データベース設計における性能評価

川越恭二，真名垣昌夫
（日本電気㈱C&Cシステム研究所）

1 はじめに

現在，多くのデータベースシステムが稼動しており，データベースは，コンピュータシステムおよびユーザの世界で定着化しつつある。また，多様化と高水準化に向けて，一層データベース技術の研究開発が進められている。多様化では，大規模データベースから，オフコン・マイコン用小型データベース[1]までという量的な面と，生産管理・販売管理等の事務応用以外に技術用，研究用などへの利用面[10]の2つのベクトルに対して研究開発が行なわれている。一方，高水準化に関しては，データ意味論，データモデルのようなデータベース基盤技術と，エンドユーザ言語のようなデータベース利用技術の2つのベクトルに対して精力的に研究されている。

このようなデータベース技術の進展にもかかわらず，データベース構築技術については，実際の場での問題が尚，残されている。その中でも重要なものとして，データベース設計がある。データベースの設計は，実業務を熟知した業務担当者と，EDPに精通したシステム設計者との共同作業によって行なわれるのが普通である。しかし，データベース設計における設計パラメータが多いため・両者の意志の不統一（情報伝達の不完全さ），設計者の経験とカンによる作業が多い事等の理由から，複雑かつ困難な仕事となっている。このような問題を解決するために，システマティックにデータベース設計を実現しようとする方法論あるいはツールがいくつか提案されている[12]。しかし，方法論によって，設計ガイドラインの認識，設計の標準化は可能であるが，設計者の負担が，大幅に減少することが可能だとは考えられず，また，依然として設計者の能力，経験を必要とするであろう。また，既存のツールは，各々が個別に独立して開発されているために，入力データ作成にかなりの時間を必要としたり，使用DBMSに適合しないモデルを対象としていたり，モデル化の仮定が明確でないためにツールの有効性が不明であるという問題をかかえる。

我々は，このような問題を解決するため，データベース設計を総合的に支援することを目的としたデータベース設計支援システムAIDS（An Infological Database Design System）を開発している[9]。AIDSでは，データベース設計過程を(i)実業務のモデル化を行なう概念設計 (ii) DBMSに依存したモデルに変換する論理設計 (iii) 格納構造をパフォーマンスの点から最適化する物理設計に分解し，各々に対して適した支援ツールを提供している。本稿では，AIDSの機能の内，論理設計におけるDB必要記憶容量と，業務処理の実行時間との見積推定するツールと，物理設計における，DB必要ファイル容量と，業務処理の実行時間を推定するツールについて説明する。尚，論理スキーマとしては，ACOS ADBS[13]で採用している CODASYL 型，物理スキーマとしては，ファイルのPOケーションパラメータを想定している。

まず，次章でデータベース設計過程の各フェイズごとの評価項目をのべ，第3章でデータベース性能評価モデル分類を説明する。

2 DB設計プロセスと評価

DB設計は、これまで (i) データ分析、(ii) データ構造設計、(iii) 格納構造設計の3フェーズに分けて行われてきた。第1のデータ分析では、データベース対象範囲のデータについてその内容と利用用途、特性、関連を分析し、整理・統一化するフェーズである。第2のデータ構造では、第1のデータ分析の結果を用いて、DBのデータ構造を設計するフェーズである。第3の格納構造設計は、第2のフェーズの結果に基づいて、DBの記憶媒体上の記憶領域の構造と、その中でのデータ格納状態を決定するフェーズである。最近では、上記第一のフェーズを要求分析とビュー統合化の2つのフェーズに分割する方向にある[1,2]。前者は、DB対象における種々の業務の情報分析と処理分析を行ない、後者は、その分析結果内の矛盾を検証、修正したのち、ビューを統合化し、DB化のための統一的データ分析結果を出力する。

上記、設計プロセスにおける各フェーズと設計項目との関連を図1に示す。データ分析フェーズでは、ユーザ要求の充足性、無矛盾性の検証が必要である。これに関する支援ツールとしては、エンドユーザにより容易にシステム化のできる簡易DBシミュレータ等が適している。データ構造設計では、理想的には性能面からの最適化は不必要であるが、現実には、CODASYL型スキーマであれ、関係DBのベースリレーションであれ、そのデータ構造の選択が少なからず、業務処理に影響を与えるために、性能評価ツールを必要とする。この段階での評価は、スキーマ間の良し悪しを決定するに各分であればよく、絶対評価は必要としない。評価項目は、処理時間とDB必要記憶容量である。格納構造設計では、性能向上を目指して、格納構造を最適化するため、DBの平均的格納状況に対する、処理時間とファイルの必要容量の評価項目が必要である。

上記以外に、機密保護、エラー保護、障害復旧、運用評価等の評価が必要であるが、データ構造/格納構造に影響を与える可能性はあるが、運用設計からのフィードバックによる結果であり、データベース設計には直接は寄与しないものと考えられる。

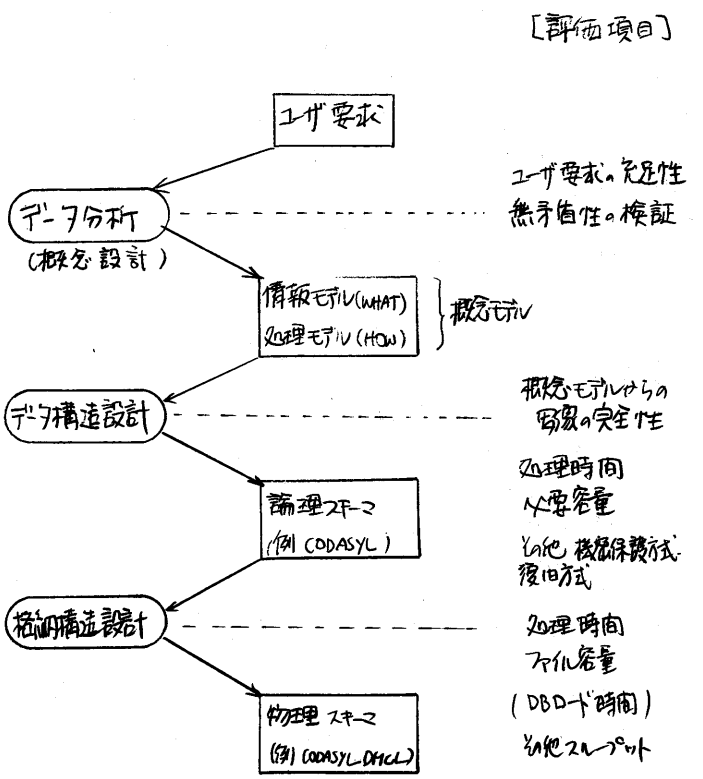


図1 データベース設計と評価

3 性能評価モデル

2で記述したように、DB設計評価ツールに必要とされるのは、記憶容量の推定と、処理時間の推定である。この推定方式としては、これまで、容量算出に関しては、(1)制御データ量＋実データ量 方式 と (2) 1ページ内平均占有量 方式 が考えられている[1]。

処理時間推定に関しては、図2に示すような4方式がある。これらの方式の何れを採用するかは、評価要求におけるモデルの詳細度、入力データ定義と取得可能性、入力データに対する感度、コスト等に依存する。

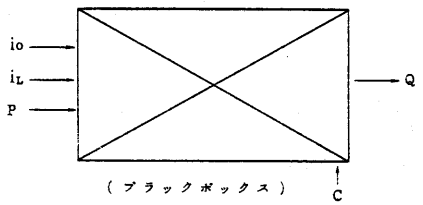
処理時間推定に関する内部モデルを、モデルのタイプ、内部状態の有無、入力データの点から分類すると、表1、図3に示す4モデルに分類される。

	モデルタイプ	状態	入力データ
モデル1	ブラックボックス	無	格納構造不要
2	ホワイト 〃	無	〃 要
3	〃 〃	有	〃 〃
4	〃 〃	有	格納順序データ

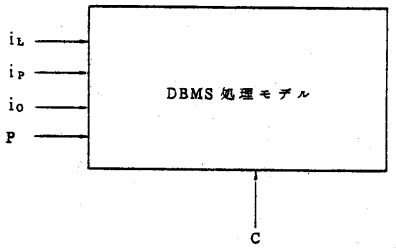
表1 モデルの分類

手法	細分類	ツール
解析モデル	単純数式	DBD-DSS[6]
	(待ち行列論)	QM-1
シミュレーション	イベントドリブン	ARTS/DB[3]
解析＋シミュレーション		DBDE[1][8] PEAL[2]
模疑データによる実シミュレーション		

図2 処理時間推定方式分類

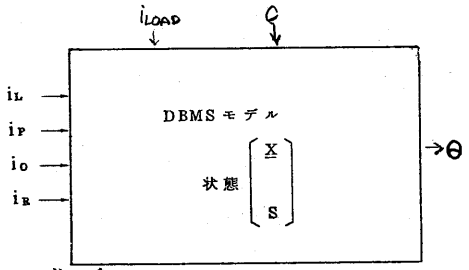


モデル1: $Q = f(i_L, i_o, C)$

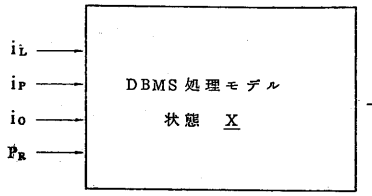


モデル2: $Q = f(i_L, i_P, i_o, C)$
ADBS

i_L : 論理スキーマ
i_P : 物理スキーマ
i_o : オカレンス数
C : 統計データ
P : 処理記述 $P_1 P_2 \cdots P_n$
Q : 処理時間
X : バッファ, ロックテーブル, ログ等の内部状態
i_{LOAD} : データベースロード状況
S : ページごとのレコード作成分布状態



モデル4:
$Q = \Sigma Q(R)$
$Q(R) = f_R(i_L, i_P, i_o, X_R, S_R, C, P_R)$
$X_{R+1} = g_R(i_L, i_P, i_o, X_R, S_R, C, P_R)$
Q_R : IDML動詞 P_R 実行に要する時間
$S_0 = i_{LOAD}$
$S_{R+1} = h_R(i_L, i_P, i_o, X_R, S_R, C, P_R)$



$Q = \Sigma Q(R)$
$Q(R) = f_R(i_L, i_P, i_o, X_R, C, P_R)$
$X_{R+1} = g_R(i_L, i_P, i_o, X_R, C, P_R)$
Q(R): 1 DML動詞実行に要する時間

図3 モデル1～モデル4

[モデル1] モデル1は、論理スキーマを入力データとし、DBMS自体は、ブラックボックスとして扱い、各種統計データを用いて、設計に算出式により入出力関係によって表現されるモデルである。このモデルは、その簡明さ、必要情報の少なさから、論理スキーマ評価用ツールのモデルとして適している。精度に関しては、あまり期待できない。

[モデル2] モデル2は、物理スキーマを入力に加え、状態をもたないDBMS内部カプセルモデルにより構成される。状態をもたないために、アクセスパス内のDMLコマンドシーケンスにおける順序を無視できる範囲で有効である。Gemiysenの DBD-DSS のモデル[5]は、このタイプに属する。

[モデル3] モデル3は、DBMSのモデルとして、状態を有するモデルを含んでいる。状態を持つために、この部分はシミュレータとして動作する。アクセスパスのシーケンスに応じてモデルが動作する。このモデルの実現例として、PEAL[2], DBDE[1] がある。

[モデル4] モデル4は、入力データとして、データの格納順序を加えて、内部的にDBの格納イメージをもつ。DBの動的なシミュレーション、DBのロード時間の推定 が可能となる。DBPROTOTYPE[4]がこのタイプに属する。

処理時間の推定には、業務処理を記述する言語を必要とする。理解の容易さ、記述の容易さ 等の点から考えればこの言語はDBMS 固有のデータ操作言語に近い形となる。

[4] スキーマ評価ツールの基本方針

データベース設計支援システム（AIDS）では、データ構造（論理スキーマ）格納構造（物理スキーマ）設計における評価ツールとして、各々 モデル1，モデル3 を使用している。

論理スキーマ設計では、格納構造が不明であるため、DBMSのモデル化は不可能であり、データ構造に陽に表現されている情報のみを使用しなければならない。このため、モデル1の使用は妥当であると考えられる。

物理スキーマ設計では、[3]のモデルの内、モデル2かモデル3の使用が考えられるが、業務処理論理のDBアクセス列が性能に与える影響を考慮する必要があることから、モデル3を使用している。しかし、モデル3では、処理をシミュレートするが故に、計算時間の長で問題が発生する可能性もなくはないと考えられる。これについては、内部モデルの状態遷移を無視することで、容易にモデル2として使用できると考えられる。

業務処理論理を表現するものをアクセスパスと呼ぶが、これには、次の要件が存在する。

- DML動詞との対応が必要である。（従って 検索処理のみでは不充分）
- DML動詞のパラメータの定義が可能でないといけない。
- 論理／物理評価ツールに共通のアクセスパスの使用が可能である

上記要件を考慮することで、コマンド＋パラメータ形式に構造体の定義を可能にしたアクセスパスコマンド言語を設計した

アクセスパスコマンドの例については、[7]章で触れる。

5 論理スキーマ性能評価方式

論理スキーマの評価は、(1) 必要記憶容量 (2) 処理時間 である。AIDSで使用している方式を以下に説明する。

[必要記憶容量]

$$\sum_{\text{レコード}} \{(\text{コントロール長}) + (\text{実データ長})\} \times (\text{レコード件数})$$

上式によって得られた値は、DBの必要記憶容量の下限を示すものである。

[処理時間]

過去の入出力データから、入出力関数パラメータを近似によって推定したのち、評価段階でその関数値を外挿法により計算することで、処理時間を推定する。

入力は、DMLタイプ、レコード件数、アクセス回数であり、出力は処理時間である。

関数例としては、

$$O_T = \sum_i f_{Ci}(N) F_i$$

但し、N: レコード件数　　O_T: 処理時間

F_i: i番目のアクセスコマンドのアクセス回数　　C_i: i番目のアクセスパスコマンド

なる関数において、$f_j(x) = a_j x + b_j$ が考えられる。実用上、この種の簡単な関数で充分であろう。

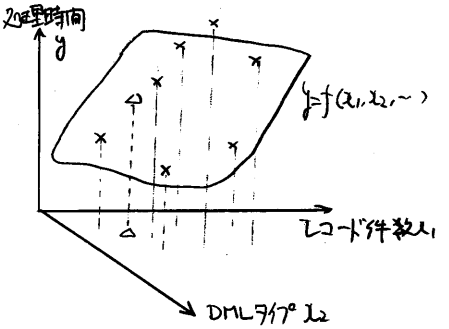


図4 論理スキーマの処理時間推定の考え方

6 物理スキーマ性能評価方式

6-1 基本思想

評価方式の詳細を説明する前に、本方式の基本的考え方を説明する。まず、ファイル容量推定方式であるが、図5に示すように、本方式では、格納状況に関する、容量分布の平均値を推定するものである。本方式で求めた値を使用すれば、平均的格納状態であればこれ以上積み込むことの不可能であるような容量値を求めることができる。実際には、算出値に余裕量を加え、増加分を加えた値にしなければならない。

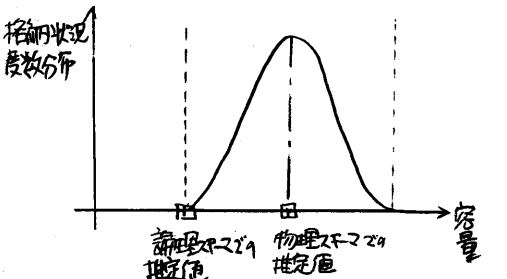


図5 物理スキーマ ファイル容量推定値の意味

処理時間の推定方式に関しては、下記の要件を満たすようにモデル化を行なっている。

a) 下記の変数の値の変化に対する処理時間の影響を考慮する
- バッファサイズ
- ブロックサイズ（ページサイズ）
- ファイルのディスクへの配置
- ブロック数
- カレントオーナレコード数

b) 再編成後のデータベースの格納状況に関する処理時間の推定を行なう。
c) 論理スキーマ、レコード件数、a)の変数を使用データとする。
d) 処理論理の変更に対して、妥当な推定値を得る。
e) 誤差率 20% 程度
f) 動的なDBの変化のシミュレートは不要である。

上記要件から、DBMS（CODASYL型）のモデル化を行なった結果、図6に示すモデルを得る。図7に想定したDBMSの基本処理モデルを示す。

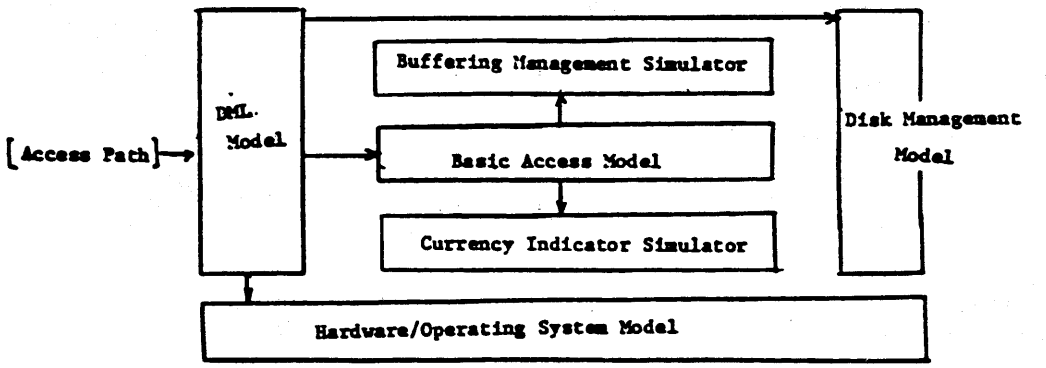


図6 DB性能評価用内部モデル

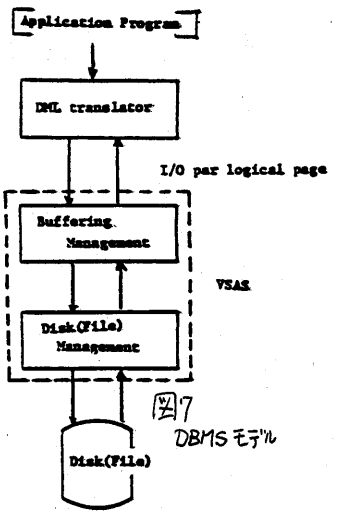


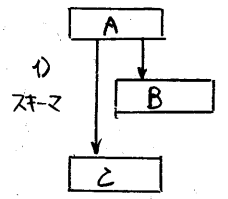
図7 DBMSモデル

図6に示すモデルを簡単に説明する。

1) 基本アクセスモデル：DML FIND FIRST、STORE 等のアクセスごとにDBMSの内部処理をモデル化。I/O回数を算出。
2) DBバッファモデル：基本アクセスモデルからのバッファエントリー登録参照要求により、バッファ状態を更新。
3) ディスクアクセスモデル：内部データとして持つ転送時間、シーク時間等から ブロック単位の1I/Oアクセス時間を算出。
4) ハード/OSモデル：各DMLごとに、そのDMLを実行する際のハード要因、OS要因を組み込む。
5) カレンシーインジケータモデル：基本アクセスモデルからのセットカレンシーインジケータ更新参照要求により、インジケータの状態を更新
6) DMLモデル：上記5モデルを統合、アクセスパス解釈、基本アクセス要求

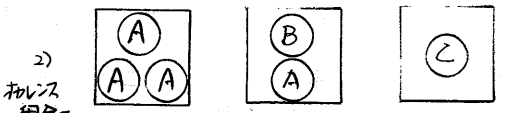
6-2 必要ファイル容量推定

推定方式を説明するために、図8を例として使用する。図8(1)に示すスキーマにおいて、1とエサイズが3.2Kバイトであるとする時、1とエ内の配置形態は図8(2)に示す3通りである。この各々のタイプの生起確率を求めれば、1とエ内の平均レコード件数を推定することができる。A, B, Cのレコードが、レコード件数の比でランダムに発生するとすると、タイプi ($i=1,2,3$) の生起確率 P_i は $P_1 = {}_{300}C_3 / ({}_{1000}C_3 \times \alpha)$, $P_2 = {}_{300}C_1 \times {}_{700}C_1 / ({}_{1000}C_3 \times \alpha)$, $P_3 = {}_{700}C_1 / ({}_{1000}C_1 \times \alpha)$ である。但し、$\sum_{i=1}^{3} P_i = 1$, 計算によると、平均レコード数は、A: 0.567, B: 0.423, C: 0.530 となる。



1) スキーマ

	レコード長	レコード件数
A	1000	300
B	2000	400
C	3000	300



2) オカレンス組合せタイプ

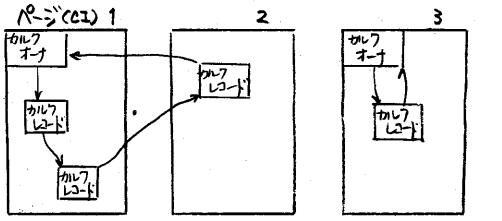
① ② ③

(1とエ = 3200バイト)

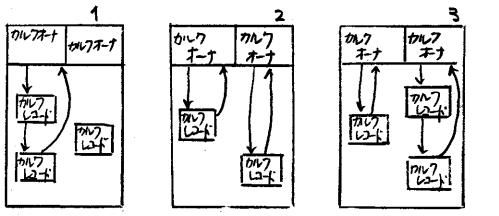
図8 例 1) スキーマ 2) オカレンスの組合せタイプ

この1とエ内平均レコード件数を用いて、実際のスキーマのクラスタリング配置を考慮して必要とエ数を求めれば、平均的必要ファイル容量が算出される。

とエ内でのレコードの格納方法としてはカルフオーナレコードの違いにより、図9に示す2つの型に分かれる。このクラスタリング配置ごとのファイル容量算出アルゴリズムを以下に示す。



(A) カルフオーナレコードを複数ページに1個設定する場合



(B) カルフオーナレコード複数個を1ページに設定する場合

図9 格納状況の場合分け

[アルゴリズムA]

step 1) とエ数の初期値(m_0)の設定

$$m_0 = \sum_{i=1} l_j N_j / \text{とエサイズ}$$

但し、l_j はレコード R_j のレコード長
N_j はレコード R_j のレコード件数

step 2) N_i を用いて、1とエ内のレコード件数の期待値(n_j)を求める。

step 3) $m_i \leftarrow m_{i-1} + \Delta M$ (とエ数の増加)

step 4) 1カルフオーナに接続されているレコード件数 θ_j を求める

step 5) θ_j / n_j を越えない最大の整数を P_j とする。

step 6) $\hat{N}_j = N_j - P_j n_j$, $P_j \leftarrow P_j + 1$

step 7) $P_j \leq n$ (nとエに1個カルフオーナを設定)
$\hat{N}_j \geq 0$ なら $P = \max P_j$ とし満足しなければ step 2) へ

step 8) トータル容量 Q は
$Q = P \times (\text{とエサイズ}) \times ([m_i/n]+1)$

[アルゴリズムB]

step 1) ~ step 4) アルゴリズムAと同じ

step 5) $P_j = [N_j / n_j] + 1$

step 6) $P_j \leq m_j$ であれば
$P = \max P_j$
満足しなければ step 3) へ

step 7) $Q = P \times (\text{とエサイズ})$

6-3 処理時間推定

6-1で述べたように、処理時間を推定するためのDBMS内部モデルは6つのサブモデルから構成されているが、その基本的算出式は下の式で表現できる。

$$T = t_1 p_1 + t_2 p_2 + t_3 p_3 + \sum_j CPU_j$$

但し、t_1, t_2, t_3 は各々ランダム、シーケンシャルREAD, WRITE1回に要するI/O時間
p_1, p_2, p_3 は各々ランダム、シーケンシャルREAD, WRITE の回数
CPU_j は処理における j 番目のDML命令の実行に要するCPU時間

上記、t_i を求めるためのモデルがディスクアクセスモデルであり、記憶媒体ごとの平均データ転送時間、回転待時間、シーク時間から t_i を算出する。一方、p_i を求めるためのモデルが基本アクセスモデル、バッファ管理モデル、カレンシーインジケータモデルである。

基本アクセスモデルは、図10に示す場合分けを行ない、個々の場合について2のDBMSの内部処理をモデル化したものである。このモデルでは、カレンシインジケータ、バッファ内のC工有無を他のモデルから得る。バッファ管理モデルは、LRU方式をシミュレートする。カレンシーインジケータモデルは、カレンシーインジケータ内部状態をシミュレートする。基本アクセスモデルは、以下の例のような処理で記述される。

FIND-NEXT処理で、カレントレコードがFIRSTにあり、セットのメンバレコードがVIAレコードである場合。

step 1) RRに 1-REFBUF を加える。

step 2) SRに CISETE *(LN) を加える

step3) 処理FBUFを実行する。

但し、RRはランダムRead回数カウンター、SRはシーケンシャルRead回数カウンタ、REFBUFはバッファ内に対象レコードタイプが存在する(1)かそうでない(0)を示す関数、CISETEはセットの1ポインタアクセスに要するページアクセス回数。FBUFはバッファエントリーとしてのWRITE要求を示す。

FIND/CONNECT/DISCONNECT

VIAセットのアクセス (同一エリア)
VIAセットのアクセス (別エリア)
NON VIAセット アクセス

FIND/CONNECT/DISCONNECT

オーナポインタ, プライアポインタあり
オーナポインタなし, プライアポインタあり
オーナポインタあり, プライアポインタなし
オーナポインタなし, プライアポインタなし

FIND

カレントレコードが オーナ
〃 FIRST
〃 LAST
〃 上以外

→カレンシーインジケータモデルを用いて内部的にモデル分割

CONNECT

INSETION FIRST
〃 LAST
〃 PRIOR
〃 NEXT
〃 SORTED

STORE

カルクレコードをstore
カレントレコードをstore (VIAレコード)
セット選択によりstore

ERASE

単一レコードのみERASE
ERASE ALL

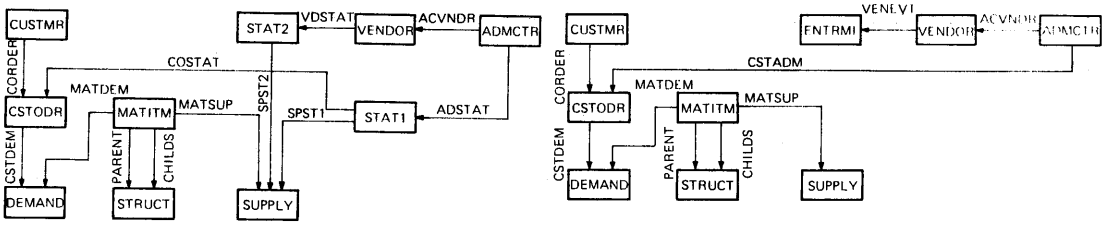
他GET, MODIFY, ACCEPT

図10 基本アクセス分割

7 評価例

7-1 論理スキーマ評価例

本方式の有効性を確めるために、図11に示す2つのスキーマ（生産管理用，機能は同等）に対して、本論理スキーマ評価ツールを適用した。アクセスパスの一例を図12に示す。比較結果を表2に示す。表2から明らかなように、スキーマBを使用した方が、性能が格段にスキーマAより向上する。このことから，事前に、本ツールを用いたスキーマの性能比較が必要である事がわかる。また，両スキーマを用いた実際のプログラムの実行結果によれば 平均して3倍，スキーマAに比してスキーマBの性能が高かった。この実測値との比較によっても本ツールの有効性がうかがわれる。



スキーマ A　　スキーマ B

図11 評価例として使用したスキーマ

```
FIND   CALC   ADMCTR
GET           ADMCTR
BLOCK         100
FIND   NEXT   ADSTAT
GET           STAT1
FIND   CALC   MATITM
GET           MATITM
FIND   USING  MATSUP
GET           SUPPLY
STORE         SUPPLY
FIND   USING  SPST1
CON           SPST1
BEND
END
```

（構造体を定義するものとして
他に CASE～OF～OF～ END がある）

図12 アクセスパス記述例

項目		スキーマA	スキーマB
必要記憶容量	Total	2481 K bytes	2662 K bytes
	実データ部	1862 K bytes	2022 K bytes
	コントロール部	619 K bytes	641 K bytes
オーダ手配計画		3825 msec	472 msec
資材所要計画*		626 msec	6.2 msec
管理節内別オーダ状況問いあわせ		3599 msec	476 msec
供給オーダ手配し		3.99 msec	3.99 msec

＊中核処理のみ

表2 比較結果

7-2 物理スキーマ評価例

ある格納構造の下で処理時間を、本物理スキーマ評価ツールを用いて推定した結果、1.67 秒の値を得た。計算時間は、アクセスパスの解釈から出力まで1分弱（ACOS 400）であった。また、ファイル容量は 242.5 K バイトの結果を得た。これは、論理スキーマの評価結果の約1.6倍であった。

本ツールと実測比較等の有効性の検証は今後の課題である

[8] おわりに

本報告では、データベース設計において欠くことのできない性能評価に関して、データベース設計支援システム（AIDS）で使用している評価方式について説明した。

AIDSでは、データ構造を決定する論理スキーマ設計フェーズ、格納構造を決定する物理スキーマ設計フェーズの各々に関して、(1) どこまでDBMSのシミュレートを行えばいいか、(2) 入力データからどこまで精密化が実現できるか、(3) CODASYL型DBMSにどこまで共通になっているか (4) 業務処理をどこまでモデル化できるか (5) どこまで拡張性を保てるか という項目を検討を加えた結果、処理時間とMS要記憶容量の推定が可能な支援ツールを提供している。

AIDSは、DB設計過程における概念設計から、格納構造設計を総合的に支援するシステムであるが、今後、更に操作性の面からの改良、設計方法論との一体化を計る予定である。

[参考文献]

[1]：L. Oberlander, T.J. Teorey (DBDE) "DBDE II: Design. Spec." Working Paper DE72-3. Univ. of Michigan (1977)

[2]：米田, 山口 (PEAL) "汎用DBMSの性能推定システム" 情処論文誌 vol.20, No.4 (1978)

[3]：C. Hutten, L. Soderland (ARTS/DB) "A Simulation Model for Performance Analysis of Large Shared Data Bases," Proc. 3rd VLDB pp524 (1977)

[4]：DB PROTOTYPE II "IBM Manual DBPROTOTYPE General Information Manual" GH20-1272-0

[5]：山口, 米田 (SPEAR) "オンラインデータベースシステム性能評価用シミュレータについて" 情処全国大会予稿集 (S54)

[6]：T.J. Gambino, R. Gerritsen (DBD-DSS), "A Data Base Decision Support System" Proc. of VLDB (1977)

[7]：武藤, 西田他, "オンラインデータベースシステムの簡易事前性能評価の一手法" 情処全国大会 (S54)

[8]：Teorey 他 (DBDE), "Network Database Evaluation using Analytical Modeling" Proc. of NCC (1978)

[9]：I. Miyamoto, "Hierarchical Performance Analysis Models for Data base System" Proc. of VLDB75 (1975)

[10]：例えば E. Wayman : Data structure, database for CAD/CAM CAD/CAMの現状と将来に関するセミナー '80.10 (1980)

[11]：例えば MDBS Micro Data Base System Inc. Auerbach Report 1980

[12]："Database Design Workshop Report" Proc. of VLDB 79 (1979)

[13]：ADBS マニュアル OFC06 ADBS言語説明書

[14]：川越, 真名垣 "AIDSのデータベース論理スキーマ評価法" 電子通信学会 情報システム部門全国大会予稿集 (1979)

[15]：川越, 久野, 真名垣 "AIDSにおけるDB格納構造評価方式" 電子通信学会 総合全国大会予稿集 (1981)

[16]：M. Manogaki, K. Kawagoe, "A Database Design System with Conceptual Model Description Language". Proc. of COMPSAC'79 (1979)