データベースシステム 91−11
(1992. 11. 5)

マルチメディアデータベースにおける同時実行制御手法の考察

坂上 雄一† 宮尾 淳一‡

† 広島大学大学院工学研究科 ‡ 広島大学総合科学部

マルチメディアデータベースにおける同時実行制御手法と，その手法のシミュレーション実験による評価について述べる．マルチメディアデータ処理を行う場合，トランザクションによっては中止・再実行が不可能なものと可能なものが考えられる．これらのトランザクションのタイプをそれぞれtype1，type2と呼び，トランザクションを区別した．また，トランザクション実行により生じる不整合をある程度許容する許容不整合の概念により，従来の直列化可能性を拡張した緩和直列化可能性の概念を定義した．そして，緩和直列化可能性の概念，および，トランザクションのタイプ分けを行うことで同時実行制御の並列性の向上を目指した．さらに，この方法に基づいたスケジューリングアルゴリズムTYPEを試作し，ランダムデータを用いたシミュレーション実験により，2PLによる結果と比較した．その結果，処理の複雑さのためにオーバーヘッドは2PLの44%増であったが，13%の並列制の向上が認められた．また，2PLに比べて平均20%程度ロック要求数が減少することから，処理を要求してから開始するまでのブロッキングが減少していることも分かった．

A Study on Concurrency Control Methods for Multimedia Database

Yuichi SAKAUE † Jun'ichi MIYAO ‡

† Graduate School of Engneering, Hiroshima University
‡ Faculty of Integrated Arts and Sciences, Hiroshima University

In processing of multimedia data, there exist two types of transactions, i.e. , non-abortable and abortable. We call them as type1 and type2, respectively. Moreover, we introduce a concept of permissible inconsistency that permits some inconsistency which is produced in transaction execution. Based on it, we define a permissible serializability(PSR) that extends ordinary serializability. By using type division and PSR, we develop a concurrency control algorithm to improve parallelism, and show some simulation results compared with 2PL.

1 まえがき

データベースシステムにおいて，画像データや音声データを含むマルチメディア化が進んできている．このようなデータは，それ自身の持つ性質上，大規模で実時間性を有する．これらを従来データと同様に取り扱うことは，実行効率の大きな低下を招くと考えられる．また，実時間性を有することから，従来の同時実行制御とは異なり，処理途中の内部データの提示を行いながら処理を進めていく必要が生じる．このような実行形式では，従来手法のようにアボート再実行 (abort-resubmit) を行うことは不可能となる．

また，実行が必ずしも直列化可能でなくとも許容でき，処理を正しいものと認めることができる場合がある．例えば，再生 (read only transaction) に代表されるように，人間に対するデータの提示においてある程度の不整合を許す処理が考えられる．たとえ直列化可能ではない処理を行ったとしてもそれを認識することができないという意味において正しさを保証している．また，統計処理のようにデータの平均化が行われるような場合にも不整合が許容できることが考えられる．

本稿では，マルチメディアデータモデル (画像) を考慮した同時実行制御手法を提案する．提案手法では，動画像・音声などを含むマルチメディアデータの特徴として次のC1，C2に注目する．

(C1) 非常に大規模なデータがある．
(C2) 処理を開始すると中止 (abort)・再実行が困難なものがある．

このような特徴をもつデータに従来の同時実行制御を適用すると，基本的に次のような問題点P1，P2が生じる．

(P1) 悲観的手法において，大規模データにロックを掛けると並列性が著しく低下する．
(P2) 楽観的手法では，処理途中で中止させられる可能性がある．

本手法では，悲観的手法と楽観的手法を組み合わせ，さらに，トランザクションを中止可能なもの (type2) と，中止不可能なもの (type1) に分けること，及び許容不整合の概念を導入することにより，上述の問題点P1，P2の解消を図る．ここで，許容不整合とは，トランザクション処理の交差実行により，データベースの処理が不整合 (inconsistent) 状態になったとされる場合でも，その程度によりまったく影響を与えないか，あるいは影響が軽微であり不整合が無視できる場合である．例えば，動画像再生などの場合には連続した複数のフレームに誤りが混入しなければ無視できる場合が多いため，許容不整合となる可能性が高いと考えられる．

本手法の概要を説明する．type1 トランザクションは拡張した2相ロックで処理を行う．type2 トランザクションは読み書きするデータセットに type1 とは異なるロックを行い，拡張した直列化可能グラフ (Serialization Graph) を構成しつつ処理を行う．但し，SG グラフでサイクルが生じる場合は，中止か不整合無視を適当なトランザクションに問い合わせる．

許容不整合と同様な概念は文献 [1] で述べられているが，[1] の条件判定はスケジューラが静的に行うのに対し，本手法では，トランザクションがその状態において動的に判断する．これにより，正確な判断と並列性向上が期待できる．また，文献 [3]，[4]，[5] では直列化可能性を保証するより広いクラスを定義することで並列性向上が期待できるが，手続きが複雑になること，およびマルチメディア処理が考慮されていないことなどの問題点がある．

2 準備

ここでは同時実行制御に関する準備を与える ([2]参照)．

[定義1] トランザクションはオペレーション列として定義される．ここで，オペレーションとは書き込み (Write,w)，読み込み (Read,r)，コミット (Commit,c)，アボート (Abort,a)，無視 (Ignore,i)，の5種類で構成される．トランザクション T_iのオペレーションの集合を O_iとすると，

$$O_i \subseteq \{r_i[x], w_i[x]\} | x \text{ はデータ項目 } \} \cup \{c_i, a_i, i_i\}$$

となり，T_iのオペレーションには全順序 (total order) $<_i$ が付いている． □

トランザクションの実行形式 (execution) は，以下のようにトランザクションの原子性 (atomicity) を保障しなければならない．

1. 各トランザクションをバッチ型式で直列に実行した場合，どの実行系列でもデータベース状態は正しい．
2. もしトランザクションが正常終了するならば，その影響は永続性を持ち，もしトランザクションが正常終了しなければ，全く影響を残さない．

また，トランザクションの型によってはある程度の不整合状態を許すトランザクションが存在する．そのようなトランザクションが不整合性を含んで実行された場合には，以下のことが成立しなければならない．

3. もしトランザクション自身がアボートしなければ，トランザクションの出力結果と実行結果としてのデータベース状態 (database state) は正しい．

[定義2] 2つのオペレーション p，q が以下の条件を満足するとき競合オペレーション (conflicting operation) であるとする．

1. それぞれ異なるトランザクションに属する．

2. 同一のデータ項目にアクセスする．

3. p，qの内の少なくとも1つが書き込みである．

□

次に，完全ヒストリ (complete history) の定義を以下に示す．

[定義 3] T上のヒストリHは以下の順序関係$<_H$を持つ半順序である．ここで，T= $\{T_1, T_2, \ldots, T_n\}$ はトランザクションの集合である．

1. H= $\cup_{i=1}^{n} O_i$

2. $<_H \supseteq \cup_{i=1}^{n} <_i$

3. 任意の競合オペレーション p,q ∈ H に対して，p $<_H$ q または q $<_H$ p となる． □

一般に，ヒストリとは完全ヒストリのプリフィクスである．つまり，ヒストリ中に現われるトランザクションは必ずしもコミットオペレーションを含んでいるとはいえない．もし，$c_i \in H (a_i \in H)$ ならば，トランザクション T_i は committed (aborted) であるという．もしトランザクション T_iが committed , aborted でもなければ active であるという．ヒストリHが与えられたとき，Hの committed projection C(H) とは，Hのコミット済みのトランザクションに属していないすべてのオペレーションを取り除くことでHから得られるヒストリである．

ここで，直列化可能の定義を説明するために，ヒストリの等価性と，直列化可能ヒストリについて説明する．

次の2つの条件を満足する場合，2つのヒストリH，H'は等価 (equivalent) であるとする．

1. それらは同一のトランザクション集合上に定義されており，同一のオペレーションを持っている．

2. それらは同一方法で，アボートされていないトランザクションの競合オペレーションを順序付ける．つまり，トランザクション T_i, T_j $(a_i, a_j \notin H)$ にそれぞれ属している競合オペレーション p_i, q_j に対して，もし $p_i <_H q_j$ ならば $p_i <_{H'} q_j$ となる．

[定義 4] H中に現われるあらゆる2つのトランザクション T_i, T_j に対して，T_i の全てのオペレーションが T_j の全てのオペレーションの前に現われるか，または，その逆が言える場合に，完全ヒストリHは直列 (serial) である．つまり，直列ヒストリは異なるトランザクションのオペレーションの交差実行が存在しないものをいう．そして，committed projection C(H) が直列ヒストリ H_s に等価であるなら，そのヒストリは直列化可能 (SR) であるという．厳密に言えば競合直列化可能 (Conflict SR : CSR) であるという． □

直列化グラフ (Serialization Graph : SG) と呼ばれる，ヒストリから導出できるグラフにより，ヒストリが直列化可能であるかどうかを決定することができる．

[定義 5] H を $T = \{T_1, T_2, \cdots, Tn\}$ 上のヒストリであるとする．H に対する直列化グラフ (SG(H) と記す) は，次のように定義される有向グラフである．

1. 節点はH中のコミットされたトランザクションである．

2. T_i のオペレーションの1つが T_j のオペレーションに対して先行し，かつ，競合している場合，$T_i \rightarrow T_j$ $(i \neq j)$ の枝を持つ． □

[定理 1] (直列化可能定理 (*SerializabilityThcorem*))[2]
ヒストリが直列化可能 $\Longleftrightarrow$ SG(H) は acyclic □

SGを用いてSRを保証しながら実行を進めていく場合，上述のSGの定義のように全てコミットされたトランザクションのオペレーションを対象とするわけにはいかない．実際 active トランザクションの実行過程においてSGを作っていかなければならない．これをSGと区別して Stored SG (SSG) と呼ぶ．

本稿で用いる labeled-SG(ラベル付き直列化可能グラフ) について説明する．labeled-SG は従来のSGの枝に処理内容 (r または w) を表すラベルの付いたものである．すなわち，枝の終点のトランザクションの競合オペレーションが読み込みの場合 r(r-labeled edge)，書き込みの場合 w(w-labeled edge) となる．ただし，r と w が同時に付く場合にはrのみを付ける．また，実際にアルゴリズムで用いられるのは labeled-SSG となる．

[例 1] 次のヒストリHについて考える．
$H = w_1[x]w_2[z]r_2[x]w_2[y]w_1[y]r_1[z]w_1[z]c_1c_2$
H上の $w_1[y]$ までの実行により

となり，さらに $r_1[z]$ の実行によりラベルrが付く．ここで，rとwが付くが，rのみを残して，

と labeled-SSG が変更される．r-labeled edge では連鎖アボート (cascading abortion) により，あるトランザクションのアボートが他のトランザクションのアボートを引き起こすことがある．連鎖アボートを考慮に入れるために，r-labeled edge に変更する必要が生じる． □

[定義 6] 緩和直列化可能 (Permissible SR : PSR) について定義する．ヒストリ中の任意の競合オペレーションを $p_i[x], q_j[x] (i \neq j)$ とする．ヒストリの半順序を $<_H$ とする時，$p_i[x] <_H q_j[x]$ または $q_j[x] <_H p_i[x]$ のいずれにしてもデータベース状態が変化しない時，$q_j[x], p_i[x]$ は交換可能であるという．ヒストリが競合

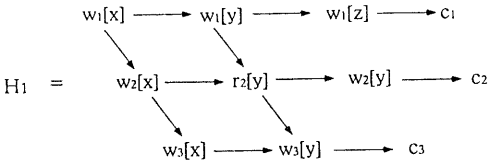
直列化可能に属していない（SGにサイクルを含む）場合，任意の交換可能オペレーションを交換することにより競合直列化可能を満足するなら，ヒストリHは緩和直列化可能であるという．□

3 クラス階層

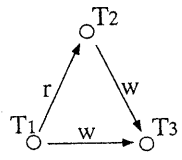
3.1 競合直列化可能と緩和直列化可能

ここで，競合直列化可能(CSR)，緩和直列化可能(PSR)について例を用いて説明する．

次のヒストリH_1を考える(図1参照)．



(a) H_1



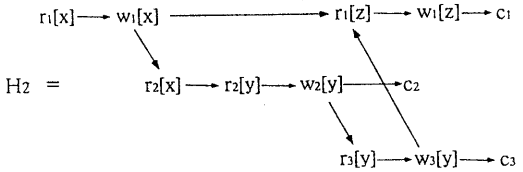
(b) labeled-SG(H_1)

図1 ヒストリH_1とlabeled-SG(H_1)

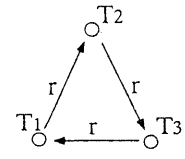
$H_1 = w_1[x]w_2[x]w_1[y]w_3[x]r_2[y]w_2[y]w_3[y]w_1[z]c_1c_2c_3$

この場合，H_1の全てのプリフィクスH_1'に対して，labeled-SSG(H_1')にはサイクルは存在しない．よって，常に競合直列化可能の定義を満足している．

また，次のヒストリH_2を考える(図2参照)．



(a) H_2



(b) labeled-SG(H_2)

図2 ヒストリH_2とlabeled-SG(H_2)

$H_2 = r_1[x]w_1[x]r_2[x]r_2[y]w_2[y]r_3[y]w_3[z]r_1[z]w_1[z]c_1c_2c_3$

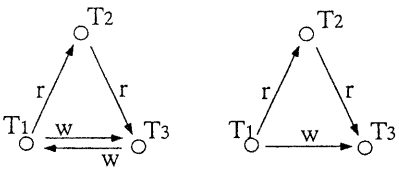
もし，$w_3[z]$がzの初期値と等しい値を書き込んだ場合，$w_3[z]$と$r_1[z]$は交換可能となる．$w_3[z]$と$r_1[z]$を交換したヒストリをH_{21}とする．

$H_{21} = r_1[x]w_1[x]r_2[x]r_2[y]w_2[y]r_3[y]r_1[z]w_3[z]w_1[z]c_1c_2c_3$

またもし，$w_3[z]$と$w_1[z]$が同一の値を書き込む場合，$w_3[z]$と$w_1[z]$は交換可能となる．$w_3[z]$と$w_1[z]$を交換したヒストリをH_{22}とする．

$H_{22} = r_1[x]w_1[x]r_2[x]r_2[y]w_2[y]r_3[y]r_1[z]w_1[z]w_3[z]c_1c_2c_3$

ここで，H_{21}とH_{22}のlabeled-SGを図3に示す．同図から分かるように，H_{21}ではサイクルが存在するために競合直列化可能とはならないが，H_{22}ではサイクルが存在しないので競合直列化可能である．よって，H_2は交換可能なオペレーションを交換することにより競合直列化可能ヒストリH_{22}に変換できるので，緩和直列化可能である．

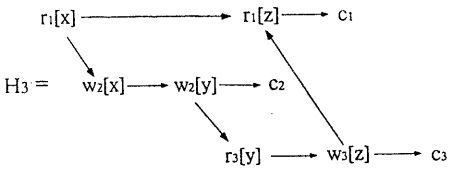


(a) labeled-SG(H_{21}) (b) labeled-SG(H_{22})

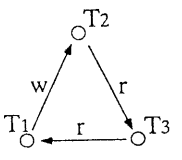
図3 ヒストリH_{21}とH_{22}のlabeled-SG

変換可能なオペレーションは読み出し専用トランザクション(read only transaction)に対して顕著である．実際，読み出しオペレーションはトランザクション自身の順序を守る必要がある（ヒストリの定義より）だけで，ヒストリ中のどの部分に配置されたとしてもデータベース状態が変化することはない．つまり，読み出し専用トランザクションにおける読み出しオペレーションは，それぞれに対する競合オペレーションと常に交換可能である(実際に読み出された値が有効かどうかは別問題である)．よって，サイクル中に読み出し専用トランザクションが存在する場合，そのオペレーションをそれらが競合するオペレーションと交換することで必ず競合直列化可能とすることができる．

その例として，次のヒストリH_3を考える(図4参照)．



(a) H_3



(b) labeled-SG(H_3)

図4 ヒストリH_3とlabeled-SG(H_3)

$H_3 = r_1[x]w_2[x]w_2[y]r_3[y]w_3[z]r_1[z]c_1c_2c_3$

ヒストリH_3は読み出し専用トランザクションT_1のオペ

レーション $r_1[z]$ の実行により競合直列化可能とはなっていない．しかし，$w_3[z]$ と $r_1[z]$ は常に交換可能であるので，$w_3[z]$ と $r_1[z]$ を入れ替えることにより，

$H_{31} = r_1[x]w_2[x]w_2[y]r_3[y]r_1[z]w_3[z]c_1c_2c_3$

のヒストリを作ることができる．このヒストリ H_{31} は競合直列化可能を満足しているので，ヒストリ H_3 は緩和直列化可能である(図5参照)．読み出した値についての保証（読み出した値が有効かどうかの決定）はトランザクション自身においてなされるものとする．

ここで，CSR，PSRのクラス階層について説明する(図6参照)．

CSRはlabeled-SGにサイクルを認めない．そして，PSRはlabeled-SGにサイクルが存在する場合，交換可能なオペレーションの入れ替えによりCSRとなるヒストリの集合を表しているので，$CSR \subseteq PSR$ は明らかである．さらに，H_2，H_3 が存在することにより，次の定理が成立する．

[定理2]

競合直列化可能 $(CSR) \subsetneq$ 緩和直列化可能 (PSR) □

3.2 許容不整合の形式的判定

不整合が許容不整合であるかどうかの判定は一般に容易ではないが，次のようにして形式的に容易に判定する方法が考えられる．すなわち，許容不整合に制限を付け，それを越えた場合は不整合にすることを考える．

1. 整合性を満足するデータ間に存在する不整合データの個数を制限する

ここでは，読み込みトランザクションについて考える．

データ項目X（$X[f_0], X[f_1], \ldots, X[f_n]$）にアクセスするトランザクションを T_a とする．トランザクション T_a がactiveになった時刻を t_a，コミットされた時刻を t_c とする．また，$X_0[f_i](i = 0, \cdots, n)$ をトランザクション T_a がactiveになった時点のデータ項目Xのiフレームの値とする．トランザクション T_a は第iフレームと第jフレームにおいて値 X_0 にアクセスし，その間のデータ項目に関しては何度かの更新の後にアクセスしたものとする．図7では，第kフレームにm回の更新が行われた後に T_a がアクセスしたことになる．ある区間δを設定し，$\delta \leq j - i$ となる場合は不整合とする．

2. 不整合データの総個数を制限する

ここでは，不整合読み込みデータの総個数に制限を付ける．各不整合が許容されたからと言って，不整合全てが許容されるとは限らない．複数のデータを集めて処理を行う場合，読み込んだ各データが許容されたとしても，そこで生じる各誤差が全体として許容制限を越えてしまうことが考えられる．例えば，複数のデータを取り込んでその総計を求める場合，いくつかのデータが他のトランザクションにより更新されていたとする．更新幅が許容誤差として認められたとしても加算された誤差が書き込みの時点でトランザクションの許容制限を越える場合に不整合とする．

4 システムモデル

4.1 マルチメディアデータモデル

本稿では，個々のマルチメディアシステムに捕われることなく同時実行制御を議論するため，抽象化したデータモデルを採用する．以下では，本稿の記述に関連したデータモデルを示す．

(1) 単純データx：従来のデータベースで扱うデータで，データ項目xは値V(x)を持つ．

(2) 時系列データX：動画像，音声などの一連のデータであり，Xはさらにフレーム $X(f_1), X(f_2), \cdots, X(f_n)$ から構成され，各フレーム単位で読み込み，書き込みなどの操作を行うことができる．

4.2 データベースシステムモデル

ここでは，図8に示されるようなシステムモデルを考える．通常のモデルとは異なり，トランザクションマネジャ（TM）とスケジューラを統合したスケジューラ（Integrated

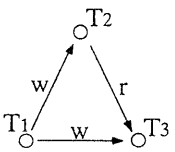


図5 labeled-SG(H_{31})

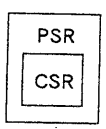


図6 CSRとPSRのクラス階層

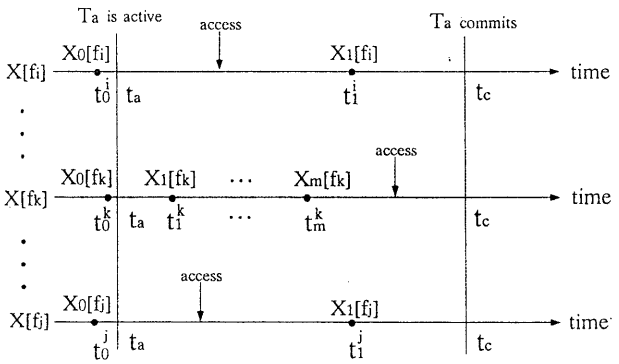


図7 許容不整合の個数制限

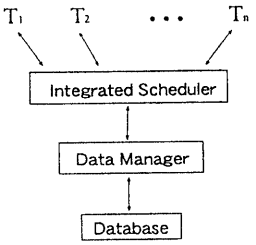


図8 データベースシステムモデル

Scheduler) を考える．トランザクション・スケジューラ間はメッセージ通信により処理を行なう．

4.3 トランザクションモデル

トランザクションは，中止(abort)の可能・不可能，許容不整合の無視(ignore)可能・不可能の２つの概念により，４種類のタイプのトランザクションに分類される．これらを本手法では，表１に示されるように，type1， type2 の２種類に分ける．

中止可能・不可能というのは，アボートを実行することができるかどうかを表している．無視可能・不可能というのは，許容不整合を認めることができるか，全く認められないということを表している．

表1 トランザクションタイプモデル

abort 可能	ignore 可能	type
False	False	type1
True	False	type2
False	True	type2
True	True	type2

4.4 メッセージ構造

メッセージにはトランザクションからスケジューラへのメッセージ(TS-message)と，スケジューラからトランザクションへのメッセージ(ST-message)の二種類が存在する．TS-message は(1)start-up-message，(2)operation-message であり，ST-message は(3)conflict-message である．

(1) start-up-message はトランザクションが実行を開始する場合のメッセージであり，read-set（読み読むデータ集合），write-set（書き込むデータ集合）を宣言する．
(2) operation-message は各データ項目に対する操作，中止・無視などの判定を行う．
(3) conflict-message は不整合が発生したことを，トランザクションに通知し，無視あるいは中止のいずれかの operation-message を要求する．

5 スケジューリングアルゴリズム

ここでのスケジューリング手法はロッキングを基本にしたものを提案する．type1 トランザクションの実行方針はアクセスするすべてのデータ項目に対するロックを最初に一括して行なうことであり，type2 トランザクションの実行方針はアクセスするすべてのデータ項目に対するチェックを最後に一括して行なうことである．ロックとチェックの違いは，アクセス透過性の違いを表している．チェックは他の操作に対してブロックされにくい，つまり，アクセス透過性が高いと言える．Lock & Check 両立性行列は表2のようになる．

スケジューラ TYPE

S1: スケジューラはキューからメッセージを取り出す．メッセージが start-up-message の場合 S2 へ進む．メッセージが operation-message の場合 S3 へ進む．
S2: 発信元のトランザクションを active トランザクションとして登録する．もしそのトランザクションが type1 の場合，読み込みデータ集合，書き込みデータ集合の情報をもとにアクセスする全データに対してロックを行う．ロックができなければメッセージをキューに戻す．
S3: 発信元のトランザクションが type1 の場合 S4 へ，type2 の場合 S5 へ進む．
S4: labeled-SG を作り，オペレーションを実行する．オペレーションの実行後，これ以後アクセスされることのないデータ項目についてはロックを解放する．オペレーションがコミットの場合は，終了するトランザクションに関する全情報をスケジューラ内から削除する．
S5: アクセスするデータ項目に対してチェックを行う．チェックが獲得できなければ，獲得できるまでブロックされる．チェックが獲得できれば，オペレーションの実行前に labeled-SG にサイクルができるかどうか検査する．サイクルができなければオペレーションを実行する．サイクルができた場合，サイクル中の type2 トランザクションの中で実行開始時刻が最も遅いトランザクションに conflict-message を送る．オペレーションがコミットの場合は，全てのデータ項目に対するチェックを解放し，終了するトランザクションに関する全情報をスケジューラ内から削除する．

表2 Lock & Check 両立性行列

		holder rc	wc	rl	wl
requester read	check	○	○	○	○
requester write	check	○	○	×	×
requester read	lock	○	×	○	×
requester write	lock	○	×	×	×

5.1 正当性

ここに出てくる記号について説明する．$p_i^a[x]$ はデータ

項目 x に操作 p を実行する type-a トランザクション T_i のオペレーションを表しているとする．また，pl はオペレーション p のロック (lock)，pc はオペレーション p のチェック (check)，pr はオペレーション p のロックまたはチェックの解放 (release) を表している．A(H)(ヒストリ H に対する active projection) はヒストリ H 中の active トランザクションのみを取り出す操作である．トランザクションの実行形式およびロック & チェック両立性行列より次の性質が満足される．

[性質 1] H をスケジューラ TYPE によって作られたヒストリであるとする．$p_i^1[x]$ と $w_j^2[x](i \neq j)$ が H 中の競合オペレーションであり $w_j^2[x] < p_i^1[x]$ であるとすると，$c_j^2 < pl_i^1[x]$ となる． □

[性質 2] H をスケジューラ TYPE によって作られたヒストリであるとする．$p_i^1[x]$ と $w_j^2[x](i \neq j)$ が C(H) 中の競合オペレーションであるとすると，その時，$pr_i^1[x] < wc_j^2[x]$ または $wr_j^2[x] < pl_i^1[x]$ となる． □

[補題 1] H をスケジューラ TYPE によって作られた，type1 トランザクションのみからなるヒストリであるとする．その時，labeled-SG(H) にはサイクルが存在しない．

(証明) 2PL に従っているので明らか． □

[補題 2] H をスケジューラ TYPE によって作られたヒストリであるとする．labeled-SSG(A(H)) にサイクルが存在し，サイクル中に少なくとも 1 つの type1 トランザクションが存在するなら，サイクル中に少なくとも 1 本の type2 トランザクションを始点とする w-labeled edge が存在する．

(証明) 補題 1 より，サイクル中には少なくとも 1 つの type2 トランザクションが存在する．サイクルが存在し，そのサイクル中に type2 トランザクションを始点とする w-labeled edge が存在しないと仮定する．サイクル中には type1 トランザクションが含まれているので，その type1 トランザクション (T_j^1) を終点，type2 トランザクション (T_i^2) を始点とする r-labeled edge が存在する．ここで，条件より T_i^2 と T_j^1 は active である．また，競合オペレーションは $w_i^2[x]$ と $r_j^1[x]$ となり，$w_i^2 < r_j^1$ であることから，性質 1 より $c_i^2 < rl_j^1[x]$ となる．これは T_i^2 と T_j^1 が同時に active であることに矛盾する．よって，サイクル中には type2 トランザクションを始点とする w-labeled edge が存在する． □

[補題 3] H をスケジューラ TYPE によって作られたヒストリであるとする．その時，labeled-SSG(A(H)) には type2 トランザクションから type1 トランザクションへの r-labeled edge は存在しない．

(証明) labeled-SSG(A(H)) に $T_i^2 \xrightarrow{r} T_{i+1}^1$ が存在すると仮定する．A(H) に対するグラフであるので，T_i^2, T_{i+1}^1 ともに active である．ここで，ts(T_i) をトランザクション T_i のタイムスタンプと呼び，T_i の実行開始時刻を表しているとする．

(1) ts(T_i^2) < ts(T_{i+1}^1) の場合

競合オペレーションは w_i^2[x]，r_{i+1}^1[x] となる．性質 1 より，$c_i^2 < rl_j^1$[x] となる．つまり，T_i^2 がコミットされた後で T_{i+1}^1 のオペレーション r_{i+1}^1[x] は実行される．ゆえに，labeled-SG(A(H)) に T_{i+1}^1 が現われる前に T_i^2 はコミットされていることになり，両トランザクションが同時に active であることに矛盾する．よって，$T_i^2 \xrightarrow{r} T_{i+1}^1$ は存在しない．

(2) ts(T_{i+1}^1) < ts(T_i^2) の場合

同様に，競合オペレーションは w_i^2[x]，r_{i+1}^1[x] となる．仮定より，w_i^2[x] < r_{i+1}^1[x] である．性質 2 より，rr_{i+1}^1[x] < wc_i^2[x] となる．よって，r_{i+1}^1[x] < w_i^2 となり，w_i^2[x] < r_{i+1}^1[x] と矛盾する．ゆえに，$T_i^2 \xrightarrow{r} T_{i+1}^1$ は存在しない．

(1)，(2) より，type2 トランザクションから type1 トランザクションへの r-labeled edge は存在しない． □

[定理 3] H をスケジューラ TYPE によって作られたヒストリであるとする．H は緩和直列化可能である．

(証明) labeled-SSG(H) にサイクルが含まれていると仮定する．まず，中止可能な場合について考える．

(1) サイクルがすべて type2 トランザクションの場合
サイクル中の任意のトランザクションを中止する．これで，labeled-SSG(H) のサイクルを切断できるので，ヒストリは直列化可能となる．よって，緩和直列化可能となる．

(2) サイクルに type1 トランザクションを含む場合
補題 2 より，w-labeled edge の始点となる type2 トランザクションが存在するので，そのトランザクションを中止することで，ヒストリは直列化可能となる．よって，緩和直列化可能となる．補題 3 より，type2 トランザクションから type1 トランザクションへの r-labeled edge は存在しないので，任意に type2 トランザクションを中止しても，連鎖アボートにより type1 トランザクションが中止されることはない．

次に，無視可能な場合について考える．もし，トランザクションが不整合を無視した場合，そのトランザクションの実行の原子性の条件 3. より緩和直列化可能である． □

6 シミュレーション結果

シミュレーションモデルを図 9 に示す．スケジューリングアルゴリズム TYPE は C 言語を用いて SPARCstation IPX 上で実現した．トランザクション数 10，オペレーション数 500 のランダムに発生させたデータを用いてシミュレーションを行った．同時実行制御のオーバーヘッドを表 3 に，出

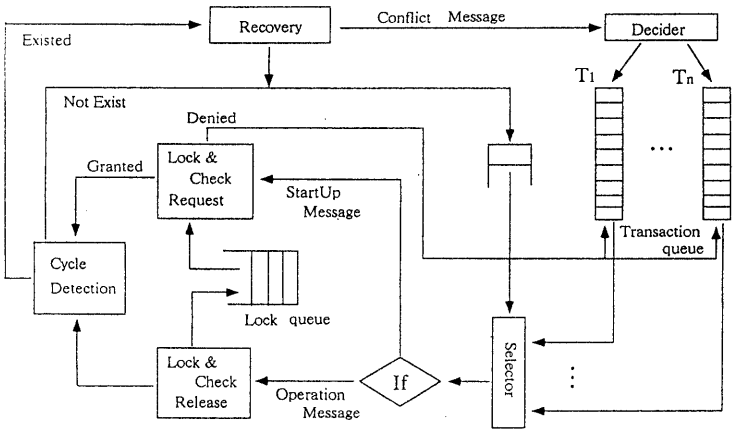


図9 シミュレーションモデル

力結果であるヒストリの論理時間を表4に，そして，ロック要求回数を表5に示す．

ここで，論理時間およびそれを求めるためのスロットについて説明する．スロットとはオペレーションを埋め込むための区間(interval)を表している．そして，スロット内のオペレーションの順序は各トランザクションおよびヒストリのオペレーションの全順序に従う．また，同一スロット内には，同一トランザクションおよび同一データ項目をアクセスするオペレーションを埋め込むことはできない．スロットにヒストリを埋め込んだときの最大のスロットの長さを論理時間とする．

表3：実時間

タイプ	2PL	TYPE (sec)
type1 only	0.14	0.18
type2 only	0.14	0.13
type1 & type2	0.14	0.20

表4：論理時間

タイプ	2PL	TYPE (slot)
type1 only	258	258
type2 only	258	193
type1 & type2	258	228

表5：ロック要求回数

タイプ	2PL	TYPE (回)
type1 only	1093	1093
type2 only	1093	658
type1 & type2	1093	864

まず，オーバーヘッドについて考察する．トランザクションがtype1のみから成る場合，TYPE法ではcycle detectionにかかる時間が余分に必要となる．type2のみから成る場合，ロック獲得により生じるブロッキングが減少するため実行効率が改善されるが，アボートによる回復のためにかかる時間により相殺されてしまい，大きな改善とはなっていない．type1，type2が混在する場合，TYPE法が2PLに比べて効率が悪くなっているのは，type2のアボートおよびtype1，type2間のブロッキングのために起こっている．

次に，論理時間を見た場合，type2トランザクションの存在のためにTYPE法のスロット数が少なくなっている．

以上より，TYPE法はアボートおよびcycle detectionにかかる時間のためにより時間がかかっているが，スロット数からわかるように，並列性の向上が見込める．

CPU性能の向上およびマルチプロセッサ環境の充実により，この時間の差は改善できると考えられる．また，画像や音声などは主に再生などの読み出しの方が頻繁に起こると考えられるので，Ignore可能なtype2トランザクションの数が増えることにより，さらに処理速度の向上が期待される．

7 あとがき

本稿では，マルチメディアデータモデルを考慮した同時実行制御手法を提案した．また，許容不整合の概念を取り入れた緩和直列化可能(PSR)を定義した．さらに，トランザクションをtype1，type2にタイプ分けすることで並列性を向上させるスケジューラTYPEを提案した．今後の課題としては，各トランザクションのデータ衝突を設定したシミュレーションにより，実際的な評価を行うことが挙げられる．

なお，本研究の一部は平成4年度文部省科学研究費補助金一般研究(B)(課題番号 04452195)分担課題による．

文献

[1] Kun-Lung Wu, et al. : "Divergence control for epsilon-serializability", Proc. Data Engineering, pp.506-515(1992).

[2] P. A. Bernstein, et al. : "Concurrency Control and Recovery in Database Systems", Addison-Wesley(1987).

[3] T. Ibaraki, et al. : "Serializability with constraints ", ACM Trans. Database Syst. vol.12, no.3, pp.429-452(1987).

[4] T. Ibaraki, et al. : "Cautious transaction schedulers for database concurrency control", IEEE Trans. Software Eng., vol.14, no.7, pp.997-1009(1988).

[5] N. Katoh, et al. : "Cautious transaction schedulers with admission control", ACM Trans. Database Syst., vol.10, no.2, pp.205-229(1985).