻印に伴う制約のみであるが，時刻印の変更の結果そのような出枝が新しく加わることはない．一方，Tに入る枝の中には，新しいものも存在するが，取り消し異常を持たないCS(WW)と$\overline{\text{CS}}$(WRW)では，その様な枝は時刻印によるものに限ることを示せる．さて，ATIO'$_i$の変更後新しくTを通る閉路を生じたとしよう．この閉路中，Tの前にTa，後にTbがあるとすれば，上の議論から枝(Ta,T)と(T,Tb)はともに時刻印の制約枝でなければならない．しかし，時刻印の制約枝の定義から，これは制約枝(Ta,Tb)の存在を意味し，Tを経由しない閉路が存在していたことになる．よって閉路はTを移す前から存在していたことになり矛盾が生じる．従って，CS$_i$(C)はTの再登録を受理する．□

補題4　このアルゴリズムによって制御されている系全体のTIOグラフ　TIOは，DITSを持つ．

証明　各サイトはCS$_i$(C)によって制御されているのでTIO$_i$はDITSを持つ．TIOは各サイトに現われる全域的なトランザクションで，同一のものを同一節点に対応させるようにTIO$_i$を重ねたものである．つまり，全域的なトランザクションの節点を時刻印軸上に並べ，局所的なトランザクションの節点をTIO$_i$通りにそれらに接続したものと考えることができる．

D$_i$に関する仮定から，TIO$_i$中のあるインターバル（同じ節点から出て，同じラベルを運ぶ枝の集合）を考えるとそれと同じラベルを持つインターバルは$i \neq j$なるTIO$_j$中には存在しない．つまり，各TIO$_i$はDITSをもち，全域的なトランザクションは時刻印によって順序づけられているので，TIO$_i$における順序とTIOにおける順序が矛盾することはない．換言すれば，このようにして作られたTIO中のインターバルはすべて，左から右へ向かい，かつ重ならないようになっている．従って，TIOはDITSを持つ．□

定理2　分散型データベースに対する第6章の先読みスケジューラのアルゴリズムは，常に直列可能スケジュールのみを出力し，しかも，デッドロックを決して生じず，また，システムに入力された全てのトランザクションを有限時間内に実行するという性質を持つ．

証明　補題1により，システムはデッドロックを起こさない．又，補題2と3により，トランザクションは必ず仮登録でき，さらにそのようなトランザクションは必ず本登録され，その後実行されることが分かった．更に補題4によりその実行結果は，システム全体として直列可能スケジュールをあたえることが証明で

きた．従って，このアルゴリズムは全システムを矛盾なく制御する．□

8　むすび

この論文では，先読みスケジューラを用いた分散型データベースシステムの並行処理制御の方法を提案し，その正当性を証明した．先読みスケジューラに対する"仮登録"を用いることによりトランザクションは拒否され続けることが無くなった．又，時刻印を用いることによりデッドロックや直列可能性の問題を解決できた．

ところで，分散型データベースシステムでは，大部分のトランザクションが局所的であり，全域的なものは例外的に生じると考えるのが普通であろう（そうなるようにデータ項目の配置を定める）．この様な状況下では，

a) 全域的トランザクションが生起しない場合は，n個のサイトが独立に，集中型の場合と同じ効率で動作する，

b) 全域的なトランザクションの介入による局所的トランザクションの処理効率の低下を極力避ける，

ような設計が望ましい．従来の2相ロックや時刻印方式では，とくにb)を保証することが難しいと思われる．本報告の方式では，各サイトのCSi(C)において，全域的トランザクションのみが，時刻印による制約を受けるので，局所的トランザクションの並行処理効率に大きな影響を与えることはなく，a)とb)の性質を保証することができ，実用性の観点からも意味があると考えられる．

今後，スケジュールの効率や障害回復機構など実際上の問題を詳細に検討し，実用の可能性を明らかにしていくことが必要であろう．

謝辞

本研究を行うに当たり熱心に検討頂いた山下雅史博士（広島大）に深く感謝します．なお，本研究は一部文部省科学研究費によるものである．

参考文献

[BeG 81] P.A.Bernstein, N.Goodman: Concurrency Control in Distributed Database Systems, ACM Computing Surveys, vol.13 No.2, June 1981 pp.185-224.

[IKK 85] 茨木俊秀，亀田恒彦，加藤直樹： Cautious Schedulerによる並行処理制御，情報処理学会　データベース研究会　1985年7月．

[IKM 83] T.Ibaraki, T.Kameda, T.Minoura: Disjoint-interval topological sort: A useful concept in serializability theory, Proc. 9th International Conf. on VLDB, Florence Italy,89-91, Oct/Nov 1983.

[Koh 81] W.H.Kohler: A survey of Techniques for Synchronization and Recovery in Decentralized Computer Systems, ACM Computing Surveys, vol.13 No.2, June 1981, pp.149-183.