Vol. 26 No. 3 情報処理学会論文誌 May 1985

CODASYL 型データベース設計用性能推定の一方式†

橋本正明†† 山多 昭†† 中野勝之††

CODASYL 型データベースに関する性能推定を目的とした解析法を用いた CPU 時間推定方式について報告する．性能要求が強いシステムの設計においては，要求されるスループットが高い精度で与えられるので，精度のよい CPU 時間推定が必要である．CPU 時間をできるだけよい精度で推定するには，データ操作命令の処理中に DBMS, OS 内で順次実行される各プログラムごとによい精度で CPU 時間を推定し，積算するのが現実的である．しかし，解析法を用いた従来の推定方式における CPU 時間算出式は，解析モデルの簡略化のため，データ操作命令ごとの CPU 時間を決める主要パラメータのみを反映する概算式になっている．本論文では，DBMS, OS 内の各プログラムのプログラム・パス対応に作成される CPU 時間算出式の上に，プログラム呼出し関係，および，プログラムの処理対象レコード名やセット名の決め方を表現することにより，各プログラムごとに CPU 時間を推定し，積算できる CPU 時間推定方式の内容が示される．また，本方式が精度のよい CPU 時間推定に有効であることが示される．本方式を適用したデータベース性能推定プログラム DBDESIGN は，約 800 個の算出式を含み，CPU 時間の推定実施例における推定誤差は 20% 以内であった．

1. はじめに

データベース設計においては，データベース設計内容の性能評価，および，データベース・システムの処理能力の把握のため，データベースに関する性能推定が実施される．その推定作業の容易化，期間短縮をおもな目的として，データベース性能推定プログラム（推定プログラム）が開発されてきた．推定プログラムは，業務処理プログラムのデータベース・アクセス要求を処理するのに必要な，①データベース管理システム (DBMS) やオペレーティング・システム (OS) の CPU 時間，②ディスクのシーク時間，サーチ時間，データ転送時間の和で表される I/O 時間，③データベースの格納に必要な記憶容量等の推定に用いられる．推定プログラムは解析法またはシミュレーション法を用いるものに分けられる[1]．シミュレーション法の場合は，DBMS, OS がシミュレーション・モデルとして詳細に表されるので推定精度はよいが，シミュレーションに必要な計算機時間が長い．解析法の場合は，DBMS, OS が解析モデルとして簡略に表されているので推定精度は犠牲になっているが，解析に必要な計算機時間は短いので，簡便に推定ができる．解析法を用いた推定プログラムとして，T. J. Teorey らの DBDE[2]，米田らの PEAL[3]，真名垣らの AIDS[4]，M. T. Pezarro らの SEER[1]等が報告されている．そのなかには，推定精度向上のため，バッファなどに限定したシミュレーション法を付加しているものが多い．それらの推定誤差は，I/O 回数，I/O 時間，および，I/O 時間と CPU 時間の和で表される処理時間に関してはおおむね 10〜25% 以内と報告されている．一方，CPU 時間に関しては報告されていない．また，CPU 時間は処理時間の小さな部分しか占めていないので，処理時間の推定誤差から CPU 時間の推定誤差は推測できない．精度のよい CPU 時間推定は I/O 時間推定よりも困難と考えられる．しかし，性能要求が強いシステムの設計においては，要求されるスループットが高い精度で与えられるので，精度のよい CPU 時間推定が必要である．

CPU 時間をできるだけよい精度で推定するには，データ操作命令の処理中に DBMS, OS 内で順次実行される各プログラムごとによい精度で CPU 時間を推定し，積算するのが現実的である．しかし，従来の推定プログラムに用いられている CPU 時間算出式（算出式）は，解析モデルの簡略化のため，データ操作命令ごとの CPU 時間を決める主要パラメータのみを反映する概算式になっている．DBMS, OS 内の各プログラムごとに CPU 時間を推定する方法としては，プログラムの制御フローを表す有向グラフを用いた B. Beizer のプログラム解析法[5]がある．その解析法における推定精度向上には，制御フローの正確な分岐確率が必要とされている．

筆者らは，DBMS, OS 内の各プログラムのプログラム・パス対応に作成される算出式の上に，①プログラムの実行環境を反映した制御フローの分岐確率を表すパラメータ，②プログラム呼出し関係，③プログラ

† A Performance Prediction Method for CODASYL Database Designs by MASAAKI HASHIMOTO, AKIRA YAMADA and KATSUYUKI NAKANO (Yokosuka Electrical Communication Laboratory, Nippon Telegraph and Telephone Corporation).

†† 日本電信電話株式会社横須賀電気通信研究所

ムの処理対象レコード名やセット名の決め方を表現することにより，各プログラムごとに CPU 時間を推定し，積算できる方式を考案した．本論文の目的はその方式の内容と適用結果を報告することである．以下，第2章に，推定精度向上が，解析法を用いた CPU 時間推定方式の課題であることを述べ，第3章に，筆者らが考案した CPU 時間推定方式を示し，第4章に，本方式を適用した推定プログラム DBDESIGN (Data-Base DESIGN system) の概要と推定精度例を示す．

2. 方式の課題

CODASYL 型データベース設計における CPU 時間推定の役割，および，解析法を用いる CPU 時間推定方式の課題を述べる．

2.1 CPU 時間推定の役割

データベース設計は要求分析，概念設計，論理設計および物理設計の工程に分けられる．そのなかで，おもに論理設計および物理設計においてデータベースに関する性能が推定される．論理設計では，概念設計で作成されるデータベースの情報構造をもとに，論理スキーマ[6]，および，業務処理プログラムのデータベース・アクセス要求内容を表すデータ操作命令の系列が作成される．さらに，これをもとに①レコード・アクセス回数，②ポインタなどの制御情報を除いたデータ量等が推定・分析され，論理スキーマおよびデータ操作命令系列の性能改善が図られる．

物理設計では，格納，物理スキーマ[6]が作成される．さらに，これをもとに① I/O 回数および I/O 時間，②DBMS，OS のダイナミック・ステップ (DS) 数および CPU 時間，③データベースの格納に必要な記憶容量等が推定・分析され，スキーマおよびデータ操作命令系列の性能改善が図られる．また，システムのスループット，応答時間およびディスク台数などが①～③等を用いて推定され，設計条件が確認される．このように，CPU 時間推定は物理設計において実施され，その役割は，①スキーマおよびデータ操作命令系列の改善に必要な性能分析手段の提供，②スループットおよび応答時間の推定に必要な CPU 時間推定値の提供にある．

2.2 CPU 時間推定方式の課題

性能要求が強いシステムの設計においては，要求されるスループットが高い精度で与えられ，さらに，スループット改善のために，CPU 時間が少しでも短くなるデータベースが設計されるので，精度のよい CPU 時間推定が必要である．解析に必要な計算機時間が短いので，簡便に推定を実施できる解析法を用いた推定プログラムにおいても，CPU 時間推定精度の向上が必要である．

DBDE[7] の IDS 用解析モデルにおける算出式は，以下の両者の和で表される．①スタート I/O 処理および I/O 割込み処理からなる I/O 処理の単位時間と I/O 回数の推定値の積．②主記憶内のデータ転送の単位時間と主記憶内のデータ転送量の推定値の積．PEAL[3] の算出式はデータ操作命令ごとに，たとえば，$aX+bY+cZ+d$ の形で表される．X, Y, Z は空きエリア探索などの処理の繰返し回数の推定値である．係数 a, b, c, d は繰返し1回分に相当する単位時間である．このように，従来の算出式はデータ操作命令ごとの CPU 時間を決める主要パラメータのみを反映する概算式になっている．DBDE では I/O 処理および主記憶内のデータ転送処理以外の CPU 処理は表せず，PEAL では繰返しごとの処理の変化等は表せないと考えられる．概算式に代わる詳細な算出式を用いて推定精度向上を図ることは，解析法を用いた CPU 時間推定方式の一課題である．

3. 方　　式

CODASYL 型データベース対象の CPU 時間推定用の解析モデルとして，データベース・モデル，プログラム実行環境および制御フローのモデル，算出式表現法を述べ，さらに解析モデルに基づく CPU 時間算出法を述べる．

3.1 データベース・モデル

推定対象データベースのモデルを以下に述べる．

(1) データベース定義

スキーマおよびデータ操作命令に関しては，使用頻度が小さいか，または，CPU 時間への影響が小さいものを除く仕様が推定へ反映される．データ数に関しては各レコード名のレコード件数，および，各メンバ・レコード名のメンバ比率が推定へ反映される．メンバ比率は同じオーナ・レコード・オカレンスをもつメンバ・レコード・オカレンスの件数を表す．

(2) レコードおよびセットのオカレンス配置

各レコード名のレコード・オカレンスは，レコード副記述項で指定されたレルム内のオーバフロー・ページを除くページに均等に配置される．各セット名のセット・オカレンスは，セット副記述項およびレコード配置モードを考慮したうえで，ページをまたがる確率

が最小となるように配置される.

(3) レコード・オカレンスへのアクセス

インデクスを持たないセットへのデータ項目内容一致検索においては，セット・オカレンスの中央のレコード・オカレンスが検出される．インデクスへのアクセスにおいては，インデクスを格納するレコードの中央のインデクス・エントリが検出される．セット内の FIRST, NEXT などの相対位置によるアクセスにおいては，次節に述べるセット・カレンシ・インディケータに基づいてレコード・オカレンスが検出される.

3.2 プログラム実行環境モデル

DBMS 内のプログラムの実行環境モデルを述べる.

(1) データベース

プログラム実行環境としてのデータベース・モデルには，一つのデータベースのスキーマおよびデータ数，一つのデータ操作命令系列の情報が含まれる.

(2) DBMS 実行状態

シミュレーション法で擬似されるおもな DBMS 実行状態を以下に示す．①OWNER, FIRST, LAST, その他の状態で擬似される[4]セット・カレンシ・インディケータ．②バッファ・ページに関するページの使用／未使用の区別，ページ内のレコード名[7]，ページ内のインデクスの有無．③データベースのロック・ページ数．④STORE 命令処理における格納レコード名と，セット選択経路の最下段セット名で表されるセット選択対象.

(3) プログラム処理対象レコード名，セット名

DBMS の実行に伴い，DBMS 内のプログラムの処理対象レコード名またはセット名が切り替わる．たとえば，セット選択の処理においては，多数のレコード名またはセット名がプログラムの処理対象として順次現れる場合もある．DBMSなどの大規模システムは単機能をもつ多数のプログラムから構成されるので，本稿では処理対象レコード名およびセット名がおのおの，たかだか一つのプログラムから構成される DBMS の解析モデルを作成している．プログラムの処理対象レコード名およびセット名の決め方は，後述のように算出式に表される.

(4) そ の 他

業務処理プログラムはシングル・スレッドで実行される．ただし，データベース・アクセスに伴う排他制御の平均的な CPU 時間は推定へ反映される.

プログラム実行環境を CPU 時間推定へ反映する手段としてパラメータを設け，DBMS 実行状態を更新する手段としてトリガを設け，おのおの，算出式の上に表現する．パラメータはプログラム実行環境の特定の性質に対応しており，その性質に関する値を返す．パラメータは一般的には以下の関数で表される.

$P_j(R, S, I)$, $j=1, 2, 3, \cdots, n$

ここで，R, S, I はおのおの，たかだか一つのレコード名，セット名，データ操作命令系列内の命令の順番を値にとる変数である．以後も同様である．関数 P_j はプログラム実行環境に基づいて，レコード名 R，セット名 S，順番 I のデータ操作命令等の特定の性質に関する値を返す.

トリガは DBMS 実行状態の種類，更新内容に対応しており，一般的には以下の関数で表される.

表 1 パラメータおよびトリガの例
Table 1 Examples of parameter and trigger.

分類	関数	説明
パラメータ	BUFNO	バッファのページ数を返す.
	SELOWNID(S)	セット選択対象のレコード名とセット名で決まるセット選択経路において，セット名 S に関するCURRENT, SYSTEM 等の選択方法を返す.
	P4PR (R, S)	セット名 S において，メンバ・レコード名 R と隣接メンバ・レコードが同一ページに存在する確率値を返す.
	OPPTR (R, S)	セット名 S において，レコード名 R のオーナ・ポインタの有無の区別を返す.
	PLLNOAP	データベースのロック・ページ数を返す.
	VS1 (R, S)	メンバ・レコード名 R に関するセット名 S のセット選択経路上のセットの集合を返す.
	VS3 (R)	レコード名 R が AUTOMATIC の記憶属性をもつメンバ・レコードとなるセット名の集合を返す.
トリガ	SETSEL (R, S)	セット選択対象として，メンバ・レコード名 R とセット名 S の対を登録する.
	BUFDR (R)	バッファの1ページを使用状態にして，そのページにレコード名 R を登録する.
	TPLNOAP	データベースのロック・ページ数に，一つのデータ操作命令の処理中にロックされたページ数を加える.

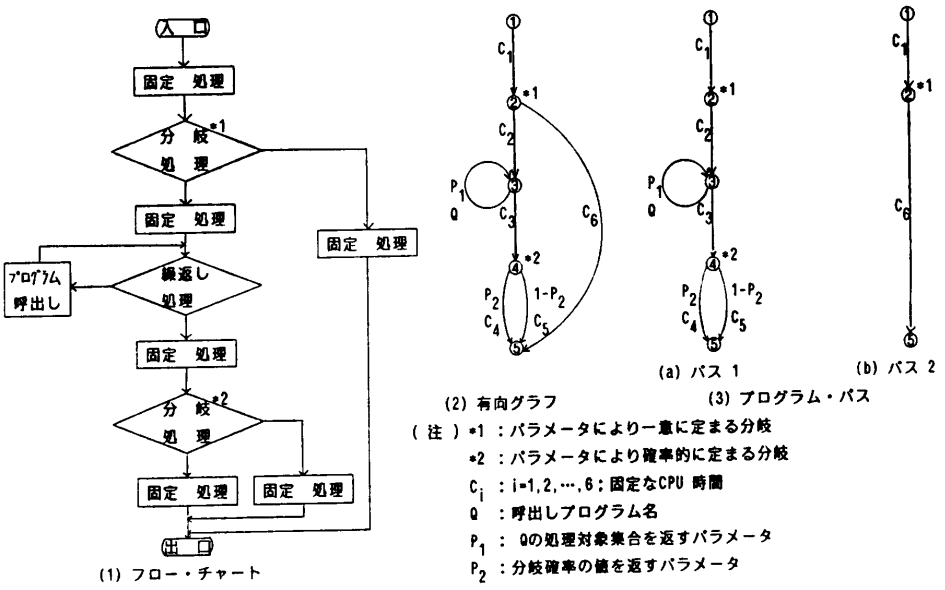

図 1 プログラム制御フローのモデル化
Fig. 1 Program control flow modelling.

$T_k(R, S)$, $k=1, 2, 3, \cdots, o$

関数 T_k はプログラム実行環境に基づいて，レコード名 R，セット名 S 等に関する特定の DBMS 実行状態を特定の内容へ更新することを表す．後述の適用例におけるパラメータとトリガの例を**表 1** に示す．

3.3 プログラム制御フロー・モデル

プログラム実行環境モデルにおいては，データベースのレコードおよびセットのオカレンス配置は確率的に定まり，その他は確定的に定まる．したがって，DBMS 内のプログラムの制御フローを表す有向グラフの中のノードは，たとえば，表 1 に示す確率値を返すパラメータ P4PR (R, S) 等を用いて確率的に分岐が定まるノード，および，OPPTR (R, S) 等を用いて一意に分岐が定まるノードに分けられる．このため，**図 1** のように，有向グラフは分岐が確率的に定まるノードと，繰返しのノードのみを含むいくつかのプログラム・パス（パス）へ分割される．本稿では，パスは分岐および繰返しを含みうるとしている．

DBMS, OS 内のプログラムに CPU 時間算定対象パスが 1 本のみ存在する場合は，プログラム名をパス名とする．算定対象パスが 2 本以上存在する場合は，プログラム中のパスを識別する記号を設け，プログラム名にその記号を添字として付加したものをパス名とする．

算出式はパスに対応しており，一般的には CPU 時間を返す以下の関数で表される．

$E_l(R, S, I)$, $l=1, 2, 3, \cdots, q$

ここで，算出式に対応するパス名を関数名とする．関数 E_l は順番 I のデータ操作命令の処理において，処理対象がレコード名 R，セット名 S の場合のパス E_l の実行，および，パス E_l から順次呼出されるすべてのパスの実行に必要な CPU 時間の合計値を返す．算出式は次節の表現法を用いて，具体的に定義される．

3.4 算出式の表現法

まず，算出式のおもな表現要素を以下に述べる．

（1） CPU 時間が固定な処理

パス内の CPU 時間が固定な処理の部分は，CPU 時間を表す定数 C で表現される．たとえば，CPU 時間の変動が少ないプログラム前処理や後処理は定数 C に近似されて表現される．

（2） CPU 時間が固定な処理の繰返し

CPU 時間が固定な処理の繰返しは，繰返し回数を返すパラメータ $P_j(R, S, I)$，1 回あたりの処理の CPU 時間を表す定数 C，および，積記号“＊”を用いて以下のように表現される．

$P_j(R, S, I) * C$

たとえば，項目数が可変な制御表の全項目チェック処理の表現に用いられる．CPU 時間が固定な処理の実行が確率的に定まる場合は，P_j が確率値を返す．

（3） プログラムの呼出し

CPU 時間算定対象パスが 2 本以上存在するプログラムの呼出しは，プログラム名 Q，および，プログラム

内のパスを識別する添字を返すパラメータ $P_j(R,S,I)$ を用いて以下のように表現される.

$Q_{P_j(R,S,I)}(R,S,I)$

たとえば，OWNER ポインタの有無に対応して，2本のパスをもつ OWNER レコード・アクセス・プログラムの呼出しの表現に用いられる．パスを1本のみもつプログラムの呼出しは，たんに算出式を表す関数 $E_l(R,S,I)$ で表現される.

（4） プログラム呼出しの繰返し

処理対象レコード名またはセット名が順次切り替わる同一プログラム呼出しの繰返しは，(3)の表現要素，処理対象レコード名またはセット名の順序付けられた集合を返すパラメータ $P_m(R,S,I)$ および，総和記号 "Σ" を用いて以下のいずれかにより表現される.

$$\sum_{R'\in P_m(R,S,I)} Q_{P_j(R',S,I)}(R',S,I),$$

$$\sum_{S'\in P_m(R,S,I)} Q_{P_j(R,S',I)}(R,S',I),$$

$$\sum_{R'\in P_m(R,S,I)} E_l(R',S,I),$$

$$\sum_{S'\in P_m(R,S,I)} E_l(R,S',I)$$

たとえば，STORE 命令の処理において，自動的にレコードと結合されるセットは複数存在しうるので，セットへレコードを結合するプログラムの呼出しの表現に用いられる．処理対象レコード名およびセット名が呼出し側と同じプログラムの呼出しの繰返しは，(3)の表現要素，繰返し回数を返すパラメータ $P_j(R,S,I)$，および，積記号 "＊" を用いて以下のように表現される.

$P_j(R,S,I)*E_l(R,S,I)$

プログラム呼出しが確率的に定まる場合は，P_j が確率値を返す.

（5） DBMS 実行状態の更新

トリガ $T_k(R,S)$ で表現される.

算出式は，上記の表現要素を加算記号 "＋" で結合した右辺，および，算出式の名前となる関数名および変数をもつ左辺で表現される．パス内の多重繰返しの表現等に用いられる "(", ")"，および，算出式の作成時に定まる確率値の表現に用いられる小数点定数も算出式に使用できる．算出式の右辺の各表現要素は，その要素対応の処理が実行される順序に従って右辺の左端から順次並べられる．図1のパスに対応する算出式を**図2**に示す.

DBMS，OS は算出式の集合として表される．一つ

$$E_1(R,S,I)=C_1+C_2+\sum_{S'\in P_1(R,S,I)} Q_{P_3(R,S',I)}(R,S',I)$$
$$+C_3+P_3(R,S,I)*C_4+(1-P_3(R,S,I))*C_5$$

注） P_3；プログラム Q のなかでパスを識別するパラメータ

（1） 図1(3)のパス1の算出式
$E_3(R,S,I)=C_1+C_6$

（2） 図1(3)のパス2の算出式

図 2 算出式の例

Fig. 2 Estimation expression.

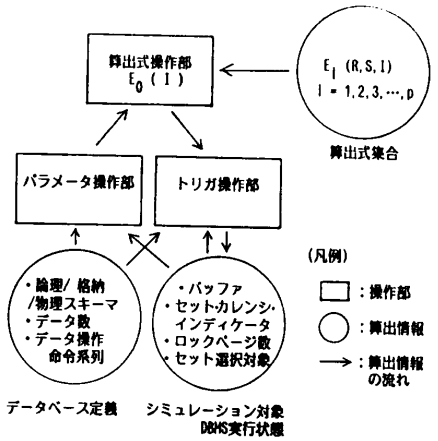

図 3 算出機構の概念図

Fig. 3 Outline of calculation method.

のデータ操作命令対応の CPU 時間算出は算出式 $E_0(I)$ から開始され，命令の内容をパラメータで反映しながら，必要な算出式が順次選ばれる．DBMS, OS 内で順次実行される各プログラム中の制御フローの分岐や繰返しを反映できる本方式の算出式は，データ操作命令ごとの CPU 時間を決める主要パラメータのみを反映できる従来の算出式より，CPU 時間に関する詳細な表現が可能である.

3.5 CPU 時間算出法

図3の算出機構により CPU 時間が算出される．パラメータ操作部は算出式操作部からのパラメータ P_j による依頼に基づき，データベース定義および DBMS 実行状態を参照しながら P_j の値を生成して，その値を算出式操作部へ返す．トリガ操作部は算出式操作部からのトリガ T_k による依頼に基づき，データベース定義および DBMS 実行状態を参照しながら T_k の値を生成して，その値により DBMS 実行状態を更新する.

算出式操作部は算出式 $E_0(I)$ から操作を開始して，以下の手順に従い，CPU 時間を算出する.

(1) 算出対象データ操作命令の決定

データ操作命令系列の先頭から順次取り出された命令が算定対象に決められる．その命令の順番 i を算出式 $E_0(I)$ の変数 I へ代入して(2)へ進む．算出対象の命令がなくなれば操作は終わる．

(2) 算出式の要素の操作

算出式 $E_0(i)$ の右辺において，定数および "+", "*", "(", ")" の演算記号を除く最左端に現れる各要素に対して，以下の操作を繰り返す．右辺が定数および演算記号のみになれば，数値計算を行い，その結果を算出対象命令の CPU 時間とする．そして(1)へ帰る．

$$P_j(r,s,i) \tag{2-1}$$

パラメータ操作部へ依頼して P_j の値を得て，$P_j(r,s,i)$ をその値で置き換える．P_j の値が数値0の場合は，その P_j と直接または "(", ")" を介して積記号 "*" で結合されているすべての要素を消去する．これは，P_j が0の場合は，本来，実行されないはずの処理に対応しているトリガが，DBMS 実行状態を更新しないようにするためである．ここで，r, s は変数 R, S の値を示す．以後も同様である．

$$E_i(r,s,i) \tag{2-2}$$

算出式集合から算出式 E_i を取り出して，変数 R, S, I へおのおのの値 r, s, i を代入し，その右辺の両端へ "(", ")" を付加したもので $E_i(r,s,i)$ を置き換える．

$$Q_{P_j(r,s,i)}(r,s,i) \tag{2-3}$$

パラメータ操作部へ依頼して P_j の値 p を得て，$Q_{P_j(r,s,i)}(r,s,i)$ を $Q_p(r,s,i)$ で置き換える．

$$\sum_{R' \in P_m(r,s,i)} Q_{P_j(R',s,i)}(R',s,i), \tag{2-4}$$

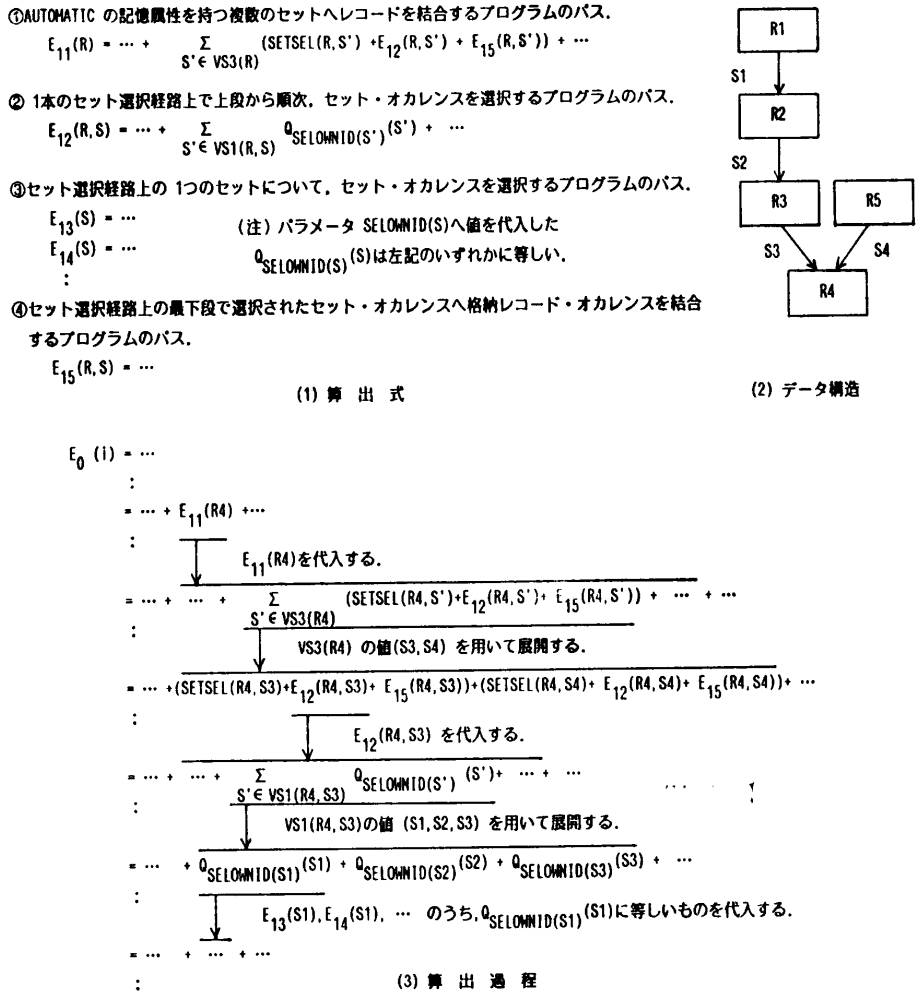

図 4 STORE 命令に関する算出式および算出過程の例

Fig. 4 Examples of estimation expression and calculation of STORE operation.

$$\sum_{S' \in P_m(r,s,i)} Q_{P_j(r,S',i)}(r, S', i),$$

$$\sum_{R' \in P_m(r,s,i)} E_l(R', s, i)$$

$$\sum_{S' \in P_m(r,s,i)} E_l(r, S', i)$$

パラメータ操作部へ依頼して P_m の値 p を得て，式を展開する．たとえば，$p=(r_1, r_2, \cdots, r_t)$ の場合，上記の第一式は以下のように展開される．

$$Q_{P_j(r_1,s,i)}(r_1, s, i)+Q_{P_j(r_2,s,i)}(r_2, s, i) +\cdots+Q_{P_j(r_t,s,i)}(r_t, s, i)$$

$$T_k(r, s) \qquad (2\text{-}5)$$

トリガ操作部へ T_k による DBMS 実行状態の更新を依頼するとともに，T_k を数値0に置き換える．

算出式の右辺の各表現要素は，対応する処理の実行順序にしたがって，左端から順次並べられている．本算出法では，算出式の右辺の左端から順次，要素が操作されるので，プログラム呼出し，DBMS 実行状態の更新および参照の順序は，実際の DBMS における実行順序とおおむね対応する．

3.6 算出式および算出過程の例

STORE 命令の処理においては，レコードの格納位置決定，格納空間確保，格納，および，レコードとセットの結合の処理が実行される．結合処理に関する算出式の一部を**図4** (1)に示す．図4 (2)のデータ構造のレコード R4 の STORE 命令に関して，結合処理の CPU 時間算出過程の一部を図4 (3)に示す．なお，セット S3 のセット選択経路は S1, S2, S3 から構成され，セット S4 のセット選択経路は S4 のみである．また，レコード R4 はセット S3, S4 に関して，AUTOMATIC の記憶属性をもつ．

図4のように，DBMS の実行が複雑になるセット選択処理に関しても，プログラムの呼出し関係，および，プログラムの処理対象レコード名やセット名の決め方が算出式に表現され，CPU 時間算出が可能である．

4. 適 用 例

電電公社の DEIMS-3[8] の CODASYL 型データベースを対象として，本方式を適用した DBDESIGN を開発した．DBDESIGN においては，CPU 時間は DS 数と CPU 平均命令実行時間の積で表され，算出式は DS 数の算出式として実現されている．算出式は約800個，パラメータとトリガは約100種である．算出式操作部は，算出式を直接実行するインタプリタとして実現されており，算出式集合は式表現のまま，データベースで管理されている．このため，個数は多いが算出式の作成，保守は困難ではない．なお，実現の容易性を考慮して算出式の要素の表現は**表2**のように代えられている．

表2 DBDESIGN における算出式要素の表現法

Table 2 Expression elements of estimation equation in DBDESIGN.

項番	算出式の要素	DBDESIGN における表現法
1	$P_j(R, S, I)$	P_j
2	$T_k(R, S, I)$	T_k
3	$E_l(R, S, I)$	E_l
4	$Q_{P_j}(R, S, I)(R, S, I)$	$Q!P_j$
5	$\sum_{R' \in P_j(R,S,I)} E_l(R', S, I)$ / $\sum_{S' \in P_j(R,S,I)} E_l(R, S', I)$	$E_l \$ P_j$
6	$\sum_{R' \in P_j(R,S,I)} Q_{P_k(R',S,I)}(R', S, I)$ / $\sum_{S' \in P_j(R,S,I)} Q_{P_k(R,S',I)}(R, S', I)$	$Q' \$ P_j$ (注1) / $Q'=Q!P_k$

(注1) 項番 4, 5 の要素に分解して表現する．
(注2) 定数，"*", "+", "(", ")"の表現法は DBDESIGN においても同じ．

図5に示す社内データベースを対象として，DBDESIGN の推定誤差を調べた．図5のデータ構造中の大部分のセットのセット順序は SORTED であり，セット選択は一段または二段である．最上位のセット以外のセットは NEXT ポインタと OWNER ポインタをもち，セット S11, S12 はインデクスをもつ．3.1節のデータベース・モデルを条件として DEIMS-3 上に構築されたデータベースを稼動させて，命令語トレーサで DS 数を実測した．この DS 数へ CPU 平均命令実行時間をかけて CPU 時間の実測値を得た．DBDESIGN による推定値 P，および，実測値 M を用いて，推定誤差を $|(P-M)/M| \times 100(\%)$ で表す．

おのおののデータ操作命令に対する推定誤差は図5のように20%以内であった．DBMS がバッファ上のレコードのみへアクセスする GET 命令，および，キーでないデータ項目に対する MODIFY 命令の推定誤差は数 % 以内であった．その他の命令種別と推定誤差の間に関係は見られなかった．DBMS が命令処理中にアクセスするレコード・オカレンス数と推定誤差の間にも関係は見られなかった．データ操作命令系列に対する推定誤差は10%以内であった．筆者らは，3.1節のデータベース・モデルの環境のもとで，

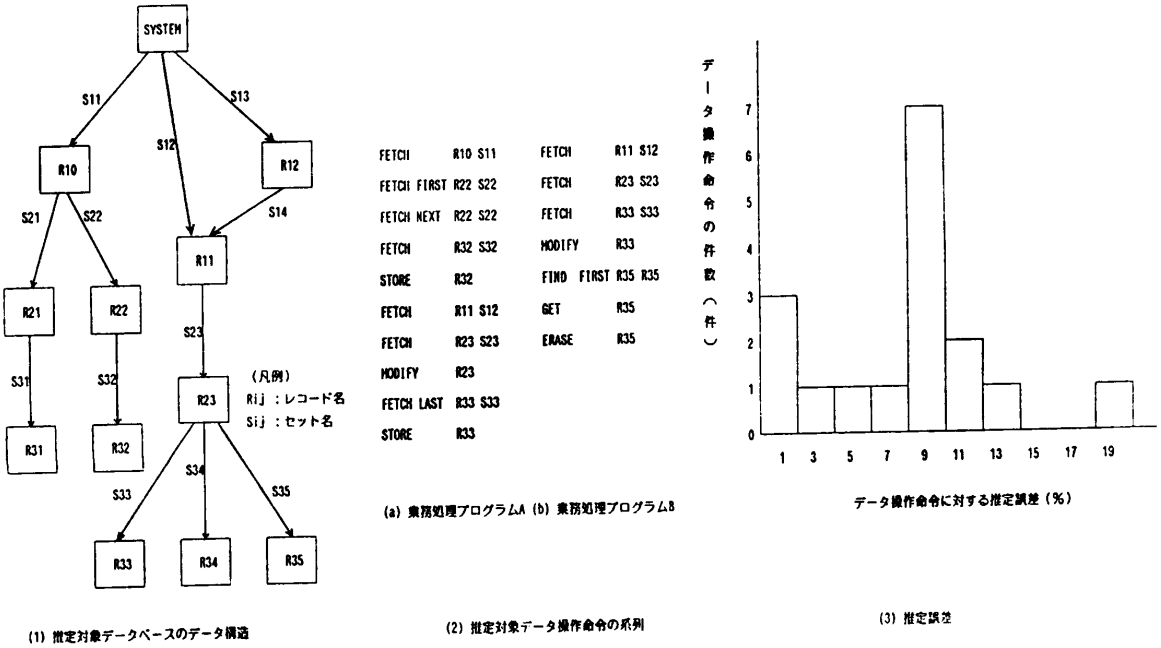

図 5 推 定 例

Fig. 5 Experiments.

DBDESIGN の CPU 時間推定誤差は 20% 以内であり，本方式が精度のよい CPU 時間推定に有効であると考えている．

5. おわりに

解析法を用いた CODASYL 型データベース対象の CPU 時間推定方式，および，適用例を報告した．そのおもな内容は以下のとおりである．①データ操作命令の処理中に，DBMS，OS 内で順次実行される各プログラムごとによい精度で CPU 時間を推定し，積算することができる．②適用例における CPU 時間推定誤差は 20% 以内であった．このことは本方式が精度のよい CPU 時間推定に有効であることを示している．③インタプリタで直接実行される算出式は，式表現のまま，データベースで管理できるので，適用例では約 800 個になった算出式の作成，保守は困難ではない．

本方式に関する今後の課題は以下のとおりである．プログラム制御フローを表す有向グラフの上のあるノードから非零の確率値で分岐が定まる相互に排他的な処理は，あたかも，算出式に表現された順序で直列に実行されているかのように，DBMS 実行状態は参照，更新されている．解析法とシミュレーション法の完全な融合が望まれる．

DBDESIGN に関しては，今後，性能面からデータベース設計内容の改善を支援するエキスパート・システムへ拡張する予定である．

謝辞 本研究についてご指導，ご協力いただいた高村部長，寺島室長，藤調査役，森室長，データ応用研究室員，DBDESIGN 関係各位へ厚くお礼申し上げる．

参 考 文 献

1) Pezarro, M. T.: Analytic Evaluation of Physical Database Designs: Two Case Studies, *Computer Performance*, Vol. 2, No. 2, pp. 53-64 (1981).
2) Teorey, T. J. and Oberlander, L. B.: Network Database Evaluation Using Analytical Modeling, Proc. NCC, pp. 833-842 (1978).
3) 米田 茂，山口康隆：汎用 DBMS の性能推定システム，情報処理学会論文誌，Vol. 20, No. 4, pp. 314-321 (1979).
4) Kawagoe, K. and Managaki, M.: Performance Evaluators for Designing Database Application Systems, *NEC R. & D.*, No. 61, pp. 79-88 (1981).
5) Beizer, B.: Analytical Techniques for the Statistical Evaluation of Program Running Time, Proc. FJCC, pp. 519-524 (1970).
6) Toh, T., Kawazu, S. and Suzuki, K.: Multi-Level Structures of the DBTG Data Model for

an Achievement of the Physical Data Independence, Proc. 3rd VLDB, pp. 403-413 (1977).
7) Oberlander, L. B. and Teorey, T. J.: The Database Design Evaluator Design Specifications, Working Paper DE 7. 2-3, Graduate School of Business Administration, University of Michigan (1977).
8) 鈴木健司，岡田静夫，伊藤健治，田中 豪：分散形データベース管理システム (DEIMS-3) の構成，情報処理データベース・システム研究会資料，37-1 (1983).

(昭和59年 8 月 6 日受付)
(昭和59年11月15日採録)