関係スキーマ上のCODASYLインターフェイスの設計とその応用

増永良文（東北大学電気通信研究所）

Ⅱ. はじめに

本稿はデータの関係モデルで記述された概念スキーマ，これを単に関係スキーマと呼ぼう，上にCODASYLデータモデルで記述された概念スキーマ，これをCODASYLスキーマと呼ぶ，をサポートするためのインターフェイス設計とその応用について述べる。

本研究には少くとも次の三つの意義を挙げることができる。

（1）異なったデータモデルで定義された（概念）スキーマ間の等価性を明らかにすることができる。従来関係データモデルが提案されてそれ程遅くなく，関係スキーマをCODASYLスキーマに変換する問題が議論されたが，これら変換論はスキーマ間の情報保存を主たる目的としている意味で，静的な変換理論であった。本稿で展開しているインターフェイス設計は単に情報保存のみならず，サポートされるCODASYLスキーマ上でのデータベース更新をも可能とするスキーマ変換理論であるという意味で，動的な理論である。一方，本稿と逆の立場にあるCODASYLスキーマ上の関係インターフェイスの設計については，近年上述と同じ意味で動的な変換理論が発表されるにいたっている。したがって，本稿の結果とそれらの結果をつき合わすことにより，関係スキーマとCODASYLスキーマの動的な等価性の議論を展開することが可能となる。

（2）貴重なデータベース資源を有効に共同利用するためには，スキーマ変換理論に基礎はけられた多データモデルデータベースシステムを開発する必要がある。このシステム実現のためには（i）スキーマの直接変換アーキテクチャによる方法と，（ii）共通データモデルアーキテクチャによる方法の二つがある。前者は後者に比べてより現実的であり，明らかにスキーマ変換によるオーバーヘッドが最小で応答性が良い。この アーキテクチャで例えば関係データベースをより広汎な利用に提供するため，CODASYLインターフェイスを設けて，利用者にあたかもデータベースがCODASYL型であるように提供するシステムを構築するには本稿での議論が必要不可欠である。

（3）本稿の議論は関係モデルによるCODASYL型データベースの設計を可能とするものである。つまりDBTG提案に準拠したCODASYL型データベースとその管理システムは現在広く世界に普及しているが，そのデータベース設計においてはデータ構造の基本であるレコード型，親子集合型，メンバーシップクラス，アクセスパス属性，等多くの概念を理解する必要があり，いわゆるデータベースの非専門家がこのデータモデルを使ってデータベース設計を行なう際の大きな障害となっている。一方，関係データモデルは対象とする実世界の情報構造を関係表の集まりとして捉えることでデータベース設計が行なえるとされているため，CODASYL型データモデルを用いて設計を行なうことに比べて，いわゆるデータベースの非専門家向きのモデルと考えられている。この立場からは，本稿で議論している関係スキーマ上のCODASYLインターフェイスの設計手法は，設計者が関係モデルを用いてデータベース設計を行なってそれを入力すれば，それと静的のみならず動的にも等価なCODASYLスキーマを生

かしてくれるスキーマ生成器の仕様をまとめていることが判る。

2. 関係スキーマ

2.1 関係スキーマ記述

図－1に本稿での関係スキーマ記述の一例を示す。本稿の定義では関係スキーマは次の4つの句からなる。

a. DOMAIN句
b. RELATION句
c. ALIAS句
d. INTEGRITY句

またRELATION句はKEY副句を含んでいる。

```
DOMAIN      eno          NUMERIC(6)
DOMAIN      ename        CHARACTER(20)
DOMAIN      birthyear    NUMARIC(4)
DOMAIN      assignment   NUMERIC(10)
DOMAIN      edept        NUMERIC(4)
DOMAIN      dno          NUMERIC(4)
DOMAIN      location     CHARACTER(5)
DOMAIN      mgr          NUMERIC(6)
DOMAIN      jid          NUMERIC(10)
DOMAIN      title        CHARACTER(10)
DOMAIN      salary       NUMERIC(6)

RELATION    EMP(eno,ename,birthyear,
                assignment,edept)
            KEY(eno)
RELATION    DEPT(dno,location,mgr)
            KEY(dno)
RELATION    JOB(jid,title,salary)
            KEY(jid)
RELATION    ALLOC(dno,jid,number)
            KEY(dno,jid)
RELATION    QUAL(jid,eno)
            KEY(jid,eno)

ALIAS       assignment;jid
ALIAS       edept;dno
ALIAS       mgr;eno

INTEGRITY FOREIGN KEY CONSTRAINT
```

Fig. 1. A Sample Relational Schema

ここでこの定義の特徴を述べる。

(1) 本来，関係の属性（attribute）とドメインは明確に区別されねばならない概念であるが，本定義では両者を同一視してしまう因習に従っている。（この因習では異なった関係に表われる同一の属性名は同一のドメインを持つということを保証してくれるのみならず，同じセマンティックスを持っていることをも保証してくれているようである。これは次にALIAS句を定義するときに大変役に立つ。）

(2) 属性の名前は異なるが，本来同一ドメインをとる二つの属性AとBが次の条件を満すとき，ALIAS A; B と表わす：すなわち $dom(A)$，$dom(B)$ で属性A，Bのドメイン，M_A と M_B で $dom(A)$，$dom(B)$ 上の一変数時変述語（$x \in dom(A)$ に対して $M_A(x)$ が真となるのはその時刻において x が属性Aの値であるとき及びそのときのみと定義される）とするとき，ALIAS A; B であるのは $(\exists m : dom(A) \to dom(B))(\forall x \in dom(A), M_A(x) \supset M_B(m(x)))$ が成立するとき及びそのときのみである。ALIAS A; B，つまりAはBの別名であるという関係は反射的，推移的であるが対称的ではない。属性Cの別名関係による反射的，推移的閉包を C^* で表わすことにする。別名であることの具体的意味は例えば図－1のALIAS mgr; eno 等の実例から容易に把握されるであろう。

(3) 従来の関係スキーマ記述では保全制約で外部キー制約が常に陽に指定されることはないが，本稿では関係スキーマ上にサポートされるCODASYLデータベースでの更新をも許すインターフェイスを設計するために常に指定することが必要である。この制約の具体的意味については次節で述べる。

2.2 外部キーと外部キー制約

まず外部キーを定義することから始める。現在外部キーの定義は多様であるが本稿では外部キーを単に値の制約だけでなく，上に導入した別名の概念を積極

的に使い，その意味での概念の含意の関係として捉えている。関係Rの(主)キーはそのKEY副句に指定されている属性(集合)である。K(R)で関係Rのキーを表わす。このとき外部キーを次のように定義する：

『外部キーの定義』 関係Rの属性の(集合)Fが(それの定義されている関係スキーマ中で)外部キーであるとは，(i) F ≠ K(R)，かつ(ii)そのスキーマのある関係S(Sは必ずしもRと異なる必要はない)が存在して，F ∈ K(S)* であるときおよびそのときのみをいう。(このときFをRのSに関する外部キーということにする)

図-1の例ではassignmentやALLOCのdno等が外部キーである。本稿の定義する外部キーの関係は属性間の存在従属の関係(の一部)になっている。

次に外部キー制約について述べる。

『外部キー制約の定義』 FをRのSに関する外部キーとする。このとき外部キー制約とは R[F] ⊆ S[K(S)] が常に成立することを要求する制約である。ここにR[X]は関係Rの属性(集合)X上の射影を表わす。

外部キー制約はデータベースの保全制約であるので，データベースが更新により異常を発生することのないよう管理する基準を暗示している。具体的には次に示すとおりである。

『更新波及』 FをRのSに関する外部キーとし，データベースに外部キー制約が課せられているとする。(a)このときSのタップル t を削除したとすると，同時に t[K(S)]値に相当する値をF値として持つ全てのタップルをRから削除しなければならない。また(b)Rへタップル t を挿入するときには，t[F]値に相当するK(S)値を持つSのタップルが既に存在しなければならない。

ここに相当するという単語は二つの属性が別名であることを定義するときに導入された写像m(の合成)により値が決まるという意味である。またこの更新波及はデータベース全体に再帰的に波及する。

3. CODASYLインターフェイス

3.1 スキーマ変換

前章で定義された関係スキーマをCODASYLスキーマに変換する方法を論ずる。図-1の例をとり具体的に説明する。

(ステップ1) 関係スキーマのKEY副句，ALIAS句の情報を使い，外部キーの全てを求める。FがRのSに関する外部キーであることを FOREIGN KEY R.F ; S.K(S) と表わすことにする(自名の場合関係名を省略することもある)。図-2に図-1の全ての外部キーを示す。なおこの例では出現しないが，FOREIGN KEY A ; B かつ FOREIGN KEY B ; C から推移的に誘引される外部キー句 FOREIGN KEY A ; C はリストしないこととする。

(ステップ2) 関係スキーマを構成している関係名をノード名とし，FOREIGN KEY R.F ; S.K(S) なるとき，ノードSからノードRへアークを引き，そのアークに属性名Fを R-F のように付随させた

```
FOREIGN KEY assignment ; JOB.jid
FOREIGN KEY edept      ; DEPT.dno
FOREIGN KEY mgr        ; EMP.eno
FOREIGN KEY ALLOC.dno  ; DEPT.dno
FOREIGN KEY ALLOC.jid  ; JOB.jid
FOREIGN KEY QUAL.jid   ; JOB.jid
FOREIGN KEY QUAL.eno   ; EMP.eno
```

Fig. 2. Foreign Keys of the Sample Relational Schema

グラフを定義する。 図-3に本稿の例を示す。

(ステップ3) [キー返還] 入線のある全てのノードについて，そのノードの関係Rの属性からそのノードの入線に付随している属性名（R-Fの場合はF）を全て消去するし，関係名をR*とする。もし属性名が全て消去されたなら，その関係の属性は中で表わす。 この操作によりキーが変化する可能性があるが，それはK(R)と消去されずに残った属性との共通集合になる。 非空なら新しいキーにアンダーラインを引く。 以上の操作をキー返還という。

(ステップ4) 新しいノードSYSTEMを導入し，ステップ3までで得られたグラフに入線の一つもないノードがあればSYSTEMノードからそのノード（ノード名をRとする）にラベルR-SETを付随させたアークをひく。 入線のないノードは一つもない場合は適当なノードを少なくとも一つえらび同様操作する。

(ステップ5) グラフ中の全てのループには*印をつける。 サイクル（周期2以上）がなしているときはそれを構成しているアークを少なくとも一つ選び*印をつける。

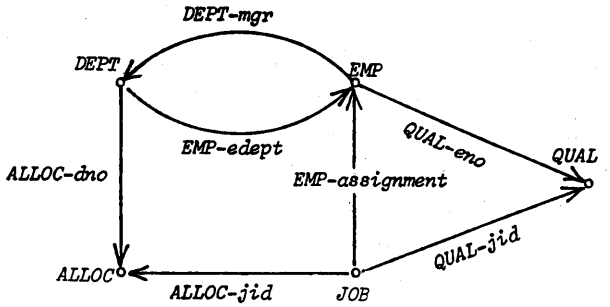

Fig. 3. A Graph Representation of the Sample Relational Schema with Foreign Key Connectivity

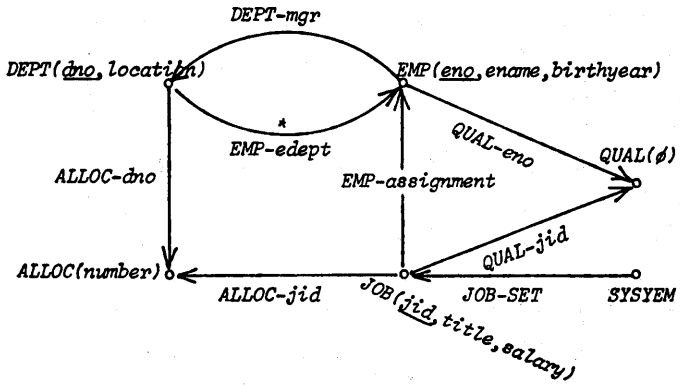

Fig. 4. The Final Graph Representation of the Sample Relational Schema

この結果得られたグラフは以下に示す解釈のもとでCODASYLスキーマを表現している。 図-4に本稿の例での最終グラフ表現を示す。

『ノードの解釈』 ノードはCODASYLのレコードと解釈する。 たとえば

```
RECORD NAME IS DEPT
  WITHIN ANY AREA
  KEY DEPT-KEY IS dno
    DUPLICATES ARE NOT ALLOWED
  02  dno
  02  location

RECORD NAME IS ALLOC
  WITHIN ANY AREA
  02  number
```

Fig. 5. Interpretation of Nodes -Examples-

```
SET NAME IS DEPT-mgr
OWNER IS EMP
ORDER IS TEMPORARY INSERTION IS SYSTEM-DEFAULT
MEMBER IS DEPT
INSERTION IS AUTOMATIC RETENTION IS MANDATORY
DUPLICATES ARE NOT ALLOWED FOR dno
SET SELECTION IS THRU DEPT-mgr
OWNER IDENTIFIED BY APPLICATION
```

Fig. 6. Interpretation of Arcs -Example-

DEPTやALLOCは図-5に示すとおりである。

『アークの解釈』 アークはCODASYLのセットと解釈する。 たとえばDEPT-mgrについては図-6に示すとおりである。 EMP-deptセットの記述は同様であるが，アークが*印付きのために，INSERTIONはMANUALとなる。 またJOB-SETの記述も同様であるが，OWNER IS SYSTEM，SET SELECTION IS THRU JOB-SET OWNER IDENTIFIED BY SYSTEMとなる点が異なる。 INSERTIONとRETENTIONの指定は同様AUTOMATICとMANDATORYである。

これら二つの解釈により図-4の最終グラフ表現は全てCODASYLスキーマ記述に変換できることになる。 この変換は情報保存である。

3.2 更新サポート

このインターフェイスにより変換され利用者に提示されるCODASYLデータベースにどのような更新操作がサポートされうるであろうかを議論する。 本節では更新機能としてとくにERASE命令をとりあげて，更新サポート能力を明らかにする。

さて，ERASE命令には補助語を使い4つの種類があり，それは次のとおりである。（ここに[ ]は0もしくは1の選択を表わす。）

ERASE [レコード名] [PERMANENT / SELECTIVE / ALL]

このうちERASE PERMANENTは実行単位の現在レコード(current of run-unit)をデータベースから消去するが，このとき二のレコードを親レコードとする子レコードのメンバーシップクラスがMANDATORY(FIXED)なら，その子レコードに再帰的にERASE PERMANENTがかかる（従ってその子レコードも消去される）ように機能する。 ERASE SELECTIVEはERASE PERMANENTと本質的に同じであるが，メンバーシップクラスがOPTIONALの子レコードの消去の取り扱い方が異なっている。 ERASE ALLは名のとおり現在レコードにぶらさがる全ての子レコードを再帰的に全て消去する。 単にERASEはセットオカレンスが空の現在レコードのみ消去する。

そこで，与えられた関係スキーマに二つの関係R(A, F)，S(B, A, E)があったとする。 このときFOREIGN KEY S.A; R.Aであり，前述のスキーマ変換アルゴリズムにより，R(A, F)を親，S(B, E)を子レコードとし，AUTOMATIC/MANDATORY属性を持つセットに変換される。 図-7に関係と対応するセットのインスタンス(オカレンス)の一例を示す。

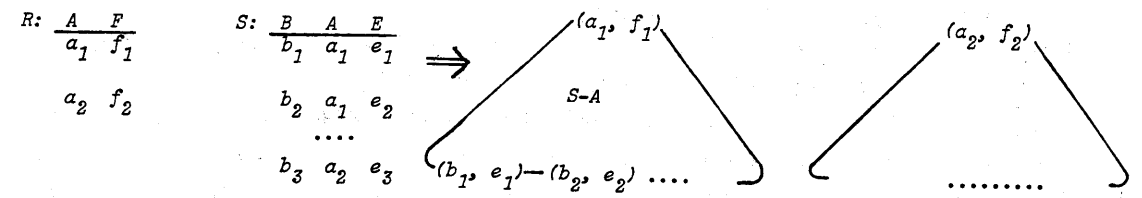

Fig. 7. A Typical Schema Translation and instances (Occurrences)

このとき関係Rからタップル (a_1, f_1) を DELETE したとする。 するとこの関係データベースに外部キー制約が課せられているので，2章で明らかにした更新波及の効果により，関係Sの A 値が a_1 を有する全てのタップル (b_1, a_1, e_1)，(b_2, a_1, e_2)，… も同時に DELETE され，この効果は再帰的にデータベース全体に及ぶ。 さて，この効果は CODASYL インターフェイスを通して，利用者に見えている CODASYL データベース上では，あたかもセット S_A の親レコード (a_1, f_1) を持つオカレンスのそのレコードに ERASE PERMANENT が発せられた場合に相当する。 この効果は他の ERASE 命令のそれではないことは容易に理解できる。 逆にサポートされている CODASYL データベースに ERASE PERMANENT を発すると，メンバーシップクラスが MANDATORY であるので，消去の効果が再帰的に波及してゆくが，この効果が基底となっている関係データベースに外部キー制約が課せられていることと，スキーマ変換の定義の仕方から，確かに実現できるものであることが判る。 つまり，具体的には同一の例を借りれば，S_A のオカレンスに発せられたレコード (a_1, f_1) の ERASE PERMANENT は関係データベースの R からのタップル (a_1, f_1) の DELETE に変換する。 もしレコード (b_1, e_1) を親レコードとするセットオカレンスでそのレコードの ERASE PERMANENT が発せられれば，それはその親レコードが A 値として a_1 を持つことを認識して，S からのタップル (b_1, a_1, e_1) の DELETE に変換される。

図-8 は以上の議論を総括したものである。 ERASE 命令を例として，更新サポートの限界が明確にされている。

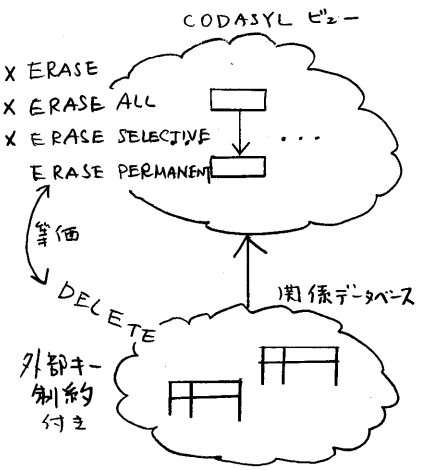

図-8. 更新サポート概念図 (ERASEの例)

4. おわりに

(1) 一般に m 対 n の関係を表わすレコード間の関係も例えば ALLOC 関係のように問題なく，リンクレコードタイプとして変換されている。

(2) スキーマ変換にあたっては，ALIAS 関係に基づき定義された外部キーの概念が重要な役割を担っている。 従って CODASYL モデルで許されている Repeating Group をサポートするには議論の拡張が必要である。

(3) ALIAS の定義は本稿では同一の属性名は同じ意味論で使用されるという前提のうえで手こずっている（このため議論が明解）が，そうでない場合には，定義の拡張が必要である。

[謝辞] これまで関連する発表にいただいた有益な御討論，御批判に深謝する。なお，本研究は文部省科学研究費の補助をえたことを付記する。

[文献] (1) 増永，多データモデルデータベースシステム構築のためのスキーマ変換基礎論，IPSJ 23回全国大会 3F-2 (1981)，(2) C. Zaniolo, Design of Relational Views ---, Proc. ACM-SIGMOD (1979)，(3) R. El-Masri et. al., Data Model Integration ---, Proc. ACM-SIGMOD (1979)，(4) E. Wong et. al., Logical Design ---, Proc. E-R Approach (1979)，(5) J. A. Larson et. al., Multimodel Data Base ---, Proc. Sperry Univac Fall Technical Symp. (1978)，(6) J. L. Jones et. al., Report of the CODASYL ---, Inf. Syst. (1978) (7) T. W. Olle, The Codasyl Approach ---, Wiley (1978) (西村・植村監訳)，(8) 増永，野口，多データモデル…，IECEAL 81-69 (1981)。