データベースにおけるDirectoryの細かさとConcurrency度との関係

松下 温　吉田 誠
（沖電気工業株式会社）

1．はじめに

データベースシステムにおいて、Transaction間でconcurrentアセクセスによるデータの共有があるとき、各Transactionに対するデータのconsistency（整合性）を維持することが必要となる。

Consistencyの維持という場合、2種類のconsistency[1],[2],[5]つまりCell-orientedなconsistencyとAttribute-orientedなconsistency　が考えられる。Cell orientedなconsistencyの維持とは、整合性の管理対象をデータセル単位ごと（アクセス単位ごと）に置いたものであり、各セル使用前のAssertionを保証することである。つまり、あるTransactionのあるデータセル使用時に、他のTransactionからのアクセスによるデータの破壊を防ぐことを意味する。一方、Attribute orientedなconsistencyの維持とは、整合性の管理対象をTransaction単位（Transactionのアクセスするすべてのセルの集合）に置いたものであり、Transaction処理の前後におけるすべてのデータのAssertionを保証することである。一般に後者は前者に比べて、忘れられる場合が多いし、また、consistencyの程度もアプリケーションに依存するところがある。

本論文では、レコードがあるP-attribute（Primary attribute）によって順序付けられ、ハードウェアセル単位に分割されて二次記憶に格納されているときの、S-attribute（Secondary attribute）アクセスに伴うconsistencyの問題（Cell orientedなconsistencyとAttribute orientedなconsistency）とconcurrencyの向上が、S-attributeによるアクセスのためのDirectoryといかに関係するかを論じる。また、シミュレーションにより得られた、Directoryの分割数（partition数）と対象アクセスセル数の関係、Directoryのpartition数とその更新頻度の関係を利用してConcurrency関数を最大にするDirectoryのpartition数と分割セル数との関係を明らかにする。

2．Consistencyの維持

Consistency維持におけるCell orientedなconsistency維持とAttribute orientedなconsistencyの維持について述べる。

図-1のようにレコードがP-attributeの値によって順序付けられ、ハードウェアセル単位に分割されて2次記憶装置に格納されている場合を仮定し、Si-attributeのためのDirectory構成を考える。この例では簡単のため、Si-attributeの値を1～6と仮定する。Si-Directoryの各partitionには、そのattributeの値の対象となるレコードを含むセル番号が格納されている。あるSi-attributeに関するTransaction要求があると、対象となるpartition中のアクセス対象セルがサーチされ、セルがアクセスされる。アクセス後、セル中の対象レコードがサーチされ、対象レコードが更新される。図-1のような環境下でCell orientedなconsistencyを維持することは、セルごとのmutual exclusionを保証することであり、一方、Attribute orientedなconsistencyを維持することは、attribute partitionごとのmutual

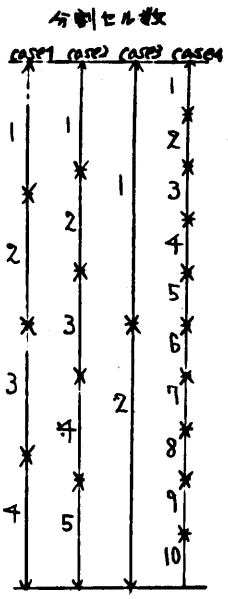
exclusionを保証することである。

Si-Directory 構成

	Si-attribute	CASE 1	CASE 2	CASE 3	CASE 4
方法1	1	1,3,4	1,3,4,5	1,2	1,6,7,8,10
	2	1,4	1,4,5	1,2	2,8,9
	3	2,4	3,5	1,2	5,10
	4	1,2	1,3	1	4,5
	5	2,3,4	2,3,5	1,2	3,4,6,9
	6	1,2,3	1,2,4	1,2	2,3,4,7
方法2	1～2	1,3,4	1,3,4,5	1,2	2,6,7,8,9,10
	3～4	1,2,4	1,3,5	1,2	1,5,10
	5～6	1,2,3,4	1,2,3,4,5	1,2	2,3,4,6,7,9
方法3	1～3	1,2,3,4	1,3,4,5	1,2	2,5,6,7,8,9,10
	4～6	1,2,3,4	1,2,3,4,5	1,2	1,2,3,4,6,7,9

	Si-attribute partitionの数
方式1	6
方式2	3
方式3	2

	分割セル数
CASE1	4
CASE2	5
CASE3	2
CASE4	10



P-attribute	S-attribute
1	1
2	4
3	2
4	6
5	6
6	5
7	6
8	5
9	4
10	3
11	1
12	5
13	1
14	6
15	1
16	2
17	2
18	5
19	3
20	1

(Logical File structure)

図1 S-Directory の構成とハードウェアセルの関係

Attribute-oriented consistency を保証しなければ、ユーザ view におけるデータの整合性は保証されなくなる。たとえば、1つの Transaction は図2のようなプロセス過程をたどるとする。

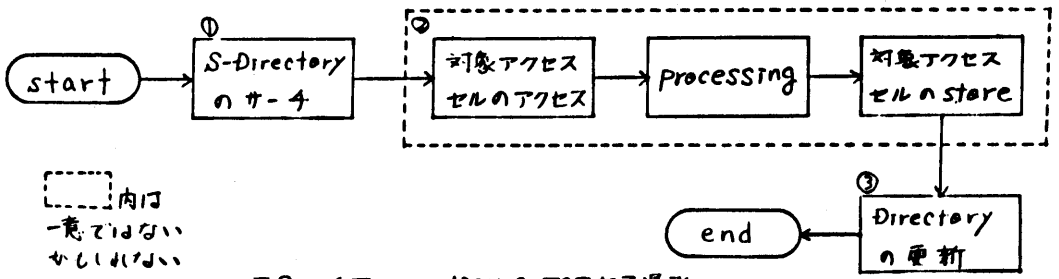


図2 1 Transaction のプロセス過程

ある Transaction が Si-attribute 値1のレコードの Si-attribute 値を2に変更する要求であり プロセス過程②を終了し③の前であったと仮定する。図-1の case4-方法1の Si-Directory を使用しているとすると以下のような矛盾が生ずる。

(i) partition 1 におけるアクセス対象セル1を削除する前に他 Transaction からの partition 1 の Directory READ があると、その Transaction はセル1をもとにアクセスする。

(ii) もし 他の Transaction が Directory の partition 2 を READ すると、現在変更になったセル番号1の対象レコードをアクセスできる。

本研究では，上述のような Cell oriented consistency を維持するために 各 attribute partition に対するロック，及び Attribute oriented consistency を維持するために セルごとのロックを設け 図-3のような制御を行うことにする。 図-4に S-Directory の構成を示す。

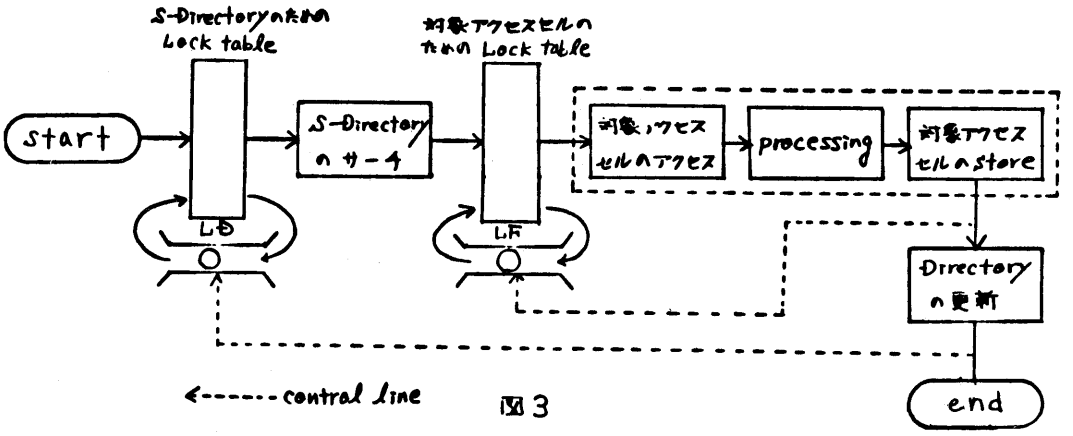


図3

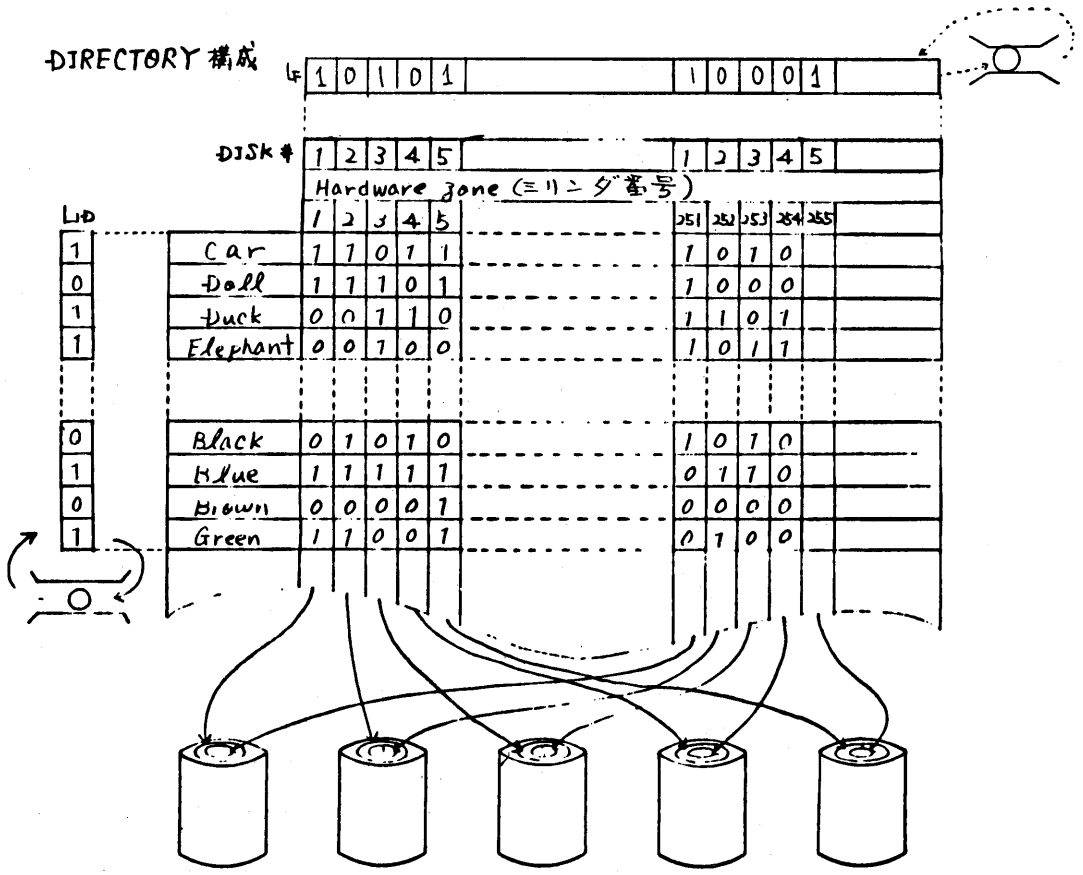


図4

3. Concurrency 関数

四一千のモデル上で Concurrency 関数を以下のように定義する。

定義：Concurrency 関数……Concurrency 関数とは Transaction が対象 Attribute portion のロックテーブルを通過でき（ロックテーブルの対象 attibute portion が UNLOCK 状態のとき通過できる），かつ 対象アクセスセルのロックテーブルを通過できる可能性を表わす関数である。

上記定義より，Concurrency 関数を以下のように定義する。

$$F(N,d,x,C_1,C_2) = \left(1-\frac{1}{100}d\right)^{x}\times\left(1-\frac{x}{N}\times\frac{1}{C_2}\right)^{C_1}$$ … Concurrency 関数

d：S- Directory の更新される割合
x：1 partition 中の平均対象アクセスセル数
N：ファイル分割セル数
C_1, C_2：Constant

4. Concurrency 関数の評価

あるレコードタイプの Primary attribute と Secondary attribute との間に Functionality がない仮定のもとで行なわれたシミュレーションの結果である。Directory の更新頻度 (d) と partition 数 (P) の関係，1 partition 中の平均対象アクセスセル数 (x) と partition 数 (P) との関係を図-5，図-6に示す。

シミュレーションで求めた d と x の値を Concurrency 関数に適用し C_1, C_2 の値を変化させて関数の変化を観察する。

図-7～図-14までに Concurrency 関数の変化を示し 表-1に各図の説明を行う。

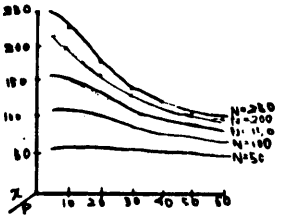


図-6 x と P の関係

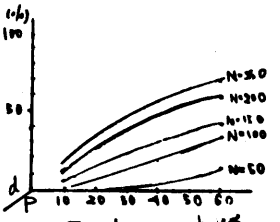


図-5 d と P の関係

図番	説明
7	図-5, 図-6 の値を $(C_1,C_2)=(1,1)$ として Concurrency 関数に適用した時の P と F との関係。
8～11	C_1, C_2 を以下のように変化させたときの P と F の関係 図-8．$(C_1,C_2)=(1,1.1)$，図-9……$(C_1,C_2)=(1,1.3)$ 図-10……$(C_1,C_2)=(1,1.6)$　図-11……$(C_1,C_2)=(1,2.0)$
12	図-8～図-11の各々において Concurrency を最大にする partition 数とセル分割数との関係
13	C_2 の値を 2.0，3.0，4.0 と変化させたとき，d の値として 2P，3P，4P の値を参照したときの P と F との関係
14	セル分割数 (N) を250に固定して C_2 を変化させたときの P と F の関係

表-1．図7～14の説明表

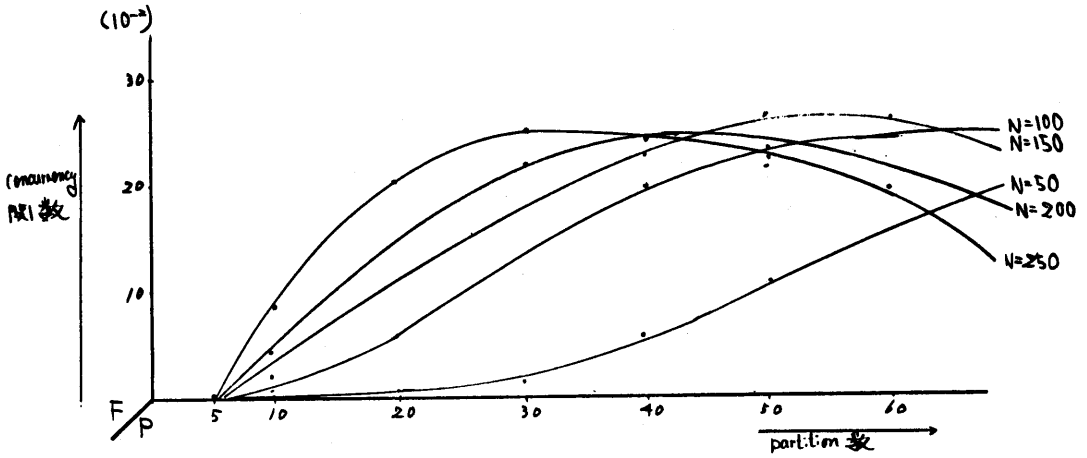


図-7. $(C_1, C_2) = (1, 1)$ のときの Concurrency 関数

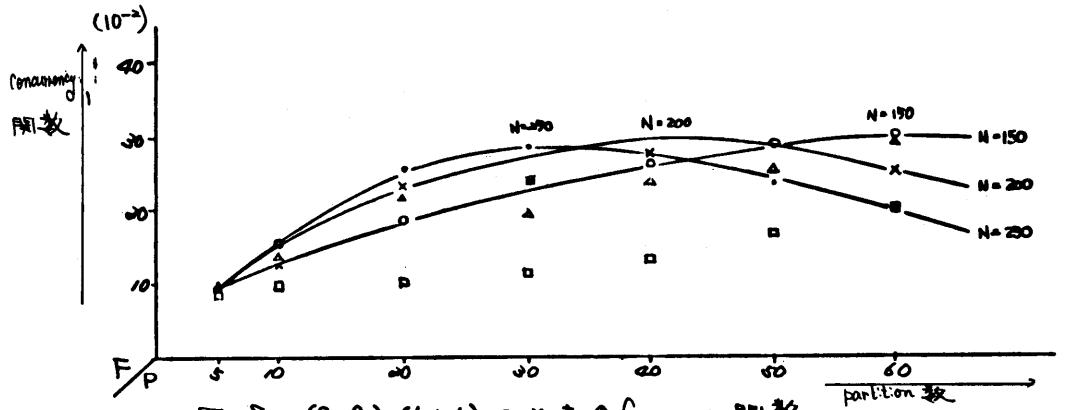


図-8. $(C_1, C_2) = (1, 1.1)$ のときの Concurrency 関数.

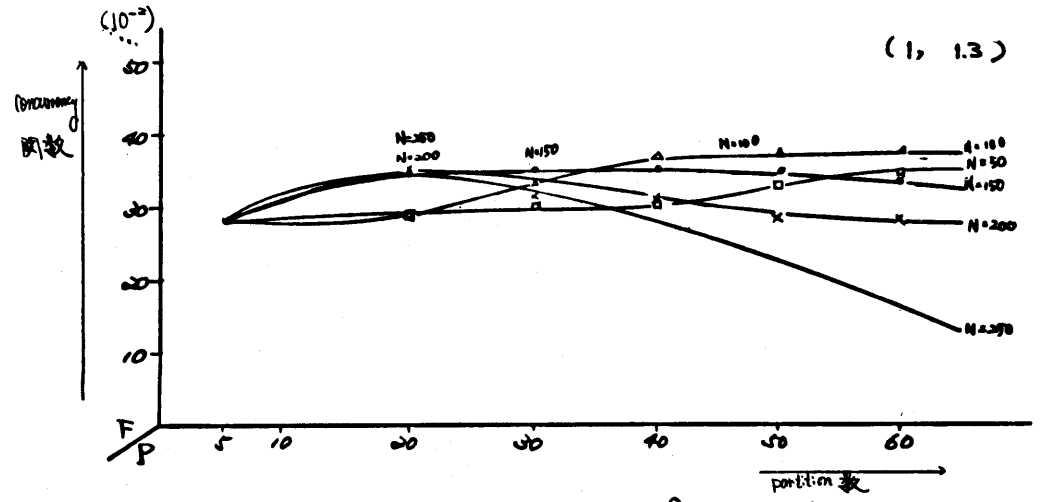


図-9 $(C_1, C_2) = (1, 1.3)$ のときの Concurrency 関数

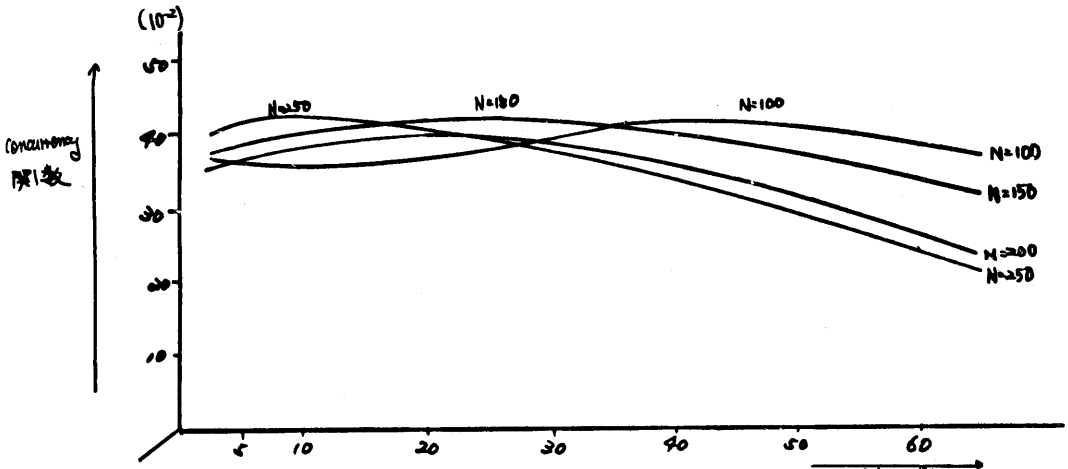


図-10 $(C_1, C_2)=(1, 1.6)$ のときの Concurrency 関数

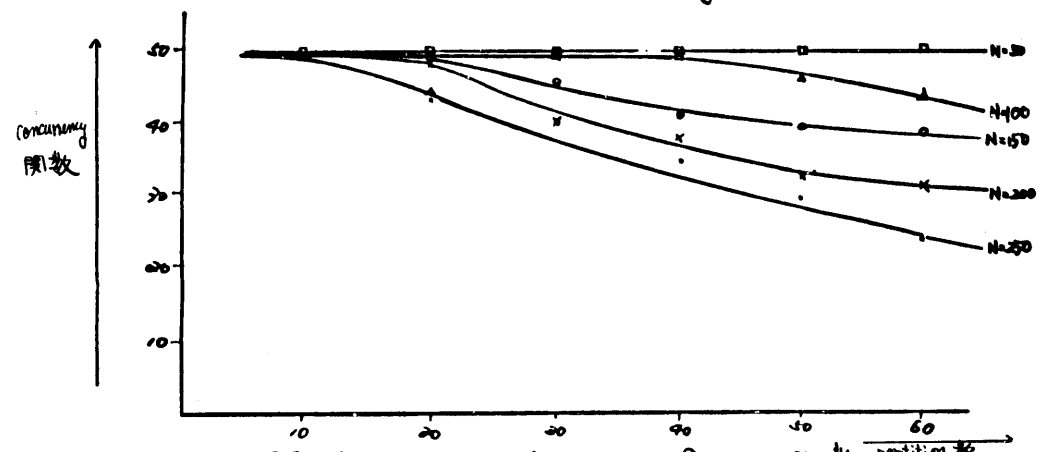


図-11 $(C_1, C_2)=(1, 2.0)$ のときの Concurrency 関数

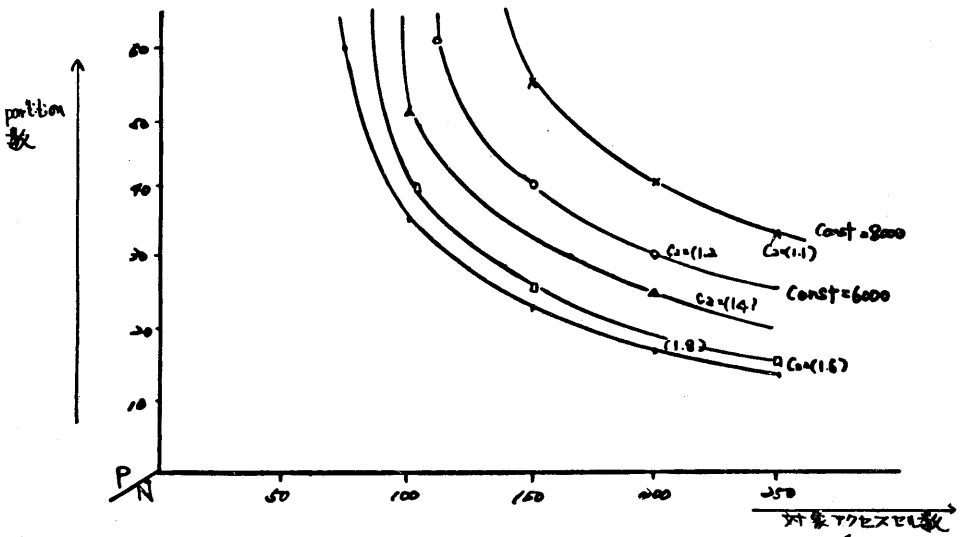


図-12 Concurrency 関数を最大とする P=N の関係.

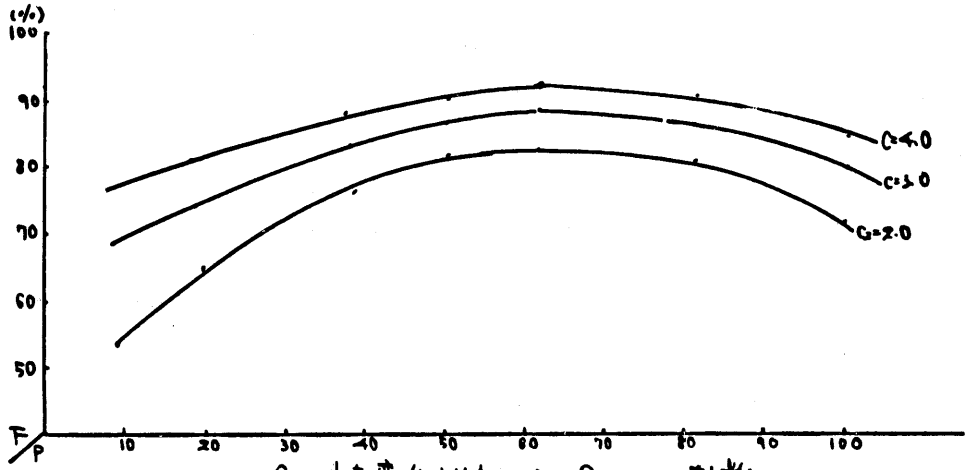


図-13 Gとdを変化させたときのConcurrency関数

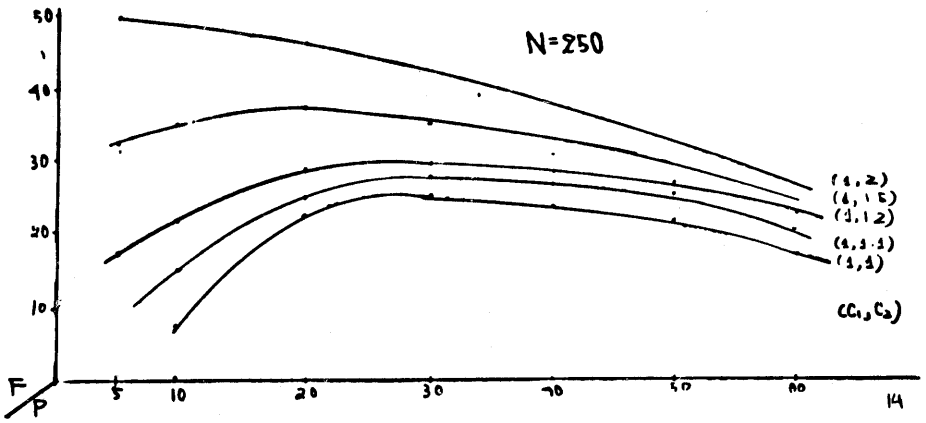


図-14 N=250でGを変化させたときのConcurrency関数

以下にConcurrency関数の評価をする。

(CASE1) 図-7に示されるように、あるレコードタイプのPrimary attributeとSecondary attributeの間にFunctionalityがなく、又レコードがprimary attributeによって順序付けられている場合には、ファイルのセル分割数が大きければ、Concurrency関数は最大となるS-Directoryのpartition数は小さくなることがわかる。

(CASE2) 図8~12は、レコードかのattribute間に何らかのFunctionalityがあれば平均対象アクセスセル数は減少することを想定して、C_2の値を変化させConcurrency関数を観察したケースであり、以下のことが言える。

(i) (分割セル数)×(Directoryのpartition数)≃Constとなるセル数とDirectoryのpartition数がConcurrency関数を最大とする。

(ii) (i)の条件下ではConstant値が小さい方がConcurrency関数の値は大きくなる。

(CASE3) 図-13は、attribute間にFunctionalityがあるとき、対象アクセスセル数を減少させるために

Secondary Indexを追加してpartition数を増した場合であり、attribute間にFunctionalityがない場合にも Concurrency関数を最大とする最適partition数が必ず存在することを意味する。

5. まとめ

Concurrency関数の評価によって、Consistencyを維持しつつ、Concurrency関数を最大とするGlobal Directoryのpartition数とファイル分割セル数との以下の関係が明らかになった。

(i) レコード中のattribute間がまったく独立の場合にも、Concurrency関数を最大とする最適Global Directoryのpartition数は存在する。

(ii) (分割セル数(N))×(Directoryのpartition数(P))≒Constとするとき、PとNがConcurrency関数を最大とする。

(iii) (ii)の条件下ではConstの値は小さい方がよい。

(iv) Concurrency関数を最大にするPartition数は、Nが大きいほど小さくなる。

上記の結果は、現在のファイルシステムにおけるある評価関数を示すとともに、ファイルシステムの設計時のDirectory構成とファイル分割数への指針を提供するものと思える。

今後の課題としては、アクセスされやすいセルとDirectoryの更新の周期を分析し、Concurrency関数を評価することである。

6. 参考文献

(1) 山崎、足田、吉田、川上松下 "分散型データベースにおける同期制御のための階層型プロトコル" 情報処理学会分散処理システム 2-1 (1979, 9, 20)

(2) 松下、吉田、脇野 "分散データベースにおけるデッドロック回避方式によるConcurrency Controlの比較評価" 情報処理学会分散処理システム研究会(1981. 11. 13予定)

(3) Martin "Computer Data-Base Organization" Prentice-Hall (1977)

(4) Wiederhold "Database Design" Mcgraw-Hall (1977)

(5) H. Yamazaki, Y. Matsushita, etal "A Hierarchical Structure for concurrency Control in a Distributed Database System" sixth Data Communication Symposium, 1979 Nov.