データベース・システム 70－1
（１９８９．３．１４）

先読みスケジューラの性能評価

河添　博之・藤原　敬子・加藤　直樹
株式会社ＱＵＩＣＫ・富士通・神戸商科大学

本論文では、データベースシステムの同時実行制御方式として提案された先読みスケジューラについて、その性能評価を計算機を使用して、シミュレーション実験を行った。このスケジューラは、トランザクション到着時にその処理単位が既知であるとの仮定の下でデッドロックやロールバックなどを起こさないものである。我々がとりあげたスケジューラは、CS(WW),CS($\overline{WRW}$)と多版方式にもとずくCS$_m$(MWW),CS$_m$(MWRW),CS$_m$(M(WR+RW$^-$),CS$_m$(M(RW+WR$^-$)をとりあげて、待ち時間とスケジューラの負荷について性能評価を行った。その結果、待ち時間はCS$_m$(M(WR+RW$^-$)とCS$_m$(M(RW+WR$^-$)が短く、負荷はどのスケジューラもあまり変わらないという結論を得た。これらの結果から、先読みスケジューラはよりよいデータベースシステムを構築するのに役立つと、考えられる。

"Perfomance Evaluation of Cautious Scheduler"

by Hiroyuki KAWAZOE(QUICK,1-28-24,Shinkawa Chuo-ku
Tokyo 104 Japan),Keiko FUJIHARA (FUJITSU) ,
Naoki KATOH(Kobe University of Commerce)

Most of the currently known transaction schedulers (e.g. those based on two-phase locking or time-stamping) use rollback. In this paper we examine practical performance of cautious schedulers,which never resort to rollback for the purpose of concurrency control under the assumption transctions on arrival predclare their read and write sets. We experiment six types of cautious schedulers CS(WW),CS($\overline{WRW}$),CS$_m$(MWW),CS$_m$(MWRW),CS$_m$(M(WR+RW$^-$)) and CS$_m$(M(RW+WR$^-$)) by computer simulation. We conclude that CS$_m$(M(WR+RW$^-$)) and CS$_m$(M(RW+WR$^-$)) are superior to others.

１．まえがき

データベースシステムにおいて、複数の命令を同時に処理するときに、なんらかの制御を行わないと、データの論理的首尾一貫性が、保たれなくなることがある。この保障を行うための制御が、同時実行制御の考え方である。この同時実行制御の手段として、現在、二相ロッキング方式や時刻印方式が、良く使用されている。しかし、これらの方式は、デッドロックやロールバックを起こす可能性がある。このような事態に一旦陥ると、データを回復するために、多大な費用と時間がかかる。

我々がここでとりあげた先読みスケジューラは、命令は到着すると同時に、そのすべての処理単位をスケジューラに知らせるという仮定のもとで、デッドロックやロールバックを起こさないというものである。

この先読みスケジューラが、優れていることは、理論上確認されているが[2,4]、実際に、計算機上、どの程度の性能を持っているのかを調べるために、いくつかのサブクラスをとりあげ、シミュレーション実験をおこなった。

２．システムモデルとスケジュールの定義

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Ｔ＝｛$T_0 T_1 \cdots T_f$｝から成る。トランザクションT_iのread operation $R_i[X]$はデータ項目Ｘの値を参照する。read operation $W_i[X]$はＸの値を更新する。ＳをＤのサブセットとするとき、readのステップ$R_i[S]=\{R_i[X] \mid X \in S\}$はＴ のread operaｰtionのこれ以上分割できない最小単位の集合である。write operation $W_i[S]$も同様に定義する。なおT_0，T_fは、初期トランザクション、最終トランザクションと呼ばれる架空のトランザクションである。T_0は、すべてのデータ項目を初期化する$W_0[D]$からなるトランザクションで、T_fは他のトランザクションがすべて完了したのちに、すべてのデータ項目の最後の値を参照する$R_f[D]$からなるトランザクションである。

トランザクションは、ステップの構成上から、次の３つのモデルに分類される。

ｉ）２ステップ モデル
　トランザクションには、２つのステップしかなく、必ず、READステップ、WRITEステップの順に到着する。
　例）$R_i[X, Y]W_i[X, Y]$

ii）READ-WRITE モデル
　トランザクションでは、READステップがすべて到着した後、WRITEステップが到着する。
　例）$R_i[X, Y]R_j[Z]W_i[X]W_j[Z]$

iii）一般モデル
　READステップ、WRITEステップが ランダムに到着する。但し、READ、WRITE両方を行うデータ項目については、必ず、READステップが、WRITEステップに先立つ。
　例）$W_i[Y]R_j[X]R_i[Z]W_j[X]$

我々が、ここで扱う先読みスケジューラは、単版先読みスケジューラと多版先読みスケジューラに分類することができる。[2,4]単版先読みスケジューラでは、データ項目Ｘについて最新のＸの値を参照するものである。一方、多版先読みスケジューラは、データ項目Ｘに過去に更新された値を持つ版の存在を許し、READ命令は、それまでに書かれている版の中から適当なものを選んで参照することができるというものである。

各トランザクションのステップの順序を保つStep(Ｔ)上の系列はlogと呼ばれる。トランザクションの集合上のlog hが与えられると、Ｔ上のスケジュールは、$s = \langle h, I \rangle$のペアで示される。ただし、$I(R_i[X])$は、hの中で$R_i[X]$に先行するwrite operationの集合へのmappingであり、Iは、interpretaｰtionと呼ばれる。もし$I(R_i[X]) = W_j[X]$なら、T_iがT_jの更新したＸの値を参照したという。特に、すべての$R_i[X] \in h$に対して、$W_j[X] = I(R_i[X])$が$R_i[X]$に先行する最新の更新操作であるとき、Iは、標準的interpretationと呼ばれ、I^*で示される。

単版先読みスケジューラの場合も〈h,I*〉として表現されるがI*がlog hの中に潜在してしまうので、一般に、hだけで示される。

同じステップ集合に対しての2つのスケジュールs＝〈h,I〉，s′＝〈h′,I′〉において、I=I′ならsとs′は等価であると言いs≡s′で示される。違うトランザクションのステップが挟まれていないlogで、しかもI*を持つスケジュールは、serialスケジュールと呼ばれる。また、serialスケジュールと等価であるスケジュールを直列可能と呼ぶ。

次に、モデルを説明をするための重要な定義を述べる。

・TIOグラフ〔1〕

sがトランザクションの集合T上のスケジュールであるとき、TIO(s)によって示されるsのTIOグラフは、ノードの集合T∪T′と枝の集合Aをもつラベル付き多重グラフである。T′、Aは以下のように定義する。もしT_jがT_iからXを参照するならXのラベルを持つread-from-arc(T_i,T_j)∈A（(T_i,T_j)：Xと示す）を持つ。もしT_jが更新したXの値を他のどのトランザクションも参照しないなら、ダミーノードT'_j∈T′を作り、(T_j,T'_j)：X∈Aを導入する。なお、W_j〔X〕はuseless writeと呼ばれる。

・DITS〔1〕

TIO(s)のノードの集合の全順序に対して次の2つの条件を満足するならDITS(disjoint interval topological sort) を持つという。

1) $T_i \ll T_j$ ならTIO(s)にはT_jからT_iへの道はない。

2)〔Exclusion rule(排除規則)〕
TIO(s)グラフ上の(T_k,T_i)：Xと(T_j,T_k)：X（ただし$T_k \neq T_j$）に対して、$T_i \ll T_j$または$T_k \ll T_k$のいずれかが成り立つ。

2)の時にT_kからT_kへパスがTIO(s)にある場合、DITSを持たせるためにラベル無しのexclusion arc(T_i,T_j)をTIO(s)に導入する。

〔定理2.1〕〔1〕
もしTIO(s)が最初のトランザクションT と最後のトランザクションT を順序付けるDISTを持つなら、スケジュールは直列可能である。

一般に与えられたTIO(s)グラフがDITSを持つかどうかを判定する問題はNP完全問題である。そこで以下のような制約を与えたサブクラスを導入する。

WW制約：あるデータ項目Xに対してW_i〔X〕＜W_j〔X〕ならT$_j$の前にT$_i$は順序付けられていなければならない。
同様に、RW制約、WR制約、RR制約を定義する。また、WR制約とRW制約の結合をWRWとして表わす。

制約集合をCとすると、TIO_c(s)はCに属するすべての制約の枝をTIO(s)に加えたグラフを示す。またTIO_c(s)にすべてのexclusion arcを加えて得られるグラフをTIO_c^*(s)で表わす。

3．先読みスケジューラ

直列可能なサブクラスCに属するスケジュールを出力する一般的なアルゴリズムCS（C）について簡単に説明する。アルゴリズムに必要な主なデータ構造は、〈P，I〉，q，PEND，DELで、〈P，I〉はすでに実行された部分的なスケジュールである。qは現在テストされるべきステップである。PENDは、最初のステップが到着しているトランザクションのうち、すでに認められているものと、現在テストされているものを除いたものの集合である。
DELは、先読みスケジューラによって遅らされたステップのリストである。DELには、1つのトランザクションからはせいぜい1つのステップしか存在しない。なぜならもしその要求が遅らされたら、トランザクションは、次の要求を送れないからである。

以下に、正確なアルゴリズムを示す。〔4〕

Procedure ＣＳ(C)

[ＣＳ０：(初期化)] P:=W_0〔X〕, q:=最初のリクエスト（最初のトランザクションＴによって発行された）ＤＥＬ:=Λ，ＰＥＮＤ:=Steps(T_1) − {q}

[ＣＳ１：(qのテスト)] 部分的なスケジュールＰｑとＰＥＮＤに完成テストを適用する。もしそれが失敗すれば、ＣＳ２へ行く。さもなければ、qを認め、ＣＳ３へ行く。

[ＣＳ２：(qは遅らされる)] ＰＥＮＤ:=ＰＥＮＤ∪{q}、現在のリクエストqによって、次のうちの１つの作業を行う。

１）q∈ＤＥＬで、しかもＤＥＬのすべてのステップがテストされ失敗したならＣＳ５へ行く。

２）q∈ＤＥＬで、１）以外の場合に、qをＤＥＬの次のステップにしてＣＳ１へ行く。

３）q∉ＤＥＬならＤＥＬの最後にqを加え、ＣＳ５へ行く。

[ＣＳ３：(qは認められた)] P:=Ｐｑでq∈ＤＥＬなら、ＤＥＬからｑを削除。

[ＣＳ４：(ＤＥＬの中から次のqを選ぶ)] もし、ＤＥＬ≠Λならｑはの中の最初のステップであるとしＣＳ１へ行く。ＤＥＬ＝ΛならＣＳ５へ行く。

[ＣＳ５：(次の到着のステップを待つ)] qは次の到着のステップになる。もしqが新しいトランザクションT_jの最初のステップなら、ＰＥＮＤ:=ＰＥＮＤ∪Steps(T_j)−{q}でＣＳ１へ戻る。

４．完成テスト

完成テストが成功するかどうかを判定するための重要な定義を以下に示す。

・ATIOグラフ〔4〕

〈P,I〉, q, ＰＥＮＤが、与えられたとき、ATIOグラフは、ATIO(〈P,I〉, q, PEND)で表わされ、ＰｑとＰＥＮＤの中にあるステップからなるトランザクションのノードとダミーノードを持つまた枝の集合は、ＡとA'からなる。Ａはread-from-arcとＰｑのステップのダミー枝である。A'はＰＥＮＤのステップの各W_i〔X〕に導入されるダミー枝(T_i, T_i'): Xと各R_i〔X〕に導入される始点の定まらない枝(−, T_i): Xの集合である。単版スケジューラの場合、Iは一意的に定まるので、ATIO(Pq,PEND)と書く。

このATIOグラフもTIOグラフ同様に$ATIO_c$(〈P,I〉, q, PEND), $ATIO_c^*$(〈P,I〉, q, PEND)が定義される。

〔定理4.1〕

Ｃ＝ＷＷの時、ＷＷ完成テストが成功するのは、$ATIO_{ww}^*$(Pq,PEND)がサイクルを持たないときに限る。

完成テストが、ＷＷと同様に$ATIO_c^*$(〈P,I〉, q, PEND)のサイクルの存在の有無で判定できるサブクラスは、表４．１に示すようなものがある。我々はこれらのスケジューラを取り上げ、性能評価を行うために、シミュレーション実験を行った。

	スケジューラ	制約枝
単版	CS(WW)	WW制約枝 〔4〕
	CS($\overline{WRW}$)	WR,RW及び一時的な枝 〔4〕
多版	CS_m(MWW)	WW制約枝 〔2〕
	CS_m(MWRW)	WRとRW制約枝 〔2〕
	CS_m(M(WR+RW))	WR制約枝とRW制約枝の一部〔5〕
	CS_m(M(RW+WR))	RW制約枝とWR制約枝の一部〔5〕

表４．１　スケジューラ

５．効率向上のための工夫

完成テストの際に導入される$ATIO_c^*$(〈P,I〉, q, PEND)はすべてのexclusion arcを必要とするので枝数が多く、ノードの数も新しいトランザクションが到着するたびに1つずつ増加していくので、グラフを保持するための領域や完成テストの処理時間が時間の経過にともない増加していくという欠点を持つ。そこで我々は、$ATIO_c^*$(〈P,I〉,PEND)グラフよりも枝数が少なく、しかもサイクルの有無に関しては等価なＳＧグラフを導入して、完成テストを行う。

ここでは表４．１のスケジューラのうちCS(WW)を取り上げ、SG_{ww}(Pq,PEND)の定義だけを行い、他は省略する。

・$SG_{ww}(Pq, PEND)$〔4〕

$SG_{ww}(Pq, PEND)$は各データ項目Xに対して、$ATIO_{ww}(\langle P, I\rangle, q, PEND)$に次の枝を加えることによって、作ることができる。

(i) WW-arc(T_j, T_ℓ)
$W_j[X]$と$W_\ell[X]$は、Pqの中にXに関して、相続くwrite operation(Pqにおいて$W_j[X]$ $W_\ell[X]$の間には、Xに関するwrite operationは無い) に対して引く枝。

(ii) WW-arc(T_k, T_j)
Pqの中にXについての最後のwrite operation $W_k[X]$から各pending write $W_j[X] \in PEND$への枝。

(iii) WR-arc(T_k, T_j)
Pqの中にXについての最後のwrite operation $W_k[X]$から各pending read $R_j[X] \in PEND$への枝。

(iv) exclusion arc(T_k, T_ℓ)
(i)のWW-arc(T_j, T_ℓ)について、T_kがT_iからXをreadしているときに、$R_k[X]$から$W_\ell[X]$へ引く枝。

(v) exclusion arc(T_i, T_j)
(ii)のWW-arc(T_k, T_j)について、T_iがT_kからXをreadしているときに、$R_i[X]$から$W_j[X]$へ引く枝。

〔定理５．１〕〔3〕
Ｃ＝WWに対して、$SG_{ww}(Pq, PEND)$がサイクルを持たないとき、かつその時に限り、$ATIO^*_{ww}(Pq, PEND)$はサイクルを持たない。

定理５．１の証明は少し複雑であるので、ここでは詳しい証明は省き、方針だけを説明する。

$SG_{ww}(Pq, PEND)$に導入するのと同様のread-from-arc, constrint arc, exclusion arcが$ATIO^*_{ww}(Pq, PEND)$にも存在するので、$SG_{ww}(Pq, PEND)$がサイクルを持つなら、$ATIO^*_{ww}(Pq, PEND)$もサイクルを持つ。逆に、$SG_{ww}(Pq, PEND)$がサイクルを持たなければ、$ATIO^*_{ww}(Pq, PEND)$もサイクルを持たない事を示さなければならない。

$ATIO^*_{ww}(Pq, PEND)$もサイクルを持たない事と$ATIO_{ww}(Pq, PEND)$がDITSを持っていることが等価であること参考文献〔4〕で証明されている。したがって、$SG_{ww}(Pq, PEND)$がサイクルを持たなければ、$ATIO_{ww}(Pq, PEND)$がDITSを持つことを証明すれば良い事になる。最初に、$ATIO_{ww}(Pq, PEND)$中のすべてのread-from-arc, constrint arcが$SG_{ww}(Pq, PEND)$の中のパスとして、表されることを示し、((i)(ii)(iii)の枝を加えることによって満たされる。)次に、$i \neq k$であるような2つのインターバル(T_i, T_j):Xと(T_k, T_ℓ):Xがお互いにoverlapしないことを示し、証明が完成する。

ＳＧグラフにおいて、トランザクションのすべてのステップが終了して、なおかつ、ＳＧグラフ上でその祖先にあたるトランザクションも同様に終了しているとき、erasableと呼ばれ、ＳＧグラフから消去することが可能である。このことにより、ＳＧグラフをコンパクトに保つことができ、完成テストの処理時間を短くすることができる。

６．シミュレーション

変数	範囲	内容
T-INT-Arr	6~16	トランザクションの平均到着間隔
MAXWSET	5~10	トランザクションがWRITEするデータ項目の最大
WRRATIO	0~	WRITEするデータ項目数とREADするデータ項目数の割合
OV	0~100	READ SETとOVER LAPしているWRITE SETのパーセンテージ

表６．１

変数	値	内容
NUM-T	600	トランザクションの発生数
S-INT-Arr	5	ステップの平均到着間隔
D-SIZE	45	データベースシステム内でアクセス可能な項目数

表６．２

我々は、表６．１のパラメタをいろいろと変化させることによって、トランザクションの待ち時間にどのような影響を与えるかを調べるために、シミュレーション実験を行う。また、表６．２は常に固定させておくパラメタである。

１からMAXWSETの間の任意の値をトランザクションが更新するデータ項目数であると決め、WRRATIO,OVの値により、参照だけ行う項目数を決める。たとえば、更新するデータ項目数＝１０、WRRATIO＝１．２、OV＝０．８の場合、更新だけ行う項目数＝２、参照だけ行う項目数＝４、更新参照の両方行う項目数＝８となる。これらの値を決め、項目数に従い、データ項目を割り当て、任意に並べることによって、トランザクションが決定される。ただし、更新と参照の両方を行う項目については、必ず、参照が更新に先立つと仮定される。

次に、トランザクションがどのように到着するかを考えてみる。トランザクション間、トランザクションのステップ間は、それぞれT-INT-Arr,S-INT-Arrを平均とする指数分布に従って到着すると仮定する

スケジューラの性能を評価するために、以下のようなシミュレーションの評価基準を考える。

ａ)平均待ち時間

トランザクションが、終了するまでに、スケジューラによって、待たされる時間（完成テストなどの処理時間は０と仮定する）。これはトランザクションより次の２つを考える。

１)トランザクション平均待ち時間

トランザクションは、その最後のステップが認められた段階で終了したものとみなされる。

２)Commit平均待ち時間

先読みスケジューラは、ロールバックされる可能性ないと最初に述べたが、ハードウェアの故障などの理由でトランザクション終了前にアボートするかもしれない。もしそのようなことが起こったら、アボートしたトランザクションからデータを読んだトランザクションをロールバックしなければならない。万一そのようなことが起こってもロールバックする必要がないことが確認されたときに初めてCommitする。Commitされた段階ではじめてトランザクションが終了したものとみなされる。

ｂ)平均トランザクション数

ＳＧ ((P,I),q,PEND)のノードの数の平均

各スケジューラを異なった環境で、実行させて性能比較を行うために、次のような３つシミュレーション実験を行った。実験１では、RWRATIO=1.2,OV=82%,MAXWSET=8として、T-INT-Arrを2~16まで変化させる。実験２では、RWRATIO=1.2,OV=82%,T-INT-Arr=8として、MAXWSETを5~10まで変化させる。実験３では、MAXWSET=6,T-INT-Arr=8として、OVを0~100%まで変化させる。(トランザクションが操作するデータ項目数の最大値を一定にするためにOVの変化にともないRWRATIOも同様に変化させる)以上の実験を2ステップ、READ-WRITE、一般の各トランザクションモデルについて行った。

まず、各実験の結果を考えてみる。実験１では、トランザクション平均到着間隔がスケジューラに及ぼす影響を考える。T-INT-Arrの値を増加させると、S-INT-Arrが一定に保たれているので、S-INT-ArrとT-INT-Arrの比が増加し、実行中のトランザクションの数が減り、トランザクションの競合が少なくなり、待ち時間が減少する。実験２では、writeするデータ項目の最大数がスケジューラに及ぼす影響を考える。MAXWSETの値を増加させると、ほとんどの場合において、待ち時間は増加する。これは、MAXWSETの値が増加すると、1つのトランザクションが操作するデータ項目数が増すことによりトランザクションの競合が多くなるからと考えられる。この実験において、MAXWSETを7から8に変化させたときに非常に待ち時間が増すのは興味深い。実験３ではREAD SET とOVER LAPしているWRITE SETのパーセンテージがスケジューラに及ぼす影響を考える。OVの値が高くなると、一般

的に待ち時間が長くなる傾向が有るが、実験1、実験2に比べると、グラフの変動が激しいので長くなるとは、断言しがたい。次に、各スケジューラの性能について考えてみる。まず最初に、他の論文でも比較的よく取り上げられているWW,$\overline{WRW}$,MWW,MWRWについてはMWRWが、ほとんどの場合に最も待ち時間が短い事がわかる。他の3つのスケジューラは、2ステップ、READ-WRITEモデルの場合には、ほとんどすべて同じ結果で、一般モデルにおいては外部環境によって、良くなったり、悪くなったりして一概にどれが良いとは言えない。次に、Commit平均待ち時間であるが、これはどのスケジューラもトランザクション平均待ち時間とほぼ同様の傾向である。一般モデルにおいてトランザクションのすべてのステップが終了してからCommitされるまでの時間は平均1秒ぐらいである。一般に、多版スケジューラにした場合、スケジューラにかかる負荷が大きくなると考えられがちなので、これについて考えてみる。これは、平均トランザクション数によって検討することが出来る。ここで、負荷を考えるのは、負荷が大きくなるといくら待ち時間が少なくても、判定グラフのノード数や枝の数が増え、完成テストなどの処理時間に悪影響を及ぼし、response timeが長くなり、スケジューラの実際の性能を悪くするためである。実験の結果、各スケジューラの平均トランザクション数は、ほとんど違いが無く、それほど大きな値でもなかった。負荷は多版スケジューラも単版スケジューラもあまりかわらず、完成テストなどの処理時間も長くならないと考えられる。次に、他の論文ではあまり取り上げられていない新しいスケジューラM(WR+RW$^-$),M(RW+WR$^-$)について考えてみる。2つのスケジューラとも今までのスケジューラより、よい結果が出ている。2ステップ,READ-WRITEモデルにおいてはM(WR+RW$^-$)の方が良く、一般モデルにおいてはM(RW+WR$^-$)の方が良いことがわかる。しかし、M(RW+WR$^-$)はトランザクションが終了してからCommitされるまでの時間が他のスケジューラに比べて長くなる場合がある。平均トランザクション数はいずれも今までのスケジューラとほとんど違いがなかった。またM(RW+WR$^-$)で注目すべき点は他のスケジューラに比べて、古いバージョンからかなりの値を参照していることである。

7．あとがき

本稿では、先読みスケジューラをいろいろな環境の下で実際にシミュレーション実験を行い、新しく提案されたCS_{RL}(M(WR+RW$^-$)),CS_{RL}(M(RW+WR$^-$))が今まで提案されている先読みスケジューラよりも優れていることを示した。しかし、我々は、処理時間などを無視して実験を行ったので、今後これらの時間も考えあわせた行わなければならないだろう。
またトランザクション実行中にステップを変更する必要が生じたときの対処方法など考えていかなければならない。

なお、詳しい実験内容は〔6〕を参照されたい。

8．参考文献

〔1〕T.Ibaraki,T.Kameda,andT.Minoura,"Serializability with constraints," ACM Trans.Database Syst.,vol.12,no.3,pp.429-452,Sept.1987

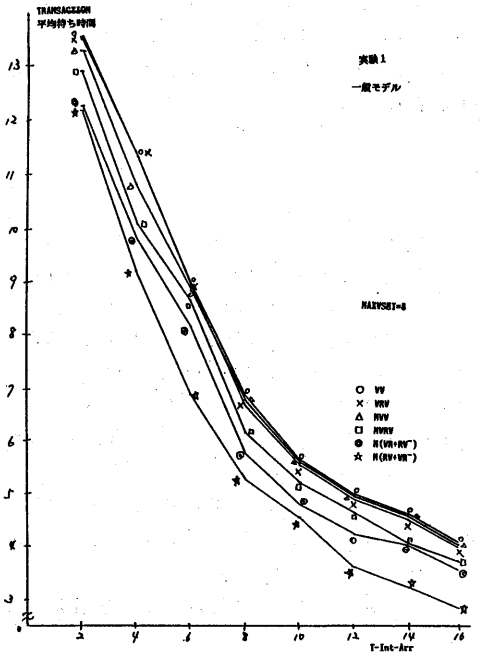
〔2〕N.Katoh,T.Kameda,and,T.Ibaraki,"A cautious scheduler for multi-step transactions,"Algorithmica,vol.1,pp.1-26,1986

〔3〕N.Katoh,T.Kameda,and,T.Ibaraki,"Efficient implementation of cautious scheduler,"Lab.Comput.Commun.Res.,Simon Fraser Univ.,LCCR Tech.Rep.(to appear),1988

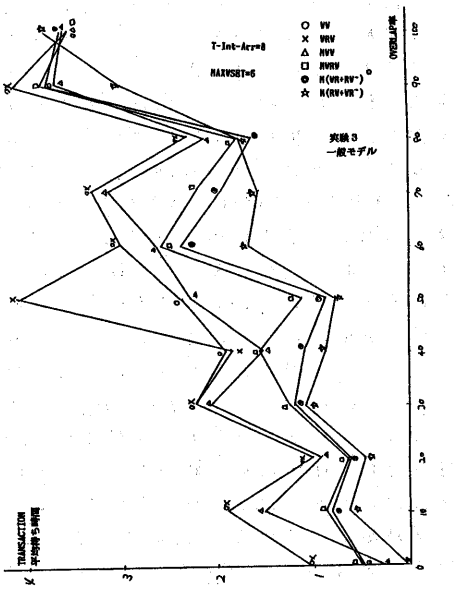
〔4〕T.Ibaraki,T.Kameda,and,N.Katoh,"Cautious transaction scheduler for Database Concurrency Control",IEEE Trans. Software Eng.,vol.14,no.7,July,1988

〔5〕加藤 直樹　未発表論文

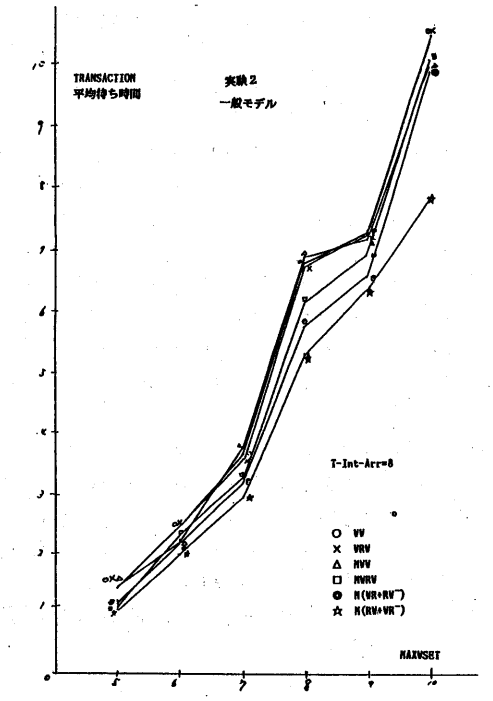
〔6〕河添 博之・藤原 敬子 神戸商科大学卒業論文(1987)



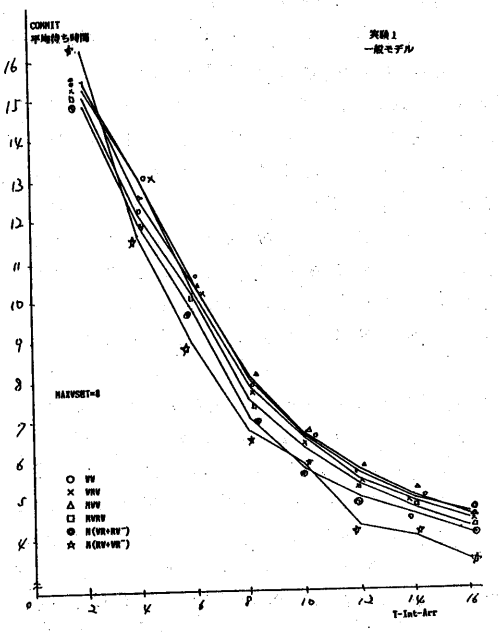
付録1
実験1・トランザクション平均待ち時間(一般モデル)



付録3
実験3・トランザクション平均待ち時間(一般モデル)



付録2
実験2・トランザクション平均待ち時間(一般モデル)



付録4
実験1・Ｃｏｍｍｉｔ平均待ち時間(一般モデル)