

LDP-V1.5 の総合評価

滝 沢 誠 横 塚 実

財団法人日本情報処理開発協会

1. 序

ローカルデータベースプロセッサ (LDP-V1.5) (TAKIM80, 82a, b) は, CODASYLデータベースシステム上の検索利用関係インタフェースシステムである。本システムは, 異種分散型データベースシステム JDDBS (TAKIM78, 79, 80, 82a, b) の異種データベースシステムとしての CODASYLデータベースシステムに対する一般ユーザ向けの共通非手続的検索利用インタフェースとともに, COBOL DMLプログラムの開発支援システムとなる。本システムは, 当協会の M-170F の AIM, A c o s - 7 0 0 の A D B S 上で 1981 年初めより既に実働している。

本論文では, LDP-V1.5 の有効性と有用性を示すために, LDP-V1.5 の性能面からと利用面からの総合的な評価について述べる。2 章では LDP-V1.5 の概要を述べ, 3 章では性能評価を, 4 章では利用面からの評価について述べる。本評価によって, 種々の QUEL 問合せに対して数秒～数 10 秒の経過時間によって, 人間が作れば数時間以上要する COBOL DML プログラムを生成できると共に, これを実行させ結果を自動的に得れることが明らかになった。

2. LDP-V1.5 の概要

LDP-V1.5 は, 以下の機能を有している (TAKIM80, 81, 82a)。

- (1) CODASYL スキーマから, 検索用の関係スキーマ (LCS) の生成
- (2) QUEL 問合せから COBOL DML プログラムの生成
- (3) 生成された COBOL DML プログラムの実行
- (4) 実行結果を関係としてユーザに出力
- (5) LCS に対するビューの階層的定義と検索演算サポート

2. 1 スキーマ変換

CODASYLスキーマ (ローカル内部スキーマ (LIS)) から, 関係スキーマ (ローカル概念スキーマ (LCS)) は, 次の様にして生成される (TAKIM81)。

(1) レコード型 $R(t_1, \dots, t_n)$ に対して, 関係スキーマ $\underline{E}(R)$ ($\underline{@E}(R)$, $a(t_1), \dots, a(t_n)$) が生成される。この $\underline{E}(R)$ は, レコード型 R に対応した関係名であり, $a(t_i)$ は t_i の属性名である。 $\underline{@E}(R)$ は, R の dbキーを値として取る仮想属性で主属性と呼ばれる。関係 $\underline{E}(R)$ 内の組は, R のレコード実現値に 1対1に対応する。 $\underline{E}(R)$ の組は, $\underline{@E}(R)$ の値, 即ち, 対応する実現値の dbキーによって識別される。 $\underline{@E}(R)$ は, $\underline{E}(R)$ の主キーともなる。関係 $\underline{E}(R)$ を \underline{E} 関係と呼ぶ。

(2) 親レコード型を A , 子レコード型を B とするセット型 S に対して, 関係スキーマ $\underline{R}(S)$ ($\underline{@E}(A)$, $\underline{@E}(B)$) が生成される。ここで $\underline{R}(S)$ は, セット型 S に対応した関係名である。 $\underline{@E}(A)$ と $\underline{@E}(B)$ は, 各々レコード型 A と B から生成された関係スキーマ $\underline{E}(A)$ と $\underline{E}(B)$ の主属性である (図 2. 1)。 $\underline{@E}(B)$ は, 主キーともなる。

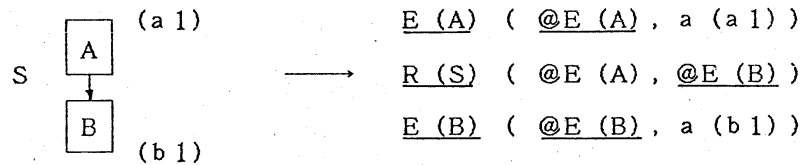


図 2. 1 セット型

(3) 図 2. 2 の様な n 個のレコード型 R_1, \dots, R_n 間の関係性を表すレコード型 (リンクレコード型と呼ぶ) に対しては, 関係スキーマ $\underline{R}(L)$ ($\underline{@E}(R_1), \dots, \underline{@E}(R_n), a(t_1), \dots, a(t_n)$) が生成される。

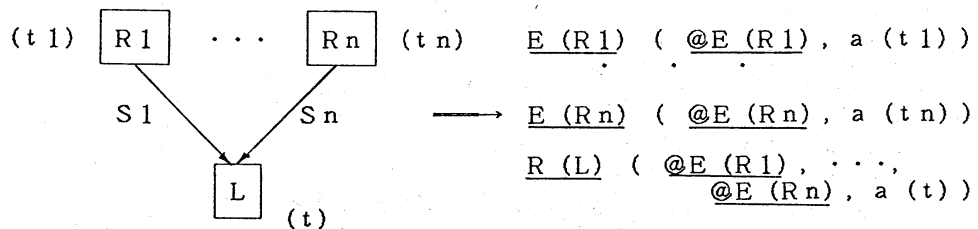


図 2. 2 リンクレコード型

図2. 3の様に $R_i = R_j = R$ の時には, $R(L)$ の主属性が $S_i @ E(R)$, $S_j @ E(R)$ となる。

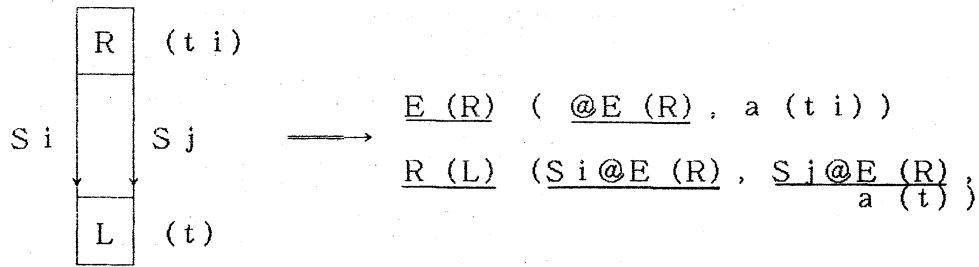


図 2. 3 リンクレコード型

図2. 4のCODASYLスキーマ (LIS) から生成される関係LCSは, 図2. 5のようになる。ここで簡単にする為に $a(t_i) = t_i$, $E(R) = R$, $R(S) = S$, $R(L) = L$ とする。

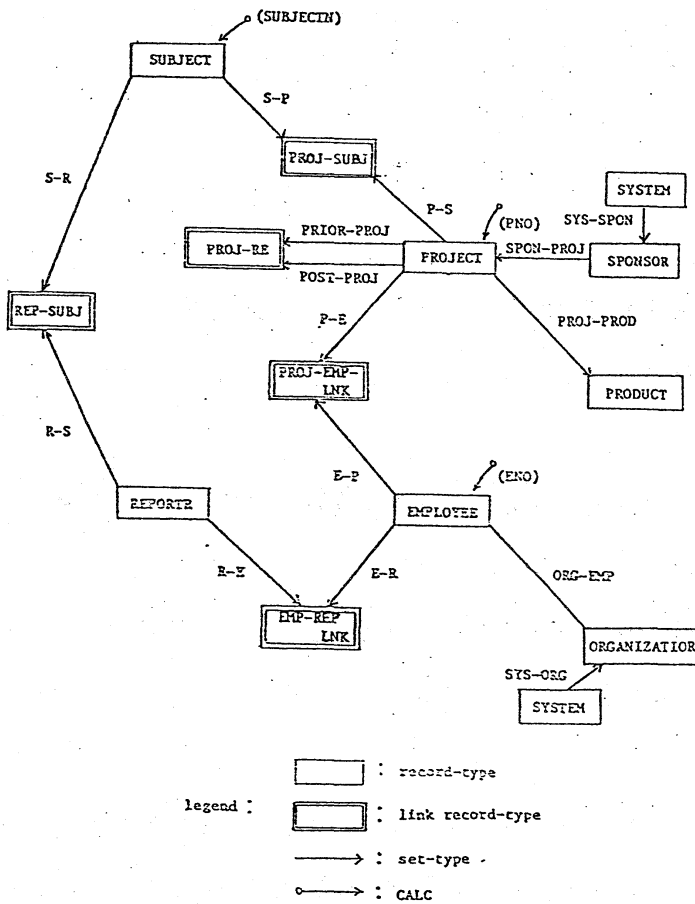


図 2. 4 PRDBSのスキーマ

```

ESR SUBJECT (@J, subjectn)
SPONSOR (@SPONSOR, sname, stype)
PROJECT (@P, pno, pname, proj-type,
          status-f, syear, eyear, budget)
PRODUCT (@PRODUCT, pd-no, pd-name, ptype,
          cost, year, language, psize)
EMPLOYEE (@E, eno, ename, efname, esex,
          birth-place, birth-year,
          birth-month, alma-mater, major,
          degree, deg-year, hired-year,
          retired-year, dept, position,
          tel, ext)
REPORTR (@R, rno, title, source-n, vol,
          numb, month, year, from-page,
          to-page, total-page,
          no-of-author, language,
          written-year, cost, rtype,
          rstate)
ORGANIZATOR (@ORGANIZATOR, org-name,
              org-type)

RSR SPON-PROJ (@SPONSOR, @P)
PROJ-PROD (@P, @PRODUCT)
PROJ-EMP-LNK (@P, @E, smonth, eyear,
              emonth, pposition, role,
              percentage)
PROJ-SUBJ (@P, @J)
REP-SUBJ (@R, @J)
EMP-REP-LNK (@R, @E, auth-no)
PROJ-RE (@P, PRIOR-PR-@P)
ORG-EMP (@ORGANIZATOR, @E)

```

図 2.5 PRDBSのLCS

2.2 問合せ変換

問合せ変換 (TAKIM80, 82b) とは、QUEL問合せから、COBOL DMLを生成し、実行させ結果を得ることである。QUELとCOBOL DMLプログラムは、次の2点が各々異なっている。

- (1) 参照するデータ構造 リレーショナル構造 CODASYL構造
- (2) アクセス単位 非手続的と手続的

この為に、QUEL問合せの意味を、CODASYLデータ構造によって、述語計算に基づいた非手続的表現を導入した。これを、CODASYL問合せグラフ (CQG) と呼ぶ。問合せ変換は、次の2つのステップから成る。

- (i) 構造変換 QUEL問合せからCODASYL問合せグラフ (CQG) を生成することによって (1) の問題を解決する。
- (ii) アクセスパス生成 CQGからアクセス手順を生成し、(2) の問題を解決する。

2. 2. 1 構造変換

図2. 5に対する『プロジェクトの開始以前に雇用されたメンバが、第一著者である論文の主題と、そのプロジェクトの主題が異なっている論文とそのプロジェクトの番号を求めよ。』は、QUELで図2. 6の様に表せる。この問合せは、図2. 7の様なグラフで表すことが出来る。これを、LCS問合せグラフ (LQG) と呼ぶ。

```

RANGE (E, EMPLOYEE) (P, PROJECT) (R, REPORTR) ;
RANGE (J, SUBJECT) ;
RANGE (PJ, PROJ-SUBJ) (RJ, REP-SUBJ) ;
RANGE (PE, PROJ-EMP-LNK) (ER, EMP-REP-LNK) ;
RETRIEVE INTO ANTI010 (P, PNO, R, RNO)
WHERE
P. @P=PE. @P AND PE. @E=E. @E AND
E. @E=ER. @E AND
ER. @R=R. @R AND R. @R=RJ. @R AND
P. @P=PJ. @P AND PJ. @J=J. @J AND
J. @J NE RJ. @J AND
P. SYEAR GE E. HIRED-YEAR AND
ER. AUTH-NO=1;

```

図2. 6 問合せ例

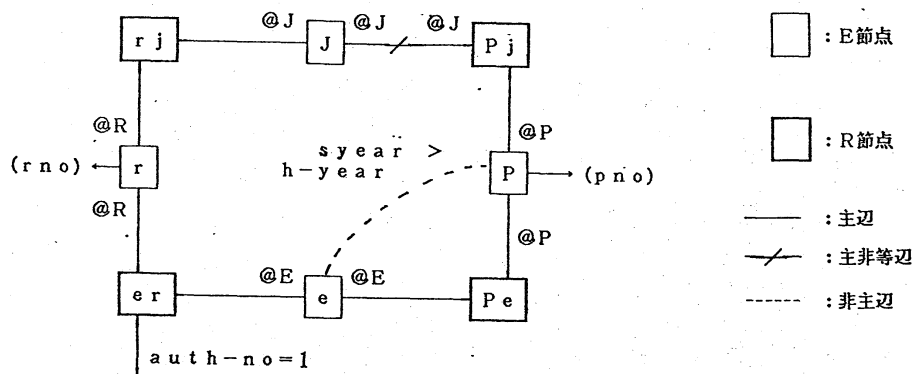


図2. 7 LQG

LQG節点は、組変数を表し、節点間の辺は結合を表す。主属性間の結合としては、 $=$ と \neq のみが許される。2. 1で述べた様に、主属性が値として、組の識別子（即ち、dbキー）を取る為に、 $<$ 、 \leq 、 \geq 、 $>$ の比較は意味がないからである。実線は主辺と呼ばれ、主属性間の等結合を表し、 \neq は非等（ \neq ）結合を表し、主非等辺と呼ばれる。点線は、主属性以外の通常の属性間結合を示し、非主辺と呼ばれる。 \perp は制限式を、 \rightarrow は目標属性を示す。LQG内の

辺と節点の表す条件式の積が問合せの条件になっている。

LQGから、表2. 1に従って変換されたグラフが、CODASYL問合せグラフ(CQG)である(図2. 8)。CQG節点は、実現値変数を示している。節点間の辺は、セット型を示している。変換は、E関係の節点を、対応するレコード型の実現値変数に変え、節点間の辺を表2. 1に基づいて対応するセット型に変えることによってなされる。2つのCQG節点xとyの間のセット辺Sは、xとyが表す実現値が同一のセット実現値内に属さねばならない条件を表している。CQGの辺と節点の表す条件式の積は、LQGに於けるものと等価である。CQGは、QUEL問合せ(図2. 6)の意味を、CODASYLデータ構造によって、非手続的に表したものである。

表2. 1 LQGとCQGの対応表

1) Yはセット型

L Q G	C Q G

2) Yはリンクレコード型

L Q G	C Q G

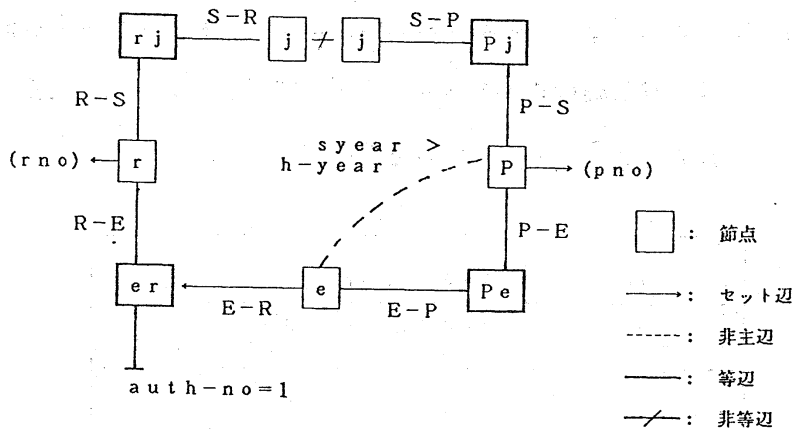


図2. 8 CQG

2. 2. 1 アクセスパス生成

アクセスパスは、CQGのセット型を表す辺を縦型に、サーチすることによって得れるアクセス木によって表される。アクセス木は、アクセスされる実現値数になるべく少なくなる様に生成される。アクセス木生成で、可能な辺が複数存在する時、次の辺は、選択度、結合度といった統計情報を用いたアクセスされる実現値数見積りに基づいて決定される。

A. 結合度

親をA、子をBとするセット型Sに対して、結合度 $cnt(S)$ と $icnt(S)$ とを次の様に定義する。

$$cnt(S) = A \text{ の各実現値が、 } S \text{ の子として持つ } B \text{ 実現値の平均数}$$

$$icnt(S) = B \text{ の各実現値が、 } S \text{ の親として } A \text{ 実現値を持つ確率}$$

アクセス木の枝l (図2. 9) に対して、

$$cnt(S) \quad \text{if } x \text{ は } S \text{ の親}$$

$$ctl = \{$$

$$icnt(S) \quad \text{otherwise}$$

Sは、枝lの表すセット型である。

B. 選択度

節点 a に対して, st_a を, a に関する制限式によって制限される実現値の割合, 即ち選択度とする。 st_a は, 制限論理式 $rf(a)$ によって表 2. 2 の様に定義される。

表 2. 2 $rf(a)$ の選択度

$rf(a)$	選択度 $st(rf(a))$
$r. a = v$	$card(a) / card(r)$ (= st_a と記す)
$r. a \neq v$	$1 - st_a$
$r. a \left\{ \begin{array}{l} > \\ < \end{array} \right\} v$	$(1 - st_a) / 2$
$r. a \left\{ \begin{array}{l} \geq \\ \leq \end{array} \right\} v$	$(1 + st_a) / 2$
$r. a = v_1 \vee \dots \vee r. a = v_n$	$n \cdot st_a$
$rf_1 \wedge \dots \wedge rf_n$	$st_{f1} \cdot st_{f2} \cdot \dots \cdot st_{fn}$ ここで $st_{fi} = rf_i$ の選択度
$rf_1 \vee \dots \vee rf_n$ (rf_i は互いに異なった属性を参照している)	$1 - (1 - st_{f1}) \cdot \dots \cdot (1 - st_{fn})$

C. 発見法

パスとして可能な候補がある場合, 次の様な発見法を用いて, 次の辺を選択する。発見法としては, 1レベルのサーチ, 2レベルのサーチ, 3レベルのサーチの3種を考えた。各サーチレベルの深さにより, それぞれ次の評価関数を用いて, 各評価関数 acc_i の値を最少とする辺 l を x の次の辺として選択する。

$$acc_1(x, l, y) = ct_l$$

$$acc_2(x, l, y) = ct_l \cdot (1 + sty \cdot \min_{(m)} (ct_m))$$

$$acc_3(x, l, y) = ct_l \cdot (1 + sty \cdot \min_{(m)} [ct_m \cdot (1 + stz_{(m)} \cdot \min_{(n)} (ct_n))])$$

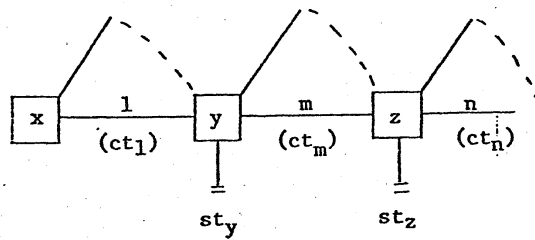


図2.9 アクセス木

3. 性能評価

本章では、LDP-V1.5の性能面からの評価について述べる。性能評価は、以下の点を目標にしている。

(1) LDP-V1.5の各構成モジュールの負荷を明らかにする。これから、性能面での隘路を求め、改良を行う。

(2) アクセスパス生成の負荷と、それによって生成されたCOBOL DMLプログラムのアクセス効率を明らかにする。

(3) 選択度、結合度といった統計情報の正しさの検証を行う。

3.1 評価環境

LDP-V1.5を、以下の環境下で評価する。

(1) LDP-V1.5を、M-170F上で実働させる。Acoss-700に対しては、M-170F上のLDP-V1.5で生成されたCOBOL DMLプログラムをADBS上で動かす。

(2) LDP-V1.5を単一利用者の下で実働させ、その経過時間 (e l a p s e t i m e) を測定する。親言語の時間取得機能 (c a l l C P T I M E) で得た経過時間は、M-170Fでは10%から20%の誤差があるが、今回は各評価問合せに対して各一回実行させる事とした。

(3) 評価用のデータベースとしては、図2.4で示したプロジェクトデータベースシステム

(PRDBS) を用いる。PRDBS内の各レコード型、セット型、リンクレコード型の実現値数とサイズを表3. 1に示す。

表3. 1 PRDBSの実現値情報

関係性名	タイプ	実現値数	レコード型	実現値数
SYS-ORG	セット	1	EMPLOYEE	68
SYS-EMP		1	ORGANIZATIO	15
SYS-SPON		1	PRODUCT	9
ORG-EMP		15	SUBJECT	136
SPON-PROJ	リンク	4	PROJECT	14
PROJ-PROD		9	REPORTR	84
E-P		62	SPONSOR	4
P-E		62		
E-R		155		
R-E		155		
R-S		625		
S-R		625		
P-S		201		
S-P		201		
PRIOR-PROJ		5		
POST-PROJ		5		

3. 2 各構成モジュールの負荷

図3. 1に、2章で述べた各構成モジュールの負荷を、各々の経過時間で示す。図から分かる様にDMLGでのCOBOL DMLプログラムを文字列として生成する為に全経過時間の70%から90%が費やされてしまう。アクセスパス生成の為に経過時間の割合は高々数%にしか過ぎない。又、QUEL問合せから内部表現としての木、グラフの生成と操作の為に10%から20%が費やされている。

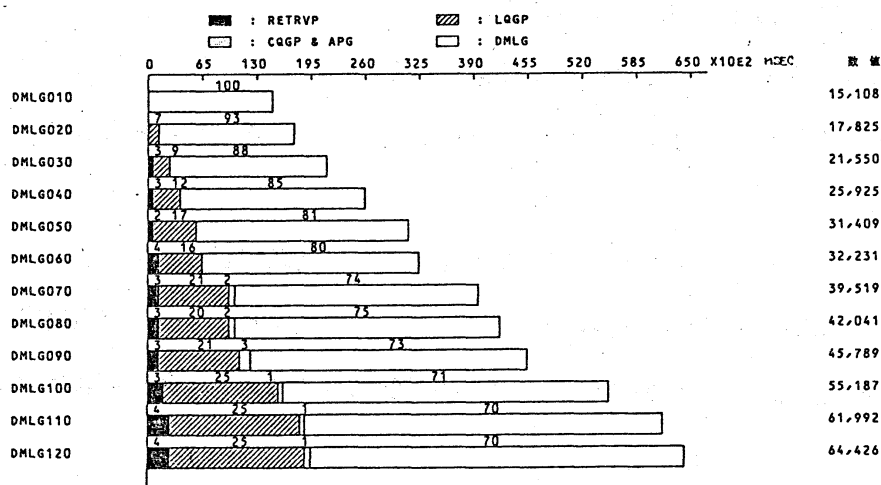


図3. 1 各構成モジュールの負荷

LDPの性能を向上させる為に、DMLGの改良を行った。DMLGでは、DMLの文字パターンを一文字ずつ左から右にサーチして、パラメータを見付けて、その置換を行っている。

例 find next \$01 within \$02.

→ find next EMPLOYEE within ORG-EMP.

この為に、以下の改良を行った。

- (i) パラメータを左づめする。
- (ii) これより変換の不要となる右端を明示する。
- (iii) コメント文を除去する。

これによって、図3. 2に示す様にDMLGの経過時間を30%から40%短縮できた。

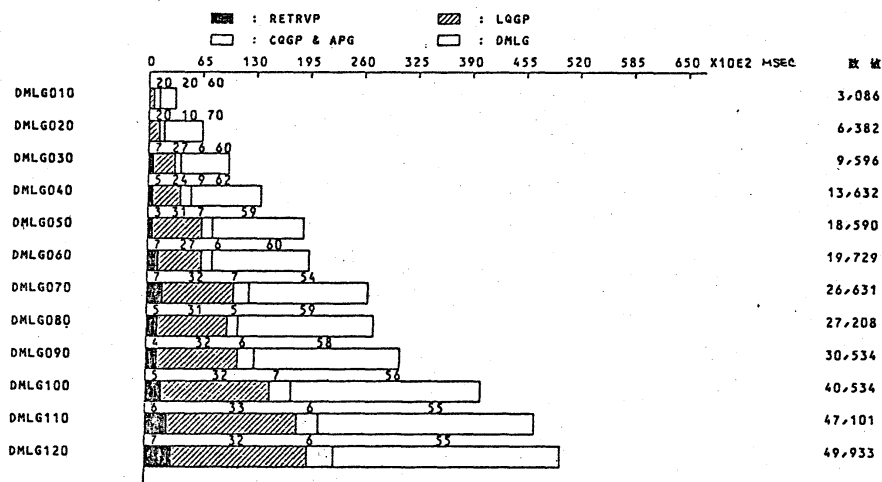


図3. 2 改良後の各構成モジュールの負荷

3. 3 アクセスパス生成の負荷

アクセスパスは、DFA 1, 2, 3の三種の発見法によって、CODASYL問合せグラフ(CQG)から生成される。アクセスパス生成に必要な経過時間を、CQGの辺数に対して、プロットしたものが図3. 3である。

DFA 1は、一レベルのサーチを行う発見法である。図から分る様にほぼ線形に経過時間が増加している事が分る。DFA 2もほぼ線形に経過時間が増加している。一方DFA 3は、辺数に対して、二乗のオーダで経過時間が増加している。

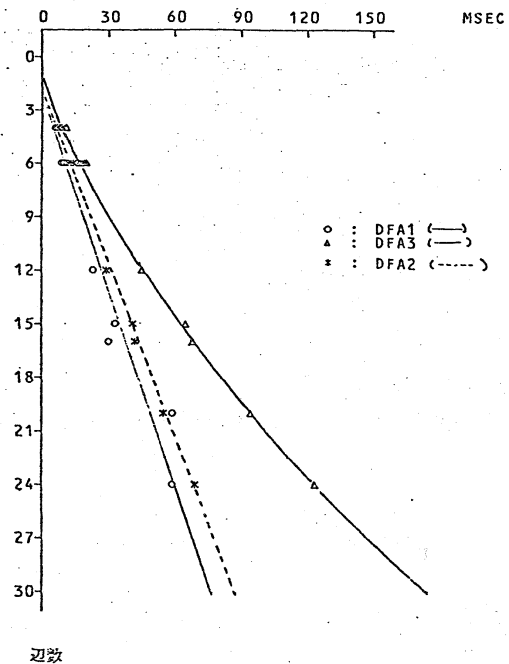


図3. 3 アクセスパス生成の負荷

3. 4 ア ク セ ス パ ス 生 成

アクセスパスは、生成されたCOBOL DMLプログラムの全経過時間を短縮する為に、アクセスされるレコード型の実現値の期待値が最少となる様に、選択度、結合度といった統計情報を基にして生成される。ここでは、COBOL DMLプログラムの経過時間がアクセスされる実現値数に単調増加するものと仮定している。図3. 4は、種々の問合せから生成されるCOBOL DMLプログラムについて、実際にアクセスされた実現値数と経過時間との関係を示している。図から分る様に、この仮定が正しい事が分る。

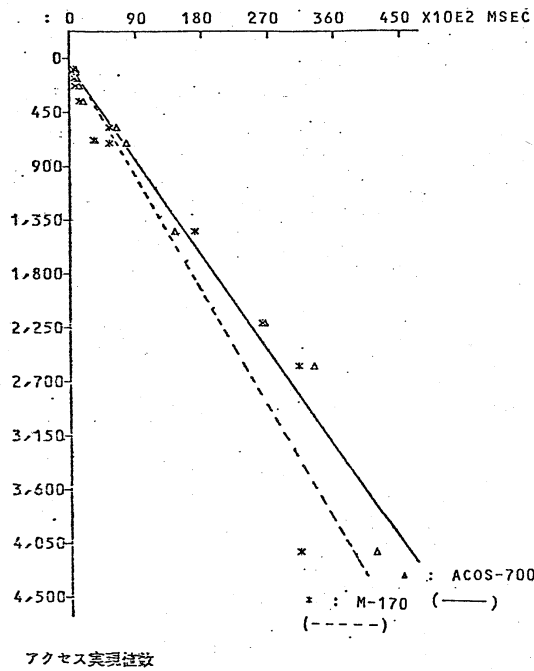


図 3. 4 アクセスされた実現値数と経過時間

アクセスパスは、選択度、結合度といった統計情報を基にして、2章で述べた様なコスト関数に基づいて見積られる。我々の用意した8個の問合せについて、見積りと、実際にアクセスされた実現値数とを表3. 2に示す。この例では、見積りと実際の値との間に、あまり相関がみられない。

表 3. 2 実現値数の見積りと実際

Query #	No. of Node	No. of Join	DFA 1		DFA 2		DFA 3	
			Est.	Act.	Est.	Act.	Est.	Act.
10	5	4	107	231	107	231	107	231
20	7	6	158	6981	158	6981	158	6981
30	5	4	31	92	31	92	31	92
40	12	12	194	954	267	993	306	2281
50	15	16	503	19267	503	34672	889	18681
60	21	24	3163	22839	3163	51251	5875	33747
70	14	15	577	4932	396	2411	396	1791
80	18	20	577	4932	549	2411	54	12090

統計情報が、データベース全体の実現値についての平均値であるのに対して、個々の問合せは、データベースの一部のみをアクセスする。このアクセスされる部分の値分布が均一でない為に、この様な現象が生じたと考えられる。従って、

(i) 一般に、データベース全体の値分布は、各アプリケーションのアクセスする値分布と一致しない。

(ii) 各アプリケーションのアクセスするデータベースの範囲は、比較的限られている。

これらの事より、見積りと実際の実現値数との関連を密接にする為に、以下の様な方法を用いる。

(i) 統計情報をアプリケーション毎に備える。データベース内で、アプリケーションがアクセスする部分についての実現値の統計情報を用意する。

(ii) 結合度をより現実と適合させる為に、次の様な動的なチューニング方法を用いる。Sをセット型とする。cnt (S), icnt (S) を各々Sの正、逆結合度とする。あるアプリケーションプログラムの実行について、ocnt (S) とmcnt (S) を、各々Sの親から子にアクセスした時の親と子のアクセスされた実現値の総数とする。oicnt (S) とmicnt (S) を各々、子から親をアクセスした時の子及び親のアクセスされた実現値の総数とする。

$$\text{cnt (S)} \leftarrow \text{if mcnt (S) } \neq 0, \text{ then} \\ \left[\text{cnt (S)} + \left(\text{ocnt (S)} / \text{mcnt (S)} \right) \right] / 2$$

$$\text{icnt (S)} \leftarrow \text{if micnt (S) } \neq 0, \text{ then} \\ \left[\text{icnt (S)} + \left(\text{micnt (S)} / \text{oicnt (S)} \right) \right] / 2$$

図3. 5は、動的チューニングによる見積りの改良を示している。

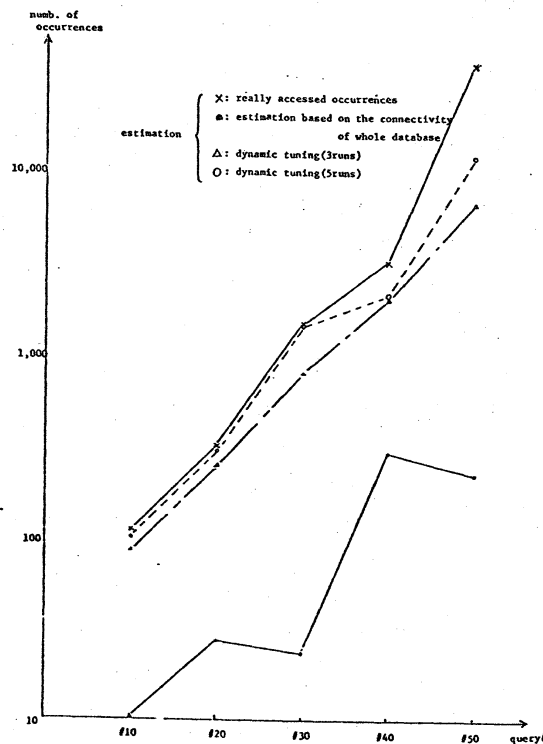


図3. 5 動的チューニングによる見積りの改良

図より、動的チューニングによって、結合度を現実の状態に近付ける事ができる事が分る。

図3. 6は、制限式を含んだ場合を示している。データベース全体の統計情報による見積りよりも、実際に近付いているが、図3. 4ほどではない。これは、選択度がアクセスされたデータベースの範囲の状況を反映していない事によっている。CODASYLデータベースシステムの選択度としては以下の点を考慮する必要がある。

- (1) レコード型の各項目の値毎に、この値を持つ実現値数を保持せねばならない。
- (2) レコード型全体と共に、各セット実現値内での子レコード実現値の値分布が必要になる。
- (3) 等制限式は、(1)及び(2)によって正しく求められるが、不等制限式($<$, \leq , \geq , $>$)に対する選択度を正しく求めるには、完全な値の分布が必要になる。

この為の情報全てを異種情報に持たすには、第1に格納空間の大きさが問題になる。(1)については、1つの方法は、よく用いる値に対してのみ情報を保つことである。しかし、(2)

については、アクセスコストの見積りの時点でセット実現値内での制限式の選択度を、反映させることは出来ない。何故なら、あるセット型でアクセスされる子実現値は、実際にアクセスされて、はじめてわかるからである。

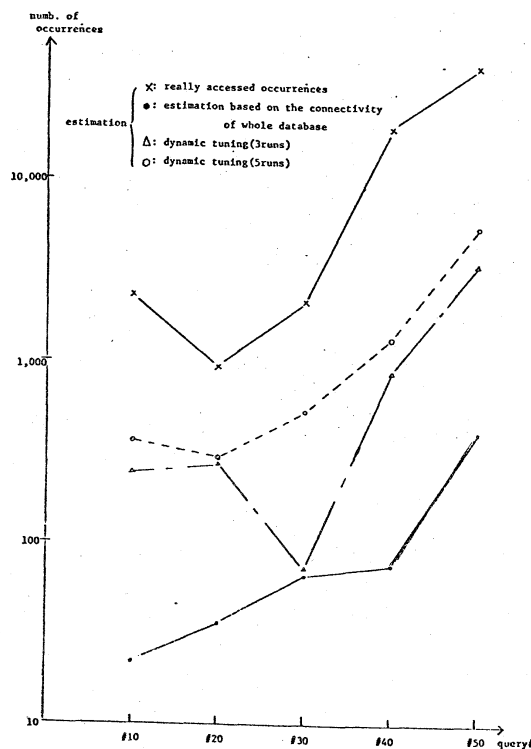


図3. 6 制限式を持った問合せに対する見積り

アクセスパスを見付ける為のサーチレベルの深さと、その効果について考える。ここでは、動的チューニングによって、統計情報が、実際の状態に合う様になっているとする。表3. 3は、この時の3つの方法DFA 1, 2, 3の見積り数と、実際のアクセス実現値数を示している。表から分る様に、サーチレベルを深めても、あまり効果はない。

表 3. 3 サーチレベルと効果

Query #	No. of Node	No. of Join	D F A 1		D F A 2		D F A 3	
			Est.	Act.	Est.	Act.	Est.	Act.
10	5	4	110	117	110	117	110	117
20	7	6	142	151	142	151	142	151
30	8	8	1903	2215	1903	2215	1903	2215
40	11	11	3799	1194	3799	1199	3801	1199
50	12	12	2318	3593	425	7585	3076	3335
60	13	14	3270	4697	6358	3343	6358	3343
70	14	15	8328	2623	5321	7656	5340	6190
80	18	20	824	3135	1563	1573	2423	8093

3. 5 COBOL DMLプログラム

生成されたCOBOL DMLプログラムの実行結果について考える。COBOL DMLプログラムは、アクセスされるレコード型とセット型の数（CQGのノード数とセット型の数）に対して、表3. 4に示す様なプログラム文数となる。

表3. 4 プログラム文数

QUERY #	NO. OF NODES	NO. OF JOIN	NO. OF COBOL STEPS		
			DATA DIV.	PROC. DIV.	TOTAL
R1	5	4	103	358	461
R2	8	8	112	426	538
R3	7	6	112	384	496
R4	16	15	137	594	731
R5	12	12	123	530	653

COBOL DMLプログラムの実行は、次の3部分から成る。

- (1) 初期化と終了
- (2) 実現値の巡航
- (3) 出力処理

これらをまとめると表3. 5の様になる。表から分る様に、初期化/終了の為の経過時間は、

ほぼ同一で次の様である。

表3. 5 COBOL DMLプログラムの経過時間

QUERY #	DBMS	COBOL ELAPSE TIME(MSEC)		
		INIT/END	NAVIGATION	OUTPUT
10	AIM	610	7220	250
	ADBS	7210	3730	430
20	AIM	640	195950	5860
	ADBS	6890	195840	37400
30	AIM	650	411240	13630
	ADBS	6750	618180	95850
40	AIM	650	45480	1200
	ADBS	6670	43350	8120
50	AIM	670	62940	1760
	ADBS	6660	10100	1800

AIM ~ 650 (ms)

ADBS ~ 6800 (ms)

4. 利用性評価

次に、LDP-V1.5のユーザから見た有用性について評価する。これは、以下の2つの場合の全ライフサイクルの経過時間を測定し、前者の有用性を経過時間として定量化する事を行う。

- (1) QUELによって問題を記述し、これをLDP-V1.5でCOBOL DMLプログラム変換し実行させ、結果を得る。
- (2) 同一の問題に対して、COBOL DMLを用いて、設計し、コーディングし、デバッグして、結果を得る。

4.1 評価方法

評価を次の様に行う。

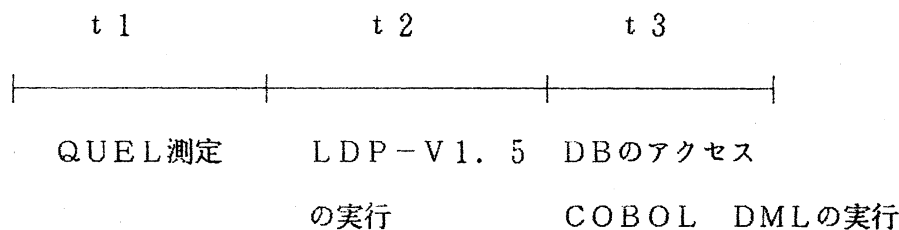
- (1) 同一の日本語の問題に対して、以下を行う。
 - (i) 計算機の素人によって、LCSに基づいて問題をQUELで記述する。

(ii) COBOLエキスパート（経験10年）によって、CODASYLスキーマに基づいて、問題をCOBOL DMLで記述する。

(2) M-170FのAIM上のPRDBSを用いる。

(3) QUEL及びCOBOL DMLによる問題解決のライフサイクルの各ステップの経過時間を測定する。〔図4.1〕

QUEL



COBOL DMLプログラム

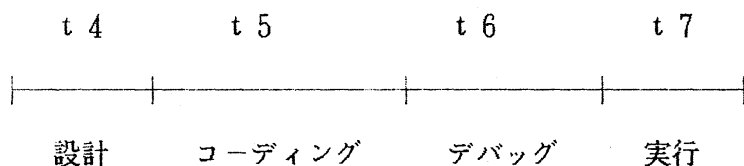


図 4.1 ライフサイクル

4.2 問題

問合せ問題を、以下に示す。

R1 各projectのメンバの書いたレポートの中で、このメンバが第一著者 (author-no=1) であるレポートの番号 (rno) とそのprojectの番号 (pno) とを求めよ。

R2 projectのテーマと同じテーマの論文を書いているprojectのメンバの番号 (eno) と、その論文番号 (rno) を求めよ。

R3 distributed database systems又は、computer

networks, 又は local networks, 又は, database systems, 又は data models を研究している project のメンバーの中で, Jipdec に属し, かつ 1970 年以降に入社 (hired year) したメンバーの名前 (ename), 番号 (eno), 及びその project 番号を求めよ。

R4 distributed database systems と, database systems と, four-schema structure と, query translation を研究している project のメンバーが書いたレポートの主題 (subjectn) と, そのメンバーの所属 (org-name) 及び, メンバーの番号を求めよ。

R5 project のメンバーの中で, project のテーマと同一のテーマの論文を書いているメンバーの名前と, そのメンバーが属している全ての project 番号及び, このメンバーが第一著者である全ての論文の番号を求めよ。

4. 3 評価結果

表 4. 1 に, 測定結果を示す。

表 4. 1 利用性評価結果

QUERY #	NO. OF RECORD TYPES	NO. OF SET TYPES	DESIGN CODING DEBUG	LDP ELAPSE TIME	NO. OF COBOL STEPS	NO. OF ACCESS OCCU.	COBOL ELAPSE TIME	
R1	5	4	Q	4.5 (M)	10 (S)	461	335	4.5 (S)
			C	10 (H)	/	104	335	3.5 (S)
R2	8	8	Q	5.3 (M)	16 (S)	538	9748	140 (S)
			C	10 (H)	/	334	4150	65 (S)
R3	7	6	Q	11 (M)	12 (S)	496	561	5.1 (S)
			C	10 (H)	/	117	205	3.5 (S)
R4	16	15	Q	14 (M)	22 (S)	731	1831	27.7 (S)
			C	17 (H)	/	256	1209	17 (S)
R5	12	12	Q	8 (M)	18 (S)	653	59709	516 (S)
			C	15 (H)	/	284	1025	18 (S)

C=COBOL, Q=QUEL, (H)=HOUR, (M)=MINUTE, (S)=SECOND

表から分る様, LDP では, 数分から数10分で結果を得れる。しかし, COBOL によっては,

結果を得るまでにエキスパートでも数時間から数10時間を要してしまう。この意味で、CODASYLデータベースシステム上に、非手続的な検索インタフェースを設ける事は、ソフトウェアの生産性の点から優れていることが分る。

アクセスパス生成では、簡単な問題に対して、LDPはエキスパート同様なアクセスパスを生成できるが、複雑な問題に対しては、エキスパートのものより劣っていることが分る。又、LDPによって生成されるCOBOL DMLプログラムの量でもエキスパートより数倍多くなっている。

アクセスパス生成でエキスパートとLDPとの差は、特にCQGがループを含む問合せにおいて、アクセス方法が異なる。LDPは、なるべく途中で中間結果を生成せずに一連の巡航アクセスによって結果を得ようとする。この為、合流節点等に於ける現在子の保持方法に注意を払っている。しかし、エキスパートは、むしろループを分割して中間結果を生成し、これを最後にまとめて処理しようとしている。

5. まとめと今後の課題

本論文では、CODASYLデータベースシステム上の非手続的な関係インタフェースシステム、ローカルデータベースプロセッサLDP-V1.5の性能及び利用面からの総合的な評価について述べた。

LDP-V1.5は、現在、当協会のM-170FのAIM及びAcos-700のADBS上に、PL/Iによって実現されている。LDP-V1.5は、PL/Iソース文数約10000、オブジェクトサイズ250kBである。

性能面の評価では、まずLDP-V1.5の各構成モジュールの負荷を経過時間によって測定し、隘路となるモジュールを明らかにした。隘路となるモジュールは、COBOL DMLプログラム文字列の生成を行うDMLGであり、全経過時間の過半数を占めていた。これの改良によって、全体に、20%~30%経過時間を短縮出来た。アクセスパス生成は全体の高々数%の負荷であり、LDP全体から見ると無視し得る程度である。現在、DMLGと共に、QUEL問合せからCODASYL問合せグラフ(CQG)を作るまでの負荷が大きな重みを占めている。ここでは、木、グラフ等の構造の非手続的な操作が行われるからである。

我々は、選択度、結合度といった統計情報によって、アクセスコストを見積り、その見積りをなるべく少なくする様にアクセスパスの生成を行っている。ここでの評価の目的は、次の3点の明確化である。

- (1) アクセスコストとして、アクセスされる実現値数とすることの妥当性
- (2) 選択度、結合度といった統計情報を用いた見積りの正しさ
- (3) サーチレベルを深めることによるLDPの負荷の増大と、その効果を明らかにする。

COBOLプログラム実行の中で、実現値を巡航する経過時間は、アクセスされる実現値数に単調増加する事が明らかになった。従ってアクセスされる実現値数を最少化する事によって、COBOL DMLプログラムの実行経過時間を短縮できる。

(2) では、データベース全体の平均値としての統計情報による見積りと、例で用いた問合せを実行した結果とはよく適合しない。この解決方法として次の2つの方法を用いた。

- (i) アプリケーション毎にこれらが用いる実現値についての統計情報を用意する。
- (ii) この統計情報の結合度情報については、各実行結果を反映させる動的チューニング方法によってアプリケーションがアクセスされる実現値の統計情報を反映させる。

これによって、見積りを、実際の値に近付けることができた。しかし、選択度の動的チューニング方法を行う為には、各属性の値毎にこの値を有する実現値数を記憶させると共にセット実現値毎の分布情報も同時に持たせねばならない。これは、格納負荷の増大をもたらしてしまう。又、スキーマレベルでのアクセスコストの見積りは、この情報を反映させる方法がない。しかし、動的チューニングは、更新が起るデータベースにおいて有効である。1つの実行によって更新効果を統計情報に反映できるからである。

サーチレベルを深めて、動的チューニングを行った結果では一般によりよいアクセスパスを生成できることは明らかではない。

問合せの結果属性には、CODASYLスキーマの構造に対応して一般に1対N、あるいはN対Mの関係性が存在する。この為、アクセスする実現値数だけでなく検索結果の出力量を最少化することが必要である。

LDP-V1.5の有用性では、同一の問題に対して、QUELによって記述しLDPで結果を得る方法と、COBOLでプログラムを組み、実行する方法とを全経過時間について比較

した。この結果、LDPでは、数分～数10分で結果を得れるのに対してCOBOLでは数時間～数10時間を要してしまう。従って、ソフトウェアの全生産性では、LDPが優れていると言える。アクセスパス生成は、一般に人間の方が優れている。人間は、スキーマレベルでの最適化を行うが、動的チューニングを行うことによってLDPは実現値レベルの情報を持つことができる。

謝辞

LDP-V1.5の実現及び設計に協力して頂いているシステク社システム部長鈴木 信氏とデータベースの保守等に助力頂いた当プロジェクトの塚本 元子女史に感謝します。又、日頃御指導して頂いている当協会開発部次長小関 重美氏に感謝します。

参考文献

(TAKIM78)

Takizawa, M., et al., "Resource Integration and Data Sharing on Heterogeneous Resource Sharing System," Proc. ICC'78, Kyoto, Japan, Sept. 1978, pp. 253-258.

(TAKIM79)

Takizawa, M., et al., "The Four-Schema Concept as the Gross-Architecture of Distributed Databases and Heterogeneity Problems," Journal of Information Processing, IPSJ, Vol. 2, Nov. 3, 1979, pp. 134-142.

(TAKIM80)

Takizawa, M., et al., "Query Translation in Distributed Database," IFIP' 80, Tokyo, Oct. 1980

(TAKIM81)

Takizawa, M., et al., "分散型データベースシステム (JDDBS-II) におけるローカルデータベースプロセッサ (LDP-V1.5) と、その評価について," IPSJ DBMS研 25-2, 1981.

(TAKIM82a)

Takizawa, M., et al., "Distributed Query Processing in Local Broadcast Networks," JIPDEC-TR 82/03, 1982.

(TAKIM82b)

Takizawa, M., et al., "The Relational Interface System to the CODASYL Database System in Heterogeneous Distributed Database Systems," JIPDEC TR 82/04, 1982.