データベースシステムにおけるPhysical Lock方式とPredicate Lock方式の評価

松下 温　辰巳 俊文　川村 克彦　小宮山 敏夫
（沖電気工業株式会社）

1. まえがき

同時実行制御（Concurrency Control）をデータベース（以下DBと称す）に許す場合には，データの整合性（Consistency）を維持するための排他制御（Exclusive Control）を必要とする。

この排他制御を実現するロック方式としていくつかの方法が知られているが，大きくはPhysical Locks［1］,［2］とPredicate Locks［3］の2つに分類することができる。

Physical Locksは，処理要求によって参照されるエンティティを含むDBのある大きさのgranuleをロックするという概念である。granuleの大きさとしては，1レコード（タプル）［4］，1ページ［5］，1セグメント［6］，リレーションのサブセット，ファイル（リレーション），DB全体［7］等が考えられる。この場合，granuleの大きさをどの程度にすれば最適な排他制御が得られるであろうか？ 一般にはgranuleの大きさを小さくとれば，同時に処理することのできる処理要求数は増大するが，ロック処理のためのオーバヘッドが増大する。逆にgranuleの大きさを大きくとれば同時に処理することのできる処理要求数は減少するが，ロック処理のためのオーバヘッドは少なくてすむ。

D.R. Rise等［3］によれば，各処理要求がかなりの数（DBの大きさの1/10～1/100）のレコードを必要とする場合には，granuleのサイズを大きく（granule数を少なく）しても，十分なスループットを保証できる。逆にgranuleのサイズを小さくしても，ロックのためのオーバヘッドが増大しスループットは改善されないと主張している。また，処理要求が極く僅かのレコードしか対象としない場合には，granuleのサイズを小さく（granule数を多く）することが望ましいと述べている［2］。

一方，Predicate Lock方式［3］,［8］では，各処理要求によって対象となる（処理要求の参照範囲を記述するpredicateを満足する）エンティティのみがロックの対象となる。すなわち，Physical LockではgranuleのサイズがDBの大きさが固定であるのに対して，Predicate Lockでは処理要求毎にgranuleの大きさが動的に変化する方式であると言える。Predicate Lockでは，ロックのための処理はpredicate同士の論理的な比較が必要であるためPhysical Lockに比べて複雑化し，より多くのオーバヘッドを発生する。それ故，十分なスループットを保証するためにはロックの数を少なく維持することが必要である。ここで，ロックの対象となるpredicateの数は，同時に処理されている処理要求数に等しいという関係がある。

松下等［9］は，Physical Lock方式における2つのLockスケジュール方式をDBに対する更新要求の発生頻度等をパラメータにして定量的に評価している。

本論文では，Physical Lock方式とPredicate Lock方式のスループットについて

定量的に比較検討を行う。

一般に処理要求（以下トランザクションまたはTrと称す）は1つ，またはそれ以上のエンティティを参照（読出しヌは更新）する。

Physical Lock方式では，処理要求によって参照されるエンティティを含むgranule（1ヌは複数）がLock tableに登録されている。新たに到着した処理要求は，参照すべきgranuleと決定され，Lock table中のエントリィとの間でconflict checkを受け，conflictが発生しなければ処理が許可される。許可された処理要求の対象granuleはLock tableに登録される。

Predicate Lock方式では，処理要求によって参照されるエンティティは1つのpredicateによって決定付けられる。Lock tableには許可された処理要求のpredicateが登録されている。新たに到着した処理要求は，処理要求の参照範囲を示すpredicateとLock table中のエントリィとの間でconflict checkを受け，conflictが発生しなければ処理が許可される。許可された処理要求のpredicateはLock tableに登録される。

以上のロック処理によって，到着した処理要求と既に処理を受けている処理要求（Active Trと称す）との間でinconsistentな状態が発生しないことが保証される。

Physical Lock方式を採れば，Lock機構を簡単に作ることができるが，反面，同時に処理できる処理要求数がgranuleのサイズに影響され，システム設計上granuleサイズの決定が大きな問題となる。

一方，Predicate Lock方式を採れば，同時に処理できる処理要求数は増加するが，反面，Lock機構が複雑であるためconflict checkのオーバヘッドが問題となる。

ここでは，この2つのLock方式をシミュレーションによって定量的に評価する。特に，トランザクションの発生頻度，更新要求の発生比率及び処理要求の複雑度等が，各々のシステムを用いたDBシステムのスループットにどのように影響するかを考察する。

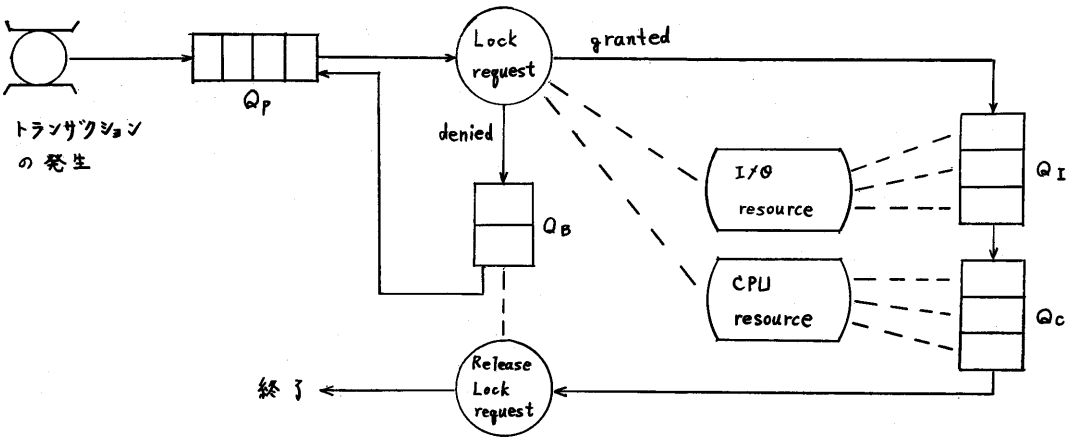
2. シミュレーションモデル

2.1 Physical Lock方式

Physical Lock方式では，トランザクションが対象とするgranuleをロックした後，DB処理を行う。

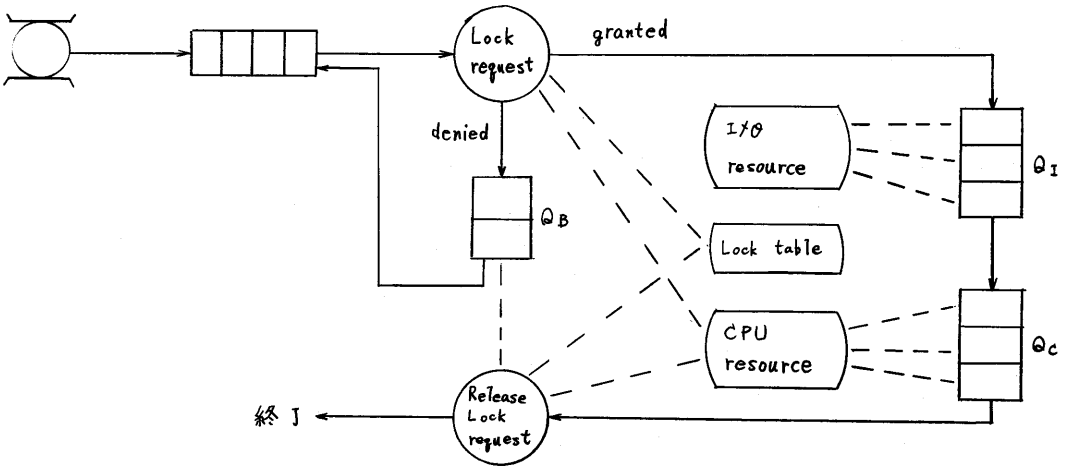
図1にこの方式のモデル図を示す。以下に本モデルによるシミュレーション方法の概略を示す。

指数分布によって発生するTrは，まずQpに入る。Qp中の先頭のTrがLock処理の対象となりconflict checkが行われ，conflictが発生しなければQEに入れられる。conflictが発生すればTrはQB中に入れられ，処理保留状態となる。Lock tableはI/Oリソース上に置かれている。



凡例　→ ： 処理の流れ
--- ： 処理要求と資源または資源とキューの関係
Q_P ： Pending キュー
Q_I ： I/O キュー
Q_C ： CPU キュー
Q_B ： Blocked キュー

図1　Physical Lock方式のシミュレーションモデル



凡例　→ ： 処理の流れ
--- ： 処理要求と資源または資源とキューの関係
Q_P ： Pending キュー
Q_I ： I/O キュー
Q_C ： CPU キュー
Q_B ： Blocked キュー

図2　Predicate Lock 方式のシミュレーションモデル

Q_I 中の先頭にある Tr が I/O 処理の対象となり，I/O 処理終了後 Q_C に入れられる。

Q_C 中の先頭にある Tr が CPU 処理の対象となり，CPU 処理終了後その Tr が対象としていた granule がロック対象から解放される。さらに，Q_B 中の先頭にある Tr が Q_P の先頭に戻される。

2.2 Predicate Lock 方式

Predicate Lock 方式は，処理中の各 Tr が参照する DB の部分を論理的な占有状態に置く方式である。占有状態に置かれている部分は Predicate（述語論理式）で決まる。

図2にこの方式のモデルを示す。以下に本モデルによるシミュレーション方法の概略を示す。

指数分布によって発生する Tr は，まず Q_P に入る。Q_P 中の先頭にある Tr がロック処理の対象となり conflict check が行われ，conflict が発生しなければ Q_I に入れられる。conflict が発生すれば，Tr は Q_B に入れられ処理保留状態となる。Lock table は主メモリ上に置かれている。

Q_I 中の先頭にある Tr が I/O 処理の対象となり，I/O 処理終了後 Q_C に入れられる。

Q_C 中の先頭にある Tr が CPU 処理の対象となり，CPU 処理終了後その Tr のロック対象を記述する Predicate が Lock table から削除され，占有状態となっていた DB の部分が解放される。さらに，Q_B 中の先頭にある Tr が Q_P の先頭に戻される。

2.3 シミュレーションの前提

2.3.1 DB システムの設定

(1) 対象 DB

本シミュレーションの対象 DB は，2ケのフラットテーブル（R_1, R_2）から成り，R_1 は10個のアトリビュート，R_2 は4個のアトリビュートから成る。但し，R_1 の a_1 と R_2 の a_1 は同一属性である。対象 DB 構成を図3に示す。

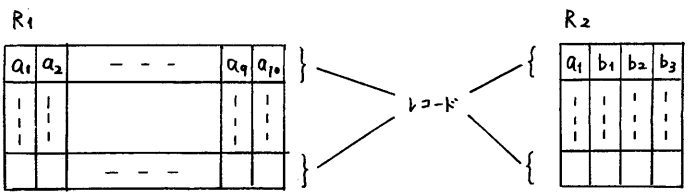


図3 対象 DB 構成

(2) granule

Physical Lock 方式では，granule 数（以下 lgran と称す）を以下のように変化させた。

lgran = 10, 25, 50, 75, 100

2.3.2 処理要求（トランザクション）

(1) Tr の発生

Tr の発生は指数分布とし，平均到着時間間隔（以下 MIAT と称す）を以下のように変化させて Tr を発生させた。

MIAT = 20, 40, 60, 100, 200, 300, ---, 1000 （単位時間）

但し，単位時間は DB の 1 エンティティに要する DB アクセスのための CPU 処理時間相当である。

(2) Tr の性質

以下の 2 種類の性質を変化させて Tr を発生させた。

① 更新要求と検索要求との割合（以下 UR と称す）

2 方式に共通のパラメータであり，

UR = 0.1（1割が更新要求），0.3, 0.5, 0.7, 0.9（9割が更新要求）

のパターンを発生させた。

② 平均参照アトリビュート数（以下 MNAR と称す）

Predicate Lock 方式の Predicate が参照するアトリビュートを表 1 のパターンで発生させた。MNAR は Tr の複雑度を示す。

表 1 MNAR の発生パターン

項番	MNAR	参照アトリビュート数 / Tr の発生確率				
		1個	2個	3個	4個	5個
1	1.47	0.70	0.20	0.05	0.03	0.02
2	1.97	0.20	0.70	0.05	0.03	0.02
3	2.88	0.05	0.15	0.70	0.07	0.03
4	3.77	0.03	0.07	0.10	0.70	0.10
5	4.44	0.03	0.05	0.07	0.15	0.70

2.4 Conflict check の方法

2.4.1 Physical Lock 方式

システム内の Active Tr の個数を K とする。i 番目の Tr（$1 \leq i \leq K$）は，L_i 個の granule をロックしている。

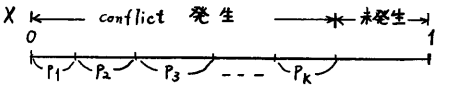
区間 [0, 1] を (K + 1) 等分し，i 番目の Tr がロックしている範囲（P_i）を

$$P_i = L_i \div lgran$$

で求める。

次に，一様乱数 X（$0 \leq X \leq 1$）を発生させ，以下の条件が成立する場合に conflict 発生と見なす。

$$X \leq \sum_{j=1}^{k} P_j$$



2.4.2 Predicate Lock 方式

システム内の Active Tr の個数をKとする。（K+1）番目に Qp から取出された Tr の j 番目のアトリビュートに関するロック範囲を，一様乱数 A（$1 \leq A \leq 1000$）によって生成する。

次に，Lock table 中の一の Predicate について，同一アトリビュートに関する占有状態にある範囲（ls）を取出し，一様乱数 B（$0 \leq B < 1000$）を発生させ，以下の条件が成立した場合に j 番目のアトリビュートの conflict 発生と見なす。

$$B \leq A + ls$$

以上の操作を（K+1）番目の Tr が参照するアトリビュート全てに対して，また，Lock table 中の全エントリィに対して行う。

2.5 入力パラメータと測定項目

2.5.1 入力パラメータ

- Tr の平均到着時間間隔（MIAT）；2.3.2 (1) 参照
- granule 数（lgran）；2.3.1 (2) 参照
- 更新要求発生頻度（UR）；2.3.2 (2) 参照
- 平均参照アトリビュート数（MNAR）；表1参照

2.5.2 測定項目

各入力パラメータに対する Tr 当りの平均処理時間（以下 MPT と称す）を求める。単位時間は，MIAT と同様，1エンティティ当りの DB アクセス CPU 時間相当である。

3. 結果と評価

3.1 開散時の平均処理時間

2方式において，全てのキューで待合せが無い状態において 1 Tr を処理するために要する時間を表2に示す。

表2 開散時の処理時間

	Physical Lock	Predicate Lock
最大	100	52
最小	10	5.4
平均	50	26.2

3.2 Physical Lock 方式の granule 数に関する評価

Tr が更新要求である割合を 30%, 90% と変化させ，平均到着時間間隔（MIAT）= 40, 60, 100（単位時間）の場合に granule 数（lgran）による平均処理時間（MPT）の影響を求めた結果を図4に示す。

図4から次のことが言える。

①開散時のMPT（=50）より短い間隔（MIAT=40）で Tr が到着する場合には，lgran が小さいとMPTが大である。即ち，lgran を大きく（同時実行可能 Tr 数を多く）することによりMPTを改善できる。但し，lgran がある程度以上大きくなると，MPTの改善効果はほとんどなくなる。

②開散時のMPT より長い間隔（MIAT = 60, 100）で Tr が到着する場合には，lgran によってMPT は影響されない。

①，②から，Physical Lock 方式において，DBを分割する granule 数を大きくすることによってスループットを向上させることのできる範囲は，処理要求の発生頻度の高い領域に限定されることが明らかである。

3.3 Predicate Lock 方式の平均参照アトリビュート数に関する評価

Tr が更新要求である割合を 30%, 70%, 90% と変化させ，MIAT = 40, 60, 100, 200（単位時間）の場合に平均参照アトリビュート数（MNAR）によるMPTの影響を求めた結果を図5に示す。

図5から次のことが言える。

①開散時のMPT（= 26.2）の 1.5 倍，2.3 倍と比較的短い間隔（MIAT=40,60）で Tr が到着する場合には，MNAR の増加に伴ってMPTも増加する。また，UR の影響も認められる。

②開散時のMPT（= 26.2）の 3.8 倍，7.6 倍と比較的長い間隔（MIAT=100,200）で Tr が到着する場合には，MNAR によってMPTはほとんど影響されない。また，UR の影響も認められない。

①は，参照アトリビュート数の増加によってロック処理のオーバヘッドが増大するためであり，Predicate Lock のロック処理オーバヘッドの問題が明らかとなった。ロック処理のオーバヘッドが顕著になるのは，処理要求の発生頻度が比較的高い領域である。

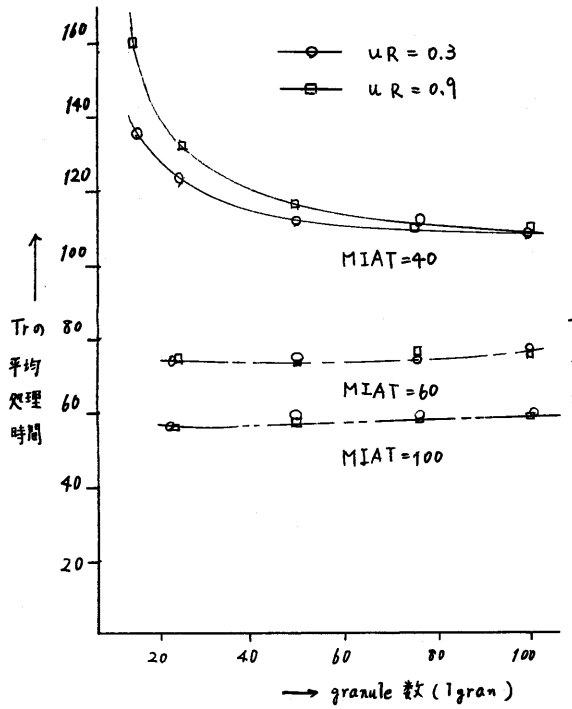


図4　lgran と平均処理時間の関係

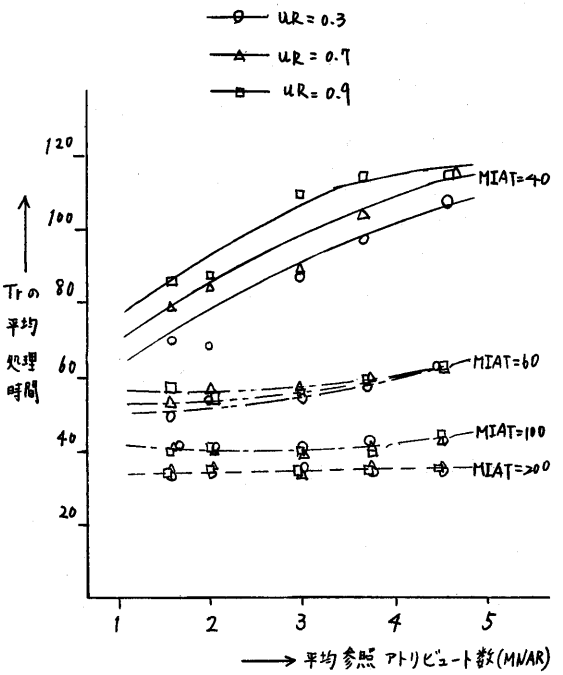


図5　MNAR と平均処理時間の関係

3.4　2つの Lock 方式の相対評価

3.4.1　更新要求発生頻度による平均処理時間の比較

MIAT = 60, 40 の場合について，Physical Lock 方式の lgran を 25, 50, 75 と変化させ，Predicate Lock 方式の MNAR を 1.47, 2.88, 4.44 と変化させて，更新要求発生頻度（UR）による2方式の MPT を求めた結果を図6，図7に示す。

図6，図7から次のことが言える。

① MIAT = 60（Physical Lock方式の関数時の MPT の 1.2 倍）程度迄の Tr 発生頻度であれば，平均参照アトリビュート数（MNAR）及び UR に無関係に，Predicate Lock方式が有利である（図6参照）。

② MIAT = 40（Physical Lock 方式の関数時の MPT の 0.8 倍）程度に Tr の発生頻度が高くなると，平均参照アトリビュート数（MNAR）または UR の値によって，有利な方式が逆転する場合がある。

(イ) MNAR が小（= 1.47）では UR に無関係に Predicate Lock 方式が有利である。

(ロ) MNAR = 2.88 では，UR の増加（同時処理可能 Tr 数が減少）に伴なって，Physical Lock 方式の方が有利となる場合が有る。

(ハ) MNAR = 4.44 では，ほとんど2方式の MPT は同じか，Physical Lock 方式の方が有利である。

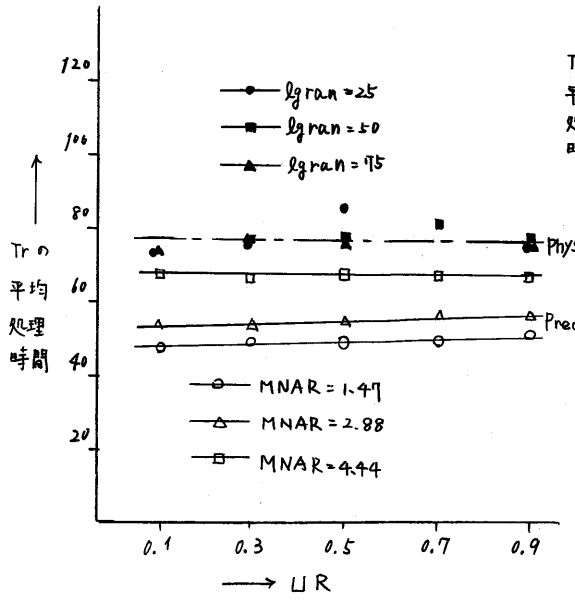


図6　MIAT=60 時の平均処理時間

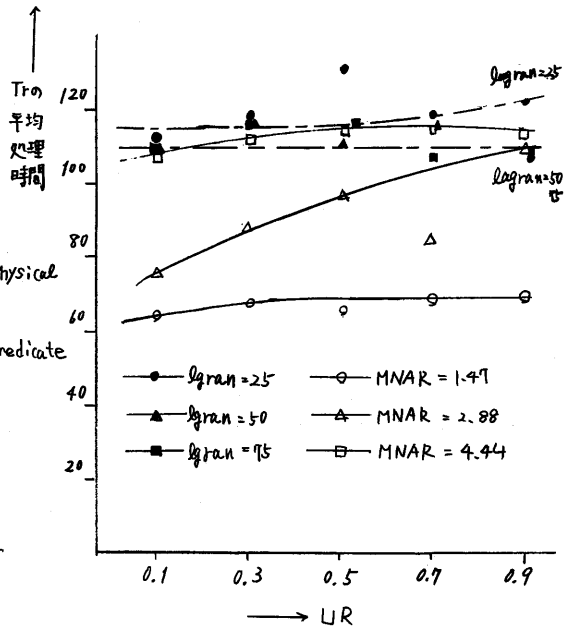


図7　MIAT=40 時の平均処理時間

②において，Predicate Lock 方式が不利になる理由は，参照アトリビュート数，または更新要求発生頻度の増加により，Lock 時のオーバヘッドが増加することにより，Predicate Lock 方式の長所である同時処理可能Tr数を大きくする効果が無くなるためであると考えられる。

3.4.2　平均到着時間間隔による平均処理時間の比較

UR=0.9 とし，両方式を3.4.1と同一条件で変化させて，MIAT による MPT の影響を求めた結果を図8に示す。

図8から，Predicate Lock 方式が大部分の領域で Physical Lock 方式より有利であることが明らかである。

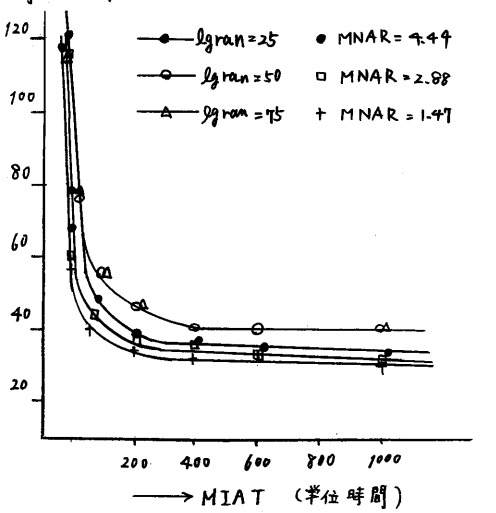


図8　MIAT に対する平均処理時間

4.　まとめ

本シミュレーションによって，Physical Lock 方式と Predicate Lock 方式のロック機構の違いによって，DBシステムへの適用範囲が以下の傾向にあることが明確にされた。

① DBシステムに対する処理要求の到着間隔が，Physical Lock方式の閉散時の平均処理時間以上であれば（トラヒックがあるレベル以下の密度），Predicate Lock 方式が有利である。

② トラヒックがあるレベル以上密になると，以下の場合においてPhysical Lock方式が有利となる。

(イ) 1つの処理要求の内容が極めて複雑（Predicateで参照するアトリビュート数が大）である。
(ロ) 1つの処理要求の内容がある程度以上複雑であり，かつ，処理要求がDBに対する更新要求である割合が高い。

しかしながら，本シミュレーションでは，平均参照アトリビュート数を5通り，更新要求の割合を5通りの場合についてのみ設定した為，上記領域の分界点が不鮮明のままであり，分界点の明確化または一般化は今後の課題である。

最後に，本研究に対してデータ収集その他に渡って多大の御協力を願った東京理科大学 工学部の関係各位に深く感謝する次第であります。

5. 参考文献

[1] D.R.Ries, M.Stonebraker. "Effect of Locking Granuality in a Database Management System," ACM Transaction on Database Systems., Vol.2, NO.3, Sept., 1977.

[2] D.R.Ries, M.R.Stonebraker. "Locking Granuality Revised," ACM Transaction on Database Systems., Vol.4, NO.2, June, 1979.

[3] K.P.Eswaran, J.N.Gray, et al "The notrom of Consistency and Predicate Locks in a Database System," Comm. of the ACM., Vol.19, NO11, NOV, 1976.

[4] M.M.Astraham, et al, "System R: Relational Appraach to Database Manegement," ACM Transaction on Database Systems. June, 1976

[5] J.N Gray et.al, "Granularity of Locks ina shared database," proc.Int. conf. on Very Large Database, Sept, 1975.

[6] M.Stonebraker, et.at. "The Design and Implementation of INGES", ACM Trans. on Database systems. Sept., 1976

[7] J.F Spizer. "portormance prototyping of data management applications" Proc of ACM., 1976

[8] J.J.Florentin. "Consistency auditing of databases," Computer Journal., Feb., 1974

[9] 松下他 "データベースシステムにおける排他制御のスケジュール方法の評価"，情報処理学会データベース管理システム研究会 1980.9