CODASYL データベースシステムに対する非手続的更新インタフェース設計の基本概念†

滝沢 誠†† 野口正一†††

既存の異種データベースシステム (DBS) から成る分散型データベースシステム (DDBS) を実現するためには，まず第一に，DDBS 全体の共通モデルに基づいて，各 DBS 上に視野（ローカル概念スキーマ）を設定する必要がある．その第一段階として，CODASYL DBS 上に共通言語の基本更新演算が行える関係型インタフェースを設計する問題を考える．そのために，次の点を論ずる．(1)実体と関連を表す2種 (E と R) の関係を定義し，それに基づいて共通モデルとしての概念モデルを定義する．(2)従来の CODASYL モデルを抽象化した論理データ構造と論理演算の記述（論理 CODASYL モデル）を与える．(3)概念モデルのR関係を新たに型の概念を用いて特性化し，論理 CODASYL モデルに対応するローカル概念 (LC) モデルを与える．そして，LC モデルのデータ構造と共通言語 RQL の基本演算とが，論理 CODASYL モデルと等価であることを示す．(4)さらに，RQL を正しく従来の COBOL DML で表す方法を与え，RQL と COBOL DML の記述等価性を示す．以上は，CODASYL DBS 上に非手続的な共通インタフェースを設計するための論理的基礎を与える．

1. まえがき

分散型データベースシステム (DDBS) 実現のための重要な問題の一つは，CODASYL[1),5)]，IMS[3)] 等の異種の既存データベースシステム (DBS) の統合問題である．本論文では，DBS の異種性をデータモデルの相違とする．統合問題を解決する基本的方法は，まず，DDBS 全体の共通モデル（概念モデルと呼ぶ）を定め，これに基づいて各 DBS 上に視野を設定することである．この視野をローカル概念 (LC) スキーマとする．次に，DDBS 全体で一意な視野としての全体概念 (GC) スキーマをこれらの LC スキーマ上に定義し，各 DBS の所在と独立にする．以上は図1の四階層構造[6),7)]として示せる．本論文では，概念モデルとして，簡単なデータ構造と非手続的操作言語を有する関係型[2)]のモデルを採用し，既存 DBS として普及している CODASYL モデルと関係型モデル間での異種性の解決を試みる．従来の CODASYL モデルとして，文献 15) 等の新仕様もあるが，論理的に本質的な問題の検討は，文献 1) と 5) で十分であると考える．四階層構造に基づく検索用関係インタフェースとして，ローカルデータベースプロセッサ (LDP-V 1.5)[8),10),14)] が設計され，Jipdec の AIM (M-170 F), ADBS (Acos-700) 上で実働している．更新用視野の設計については，セット型のメンバシップクラスをモデル化するために，文献 13) は外来キー制約[11)]を用い[13)]，12) は親レコード型の主キーを子レコード型の主キーに伝播させる方法[12)]を用いている．また，文献 16) では，関係 DBS 上に CODASYL スキーマを設ける問題を論じている．これらに対して，本論文では，実体間の関連を，更新演算の意味を与えて特性化し，2種のメンバシップクラスに対応した型を与える．

以上の考察に基づき，本論文ではまず，既存データモデルを概念化した概念モデルを定義し（2章），従来の CODASYL モデルを抽象化した論理 CODASYL モデルの記述を与える（3章）．4章では，概念モデルの実体間の関連を特性化したローカル概念 (LC) モデルの定義と，LC データ構造と述語論理形式の LC 操作言語 RQL の基本演算を与え，これらが論理 CODASYL モデルと等価であることを示す．5章では，RQL の COBOL DML による正しい表現方法を与える．

2. 概念モデル

2.1 データモデルの概念化

DBS は，データモデルに基づいたデータ構造と，この上の操作演算を利用者に提供する処理システムと考えられる．データモデルとしては，関係[2)]，階層[3)]，

† Fundamental Concepts on the Design of the Non-procedural Update Interface over the CODASYL Database Systems by MAKOTO TAKIZAWA (Japan Information Processing Development Center) and SHOICHI NOGUCHI (Research Institute of Electrical Communication, Tohoku University).

†† (財)日本情報処理開発協会
††† 東北大学電気通信研究所

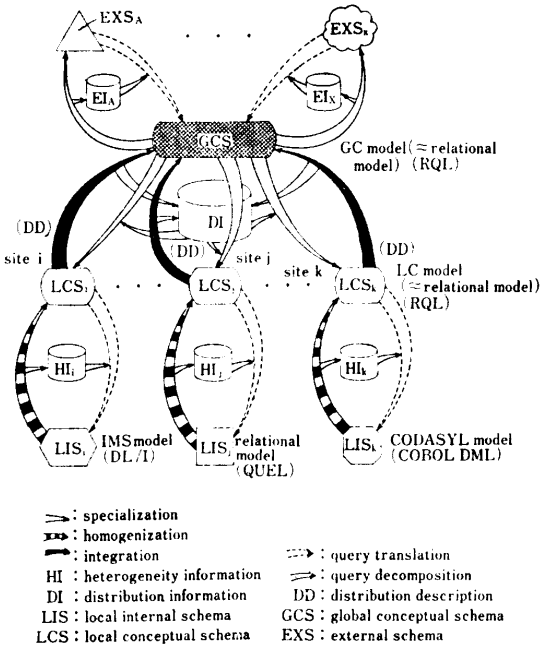


図 1 四層スキーマ構成[6),7)]
Fig. 1 Four-schema structure.

網[1),5)]の三つが著名である．本論文では，DBS の異種性を各データモデルの相違とし，データモデルはデータ構造とこれらの操作演算で定まるとする．異種 DBS の統合を行うためには，まず DDBS 全体の共通モデル（概念モデルとする）に基づいて，各 DBS 上に視野を設ける必要がある．

階層，網モデルは，実体と関連との2種の概念を有している．また，更新演算の意味を与えて，関連は特性化され，型づけされている．たとえば，網型の CODASYL モデル[1),5)]（3章）では，レコード型は実体集合で与えられ，セット型は二つの実体集合間の1対 $m(\geq 0)$ の関係で表される．さらに，セット型のメンバシップクラスの型は，更新演算の意味する制約条件で特性化される．階層型の IMS モデル[3)]では，セグメント型が実体集合，階層順序が実体集合間の1対 m 関係を表している．以上の考察より，概念モデルは，実体と関連との2種の独立な概念をもたねばならず，また各モデルのデータ構造は，概念モデルにおける関連と型の概念を用いて特性化できることがわかる．本章では，概念モデルのデータ構造と操作言語の定義を与える．

2.2 概念データ構造

概念データ構造は，時間不変な論理構造を与える概念スキーマ $\underline{S}$ と，実際にデータが与えられた(時間可変な）概念データベース S とから成る．$\underline{S}$ は，E と R の2種の関係スキームから成る．E(関係)スキーム $\underline{\boldsymbol{E}}$ は，属性集合 $\Omega_E=\{@E_1, a_1, \cdots, a_m\}$ $(m\geq 0)$ とインテグリティ条件 Γ_E から成る $(\underline{\boldsymbol{E}}=(\Omega_E, \Gamma_E))$．各属性 $\alpha\in\Omega_E$ の可能な値の集合（定義域）を $\mathrm{dom}(\alpha)$ とする．$\mathrm{dom}(@E)$ は $\underline{\boldsymbol{E}}$ に固有な識別子の集合で，$@E$ を主属性とする．Ω_E に実際に値を与えた組のなかで，Γ_E を満足する組を E 組とする．この E 組の集合を E 関係 $E=\{e \mid e\in\mathrm{dom}(\Omega_E)\wedge\Gamma_E(e)\}$ (ここで $\mathrm{dom}(\Omega_E)=\underset{\forall\alpha\in\Omega_E}{\times}\mathrm{dom}(\alpha)$) とする．$E$ 組 $e\in E$ の属性の部分集合 $A\subseteq\Omega_E$ の値を $A(e)$ とする．E 内で $@E$ の値は一意に定めてあるので，$@E$ は主キーとなる．E 関係スキーム Γ_E を省略して，$\underline{\boldsymbol{E}}(@E, a_1, \cdots, a_m)$ とも表す．

任意の $n(\geq 2)$ 個の E 関係スキーム $\underline{\boldsymbol{E}}_i(i=1, \cdots, n)$ と属性集合 $\Omega_{R'}(\Omega_{R'}\cap\Omega_{Ei}=\phi)$ があり，関係 $R\subseteq\overset{n}{\underset{i=1}{\times}}\mathrm{dom}(\Omega_{Ei})\times\mathrm{dom}(\Omega_{R'})$ が存在するとき，R（関係）スキーム $\underline{R}=(\underline{\boldsymbol{E}}_1, \cdots, \underline{\boldsymbol{E}}_n, \Omega_{R'}, \Gamma_R)$ が定義される．$\Omega_{R'}=\{b_1, \cdots, b_q\}$ $(q\geq 0)$ と Γ_R は，おのおの $\Omega_{E_1}, \cdots, \Omega_{E_n}$ より意味的に帰納された属性集合とインテグリティ条件である．Γ_R は $\underline{R}$ に与えうる値の組（R 組）を定める．この R 組の集合を R 関係 $R\triangleq\left\{r \mid r\in\overset{n}{\underset{i=1}{\times}}\mathrm{dom}(\Omega_{Ei})\times\mathrm{dom}(\Omega_{R'})\wedge\Gamma_R(r)\right\}$ とする．Ω_{Ei} で $@E_i$ は主キーなので，$\underline{R}(@E_1, \cdots, @E_n, \Omega_{R'})$，$R=\left\{r' \mid r'=@E_1\cdots@E_n\Omega_{R'}(r)\wedge r\in\overset{n}{\underset{i=1}{\times}}\mathrm{dom}(\Omega_{Ei})\times\mathrm{dom}(\Omega_{R'})\wedge\Gamma_R(r)\right\}$ とする．各 R 組 $r\in R$ に対して次の条件が成り立つ．すなわち，すべての $\underline{\boldsymbol{E}}_i$ について $@E_i(e_i)=@E_i(r)$ なる組 e_i が E_i 内に存在する．

以上の E と R 関係の集合を，概念データベース S と定義する．たんに関係（スキーム）というときは，$S(\underline{S})$ 内の E または R 関係（スキーム）を表すものとする．

2.3 概念操作言語 (CML)

概念モデルの操作言語 CML によって，S に対する検索と更新演算を，QUEL[4)] と同様な形式で表せる．

2.3.1 検索演算

検索演算は，(i)検索条件の設定と，(ii)目標関係スキームの設定との二つの手順により定義される．(i)の検索条件は，組関係論理言語[3)]により(1)のよ

うに与えられる.

$$\{(t_1,\cdots,t_p)\mid(\exists u_1)\cdots(\exists u_q)\Big(\bigwedge_{i=1}^{p} T_i(t_i)\ \bigwedge_{j=1}^{q} U_j(u_j)\wedge QL(t_1,\cdots,t_p,u_1,\cdots,u_q)\Big)\} \quad (1)$$

$\boldsymbol{X}=\{T_1,\cdots,T_p,U_1,\cdots,U_q\}$; $\boldsymbol{x}=\{t_1,\cdots,t_p,u_1,\cdots,u_q\}$ とする. $x\in\boldsymbol{x}$ は組変数で, $X\in\boldsymbol{X}$ とすると $X(x)$ は x が関係 X 内の組を値としてとることを表している. $t_i\in\boldsymbol{x}$ は関係 $T_i\in\boldsymbol{X}$ の組変数で(1)の自由変数, また $u_j\in\boldsymbol{x}$ は関係 $U_j\in\boldsymbol{X}$ の組変数で(1)の束縛変数となる. QL は述語 $\alpha(x)\ \theta\ \beta(y)$, $\alpha(x)\ \theta v$ と結合子 $(\vee,\wedge,\sim)$ から成る式である. ここで, θ は比較演算子, v は定数, $x,y\in\boldsymbol{x}$ はおのおの関係 X, $Y\in\boldsymbol{X}$ の組変数, α と β はおのおの X と Y の属性である. $\alpha(x)$ は, 関係 X の組の属性値を表す. (ii)では利用者の必要とする目標関係スキームを $\underline{T}(\Omega_T=\{a_1,\cdots,a_k\})$ とすると, Ω_T は $\tilde{f}(\Omega_{T_1},\cdots,\Omega_{T_p})=\Omega_T$ なる関係 $\tilde{f}$ を定義することで与えられる. $\tilde{f}$ から誘導される関数 f を直積 $T_1\times\cdots\times T_p$ に作用させると目標関係 T が与えられる. この f を目標関数とする.

以上をもとに, 検索演算は CML で次のように表される.

var $(x_1, X_1)\cdots(x_l, X_l)$; (2)

get into $T(\Lambda)$ where Ψ; (3)

(2)は(1)の $X_i(x_i)$ を表し, 関係 X_i に組変数 x_i を定める. (3)の Λ と Ψ はおのおの, 目標リストと条件式である. Ψ は(1)の QL を表す. (1)の $\alpha(x)$ は(3)では $x.\alpha$ と表される. Λ は目標関数 f を表し, $\Lambda=(\beta_1,\cdots,\beta_k)$, $\beta_j=a_j=\varepsilon_j$ $(j=1,\cdots,k)$ である. ε_j は, $t.\alpha$(t は自由変数), v, またはこれらの算術式で, Ψ を満たす $\{t_1,\cdots,t_p\}$ 値から目標属性 a_j 値を与える関数である. たとえば, E 関係スキーム EMP (@E, name, age, sal) に対する「30歳以上の従業員名とその給料の1.2倍を求めよ」は CQL で次のように書かれる.

var $(e,$ EMP);

get into T(e.name, sv$=e$.sal$\times$1.2) where

e.age$\geq$30; (4)

概念モデルの検索演算では, 主属性が識別子をとることから演算子として $=$ と $\neq$ のみが許され, また主属性上の算術演算は許されない.

2.3.2 更新演算

更新演算は, CML では, 目標属性集合 Ω を定める目標リスト Λ と条件式 Ψ, および Ω に対する次の六つの基本更新演算 ρ として表現される. ここで, $\underline{E}$ と $\underline{R}$ (@$E_1,\cdots,$@$E_n, b_1,\cdots,b_p$) をおのおの $\underline{S}$ 内の任意の E と R スキームとする.

(1) del(e, E) は組 $e\in E$ と同時に, @E を主属性としてもつ全 R 関係 R から @$E(e)$=@$E(r)$ なる組 r を除く. (2) del(r, R) は組 $r\in R$ を除く. (3) app(e, E) は組 e を E に加える. (4) app($r, R; e_1, E_1; \cdots; e_n, E_n$) は各 E_i が @$E_i(e_i)$=@$E_i(r)$ なる組 e_i $(i=1,\cdots,n)$ をもつならば, R に組 r を加える. (5) rep(e, e', E) は組 $e\in E$ の非主属性値を e' に変える. (6) rep(r, r', R) は組 $r\in R$ を r' に変える. Ψ と Λ は以上の e, e', r, r', e_i を定める.

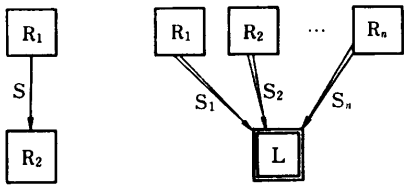
3. 論理 CODASYL モデル

3.1 従来の CODASYL モデルについて

本論文では, 従来の CODASYL DBS 上に述語論理形式の操作演算を備えた関係型視野の設計を目指している. このためには, 従来の CODASYL モデルの十分な論理的整理がまず必要になる. 従来の CODASYL モデルとして, 文献 1) のデータ構造と 5) のデータ操作言語 DML とを考察する. 文献 15) 等も考えられるが, 従来の CODASYL モデルの論理的に本質的な問題を検討するには, 1) と 5) で十分であると考える. また, 1) と 5) に準拠した DBMS (例, ADBS (日電), AIM (富士通)) が, 高性能大型 DBS に広く用いられているからでもある. 従来の CODASYL モデルは, 物理的側面と論理的側面が未分離であるとともに, 論理的な考察がほとんどなされていない. したがって, 本章では, 従来の CODASYL モデルから, データ構造を抽象化してモデル化し, これから論理基本演算を自然に導出して, 論理 CODASYL モデルを定める. また, 論理基本更新演算が従来の COBOL DML[5] によって実現できることを示し, 本論文で定めるモデルと従来モデルの論理関係(等価性)を明らかにする.

3.2 論理データ構造

論理 CODASYL モデルの論理データ構造は, 時間不変な論理構造を与える論理スキーマ $\underline{C}$ と, これに実際にデータが与えられた (時間可変な) 論理データベース C とから成る. $\underline{C}$ はレコード (R) 型とセット (S) 型とから成る. R 型 $\underline{R}$ は, 項目集合 $\Omega_R=\{$@R, $t_1,\cdots,t_m\}$ $(m\geq 0)$ とインテグリティ条件 Γ_R から成る $(\underline{R}=(\Omega_R,\Gamma_R))$. 各項目 τ の定義域を dom(τ) とする. @R はデータベース (db) キーで, dom(@R) は



(a) セット型　　(b) リンク型

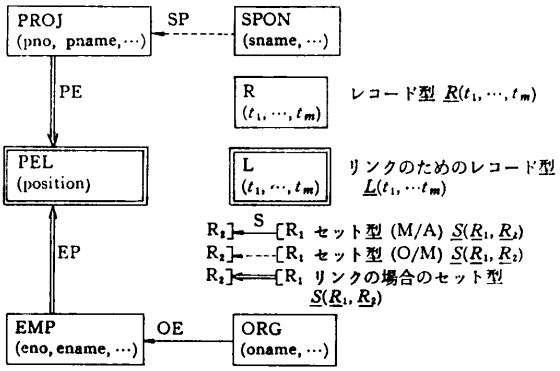
図 2 セット型とリンク型
Fig. 2 Set-type and link-type.

C 内で一意な識別子の集合である．各 t_i はデータ項目である $(i=1,\cdots,m)$．Ω_R に実際に値を与えた組のなかで Γ_R を満たす組（レコード実現値）の集合をレコード実現値(RO)集合 $R=\{r \mid r\in \mathrm{dom}(\Omega_R)\wedge\Gamma_R(r)\}$ とする．R 実現値 $r\in R$ の項目の部分集合 $I\subseteq\Omega_R$ で定まる値を $I(r)$ とする．各 $r\in R$ は一般に，データ項目値ではなく，$@R$ によって識別され，$@R(r)$ は C 内で一意である．よって，$@R$ は $\underline{R}$ の主キーとなる．利用者に $@R$ 値の情報は与えられないので，R 型を $\underline{R}(t_1,\cdots,t_m)$ (Γ_R は省略) とも書く．また各 $r\in R(r)$ の $@R(r)$ を $\delta(r)$ とも表す．

任意の R 型 $\underline{R}_2$ から $\underline{R}_1$ ($\neq\underline{R}_2$) に部分関数が定義されるときに，これをセット (S) 型 $\underline{S}=(\underline{R}_1,\underline{R}_2,\Gamma s)$ として表す(**図 2** (a))．$\underline{R}_1$ と $\underline{R}_2$ をおのおの，$\underline{S}$ の親と子 R 型とする．Ω_{R_1} と Ω_{R_2} から意味的に帰納されるインテグリティ条件 Γs を満たす組 $(r_1\in R_1, r_2\in R_2)$ (セット (S) 実現値)* の集合をセット実現値 (SO) 集合 $S=\left\{s \mid s\in \overset{2}{\underset{i=1}{\times}}\mathrm{dom}(\Omega_{Ri})\wedge\Gamma s(s)\right\}$ とする．$\underline{S}$ は部分関数を表すので，各 $r_2\in R_2$ は $(r_1,r_2)\in S$ なるたかだか一つの $r_1\in R_1$ を，また各 $r_1\in R_1$ は $(r_1,r_2)\in S$ なる $h(\geq 0)$ 個の $r_2\in R_2$ をもてる．以下，SO 集合 S を R 実現値の主キー，すなわち db キーで表すことにすると，$S=\left\{s' \mid s'=(\delta(r_1),\delta(r_2))\wedge s=(r_1,r_2)\in \overset{2}{\underset{i=1}{\times}}\mathrm{dom}(\Omega_{Ri})\wedge\Gamma s(s)\right\}$ と表現できる．また $(\delta(r_1),\delta(r_2))\in S$ で，r_1 と r_2 をおのおの，親と子 R 実現値とする．S 型を $\underline{S}(\underline{R}_1$ と $\underline{R}_2)$ (Γs は省略) とも表す．

任意の R 型 $\underline{R}_1,\cdots,\underline{R}_n(n\geq 2$，異なっている必要はない) について，$\Omega_{R_1},\cdots,\Omega_{R_n}$ から意味的に帰納されるデータ項目集合 $\Omega_{R'}$ とインテグリティ条件 Γ_L とが定義され，一般の関係 $L\subseteq \overset{n}{\underset{i=1}{\times}}\mathrm{dom}(\Omega_{Ri})\times\mathrm{dom}(\Omega_{R'})$

* 文献 1) では，S 実現値は，各 $r_1\in R_1$ に対して，r_1 と r_1 を親とする子 $r_2\in R_2$ との集合である．



(a) 論理 CODASYL スキーマ

E関係スキーム	PROJ (@P, pno, pname, ···, sal)
	SPON (@S, sname, ···)
	ORG (@O, oname, ···)
	EMP (@E, ename, ···)
P型R関係スキーム	SP (@S, @P)
T型R関係スキーム	OE (@O, @E)
G型R関係スキーム	PEL (@P, @E, position)

(b) LC スキーマ

図 3 論理 CODASYL スキーマと LC スキーマ
Fig. 3 Logical CODASYL schema and LC schema.

が定義されるとする．このとき，これを，共通の子 R 型 $\underline{R}(\Omega_R)(\Omega_R=\Omega_{R'}\cup\{@R\})$ をもつ n 個の異なった S 型 $\underline{S}_i(\underline{R}_i,\underline{R})$ $(i=1,\cdots,n)$ から成るリンク型 $\underline{L}=(\underline{R}_i,\underline{S}_i(\underline{R}_i,\underline{R})$ $(i=1,\cdots,n),\underline{R},\Gamma_L)$ と定義する(図 2 (b))．$L=\left\{l \mid l=(r_1,\cdots,r_m,r)\in \overset{n}{\underset{i=1}{\times}}\mathrm{dom}(\Omega_{Ri})\times\mathrm{dom}(\Omega_R)\wedge\Gamma_L(l)\overset{n}{\underset{j=1}{\wedge}}(\delta(r_j),\delta(r))\in S_j\right\}$ をリンク実現値 (LO) 集合とする．$\underline{S}_j$ と $\underline{R}$ をおのおの，リンクの場合の S 型と R 型とする．

以上の RO, SO, LO 集合の集合を論理データベース C とする．

S 型 $\underline{S}(\underline{R}_1,\underline{R}_2)$ には，M/A と O/M の 2 種の型を定めれる(リンクの場合も含む)．これはメンバシップクラス[1] の型であり，$\mathrm{mc}(\underline{S})$ と表す*．この意味については，3.3 節で述べる．型付けされた論理スキーマ $\underline{C}$ では，S 型を有向辺，R 型を点とした有向閉路内のすべての S 型が M/A 型であってはならず，また，リンクの場合の S 型は M/A 型であるとする．**図 3** (a) にプロジェクトと構成員，所属，出資者を表す論理スキーマ例を示す．箱は R 型，二重線の箱はリンクの場合の R 型，実線と点線の矢はおのおの，M/A と O/M 型の S 型を，二重線の矢は，リンクの場合の S

* 文献 1) では，$\mathrm{mc}(S)$ は消去に対する除去クラス $\mathrm{rc}(S)$ と追加に対する格納クラス $\mathrm{sc}(S)$ との対 $(\mathrm{rc}(S)/\mathrm{sc}(S))$ である．$\mathrm{rc}(S)\in\{M(\text{強制}),O(\text{選択})\}$，$\mathrm{sc}(S)\in\{A(\text{自動}),M(\text{手動})\}$ であるが，一般には $\mathrm{mc}(S)\in\{\mathrm{M/A},\mathrm{O/M}\}$ が用いられる．

型 (M/A 型) である.

3.3 論理基本更新演算

論理データベース C 上の論理基本演算としては,検索と更新がある.検索演算は,文献 14) で詳論するので,本論文では,更新演算について論じる.まず,C 内の任意の RO 集合を R とする.各 $r\in R$ に対して,そのすべての M/A 型 S 実現値の子 $r'\in R'$ の集合が定まる.同様に $r'\in R'$ の M/A 型 S 実現値の子 $r''\in R''$ の集合が定まる.r から始めて,このように再帰的に定義されていく全 R 実現値の集合を $M(r, R)$ (r も含める) とする.各 $r'\in M(r, R)$ を親または子とする全 S 実現値の集合を $O(r')$ とする.このとき,$\mathrm{ms}(r, R)\triangleq M(r, R)\underset{\forall r'\in M(r,R)}{\cup} O(r')$ とする.また,C 内の各 R 型 $\underline{R}$ に対して,$\underline{R}$ を子とするすべての M/A 型の S 型 $\underline{S}'(\underline{R}', \underline{R})$ と $\underline{R}'$ の対の集合を $O_S(\underline{R})$ とする.さて,$\underline{R}, \underline{S}(\underline{R}_1, \underline{R}_2), L=(\underline{R}_i, \underline{S}_i(\underline{R}_i, \underline{R})\ (i=1, \cdots, m), \underline{R})$ をおのおの,C 内の任意の R, S, L 型とすると,次の 9 種の論理基本更新演算が C 上に導ける.

(1) $\underline{\mathrm{del}}(r, R)$ は C から $\mathrm{ms}(r, R)$ を除く.(2) $\underline{\mathrm{del}}(s=(k_1, k_2), S)$ は,$\mathrm{mc}(\underline{S})=$M/A ならば $\mathrm{mc}(r_2, R_2)(\delta(r_2)=k_2)$ を,O/M ならば s のみを消去する.(3) $\underline{\mathrm{del}}(l, L)$ は $l\in L$ および,各 S_i から l を子とする $s_i\in S_i$ を消去する.(4) $\underline{\mathrm{app}}(r, R; r_1, R_1; \cdots; r_n, R_n)$ は,$(\underline{S}_i, \underline{R}_i)$ が $O_S(\underline{R})$ に含まれ $r_i\in R_i$ のとき $(i=1, \cdots, n)$,R に r を,S_i に $(\delta(r_i), \delta(r))$ $(i=1, \cdots, n)$ を加える.(5) $\underline{\mathrm{app}}(s=(\delta(r_1), \delta(r_2)), S)$ は,$\mathrm{mc}(\underline{S})=$O/M で $r_1\in R_1, r_2\in R_2$ ならば,S に s を加える.(6) $\underline{\mathrm{app}}(l, L; r_1, R_1; \cdots; r_n, R_n)$ は R を L とした(4)と同じである.(7) $\underline{\mathrm{rep}}(r, r', R)$ は $r\in R$ のデータ項目値を r' に変える.(8) $\underline{\mathrm{rep}}(s=(k_1, k_2), s'=(k_1', k_2), S)$ は,$s\in S$ で $r_1\in R_1(\delta(r_1')=k_1')$ ならば,s を s' に変える.(9) $\underline{\mathrm{rep}}(l, l', L)$ は $l\in L$ のデータ項目値を l' に変える. □

3.4 論理基本更新演算の DML 表現

従来の COBOL DML[5] (以降 DML と記す) の実行機構を抽象化し,われわれが示した論理データベース C 上で DML に対応した操作演算を実行できる抽象機械 M を定義する.また,論理スキーマ C は R 型 $\underline{R}_i(i=1, \cdots, n)$ と S 型 $\underline{S}_j(j=1, \cdots, m)$ から成るとする.M は,$3+2n$ 個のレジスタ $\mu, \nu, \kappa, \kappa_{R_i}, \beta_{R_i}$ $(i=1, \cdots, n)$,および記憶媒体Dをもつ.μ と ν にはおのおの現在処理中の R 型と S 型についての情報として,その名前がはいっている.κ と κ_{R_i} には,おのおの C と R_i 内で現在処理中の R 実現値の位置情報としての db キーがいれられる.β_{R_i} には,R_i の R 実現値のデータ項目値を記憶できる.レジスタ α の中身を (α) とする.D には C の各時刻の C がはいる.$\boldsymbol{\mu}, \boldsymbol{\nu}, \boldsymbol{\kappa}, \boldsymbol{\kappa}_{R_i}, \boldsymbol{\beta}_{R_i}, \boldsymbol{D}$ をおのおの,$\mu, \nu, \kappa, \kappa_{R_i}, \beta_{R_i}, D$ のとりうる値の集合とし,$\boldsymbol{K}=\boldsymbol{\kappa}\underset{i=1}{\overset{n}{\times}}\boldsymbol{\kappa}_{R_i}$, $\boldsymbol{B}=\underset{i=1}{\overset{n}{\times}}\boldsymbol{\beta}_{R_i}$ とする.m 上の演算 p は,関数 $p: \boldsymbol{\mu}\times\boldsymbol{\nu}\times\boldsymbol{B}\times\boldsymbol{K}\times\boldsymbol{D}\rightarrow\boldsymbol{B}\times\boldsymbol{K}\times\boldsymbol{D}$ である.p としては以下のものがある.

(i) レジスタ $\mu, \nu, \kappa, \kappa_{R_i}, \beta_{R_i}$ に値をセットする.ここでは κ と κ_{R_i} には,ある処理によって与えられるある条件式を満たす R 実現値の db キーがセットされているものとする.このセット方法については,文献 14) で別に論じる.(ii) $(\mu)=R_i$ のとき,(β_{R_i}) をデータ項目値とする R 実現値に,新たな db キーを与え,これを κ と κ_{R_i} にセットして R_i に加える.(iii) $(\mu)=R_i$ のとき,(κ) を db キーとする $r_i\in R_i$ を除く.同時に,r_i を含む全 S 実現値を除き,κ と κ_{R_i} をリセットする.(iv) $(\mu)=R_i, (\nu)=S_i$ のとき,(κ) を db キーとしてもつ $r_i\in R_i$ を子とする S 実現値を S_i から除く.(v) $(\mu)=R_j$, $(\nu)=S_i$ のとき,(κ) を db キーとする $r_j\in R_j$ を,(κ_{R_i}) を db キーとする $r_i\in R_i$ の子として S_i に加える.(vi) (κ) を db キーとする $r_i\in R_i$ のデータ項目値を (β_{R_i}) とする. □

論理基本更新演算 (3.3 節) を,以上の(i)～(vi)で表せることは示せる.この詳細は,文献 14) で示す.(ii)～(vi)はおのおの DML の $\underline{\mathrm{store}}\ R_i.$, $\underline{\mathrm{erase}}\ R_i.$, $\underline{\mathrm{disconnect}}\ R_i\ \underline{\mathrm{from}}\ S_i.$, $\underline{\mathrm{connect}}\ R_i\ \underline{\mathrm{to}}\ S_i.$, $\underline{\mathrm{modify}}\ R_i.$ に対応している.また,DML の $\underline{\mathrm{modify}}\ R_i\ \underline{\mathrm{only}}\ S_i.$ は(iii)と(iv),M/A 型S型についての $\underline{\mathrm{store}}\ R_i.$, $\underline{\mathrm{erase}}\ R_i\ \underline{\mathrm{permanent}}.$ はおのおの,(ii)と(v)と,$\mathrm{ms}(r_i, R_i)$ 内の各 R 実現値に(iii)を繰り返し適用することとによって表せる.よって,論理基本更新演算は,**表 1** に示すように DML で表せる.たとえば,$\underline{\mathrm{del}}(r, R)$ は,(1)まず κ に消去すべき $r\in R$ の db キーを記憶し $(\kappa\leftarrow\delta(r))$,(2) $\underline{\mathrm{erase}}\ R\ \underline{\mathrm{permanent}}.$ で $\mathrm{ms}(r, R)$ を除くことで表せる.また $\underline{\mathrm{app}}(r, R; r_1, R_1; \cdots; r_n, R_n)$ は,(1)追加する r のデータ項目値を β_R に記憶し,(2)各 $r_i\in R_i$ の db キーを κ_{R_i} に記憶し $(\kappa_{R_i}\leftarrow\delta(r_i))$,(3) $\underline{\mathrm{store}}\ R.$ で R に r と,各 S_i に $((\kappa_{R_i}), \delta(r))$ とを加えることで表せる.以上より次の命題が成り立つ.

表 1 CODASYL 基本演算の COBOL DML 表現
Table 1 Primitive operations and COBOL DML in the CODASYL model.

基本演算	COBOL DML 表現
(1) del (r, R);	[$\kappa \leftarrow \delta(r)$;] erase R permanent.
(2) del (s, S); ここで $s=(k_1, k_2)$	mc (S)=M/A[$\kappa \leftarrow \delta(r)$;] erase R_2 permanent mc (S)=O/M[$\kappa \leftarrow \delta(r)$;] disconnect R_2 from S.
(3) del (l, L);	[$\kappa \leftarrow \delta(l)$;] erase L.
(4) app $(r', R;\ r_1 \in R_1;\ \cdots;\ r_n \in R_n)$; (6) R はレコードまたはリンク型	[$\kappa_{Ri} \leftarrow \delta(r_i)$; $(i=1, \cdots, n)$] [$\beta_R \leftarrow r'$;] store R.
(5) app (s, S);	mc (S)=O/M[$\kappa_{R1} \leftarrow \delta(r_1)$; $\kappa \leftarrow \delta(r_2)$;] connect R_2 to S.
(7) rep (r, r', R); (9) R はレコードまたはリンク型	[$\kappa \leftarrow \delta(r)$; $\beta_R \leftarrow r'$;] modify R including $S_1, \cdots, S_n$*.
(8) rep (s, s', S); ここで $s=(k_1, k_2)$, $s'=(k_1', k_2)$	[$\kappa_{R1} \leftarrow k_1'$ $\kappa \leftarrow k_2$; $\kappa \leftarrow k_2$;] modify R_2 only S.

* 置換されるデータ項目のどれかがセット型 $S_1, \cdots, S_n$ のソートまたは DNA 項目である．CALC と項目値の置換はできない．

[命題 3.1] すべての論理基本更新演算は，従来の COBOL DML で表せる．■

従来の COBOL DML[5] の厳密な意味については，文献 14) で多テープチューリング機械によって表す．

4. 概念モデルによる論理 CODASYL モデル表現

本章では，概念モデルの R スキームを特性化し，それに従って導かれる基本更新演算を定義し，これを用いてローカル概念 (LC) モデルを与える．また，論理 CODASYL モデルと本 LC モデルとが等価であることを示す．

4.1 LC データ構造

概念スキーマ $\underline{S}$ 内の各 R スキーム $\underline{R}$ を以下のように特性化したものをローカル概念 (LC) スキーマ $\underline{\mathcal{L}}$ とする．(1) R スキーム $\underline{R}$ に，FとGとの2種の型を設ける．$\underline{R}$ が E スキーム $\underline{E}_2$ から $\underline{E}_1$ への部分関数を表すとき，$\underline{R}$ をF型とする．関数の定義域を表す $@E_2$ が主キーとなり，$\underline{R}(@E_1, @\underline{E}_2)$ と表す．ここで，下線を引いた属性は，主キーを表すとする．$\underline{R}=(\underline{E}_1, \cdots, \underline{E}_n, \Omega_R')$ が $\underline{E}_1, \cdots, \underline{E}_n$ 間の一般の関係を表すとき，$\underline{R}$をG型とする．各 $\underline{E}_i$ の主キー集合 $\{@E_1, \cdots, @E_n\}$ の値によって R 内の各組が定まるので，これを主キーとし，$\underline{R}(@\underline{E}_1, \cdots, @\underline{E}_n, \Omega_R')$ と表す．(2) F型 $\underline{R}$ にはさらに，T (全) または P (部分) の型が与えられ，おのおの更新演算に異なった意味を与える (4.2 節)．□

$\underline{\mathcal{L}}$ に実際にデータを与えたものを LC データベース $\underline{\mathcal{L}}$ と定義する．

LC と論理 CODASYL データ構造は次のように対応する．(1) R型 $\underline{R}(t_1, \cdots, t_m)$ と E スキーム $\underline{R}(@\underline{R}, t_1, \cdots, t_m)$ に対しては，データ項目 t_i と属性 t_i，db キーと $@R$ とが対応する．(2) S型 $\underline{S}(\underline{R}_1, \underline{R}_2)$ には，Fスキーム $\underline{S}(@R_1, @\underline{R}_2)$ が対応する．$@R_i$ はR型 $\underline{R}_i$ に対応した E スキームの主キーである $(i=1, 2)$．$\underline{S}(@R_1, @\underline{R}_2)$ は，S型 $\underline{S}$ が M/A のときT型に，O/M のときP型になる．(3) L型 $\underline{L}=(\underline{R}_i, \underline{S}_i(\underline{R}_i, \underline{R})$ $(i=1, \cdots, n)$, $\underline{R}(b_1, \cdots, b_q))$ はGスキーム $\underline{L}(@\underline{R}_1, \cdots, @\underline{R}_n, b_1, \cdots, b_q)$ に対応する*．$@R_i$ は R 型 $\underline{R}_i$ に対応したスキームの主キーである．$(i=1, \cdots, n)$．□

以上より，次の命題が成り立つことは明らかである．

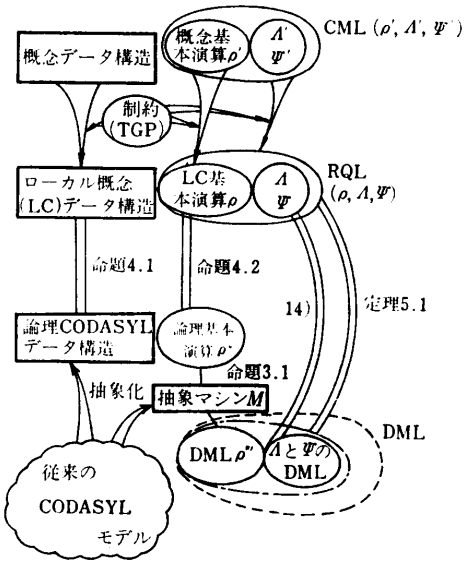


図 4 CODASYL モデルと概念モデル
Fig. 4 CODASYL model and conceptual model.

* $\underline{R}_i=\underline{R}_j$ $(i \neq j)$ の場合には，$S_i@R_i, S_j@R_j$ を属性名とする．

[命題 4.1] LC データ構造は，論理 CODASYL データ構造と1対1対応する．■

図3(b)は，(a)の LC スキーマを示している．

4.2 LC 操作言語 RQL

LC モデルの述語論理形式の操作言語 RQL (relational query language) は，目標リスト Λ，条件式 Ψ，および基本演算 ρ によって LC データベース上の演算を表す．ρ としては基本検索演算と基本更新演算がある．Λ と Ψ は2.3節で述べたものと同じである．本節では，まず基本更新演算を述べ，次に RQL 更新演算表現を述べる．

4.2.1 基本更新演算

$\underline{R}$ と R をおのおの任意の E スキームとその関係とする．命題4.1より，LC と論理 CODASYL データ構造は1対1対応するので，3.2節の $ms(r, R)$ と $Os(\underline{R})$ とおのおの等価な集合として $ts(r, R)$ と $Ts(\underline{R})$ を自然に定義できる．RスキームがPかG型だけから成る場合には，2.3.2 項の概念基本更新演算と同じである．T型の R スキームを含む場合には，次の LC 基本更新演算が自然に存在する．(i) $\underline{del}(r, R)$ は $ts(r, R)$ を消去する．(ii) $\underline{del}(s, S)$ は，$\underline{S}$ がT型のとき $@R_2(s)=@R_2(r_2)$ なる $ts(r_2, R_2)$ を消去する．(iii) $\underline{app}(r, R; r_1, R_1; \cdots; r_n, R_n)$ は，$(\underline{S_i}, \underline{R_i})$ が $Ts(\underline{R})$ に含まれ $r_i \in R_i$ $(i=1, \cdots, n)$ のとき，R に r を加えるとともに $@R_i(r_i)=@R_i(s_i) \wedge @R(s_i)=@R(r)$ なる s_i を S_i に加える $(i=1, \cdots, n)$.

以上より，次の命題が成り立つことは明らかである．

[命題 4.2] LC モデルと論理 CODASYL モデルの基本更新演算は，1対1対応する．■

[定理 4.1] LC モデルと論理 CODASYL モデルとは，データ構造と基本演算について1対1対応し，等価である．

[証明] 命題4.1と4.2より互いのデータ構造と基本更新演算とが等価であるので，明らかである．■

4.2.2 RQL 更新演算

RQL による更新演算表現について述べる．ここで，x をある組変数集合とし，Λ_x と Ψ_x をおのおの，x を参照する目標リストと条件式とする．

A. 消去演算

$\underline{X}$ を $\underline{\mathcal{L}}$ 内の消去対象の関係スキームとし，$\underline{X_i}$ を任意の関係スキームとする．関係 X への消去演算は RQL で，次のように書かれる．

$$\left.\begin{array}{l}\underline{\text{var}}\ (u, X)\ (x_1, X_1)\cdots(x_l, X_l);\\ \underline{\text{delete}}\ u\ \underline{\text{where}}\ \Psi_x;\\ \qquad (x=\{u, x_1, \cdots, x_l\})\end{array}\right\} \quad (5)$$

(5)は，$t \in X \underset{i=1}{\overset{l}{\wedge}} x_i \in X_i \wedge \Psi_x(t, w_1, \cdots, w_l)$ なる組 $t \in X$ について，LC 基本更新演算 $\underline{del}(t, X)$ することを表している．たとえば，$\underline{X}$ がE型ならば $ts(t, X)$ が消去される．

B. 追加演算

(i) $\underline{E}$ を追加対象の E 関係スキームとする．$\underline{E}$ は n 個のT型 $\underline{R_i}(@E_i, @\underline{E})$ で E 関係スキーム $\underline{E_i}$ と関連しているとする $(Ts(\underline{R})=\{(\underline{R_i}, \underline{E_i}) \mid i=1, \cdots, n)\}$．また，$\underline{X_j}$ を $\underline{\mathcal{L}}$ 内の任意の関係スキームとする．E への追加は RQL で次のように書かれる．

$$\left.\begin{array}{l}\underline{\text{var}}\ (e_1, E_1)\cdots(e_n, E_n)(x_1, X_1)\cdots(x_l, X_l);\\ \underline{\text{append to}}\ E(\Lambda_x)\ \underline{\text{where}}\ \Psi_x;\\ \qquad (x=\{e_1, \cdots, e_n, x_1, \cdots, x_l\})\end{array}\right\} \quad (6)$$

(6)は，$w=(t_1, \cdots, t_n, w_1, \cdots, w_l) \underset{i=1}{\overset{n}{\wedge}} t_i \in E_i \underset{j=1}{\overset{l}{\wedge}} w_j \in X_j \wedge \Psi_x(w)$ なる w に対する $t'=\Lambda_x(w)$ を求め，組 $(t_1, \cdots, t_n, t')$ について $app(t', E; t_1, E_1; \cdots; t_n, E_n)$ を行うことを表す．

(ii) $\underline{R}$ を追加対象の R 関係スキームとし，$\underline{X_i}$ を任意の関係スキームとする．R に対する追加演算は，RQL で次のように表される．

$$\left.\begin{array}{l}\underline{\text{var}}\ (x_1, X_1)\cdots(x_l, X_l);\\ \underline{\text{append to}}\ R(\Lambda_x)\ \underline{\text{where}}\ \Psi_x;\\ \qquad (x=\{x_1, \cdots, x_l\})\end{array}\right\} \quad (7)$$

(7)は，$w=(w_1, \cdots, w_l) \underset{j=1}{\overset{l}{\wedge}} w_j \in X_j \wedge \Psi_x(w)$ なる w について，$t'=\Lambda_x(w)$ を求め，$\underline{app}(t', R)$ を行うことを表す．

C. 置換演算

$\underline{X}$ を置換対象の関係スキーム，$\underline{X_i}$ を任意の関係スキームとする．X への置換演算は RQL で次のように表される．

$$\left.\begin{array}{l}\underline{\text{var}}\ (u, X)(x_1, X_1)\cdots(x_l, X_l);\\ \underline{\text{replace}}\ u(\Lambda_x)\ \underline{\text{where}}\ \Psi_x;\\ \qquad (x=\{u, x_1, \cdots, x_l\})\end{array}\right\} \quad (8)$$

(8)は，$w=(t, w_1, \cdots, w_l) \wedge t \in X \underset{j=1}{\overset{l}{\wedge}} w_j \in X_j \wedge \Psi_x(t, w_1, \cdots, w_l)$ なる w に対する $t'=\Lambda_x(w)$ を求め，(t, t') について $\underline{rep}(t, t', X)$ を行うことを表している．すなわち，$t \in X$ の非主キー値を t' に置換する．

5. 非手続的インタフェース設計

CODASYL DBS 上に，LC モデルの視野と非手続的操作言語 RQL とを利用者に提供するインタフェースを設けるためには，LC データ構造上の RQL 演算を，COBOL DML で実現する必要がある．本章では，RQL を COBOL DML で表せることを示す．まず命題 3.1（表 1）と命題 4.2 より次の命題が成り立つことは明らかである．

［命題 5.1］ すべての LC 基本更新演算を COBOL DML で表現できる．■

次に，4.2.2 項で述べた RQL 更新演算について考える．関係 X に対する RQL 演算が組変数集合 $x=\{y_1,\cdots,y_q\}$（y_i は関係 Y_i の組変数）を参照するとき，この RQL 演算の意味は，次の(9)の形式で表せる．

$$\text{for } \forall v(v=\lambda_x(w)\wedge\Psi_x(w)\wedge w\in Y_1\times\cdots\times Y_q)\quad \underline{\text{do op}}\ (v, X);\qquad (9)$$

op は LC 基本更新演算子である．条件式 Ψ_x を満足する組 w について，目標リスト λ_x はある目標属性集合 Ω 値の組 v を与える．たとえば 4.2.2 項の B (i)の append では，λ_x は組 w から組 $(t_1,\cdots,t_n,t')$ を与える．(9)の for…do は更新データ v の集合を導出する目標リスト Λ_x と条件式 Ψ_x から成る RQL 検索演算として表せる．RQL 検索演算を DML で正しく表す方法は示せるが，この手順は別[14]に論じる．またこの手順に従って設計されたシステムは LDP-V 1.5[8),10)] として実働している．さて，(9)の LC 基本演算 op(v, X) を命題 5.1 より DML で表せるので，RQL 更新演算の DML 表現を**表 2** のように示せる．よって，次の定理が成り立つ．

［定理 5.1］ すべての RQL 演算を COBOL DML で表現できる．■

例として図 3 (b)を考えると，消去演算「Jipdec を除け」は RQL で(10)のように表される．

```
var(o, ORG);
  delete o where o.oname="Jipdec";   (10)
```

(10)は，表 2 より COBOL DML で表現すると，次の手順に対応する．(i) oname(t)="Jipdec" なる $t\in$ORG を見つけ，t の db キーを κ に記憶し，(ii) erase ORG permanent. を行う．これによって，Jipdec の従業員も同時に消去される．

表 2 RQL の COBOL DML 表現
Table 2 COBOL DML description of RQL operations.

RQL	RQL の COBOL DML 表現
A. 消去 var $(u, X)(x_1, X_1)\cdots(x_l, X_l)$; delete u where Ψ_x; $(x=\{u, x_1,\cdots, x_l\})$	$T\leftarrow\{t\mid w=(t, w,\cdots, w_l)\in X\times X_1\times\cdots\times X_l\wedge\Psi_x(w)\}$; for $\forall t\in T$ do (i) X は E 型　$\kappa\leftarrow@X(t)$; erase X permanent. od; (ii) X は B 型　$X(@E_1, @E_2)$ (ii-1) X は T 型　$\kappa\leftarrow@E_2(t)$; erase E_2 permanent. od; (ii-2) X は P 型　$\kappa\leftarrow@E_2(t)$; disconnect E_2 from X. od; (iii) X は G 型　$X(@E_1,\cdots,@E_n, \Omega_X')$ $\kappa\leftarrow t\in X$ の db キー; erase X. od;
B. 追加 (1) E 関係 E への追加 $E(@E, \Omega_E')$ var $(e_1, E_1)\cdots(e_n, E_n)$ $(x_1, X_1)\cdots(x_l, X_l)$; append to E (Λ_x) where Ψ_x; $(x=\{e_1,\cdots, e_n, x_1,\cdots, x_l\})$ (2) R 関係 X への追加 var $(x_1, X_1)\cdots(x_l, X_l)$; append to $X(\Lambda_x)$ where Ψ_x; $(x=\{x_1,\cdots, x_l\})$	(1) $T\leftarrow\{(t_1,\cdots, t_n, t)\mid w=(t_1,\cdots, t_n, w_1,\cdots, w_l)\in E_1\times\cdots\times E_n\times X_1\times\cdots\times X_l\wedge\Psi_x(w)\wedge t=\Lambda_x(w)\}$; for $\forall(t_1,\cdots, t_n, t)\in T$ do $\beta_E\leftarrow\Omega_E'(t)$;　$\kappa_{Ei}\leftarrow@E_i(t_i)$ $(i=1,\cdots, n)$; store E. od. (2) $T\leftarrow\{t\mid w\in X_1\times\cdots\times X_l\wedge\Psi_x(w)\wedge t=\Lambda_x(w)\}$; for $\forall t\in T$ do (i) X は P 型　$X(@E_1, @E_2)$ $\kappa_{E1}\leftarrow@E_1(t)$;　$\kappa_{E2}\leftarrow@E_2(t)$; connect E_2 to X. od; (ii) X は G 型　$X(@E_1,\cdots, @E_n, \Omega_X')$ $\kappa_{Ei}\leftarrow@E_i(t)$ $(i=1,\cdots, n)$;　$\beta_X\leftarrow\Omega_X'(t)$; store X. od;
C. 置換 var $(u, X)(x_1, X_1)\cdots(x_l, X_l)$; replace $(u\Lambda_x)$ where Ψ_x; $(x=\{u, x_1,\cdots, x_l\})$	$T\leftarrow\{(t, t')\mid w=(t, w_1,\cdots, w_l)\in X\times X_1\times\cdots\times X_l\wedge\Psi_x(w)\wedge t'=\Lambda_x(w)\}$; for $\forall(t, t')\in T$ do (i) X は E 型　$X(@X, \Omega_x')$ $\kappa\leftarrow@X(t)$;　$\beta_X\in\Omega_X'(t')$; modify X. od; (ii) X は B 型　$X(@E_1, @E_2)$ $\kappa_{E1}\leftarrow@E_1(t')$;　$\kappa\leftarrow@E_2(t)$; modify E_2 only X. od; (iii) X は G 型　$X(@E_1,\cdots, @E_n, \Omega_X')$ $\kappa\leftarrow t\in X$ の db キー;　$\beta_X\leftarrow\Omega_X'(t')$; modify X including $S_1,\cdots, S_n$. od;

追加演算「鈴木を（田中と同所属で）EMP に加えよ」は RQL で(11)のように書かれる.

```
var(o, ORG) (e, EMP)(œ, OE);
append to EMP (e.¬ame="鈴木", …)
where e.ename="鈴木" and e.@E
  =œ.@E and œ.@O=o.@O;          (11)
```

(11)は COBOL DML で表すと次の手順に対応する.（ⅰ）$t_1 \in \mathrm{ORG} \wedge w_1 \in \mathrm{EMP} \wedge \mathrm{ename}(w_1)=$"田中" $\wedge(\delta(t_1), \delta(w_1)) \in \mathrm{OE}$ なる t_1 を求め，κ_{ORG} に $\delta(t_1)$ を記憶し，（ⅱ）β_{EMP} に追加する R 実現値（"鈴木", …）を記憶し，（ⅲ）store EMP. を行う. これによって，EMP に "鈴木" を加えるとともに，鈴木を田中と同じ所属にする S 実現値を SO 集合 OE に加えることができる.

6. むすび

本論文では，DDBS の統合化問題の解として，DDBS 全体の共通モデルに基づいて，従来の CODASYL DBS 上に視野を設計するための基本的問題を論じた. まず，既存のデータモデル（階層，網）を抽象化し，これらのモデルの（静的）なデータ構造を表せるモデルとして，事象と関連概念を有した概念モデルを共通モデルとして定めた. 次に，従来の CODASYL モデルを抽象化した論理 CODASYL データ構造と，それから自然に導ける論理基本更新演算とによって論理 CODASYL モデルを定義した. この論理モデルは，従来の CODASYL モデルより上位にある. また，論理基本更新演算を，従来の DML で表せることを示し，両モデルの論理的関係を明らかにした. 次に，概念モデルの R 関係スキームを型づけし，型によって基本更新演算を定めて，ローカル概念 (LC) モデルとし，この LC モデルと論理 CODASYL モデルの等価性を示した. さらに，述語論理に基づいた LC 操作言語 RQL を，LC 基本更新演算に述語論理形式の条件式 Ψ と目標リスト Λ を付加したものとして定義した. Ψ と Λ を DML で表せること[8),10),14)]と，LC 基本更新演算を DML で表せることから，すべての RQL を DML で表せることを示した. 以上によって，従来の CODASYL モデルの論理記述と，従来の CODASYL DBS 上に述語論理形式の操作言語を備えた視野を提供する論理的基礎とを与えることができたと考える.

以上の設計概念は，すでにローカルデータベースプロセッサ (LDP-V 2) として当協会の AIM (M-170 F) と ADBS (Acos-700) 上で実働している. LDP-V 2 は PL/I で作成されている（ソース約 15,000 文，実行形式 250 kB）. 現在，本システムの性能面と利用面からの総合評価を行っているが，数行の RQL の入力によって，複雑なデータベース保守を行う 1,000～3,000 行の COBOL DML プログラムを 1 分以内の経過時間 (M-170 F) で生成できる. LDP は DDBS の共通インタフェースとなるとともに，既存 CODASYL DBS の非定型利用インタフェースおよび COBOL DML プログラムの開発支援ツールとなると考えている.

参考文献

1) CODASYL DDL Committee: CODASYL Data Description Language, J. Dev., (1973)（藤中他訳：CODASYL データベース用記述言語, p. 250, 情報処理学会 (1977)).
2) Codd, E. F.: A Relational Model of Data for Large Shared Data Banks, *CACM*, Vol. 13, No. 6, pp. 337-387 (1980).
3) Date, C. J.: *An Introduction to Database Systems*, 3rd ed., p. 574, Addison-Wesley, Reading (1981).
4) Held, G. et al.: INGRES—A Relational Data Base System, AFIPS Conf. Proc., pp. 409-416 (1976).
5) Olle, T. W.: *The CODASYL Approach to Data Base Management*, p. 287, John Wiley & Sons, New York (1978)（西村，植村他訳：CODASYL のデータベース，共立出版，東京 (1975)).
6) Takizawa, M., Hamanaka, E. and Ito, T.: Resource Integration and Data Sharing on Heterogeneous Resource Sharing System, Proc. ICCC '78, pp. 253-258 (1978).
7) Takizawa, M. and Hamanaka, E.: The Four-Schema Concept as the Gross Architecture of Distributed Databases and Heterogeneity Problems, JIP (IPSJ), Vol. 2, No. 3, pp. 134-142 (1979).
8) Takizawa, M. and Hamanaka, E.: Query Translation in Distributed Databases, Proc. IFIP '80, pp. 451-456 (1980).
9) Takizawa, M.: Distribution Problems in Distributed Databases-Integration and Query Decomposition, JIP (IPSJ), Vol. 5, No. 3, pp. 139-147 (1982).
10) 滝沢 誠，横塚 実，鈴木 信：CODASYL データベースシステムに対する関係インタフェースシステム—LDP-V 1.5—の設計と実現について，情報処理学会論文誌，Vol. 23, No. 6, pp. 665-675 (1982).

11) Smith, J. M. and Smith, D. C. P.: Database Abstractions: Aggregation and Generalization, *ACM TODS*, Vol. 2, No. 2, pp. 105-133 (1977).
12) Wong, E.: Logical Design and Schema Conversion for Relational and DBTG Databases, Proc. E-R Conf., pp. 311-321 (1979).
13) Zaniolo, C.: Design of Relational Views over Network Schemas, Proc. SIGMOD, pp. 177-190 (1979).
14) 滝沢 誠, 野口正一: 非手続的グラフ問合せの DML 表現について, (準備中).
15) CODASYL DDL Committee: Report of the CODASYL Data Description Language Committee, *Inf. Syst.*, Vol. 3, No. 4, pp. 247-320 (1978).
16) 増永良文: 関係スキーマ上の CODASYL インタフェース設計とその応用, 情報処理学会データベース管理システム研究会資料 29 (1982).

(昭和57年12月 2 日受付)
(昭和58年 6 月20日採録)