データベース・システム 53－1
（1986. 5. 19）

異種DBSから成る分散型演繹DBSについて

滝沢 誠[*] 伊藤 秀昭[**] 盛屋 邦彦[**]
[*]東京電機大学 [**]日本情報処理開発協会

通信網の発展により、種々のデータベースシステム(DBS)を、統合的に利用出来るシステムが求められている。各DBSは、夫々のデータモデルを持ち、この種類の差が、統合化を行ううえでの問題となる。このため、本論文では、ネー階述語論理のネー階理論として、各データモデルをとらえ、Prologに似た論理型言語を共通言語とする方法を示している。論理型言語は、SQL等の従来の関係型言語に対して、複雑な再帰的、非決定的に定義でき、高い記述能力を有している。本論文では、Prologシステムで用いられているSLD導出を改良して、各用DBSの主系である網型DBSに対するDMLプログラムの合成方法について述べる。本システムをワークステーションに実現することにより、異種DBSに対するアクセス・インタフェースとなるものである。

Distributed Deductive Database System for Heterogeneous Database Systems

Makoto Takizawa*, Hideaki Itoh**, and Kunihiko Moriya**

*Faculty of Science and Engineering, Tokyo Denki University
**Japan Information Processing Development Center

Abstract This paper presents the design and implementation of logic language interfaces on conventional CODASYL database systems, which are used in the major parts of database applications. In this paper, we show that even CODASYL model goes with the resolutions, although the relational model is taken in almost all approaches to combining the database systems with the resolutions. First, we define a formal system of logical aspect of the CODASYL model based on the first-order theory. Second, we make clear the semantics of conventional CODASYL data manipulation language(DML) by defining an abstract machine called a VNM. Then, we show a refutation procedure called a navigational (NV) procedure in which meaningless backtrackings are removed. In the NV refutation procedure, the CODASYL database system is accessed in an interpretive manner, because the CODASYL model provides navigational(record-at-a-time) language DML and resolution is principally based on sequential access.

1. はじめに

計算機が種々の分野に普及するとともに、データベースシステム(DBS)が計算機応用の中核となってきている。又、通信網の発展により、利用者は、種々のDBSに、通信網を用いることによりアクセス出来るようになってきている。現在は、利用者は、個々のDBSを、夫々固有の方法でアクセスを行っているが、通信網上の複数の種々のDBSを、統合的にアクセス出来ることが望まれている。即ち、DBSの所在も、各DBSの相異も意識することなく、一つの巨大情報システムとして利用者がデータのアクセスを行えることが望まれている。

複数のDBSを利用するうえでの第一の問題点は、各DBSの異種性である。DBSの異種性のなかで、重要なものはデータモデルの型の差である。現在のメインフレームは、関係型[3]と網型[2),15]の両方のDBMSを提供し、パソコンでも関係型のものが提供されてきている。網型モデルは、物理と論理の両面が混在した網型データ構造と、これに対する手続的(又は、巡航的)操作言語DML[15]がCOBOL等の親言語に埋め込まれていて、定型的な高性能応用に利用されている。一方、関係型モデルは、データ独立なデータ構造と、非手続的な操作言語が提供され、非定型的な性能要求の厳しくない応用で利用されている。現在のDBSの多くは、網型であり、個人用、CAD用等の新しい応用には関係型のDBSが用いられてきている。

こうした二種の既存DBSを統合的に利用するためには、[11]に示すように、まず各DBSに、共通の型のモデル、例えば、関係型モデルを提供させる方法がある。既に、網型DBS上に、関係型モデルを提供するインタフェースは、LDP[11]として実現され、他に　　[　]の試みがある。

こうした試みに対して、従来のデータモデルを、第一階述語論理の第一階理論として考えることができる。　　既存の

DBS上に、Prolog[9]等の論理型言語を設けることの利点は、異種DBS上に共通言語インタフェースを設けることと共に、言語の表現能力として、関係型言語にはない、視野の再帰的、非決定的定義を行えることである。

本論文では、ワークステーション(WS)から、通信網を経由して種々のDBSを、論理型言語によりアクセスさせるシステムLIP(logic interface processor)について述べる。第2章ではLIPの論理構成を示し、第3章では、網型モデルの第一階理論を示す。第4章では、網型DBSに対する反駁の方法を示し、第5章には、システムの実現形態を述べる。

2. LIPの論理構成

LIPは、図2.1に示すように、各利用者毎のワークステーション(WS)と、種々のDBSが通信網で結合された形態をとっている。WSは、推論プロセッサ(IP)と関係型DBSとから成る。各DBS内のデータを事実として、WSのDBSには、利用者の必要とする視野の定義である規則が格納される。IPは、利用者からの問合せを、規則を用いて導出して、各DBS毎のDMLプログラムを合成し、これをDBSに送って実行させる。

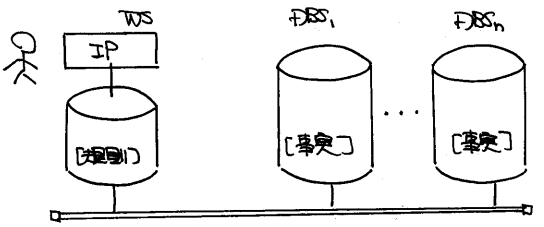

図2.1　LIPの論理構成

3. 網型モデルの論理表現

網型モデルの論理構造を抽象化したモデルの1階理論を与える。

3.1 概念網型体系

論理的な網型モデルに対する1階理論としての概念網型体系(CNS)Nは、四組 $(\mathfrak{S}, L, \Lambda, N)$ で与えられる。$\mathfrak{S}$ は概念網型スキーマ、L, Λ, N はそれぞれ、$\mathfrak{S}$ に対する1階言語、公理集合、推論規則である。概念網型スキーマ $\mathfrak{S}$ は、項目記号集合 $\mathcal{T}$, R型記号集合 $\mathcal{R}$, S型記号集合 $\mathcal{S}$ と、次の生成規則により与えられる。

R型記号 R は、生成規則

$$R = (@R, t_1, \cdots, t_m) \quad (m \geq 0)$$

で与えられる。$@R$ は此鍵、名 t_i はデータ項目である。$@R$, t_i は $\mathcal{T}$ 内の記号である。

S型記号 S は、生成規則

$$S = (R_1, R_2)$$

で与えられる。R_1 と R_2 は R型記号で、夫々の親と子という。但し、R_1 と R_2 は異なっている。

スキーマ $\mathfrak{S}$ に対する L は、1階言語で、次の四種の述語記号がある。

(1) 名項目記号 t に対する T型記号 t.

(2) 名R型記号 R に対する R型記号 R.

(3) 名S型記号 S に対する S型記号 S.

(4) この他の述語記号を V型記号とする。

これらの記号集合に対する項、論理式(wff)は、$\mathcal{S}$ と同じに定義される。

Λ は公理集合で、網型モデルの制約を示す次の特殊公理を含んでいる。

(1) 名R型記号 $R = (@R, t_1, \cdots, t_m)$ に対して、

$$\forall x_1 \cdots \forall x_m \forall k (R(k, x_1, \cdots, x_m) \rightarrow @R(k) \wedge t_1(x_1) \wedge \cdots \wedge t_m(x_m)).$$

$$\forall k \forall x_1 \cdots \forall x_m \forall y_1 \cdots \forall y_m (R(k, x_1, \cdots, x_m) \wedge R(k, y_1, \cdots, y_m) \rightarrow x_1 = y_1 \wedge \cdots \wedge x_m = y_m)$$

(2) 名S型記号 $S = (R_1, R_2)$ に対して、

$$\forall k \forall h (S(k, h) \rightarrow @R_1(k) \wedge @R_2(h))$$

$$\forall k \forall h \exists x_1 \cdots \exists x_m \exists y_1 \cdots \exists y_m (S(k, h) \rightarrow R_1(k, x_1, \cdots, x_m) \wedge R_2(h, y_1, \cdots, y_m))$$

$$\forall k \forall h \forall j (S(k, h) \wedge S(j, h) \rightarrow k = j)$$

(3)は、より一般なデータ構造に対する公理を与えている。この他に、閉定義域、単一名、名述語記号の定義の公理及び等式公理がある。

推論規則としては、分離規則と一般化規則がある。

さて、スキーマ $\mathfrak{S}$ の1階言語 L の解釈 I は (D, Φ) で、D は定義域、Φ は、L 内の定数、関数、述語記号から D への写像である。名T記号 t について、$\Phi(t)$ は t の定義域を、名R記号 $R = (@R, t_1, \cdots, t_m)$ に対して、$\Phi(R) \subseteq \Phi(@R) \times \Phi(t_1) \times \cdots \times \Phi(t_m)$ を、名S記号 $S = (R_1, R_2)$ に対しては、$\Phi(S) \subseteq \Phi(R_1) \times \Phi(R_2)$ を与える。$\Phi(R)$ を、レコード実現値(RO)集合、$\Phi(S)$ をセット実現値(SO)集合とする。公理集合 Λ を満足する解釈 I を、CNS N のモデルという。Φ を N のモデルとしたとき、Φ で定まる RO集合と SO集合の族を、概念網型スキーマ $\mathfrak{S}$ のデータベース $\mathfrak{S}$ とする。

以上みてきたように、CNS は1階理論であるので、完全で健全である。

3.2 ホーン節

CNS の wff は、ホーン節

$$A \leftarrow B_1, \cdots, B_n$$

の形式で表されるとする。ここで、A, B_i は基本式で、A を頭、$B_1, \cdots, B_n$ を本体という。頭のある節を限定節、無いものを目標節という。$n > 0$ の限定節を規則節、$n = 0$ の限定節を単位節という。$n = 0$ の目標節を空節□という。

概念網型データベース $\mathfrak{S}$ 内の名実現値に対して、具象単位節が対応する。従って、$\mathfrak{S}$ は、RO集合と SO集合を示す R型基本式と S型基本式の具象例の集合である。

例として、図3.1に示す網型スキーマを考えてみよう。箱は R型記号、矢は S型記号を示している。矢は親から子への関係を方向で示している。図3.2には、このスキーマに対する限定節集

合下を示す。5)～31)は具象単位節で、この集合は、概念網理データベースを示している。1)は、「Nは部Mのメンバである」を、2)は、「Nはプロジェクトのメンバである」を示す。3)と4)は、「Nは、プロジェクト下の関係者である」を示し、再帰的、非決定的な視野を定義している。

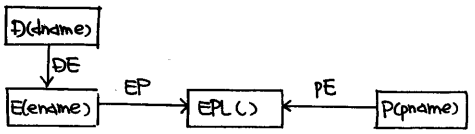

図3.1 概念網理スキーマ

```
F={1) ED(N,M)<-E(X,N),DE(Y,X),D(Y,M).
   2) EE(N,M)<-E(X,N),EP(X,Y),PEL(Y),
              PE(Z,Y),P(Z,M).
   3) PP(N,T)<-EE(N,T).
   4) PP(N,T)<-EE(N,U),EE(M,U),PP(M,T).
   5) E(1,a)<-.     6) E(2,b)<-.
   7) E(3,c)<-.     8) E(4,d)<-.
   9) D(1,c)<-.    10) D(2,a)<-.
  11) DE(1,1)<-.   12) DE(1,3)<-.
  13) DE(2,2)<-.   14) DE(2,4)<-.
  15) P (1,d)<-.   16) P (2,n)<-.
  17) PE(1,1)<-.   18) PE(1,2)<-.
  19) PE(1,3)<-.   20) PE(2,4)<-.
  21) PE(2,5)<-.   22) EP(1,1)<-.
  23) EP(1,4)<-.   24) EP(2,5)<-.
  25) EP(3,2)<-.   26) EP(4,3)<-.
  27) EPL(1)<-.    28) EPL(2)<-.
  29) EPL(3)<-.    30) EPL(4)<-.
  31) EPL(5)<-.    }
```

図3.2 限定節集合

利用者の概念網理データベースに対する問合せは、目標節により表される。例えば、「dプロジェクト員は誰々は ←EE(?N,d)」となる。ここで、変数は大文字、定数は小文字で示され、?の付いた変数は、解を求めている目標変数である。

4. 反駁法

次に、目標節と限定節集合から、解を求めるDBプログラムの合成方法について考える。

4.1 SLD反駁

Pを限定節集合、Gを目標節 $\leftarrow B_1, \cdots, B_m$ とする。代入θは、$\{x_1/t_1, \cdots, x_p/t_p\}$ の形式で、x_iは変数、t_iは項である($i=1, \cdots, p$)。Eを単一階言語の式とすると、EθはE内の変数$x_1, \cdots, x_p$を全て、夫々項$t_1, \cdots, t_p$で同時に置き換えて得られる式である。ここで、節 $C = B \leftarrow A_1, \cdots, A_k$ をP内の節とし、$B_j\theta = B\theta$ なる代入θが存在するとする。又、B_jはお計算規則R[13]により Gより選択された基本式である。このとき、

$\leftarrow (B_1, \cdots, B_{j-1}, A_1, \cdots, A_k, B_{j+1}, \cdots, B_m)\theta$

を、GとCのSLD導出形、という。Cを入力節という。P∪{G}のSLD演繹とは、目標節、入力節、代入夫々の系列 $G_0(=G), G_1, \cdots, G_n$, $C_1, \cdots, C_n$, $\theta_1, \cdots, \theta_n$ で、G_{i+1}は、G_iとC_{i+1}のSLD導出形である。空節を導くSLD演繹をSLD反駁という。PをP∪{G}のSLD反駁としたとき、代入の合成 $\theta_1 \cdots \theta_n$ を、G内の目標変数の束縛に制約した代入θを、計算結代入という。θは、問合せの一つの解を与える。

計算規則Rのもとで、P∪{G}の可能なSLD演繹は、図4.1に示すSLD木により示せる。これは、図3.2に対する目標節 ← PP(?N,d) のSLD木である。節点は目標節を、枝は入力節を示し、根からの各路は演繹を示している。空節を導く路を成功路といい、SLD反駁を示している。一方、選択基本式と単一化可能な頭を持つ入力節がない時、Xで示し、根からXまでの路を失敗路という。又、無限の長さの路を無限路といい、無限に導出が繰り返されることを示している。

問題は、SLD木から、いかに有効に成功路を見つけるかである。この手続きをSLD反駁手続きという。現在のPrologシステムの多くは、最左の基本式を選ぶ計算規則と、入力節をあらかじめ順序付けておいて、SLD木をこの順に順序化しておいて、木を縦型に探索する手続きを用いている。

以上のSLD導出を推論規則とした形式的体系は、健全で完全であることが証明されている[1),13)]。

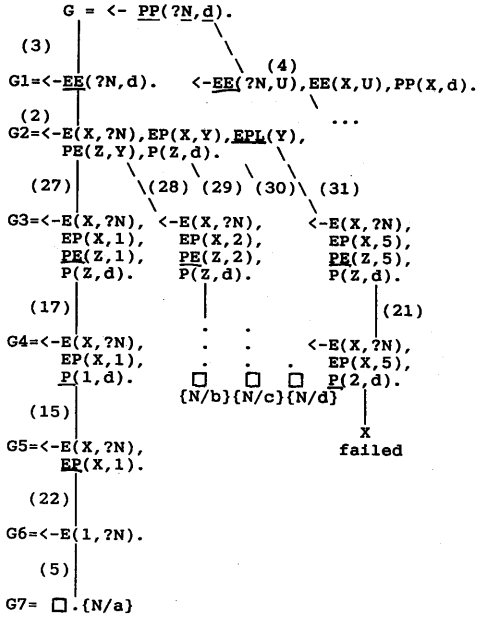

図4.1. SLD木

4.2 巡帰的反駁手続き

以上のSLD導出を基本として、拡理DBSのホルプログラムを合成する方法について考える。このために、次の基本的な方針をたてている。

(1) SLD導出では、入力節を一つずつ限定節集合Pから取り出すが、この操作は、とが具象基本式の場合は、データベースから一つの実現値を検索することである。従って、SLD導出と、ホルによるレコード単位の操作を自然に対応づけれる。

(2) 反駁に失敗したとき、又は他の計算答代入を求めるとき、SLD木の後戻りを行う。現在の節点から親へ戻り、他の入力節を見つけ導出を行う。入力節が具象基本式(即ち、単位節)の場合には、データベース内の実現値を順に検索すること。例えば、親に対する子レコード実現値を順に検索することに対応できる。一方、規則節の場合には、他の規則節を見つけ、これについて導出を行う。このため、これ以降は、全く異なった目標節と入力節の系列となり得る。従って、同じ規則節を入力節としている路の計算答代入は、データベースに対する検索操作により一括して求めれば、Pから規則節を後戻りの毎に探索する必要がなくなる。

(3) 後戻りするとき、現在の選択基本式と、これの親の選択基本式が共有した変数を有していないとき、親を飛び越して後戻りさせる。変数の共有関係がないので、親の選択基本式内の変数に、新しい代入が与えられても、現在の選択基本式内の変数に何の影響もないからである。

以上の方針に従って、SLD導出を、拡理DBSの操作に適合させることを目指した巡帰的反駁手続きを示す。まず(2)で述べた、類似した反駁を定義する。

[定義] TとT'を$P\cup\{G\}$のSLD反駁とする。TとT'は、次の条件を満足するとき、互いに類似(similar)するといい$(T\simeq T')$。

(i) TとT'は同一の長さである。

(ii) TとT'のi番目の導出で、図4.2で、

(ii-1) $i>0$ ならば、$\theta_i=\theta_{i-1}\sigma$, $\theta_i'=\theta_{i-1}'\sigma'$
$i=0$ ならば、$\theta_0=\theta_0'=e$.

(ii-2) $k=k'$, $j=j'$ (即ち、$B_j=B_j'$)
$m=m'$. ここで、$m=m'=0$のとき、BとB'は同じ述語記号の基本式で、$m=m'>0$のとき、C_iとC_i'は同じ。□

類似関係$\simeq$は、明らかに同値関係であり、$P\cup\{G\}$の路の集合を同値類に分割する。例えば、図4.1のSLD木は、同値類$S_1=\{$(3)-(2)-(27)-(17)-(5)-(22)-(5), (3)-(2)-…,…, (2)-(2)-(2)-(21)$\}$, $S_2=\{$(4)-… $\}$, … と分割できる。

ある反駁Tがみつかると、Tより、Tと類似している全反駁の計算答代入を求めることができる。例えば、図4.1の最左の反駁がみつかったとすると、これより図4.3に示すホルプログラムにより全計算答代入を求めれる。

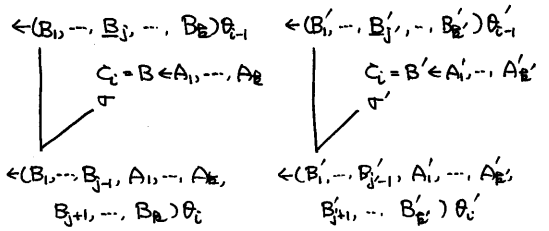

図4.2 類似関係

```
      first (EPL).   go to M1.
L1:   next  (EPL).
M1:   if ( g = F)    go to END.
      owner (PE).    go to M2.
L2:                  go to L1.
M2:   if ( g = F)    go to END.
      get(P).
      if ( P.pname~=d) go to L2.
                     go to M3.
L3:                  go to L2.
M3:   owner  (EP).
L4:                  go to L3.
M4.   if ( g = F)    go to L3.
      get(E).
      output(E.ename). go to L4.
```

図4.3 DMLプログラム例

次に、(3)の後戻りの効率化について考えてみます。基本式Aに対して、V(A)をA内の変数集合を示すものとする。このとき、代入θについて、

$\Sigma_\theta = \{ \langle x, y \rangle \mid x \in V(G),\ y \in V(y),\ x\theta = y\theta \}$

とする。又Σを用いて、目標節Gに対する目標グラフを定義する。

[定義] 目標節$G = \leftarrow B_1, \cdots, B_m$と代入θに対して、目標グラフ$GL(\{B_1, \cdots, B_m\})$は、次の手続きで得られるグラフである。

(1) 各B_iに対して、点B_iをつくる$(i = 1, \cdots, m)$。

(2) 各点の対$\langle B_i, B_j \rangle$について、$B_i\theta$と$B_j\theta$がユニファイできるならば、B_iとB_j間に辺$\langle B_i:x, B_j:y \rangle$を設ける。□

[定義] Tを、計算規則Rによる$P \cup \{G\}$のSLD反駁とする。Mを$\{\langle A, \theta \rangle \mid A$はP$\cup\{G\}$内のR-ヌはS-基本式で、T内で、Aの例が選択基本式$(A\theta)$が選ばれている$\}$とする。Tの目標グラフは、M内の要素を点とし、M内の二点$\langle A_i, \theta_i \rangle$と$\langle A_j, \theta_j \rangle$で、$A_i$と$A_j$が$\theta$で$A_i\theta = A_j\theta$なるとき、この二点間に辺を設けたグラフである。□

次に、目標グラフの特殊例として、巡航木を定義する。

[定義] 目標グラフGの巡航木Tは、次の条件を満足する木である。

(1) T内の点とG内の点には全単射の関係がある。

(2) G内で点xとy間に辺があれば、xとyは、T内で同一の根葉路内にある。

(3) T内の各点の子は、左から右に順序付けされている。□

ここで、目標点$\langle B, \theta \rangle$とは、ある目標変数Yに対して、$X\theta = Y\theta$なる変数XをもつT内で最初の点である。図4.4に、目標節$\leftarrow P(z,d), PE(y,z), EPL(y), EP(x,y), E(x,?N)$の巡航木を示す。

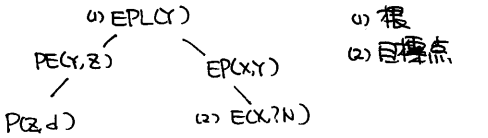

図4.4 巡航木

さて、目標節Gと限定節集合Pよりsld反駁を見つける反駁手続NVRを示す。

[NVR] 計算規則Rによる$P \cup \{G\}$の巡航反駁手続きは、次のようである。

(1) $T_0 = \Phi$, $\theta_0 = \varepsilon$, $GL_0 = GL(G)$

(2) $G_{i-1} = \leftarrow B_1, \cdots, B_m$とする。Rは、$\langle T_{i-1}, G_{i-1}, \theta_{i-1} \rangle$より、選択基本式$B_j$及び$T_{i-1}$内の点Xを与える。入力節$C_i = B \leftarrow A_1, \cdots, A_n$とし、$B\theta = B\theta_{i-1}\theta$とし、$G_i$を$G_{i-1}$と$C_i$の$\theta$についてのSLD導出形とする。$\theta_i = \theta_{i-1}\theta$。

(2-1) $n > 0$のとき、GL_iは、GL_{i-1}から点B_jを除き、ノード$A_1, \cdots, A_n$と共有変数を示す辺を加える。$T_i = T_{i-1}$。

(2-2) $n = 0$のとき、T_iは、T_{i-1}の点Xの最右の子としてB_jを加えて得られる。GL_iはGL_{i-1}より、点B_jを除いて得られる。

(3) (1)(2)を、グラフGL_iが空になるまで逐次適用する。空になったときの代入θ_iを、巡航代入とする。□

NVR内の計算規則は、次のようである。

[NV規則R] Rは、巡航木T、目標グラフGL、代入θに対して、GL内の選択基本式Bと、T内の点Xを次の手続きで与える。

(1) Xを、Tの最右最下の点とする。

(2) GL = φ なるは、(6)へ。
(3) Tから、∀(B)∈∇(X)≠φ なる点〈B,θ〉のなかで、ecost(B,θ)が最小のものを選ぶ。
(4) 見つかれば、〈B,X〉を出力する。
(5) 見つからなければ、Xが根でなければ、Xの親をXとして、(2)へ。
(6) GLから、fcost(B,θ)が最小の点〈B,θ〉を選び、〈B,X〉を出力。□

以上のNV反駁手続きにより得られた巡航木Tより、OMLプログラムを合成する。

4.3 OMLプログラムの生成

巡航木Tに対するOMLプログラムVは、有向辺で結合されたセルの集合である。T内の各点Xに対して、セルC(X)が存在し、C(X)は、二つの入力ポートF(X)とN(X)、四つの出力ポートS(X), B(X), PT(X), NT(X)がある。C(X)は、順序付けられた実現値集合と、この順序に従って操作を行えるオブジェクトである。セル間のリンクは、入出力関係を表し、四種のリンク、S, B, PT, NTがある。点Yが、T内でXの次の経理順序にあれば、S(X)からF(Y)にSリンクが設けられる。YがXの親なるは、B(X)からN(Y)にBリンクが設けられる。YがXの左/右子の最近の目標点のとき、出力ポートPT(X)/NT(X)からF(Y)にPT/NTリンクが設けられる。

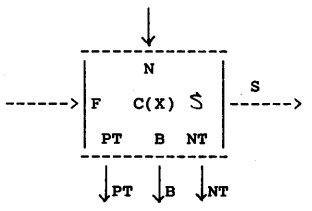

図4.5 セル構造

C(X)は、入力ポートからトークンを受信する毎に起動される。トークンは、Sリンクにより代入を運ぶ。F(X)でトークンθを受ければ、C(X)内の順序付けられたデータの先頭の実現値が得られ、θとの照合を行い、照合するまで(即ち、条件が合うまで)順々に実現値が検索される。条件が合えば、実現値の現在位置を記憶しておき、代入θ'(=θσ、σは現在の実現値の代入)を合成し、S(X)よりθ'を送信する。条件を満足する実現値のないときは、B(X)よりトークンを出力し、後戻りする。N(X)よりトークンを受信したときは、現在位置より実現値を順々に検索し、以上を行う。後戻りである。

巡航木Tより、類似した反駁の全ての計算用代入を求めるには、最右のセルC(X)でS(X)より出力されるトークンθが計算用代入なので解として出力すると同時に、Bリンクの点に後戻りする。このときの後戻りは、目標変数に対する新しい用代入を求めることが目的なので、木内の親に後戻りするだけでは新しい用代入が得られるとは限らない。このために、Xの左の最近の目標点Y(XとPTリンクで結合されている)に戻る。Yから上記の手続きを繰り返すが、Yから右のセル全てを繰り返す必要はない。Yを親とする部分木の最右のセルまでを繰り返す。そでトークンがSリンクに出力されるとき、次に起動されるのは、目標点を含む部分木の根Tで、Xの右側になるセルである。Xを工として上記を繰り返す。

以上より、ポートより送信されるトークンは〈T,θ,L〉の形式で、θは代入、Lは、上記のセルを示し、Tは下から(失敗か成功)を示す。S(X)より〈T,θ,L〉を受信したとき、T=Sなるは、θを求めX=Lなるは、〈F,θ,φ〉をS(X)より出力する。〈F,θ,φ〉を受信したなるは、Xが目標点を含む部分木の根なるは、θを求め、〈T,θ,Z〉(ZはXの部分木の最右のセル)をS(X)より出力する。そうでなければ〈F,θ,φ〉をS(X)より出力する。N(X)より、〈F,_,L〉を受けた時、θを求め、〈T,θ,Z〉を出力、〈T,_,L〉を受けたときは、〈T,θ,L〉を出力する。以上の操作により、新しい用代入だけに関するセルだけを起動出来、不要なデータベースのアクセスを減少出来る。

図4.6に、図4.4に対するOMLプログラ

```
     B
 ____________________________________
 ↓   S        ↑      ↑       ↑        ↑
EPL --> PE --> P --> EP --> E

     first (EPL).          go to M2.
L1:  next  (EPL).
M1:  if(g=F)               go to END.
     owner (PE).
     if(G=F)               go to L1.
     get   (P).
     if(P.pname~=d)        go to L1.
     owner (EP).
     if(G=F)               go to L1.
     get   (E).
     output(E.ename).      go to L1.
```

図4.6 DMLプログラム

を示す。

4.4 コスト関数

NV規則で用いられているコスト関数ecost(B,θ)とfcost(B,θ)について述べる。Bは基本式で、θは代入である。ここで、XをBが指示するRO集合とし、Xの項目tに対して、πをtの選択度とする。

(1) fcost(B,θ)=

(i) π·|X| if tθ=c で、tはCALC項目(ランダムアクセス)

(ii) |X| otherwise (逐次アクセス)

(iii) M·Σcost(G_i) if BはV基本式で、V←G_1,…,G_n と単一化可能(M≥1)

Mにより、V基本式が選択されることを防ぐ。YをBの示すSO集合とする。

(2) ecost(B,θ)=

(i) Yの平均の子実現値数 if B(t_1,t_2)θで、t_1θ=c

(ii) Yの親をもつ確率 if t_2θ=c

(iii) fcost(B,θ) if BはR基本式、V基本式、

5. まとめ

現在、DBSサーバとして、M-360RのAIM/DBを、ワークステーション(WS)としてSun2/120を用い、WSのDBSとしては、我々の開発した関係型DBMS Granadaを用いている。WSにより、各DBS毎のDMLプログラムを合成し、DBSに実行させる。本システムにより、利用者は容易に種々の複数のDBSを操作できる。

各DBSが、論理言語インタフェースを有している場合は、次のようにする。各述語記号に対して、所在するDBSが定まっているので、目標グラフをDBS毎に分割し、分割したグラフの目標節をそのDBSに送り、答代入を一つ一つ得る。AIM/DBについては、図5.1に示すような論理型言語インタフェースLIPCを、既に開発しており、現在、WSのLIPを開発中である。WS LIPでは、網型DBSに対するのと同様に Granadaの提供する組単位の操作言語TMLにより推論プロセッサとのインタフェースをとっている。

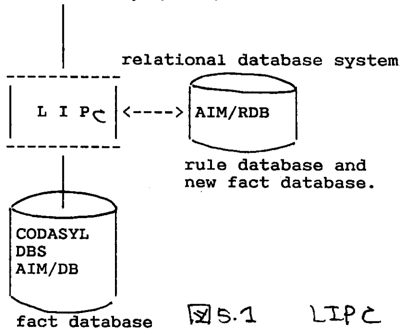

図5.1 LIPC

References

1) [APT82] Apt, R., and van Emden, M.H., "Contributions to the Theory of Logic Programming," JACM, Vol.29, No.3, pp.841-862.

2) [CODA73] CODASYL DDL Committee, "CODASYL Data Description Language," Journal of Development, 1973.

3) [CODD70] Codd,E. F., "A Relational Model of Data for Large Shared Data Bank," CACM, Vol.13, No.6, 1970, pp.337-387.

4) [DAYA82] Dayal, U., et al., "Query Optimization for CODASYL Database Systems," Proc. of the ACM SIGMOD, 1982, pp.138-150

5) [ENDE72] Enderton, H, "Mathematical Introduction to Logic," Academic Press, 1972.

6) [HENSL84] Henschen, L.J. and Naqvi, S.A.,"On Compiling Queries in Recursive First-Order Databases," JACM, Vol.31, 1984, pp.47-85.

7) [JACO82] Jacob, B.E., "On Database Logic," JACM, Vol.29, No.2, 1982, pp.310-332.

8) [KOBA85] Kobayashi, I., "Classification and Transformations of Binary Relationship Schemata," Sunno Institute of Business Administration, 1985.

9) [KOWA79] Kowalski, R., "Logic for Problem Solving," North-Holland, 1979.

10) [TAKI86] Takizawa, M., Ito, H., Moriya, K., "Logic Interface System on CODASYL DBS," 京大数理解析研. Feb. 1986

11) [TAKI80] Takizawa, M. et al.,"Query Translation in Distributed Databases," IFIP'80, Tokyo-Melbourn, 1980, pp.451-456.

12) [ULLMJ85] Ullman, J.D., "Implementation of Logical Query Languages for Databases," TODS, Vol.10, No.3, 1985, pp.289-321.

13) [LLOY85] Lloyd, D., "Foundation of Logic Programming," Springer-Verlag, 1984.

14) [OHSU83] Ohsuga, S., "Developing a Deductive Relational Database for Uniform Handling of Complex Queries," JIP of IPSJ, 1983, pp.123-137.

15) [OLLE78] Olle, T., "The CODASYL Approach to Data Base Management," John Wiley & Sons, 1978.