データ間の関係からデータベースおよびその利用者インタフェースを自動生成する方法について

4S−5

鈴木 卓治
電気通信大学大学院 情報工学専攻

小林 光夫
電気通信大学 情報工学科

1. はじめに

実用ソフトウェアの急激な需要増に対応するために，効率のよいソフトウェア自動作成技術が求められている．とくに，データベースを利用するシステムについての需要は高い．

データベースを含むソフトウェアシステムは，データベースの保守・更新を行なうためのデータ管理部と，データベースから情報を抽出し利用するためのデータ処理部とに大きく分けられる．これらの具体的な仕様は，システムで扱うデータ間の関係，すなわちデータモデル [1] が与えられれば，導くことができる．

本稿では，与えられたデータモデルから，データベースのスキーマおよびデータ管理のための利用者インタフェースを自動的に生成するための手法について述べる．

2. データモデルの記述

実体-関連モデル [2] では，データを実体の集合，実体の属性，実体間の関連，および関連の属性の四種類に分解し，これらの定義を組み合わせてデータモデルを記述する．たとえば学生データベースを例にとると，学生，学科，科目などの各集合を実体集合と考え，学生の氏名を属性，履修科目を（科目との）関連ととらえ，履修科目の成績を関連の属性ととらえる．

このような見方は，自然で理解しやすい半面恣意性を反映しがちであり，直接明らかではない隠れた関係を見のがすことがある．

DAPLEX[3], FQL[4], EFDM[5, 6] などに代表される関数データモデルでは，数や文字列など既定義の集合とデータモデルで定義される実体の集合とを区別せず，属性も関連もともに集合間の関数として表わす．属性や関連がもつ性質は，関係と正規形制約で表わすより，関数で表わす方が自然である．われわれは，モデルの記述に関数データモデルを使用する．

学生データベースのデータモデルの定義の例を図1に示す．

```
集合  学生; 自宅生, 下宿生 は 学生 の直和分割;
      学科; 科目;
関数  学籍番号:  学生 → 文字列(7,"0-9") は
                        単射, 全域関数;
      氏名:  学生 → 文字列 は 全域関数;
      住所:  学生 → 文字列;
      所属学科:  学生 → 学科;
      履修科目:  学生 → 科目 は 多値関数;
      帰省先:  下宿生 → 文字列;
      学科名:  学科 → 文字列 は 単射, 全域関数;
      科目名:  科目 → 文字列 は 単射, 全域関数;
      単位数:  科目 → 整数 は 全域関数;
      履修学生:  科目 → 学生 は
                        多値関数, 履修科目 の逆関数;
集合  履修 は 履修科目 のグラフ;
関数  成績:  履修 → 整数;
```

図 1: データモデルの定義の例: 学生データベース

Automatic Generation of a database schema and a data management interface from a given functional data model
SUZUKI Takuzi, KOBAYASI Mituo
University of Electro-Communications

記述の方式は，おおむね EFDM の方法に従っているが，これに全射と関数のグラフの概念を加え，さらに“単射”，“全射”および“逆関数”の概念を部分関数および多値関数へ拡張し利用している．たとえば，図1の定義では，履修科目の成績を表わす関数の定義域が学生の履修科目を表わす関数のグラフとなっている．EFDM では関連の属性を表わすのに多引数関数を用いるが，グラフによる記述のほうがより直接的である．

3. データ管理用インタフェースに対する要求

データベースの主な管理作業は，新しい実体の追加，実体の削除，実体の属性値や実体間の関連の変更である．この作業を対話的に行なうための利用者インタフェースを考える．窓システムを用いた実現を想定すると，その素朴なイメージは，たとえば図2のようになる．

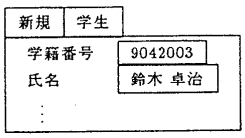

図 2: データ管理インタフェースのイメージ

一つの窓は一つの実体を表わし（これを実体窓と呼ぼう），窓の見出しには，管理作業の別（新しい実体の追加，属性値あるいは関連の変更，実体の検索，実体の指定）と，実体の属する集合の名前が表示される．窓の中には，属性および関連（すなわち関数）を表わす項目の並びが表示され，利用者は項目値の入力・変更・削除を行なうことによって，属性および関連情報の変更が可能である．画面上には複数の実体窓が表示可能であり，ある実体の追加・変更・削除の操作が他の実体の追加・変更・削除を引き起こす場合でも，これに対応することができる．項目値の入力に対しては，つぎに挙げる機能を用意する．

- 項目値として正しい集合の要素だけが入力できるようにする．
- 文字列，数，日付などに加えて，実体もまた項目値とすることができる．インタフェースはこれを支援するため，実体の検索機能や指定の機能などを備えていなければならない．
- 項目値の入力につぎのような条件をつけることができる．
  - 値はただ一つ入力可能か，複数個入力可能か（単値項目，多値項目）．
  - 項目値の入力は必須か（必須入力項目）．
  - 実体と項目値との対応が1対1か（一意対応項目）．

4. インタフェースの生成と駆動

実体集合 A の実体窓を W_A と書くことにすると，データ管理のためのインタフェース I は，実体窓の集合 $\{W_A, W_B, \cdots\}$ のことであるとしてよい．

実体窓 W のもつ項目の集合を $items(W)$ で表わし，ある項目 i の項目値として許される集合の名前を $iset(i)$ で，項目に関する制約を $icns(i)$ で，項目に対応する関数の逆関数が存在するときの逆関数に対応する項目を $invi(i)$ で，それぞれ表わすことにする．

データモデルの定義 M が与えられたとき，インタフェース I を生成するアルゴリズムはつぎのように書ける．

1. $I := \emptyset$ とする．
2. M が集合 A の定義を含んでいるなら，$I := I \cup \{W_A\}$, $items(W_A) := \emptyset$ とする．
3. M が集合 A を定義域，B を値域とする関数 f の定義を含んでいるなら，$items(W_A) := items(W_A) \cup \{f\}$, $iset(f) := B$, $icns(f) := \emptyset$ とする．
4. M が集合 A の部分集合 B の定義を含んでいるなら，$items(W_B) := items(W_B) \cup items(W_A)$ とする．
5. M が多値関数 f の定義を含んでいるなら，$icns(f) := icns(f) \cup \{$多値$\}$ とする．
6. M が全域関数 f の定義を含んでいるなら，$icns(f) := icns(f) \cup \{$必須入力$\}$ とする．
7. M が単射の関数 f の定義を含んでいるなら，$icns(f) := icns(f) \cup \{$一意対応$\}$ とする．
8. M が f の逆関数 g の定義を含んでいるなら，$invi(f) := g$, $invi(g) := f$ とする．
9. M が定義域 A, 値域 B の関数 f のグラフ G の定義を含んでいるなら，$items(W_G) := items(W_A) \cup \{f\}$, $iset(f) := A \times B$, $icns(f) := \{$必須入力, 一意対応$\}$ とする．

このアルゴリズムによって図1の定義から導かれるインタフェースを，図3に示す．

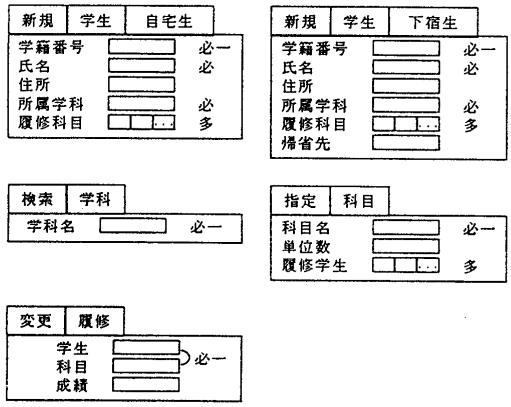

図 3: 導出されるインタフェース

実際の場面では，利用者との対話支援プログラムが情報 I を利用し，安全なデータ管理を保証する．たとえば，$f \in items(W_A)$, $g \in items(W_B)$, $invi(f) = g$のとき，A の実体 a における f の項目値が b を含んでいれば，B の実体 b における g の項目値は必ず a を含むようにデータ管理が行なわれる（逆関数の性質が保存されるようにする）．

5. データベーススキーマの生成

データをファイル上に置くことを想定して，データベースを可変長配列の集まりにより実現する．配列は集合および単値関数を表わすもの（図4(a)），多値関数およびそのグラフを表わすもの（図4(b)）とに大別される．

(a) は，学生集合および学籍番号，氏名，および住所の三つの関数を表わしている．(a) の形式の配列では，配列の添字の集合が関数の定義域である集合を表わしている．これにより関数の適用が高速に行なえる．部分集合は，配列の要素値として，部分集合に包含されていることを表わすフラグ（あるいは直和分割の各成分

学生	学籍番号	氏名	住所 F	住所
1	9042003	鈴木 卓治	T	調布市…
2	8832013	電通 太郎	F	
⋮	⋮	⋮	⋮	⋮

(a) 集合および単値関数を表わす配列

履修	学生	科目	成績 F	成績
1	1	1	T	80
2	1	3	F	
⋮	⋮	⋮	⋮	⋮

(b) 多値関数およびそのグラフを表わす配列

図 4: 可変長配列によるデータベースの実現

を表わすタグ）をもたせることで表わす．部分関数は，配列の要素値を，関数値が定義されているかどうかを表わすフラグと関数値との組とすることで表わす．全域関数ではフラグは不要である．同じ定義域をもつ関数はまとめて一つの配列として表わす．逆関数は逆引き用のデータ構造（たとえば B-Tree など）を付加することで表わす．

(b) は，履修科目とその成績の二つの関数を表わしている．(b) の形式の配列は，関係データベースにおける関係を表わす表と同じものである．多値関数の定義域側の値に対して逆引き用のデータ構造を付加すれば，関数の適用を高速化することができる．関数のグラフを用いるときは，配列の添字をグラフ集合の表現として用いることができる．また関数のグラフを定義域にもつ関数は，関数値が定義されているかどうかを表わすフラグと関数値との組を配列の要素値に付け加えることで表わすことができる．

6. おわりに

意味データモデルが実際のソフトウェア開発の場面で有効に利用でき役立つことを，具体的に示した．

今回採用したデータモデルの記法はまた洗練されておらず，改良の必要がある．また，集合値関数の導入（多値関数の廃止），定義域や値域が関数集合であるような関数の導入，既定値（デフォルト値）の設定機能の導入，桁数などの部分範囲の指定機能の導入についても検討したい．

データ処理のための利用者インタフェースの自動生成も興味深い話題である．

参考文献

[1] Hull,R., King,R.: *Semantic Database Modeling: Survey, Applications, and Research Issues*, ACM Computing Surveys, Vol.19, No.3, pp.201-260(1987).

[2] Chen,P.P.: *The Entity-Relationship Model — Toward a Unified View of Data*, ACM Transaction on Database Systems, Vol.1, No.1, pp.9-36(1976).

[3] Shipman,D.W.: *The Functional Data Model and the Data Language DAPLEX*, ACM Transaction on Database Systems, Vol.6, No.1, pp.140-173(1981).

[4] Buneman,P., Frankel,R.E., Nikhil,R.: *An Implementation Technique for Database Query Languages*, ACM Transaction on Database Systems, Vol.7, No.2, pp.164-186(1982).

[5] Kulkarni,K.G., Atkinson,M.P.: *EFDM: Extended Functional Data Model*, The Computer Journal, Vol.29, No.1, pp.38-46(1986).

[6] Gray,P.M., Kulkarni,K.G., Paton,N.W.: Object-Oriented Databases: A Semantics Data Model Approach, Prentice-Hall, 1992.