データベースシステム 101－5
（1995. 1. 26）

ビュー機能を用いた E-R モデルデータベースの操作

村田 美友紀　　掛下 哲郎
佐賀大学理工学部情報科学科

我々は、これまでの研究でRDB、OODB、入れ子RDBをビュー機能を用いて統一的に検索・操作するための枠組を提案した。ビュー機能を用いた操作は、複数のビュー間での操作(移動、追加、コピー)を定義したことで、検索機能に加えデータベースの変更も可能である。本稿では、関係論理の概念を用いて、ビューを再定義する。更に、ビュー機能を用いた操作をE-Rモデルデータベースに適用する。ビュー機能を用いたデータベース操作は、(1)複数のデータモデルを統一的に操作できる、(2)GUIに対しての親和性が高い、(3)データ一貫性が容易にチェックできる、などの利点がある。

Retrieval and Manipulation of E-R Model Database using View Function

Miyuki Murata　　Tetsuro Kakeshita

Department of Information Science, Saga University
Saga 840, Japan

large We have proposed a unified framework for data retrieval and manipulation of RDB, OODB and nested RDB using view function. A user can retrieve database entities using a view, and also he can manipulate the database using moving, adding and copying entities between two views. We update the definition of the view using the notion of relational calculus in this paper. The new view function is used to manipulate E-R model database. The view function has three advantages: (1) a user can uniformly manipulate multiple datamodels, (2) the view-based user interface is suitable for GUI, and (3) a user can easily check the database integrity.

1 はじめに

データベース (DB) 操作言語は、DB の使い易さを決定する重要な要因である。RDB には宣言型言語 SQL、OODB には手続き型言語 C++が操作言語として主に用いられる。しかし、ユーザが扱うデータモデル毎に操作言語が異なると、DB の使い易さが大幅に低下する。我々はこれまでの研究で、RDB、OODB、入れ子 RDB をビュー機能を用いて統一的に操作するための枠組を提案した [1, 2, 3]。本稿では、ビュー機能を用いた DB 操作を E-R モデル [6] の DB に適用する。ビューは DB における一般的な概念であるため、種々のデータモデルに対しても親和性が良い。このため、ビュー機能を用いた操作は、複数のデータモデルに適用できるので、種々のデータモデルが混在する連合 DB に対しても検索・変更操作が行なえる。

E-R モデル DB の操作の定義に先立ち、関係論理の概念を用いてビューを再定義する。これにより、より複雑なインスタンスの検索・変更を、ユーザが見通し良く行なえるようになる。

ビューは選択条件を満たすインスタンスを検索し、表示する機能を持つ。ビュー機能を用いた DB の変更操作は、異なるビュー間でインスタンスを操作 (移動、追加、コピー、削除) することにより実行される。ビューはウインドウ、ビュー間の操作はマウス等を用いて実現できるため、ビュー機能を用いた操作は GUI に対する親和性が高い。以下にインスタンスの入力の例を示す。

例 1 選択条件が異なるビュー $V_0, \cdots, V_3$ を定義する。ビュー V_0 でインスタンス I_1 を生成し、それを V_1 に移動すると I_1 の値は $(x, 1)$ に変更される。V_2 に対して I_1 のコピーを実行すると新しいインスタンス $I_2(=(x, 2))$ が生成される。更に V_3 へ移動することによって I_2 の値が $(y, 2)$ に変更される。なお、各ビューは選択条件だけが異なるので、V_0 のコピー及び、選択条件の修正により容易に定義できる。□

一度ビューを定義すれば、そのビューにインスタンスを移動することにより値が設定されるので、同一の値を何度も入力する場合に便利である。また、複数のインスタンスを同時に選択すると、それらを一括して操作できる。また、ビュー機能によりデータ一貫性のチェックも行なえる。

[4] は Prolog の Horn 節を用いて複合オブジェクトを検索する枠組を定義している。その検索能力は本稿の枠組と同等であるが、DB の変更操作を提供していない。[5] でもグラフとのパターンマッチングを基本操作とする複合オブジェクトの操作が提案されているが、データ操作機能が貧弱なので、DB の変更の際に複雑な操作が必要である。ビュー機能を用いた操作は、ビュー間操作を用いることによって DB の複雑な変更が可能になる。

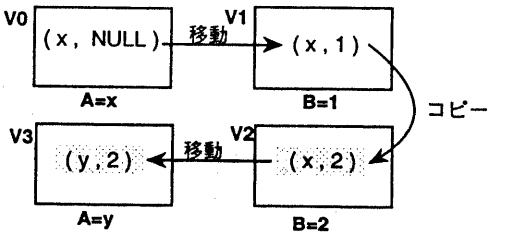

図 1: データ入力操作

本稿は次のように構成されている。2 節ではまず、種々のデータモデルを表現するための共通データモデルを定義する。次に、ビュー及びビュー間操作を定義する。3 節ではビュー機能を用いた E-R モデルデータベースの検索・変更操作を定義する。4 節では E-R モデルにおけるデータ一貫性のチェック方法をビュー機能を用いて定義する。最後に 5 節でまとめを行なう。付録ではビュー機能を用いた RDB、OODB、入れ子 RDB の操作をまとめる。

2 ビュー機能

本節では、最初に種々のデータモデルを統一的に表現するための共通データモデルを定義する。この共通データモデル上でビュー機能 (ビュー及びビュー間操作) を定義することによって、ビュー機能とデータモデルは互いに独立になる。

2.1 共通データモデル

共通データモデルのスキーマ S はノード N の有限集合である。ノードの構成要素は属性とリンクである。属性 A は基本データ型 (整数、文字列等) であり、リンク L はノードを参照するための型である。S に含まれる全ての属性の集合を $\boldsymbol{A}$、全てのリンクの集合を $\boldsymbol{L}$ と定義する。ノード N は組ノード N_T、集合ノード N_I、選択ノード N_S のいずれかである。

各ノードは以下に示す構造を持つ。

$$\begin{aligned} N_T &= <\boldsymbol{A}', \boldsymbol{L}'> \quad \boldsymbol{A}' \subseteq \boldsymbol{A}, \boldsymbol{L}' \subseteq \boldsymbol{L} \\ N_I &= <L> \quad L \in \boldsymbol{L} \\ N_S &= <\boldsymbol{L}'> \quad \boldsymbol{L}' \subseteq \boldsymbol{L} \end{aligned}$$

次にインスタンスの定義を行なう。属性Aの値はAの型に属する値、リンクLの値はLが指すノードのインスタンスの識別子の値と定義する[1]。AとLの値を次のように表す。

$$\begin{aligned} A\text{の値} &: a^0(=NULL), a^1, a^2, \cdots \\ L\text{の値} &: l^0(=NIL), l^1, l^2, \cdots \end{aligned}$$

各ノードNのインスタンスI_Nはノードの種類によって次のように定義する。

$$\begin{aligned} I_{N_T} &= <\boldsymbol{a}', \boldsymbol{l}'> \\ &\quad \boldsymbol{a}'\text{は}\boldsymbol{A}'\text{の各要素の属性値からなる組} \\ &\quad \boldsymbol{l}'\text{は}\boldsymbol{L}'\text{の各要素のリンク値からなる組} \\ I_{N_I} &= <\phi(=\{NIL\})> \text{or} <\{l^1, l^2, \cdots\}> \\ &\quad l^p\text{は}NIL\text{でない}L\text{のリンク値} \\ I_{N_S} &= <l> \\ &\quad l\text{は}\boldsymbol{L}'\text{のいずれかの要素のリンク値} \end{aligned}$$

ノードNのインスタンスI_Nの属性/リンクXの値を$I.X$で表す。スキーマSのインスタンスはSに属するノードのインスタンス全体からなる集合である。

3.2節では、この共通データモデルによってE-Rモデルを表現する。付録では、共通データモデルによってRDBやOODBも表現する。

2.2 ビューの定義

本節ではビューを定義する。これに先立ち領域変数について述べる。ノードNに対し、領域変数$D_N^1, D_N^2, \cdots$を定義する。D_N^iはNの任意のインスタンスI_N^jにマッチングできる。D_N^iがI_N^jにマッチングした時、Nの任意の属性/リンクXについて

$$D_N^i.X = I_N^j.X$$

が成立する。D_N^iの値はI_N^jの識別子の値である。

ビューは射影属性と選択条件の組によって定義される。射影属性は$D_N^i.A$で表される属性の集合であり、ビュー上に表示する属性を指定する。但しリンクは含まない。選択条件は以下に定義する論理式であり、インスタンスを検索するために用いる。

[1]ノードの各インスタンスは固有の識別子を持つ。

1. D_Nを領域変数とする。Nの属性Aについて、$D_N.A\,\theta\,c$は選択条件である。但し、θは'=', '≠', '<', '>', '≤', '≥'のいずれかである。cは定数である。

2. $D_N, D'_{N'}$を領域変数とする。N, N'は同一ノードとは限らない。N, N'の属性/リンクX, X'が比較可能である時、$D_N.X\,\theta\,D'_{N'}.X'$は選択条件である。但し、$N$が集合ノードでない時、$\theta$は'=', '≠', '<', '>', '≤', '≥'のいずれかである。またNが集合ノードである時、θは'=', '≠', '⊂', '⊃', '⊆', '⊇'のいずれかである。

3. ノードNのリンクLがノードN'を参照している時、$D_N.L\,\varphi\,D_{N'}$は選択条件である。但し、Nが集合ノードでない時、φは'=', '≠'のいずれかである。また、Nが集合ノードである時、φは'∋', '∌'のいずれかである。

4. ノードNの2つの領域変数D_N, D'_Nについて、$D_N\,\varphi\,D'_N$は選択条件である。但し、φは'=', '≠'のいずれかである。

5. 領域変数D_NとNの属性/リンクXについて、$D_N.X$='undefined'は選択条件である。

6. 'true'は選択条件である。

7. 式F_1とF_2が選択条件ならば、$(F_1 \wedge F_2)$、$(F_1 \vee F_2)$、$\neg F_1$は選択条件である。

ここで条件1, 2は、$D_N, D'_{N'}$がマッチングしたインスタンス$I_N, I_{N'}$の属性/リンクの値や、リンク値の集合が条件を満たす場合に真となる。条件3はD_NがマッチングしたインスタンスI_Nのリンク値の集合と、$D_{N'}$がマッチングしたインスタンス$I_{N'}$の識別子が条件を満たす場合に真となる。条件4はD_N, D'_NがマッチングしたインスタンスI_N, I'_Nの識別子が条件を満たす場合に真となる。条件5はD_NがマッチングしたインスタンスI_Nについて、Xが属性ならば$I_N.X = NULL$の場合に真となり、Xがリンクならば$I_N.X = NIL$ (Nが集合ノードの場合には$I_N.X = \phi$) の場合に真となる。条件6は常に真となる。

ビューVはVに属するインスタンスについて、射影属性で指定された属性値を表示する。ビューに属するインスタンスとは、ビューの選択条件を満たし、各領域変数にマッチングするインスタンスの集合で

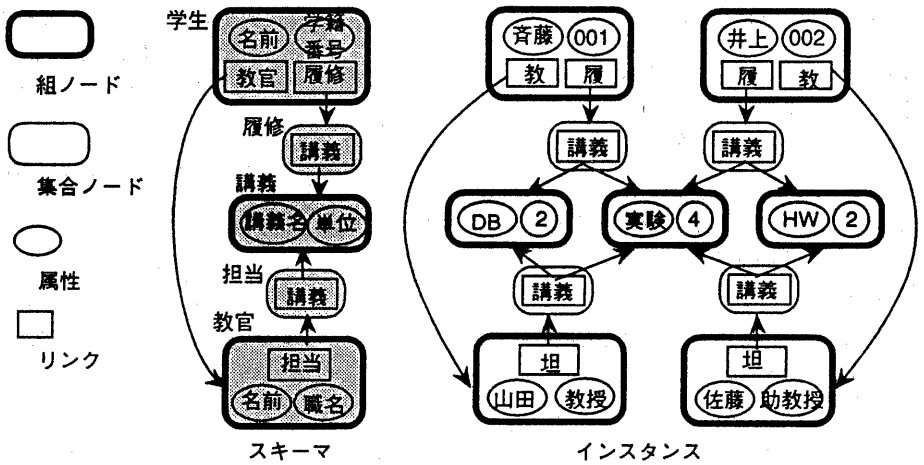

図 2: スキーマとインスタンスの例

ある。ビューはシステムによって識別子を割り当てられ、必要に応じて固有のビュー名を持つことができる。ビュー Vの選択条件を $SC(V)$、射影属性の集合を $PA(V)$ と表記する。また、$SC(V)$ または $PA(V)$ に属する領域変数の集合を $DV(V)$ と呼ぶ。以下にビュー定義の例を示す。

例 2 図2のスキーマとインスタンスを考える。次のビュー V_4を定義する。

$$DV(V_4) = \{D_S\text{: 学生}, D_L\text{: 履修}, D_C\text{: 講義}\}$$
$$SC(V_4) = (D_S.\text{名前}=\text{'斉藤'}) \wedge (D_S.\text{履修} = D_L) \wedge (D_L.\text{講義} \ni D_C)$$
$$PA(V_4) = \{D_C.\text{講義名}\}$$

V_4は、学生 '斉藤' が履修する講義名 '実験' と 'DB' を表示する。 □

2.3 移動

移動操作は、インスタンスの属性/リンクを変更するために利用される。ビュー V_s, V_dを定義し、V_s に属するインスタンス Iを V_dに移動する。この時、$D_N \in DV(V_s) \cap DV(V_d)$ を満たす領域変数 D_Nがマッチングするインスタンス $I_N(\in I)$ だけが、変更の対象となる。移動操作により、I_Nの属性/リンク Xの値が $SC(V_d)$ を満足するように変更される。$SC(V_d)$ により変更後の $I_N.X$の値が唯一に決定されない場合には、移動操作は拒否される。$I_N.X$がリンクの集合である場合には、$SC(V_s)$ を満たすリンク値だけが $SC(V_d)$ を満足するように変更される。以下に移動操作の例を示す。

例 3 図2のスキーマとインスタンスを考える。ビュー V_5, V_6を次のように定義する。射影属性は、変更結果に影響を与えないので省略する。

$$DV(V_5) = \{D_S\text{: 学生}, D_T^1\text{: 教官}, D_L\text{: 履修}, D_C^1\text{.講義}\}$$
$$SC(V_5) = (D_S.\text{教官} = D_T^1) \wedge (D_T^1.\text{名前}=\text{'山田'}) \wedge (D_S.\text{履修} = D_L) \wedge (D_L.\text{講義} \ni D_C^1)$$

$$DV(V_6) = \{D_T^2\text{: 教官}, D_K\text{: 担当}, D_C^2\text{: 講義}, D_L\text{: 履修}\}$$
$$SC(V_6) = (D_T.\text{名前}=\text{'佐藤'}) \wedge (D_T.\text{担当} = D_K) \wedge (D_K.\text{講義} \ni D_C^2) \wedge (D_L.\text{講義} \ni D_C^2)$$

V_5に属するインスタンスを V_6に移動する。この時変更対象となるのは、D_Lがマッチングするインスタンス (図 3の破線で囲まれた部分) である。移動操作により、教官 '山田' が指導する学生は、教官 '佐藤' が担当する講義を全て履修するように変更される。 □

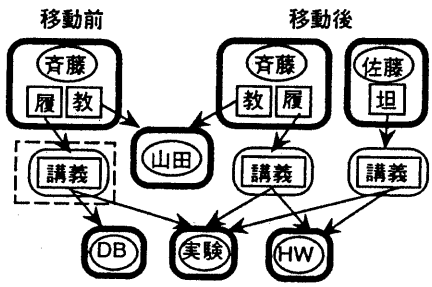

図 3: インスタンスの移動

属性 A を空値に設定する操作やリンク L を切断する操作も、選択条件 $D_N.A$='undefined' または、$D_N.L$='undefined' を持つビューへの移動操作によって実行できる。

2.4 追加

追加操作は、集合インスタンスに新たなリンク値を追加するために利用される。ビュー V_s, V_d を定義し、V_s に属するインスタンス I を V_d に追加する。追加操作は次の二条件が満足された時にのみ実行される。(1) $D_N \in DV(V_s) \cap DV(V_d)$ を満たす集合ノード N の領域変数 D_N がマッチングするインスタンス $I_N(\in I)$ が存在する、(2) スキーマ中で N のリンク L が参照するノード N' の領域変数 $D_{N'}$ が、$DV(V_d)$ に含まれる。V_d 中で $D_{N'}$ がマッチングするインスタンスの識別子の集合を ID とする。追加操作によって $I_N.L \cup ID$ が新しい $I_N.L$ の値になる。以下に追加操作の例を示す。

例 4 図 2のスキーマとインスタンスを考える。次のビュー V_7, V_8 を定義し、V_7 に属するインスタンスを V_8 に追加する。

$DV(V_7) = \{D_T^1$: 教官, D_K^1: 担当 $\}$
$SC(V_7) = (D_T$. 名前='山田') $\wedge (D_T$. 担当$=D_K)$

$DV(V_8) = \{D_T^2$: 教官, D_K^1, D_K^2: 担当, D_C: 講義 $\}$
$SC(V_8) = (D_T^2$. 名前='佐藤') $\wedge (D_T^2$. 担当$=D_K^2)$
$\wedge (D_K^2$. 講義 $\ni D_C) \wedge (D_K^1$. 講義 $\ni D_C)$

この時追加の対象になるのは、D_K^1 がマッチングするインスタンス (図 4の破線で囲んだ部分) である。追加操作により、教官 '山田' が担当する講義に教官 '佐藤' が担当する全ての講義が追加される。なお、同一のインスタンスは定義により重複して追加されないため、図では講義 'HW' だけが新たに参照されている。 □

2.5 コピー

コピー操作は任意のインスタンスをコピーして、新しいインスタンスを生成するために利用される。ビュー V_s, V_d を定義し、V_s に属するインスタンス I を V_d にコピーする。この時、$D_N \in DV(V_s) \cap DV(V_d)$ を満たす領域変数 D_N がマッチングするインスタンス $I_N(\in I)$ だけが、コピー操作の対象となる。コピー操作により、I_N と属性/リンク X の値が等しいインスタンス I'_N が生成される。I_N がインスタンス $I_M(\in I)$ を参照しており、I_M もコピー操作の対象である場合は、I'_N の対応リンクも I'_M を参照するように変更される。以上の手続きで生成された I'_N は、$SC(V_d)$ を満足するように変更される。$SC(V_d)$ により変更後の $I_N.X$ の値が唯一に決定されない場合には、コピー操作は拒否される。以下にコピー操作の例を示す。

例 5 図 2のスキーマとインスタンスを考える。ビュー V_9, V_{10} を定義し、V_9 に属するインスタンス I を V_{10} にコピーする。

$DV(V_9) = \{D_S$: 学生, D_L: 履修, D_C: 講義 $\}$
$SC(V_9) = (D_S$. 名前='斉藤')$\wedge (D_S$. 履修$=D_L)$
$\wedge (D_L$. 講義 $\ni D_C)$
$DV(V_{10}) = \{D_S$: 学生, D_L: 履修 $\}$
$SC(V_{10}) = (D_S$. 名前='森')$\wedge (D_S$. 学籍番号='003')
$\wedge (D_S$. 履修$=D_L)$

コピー操作により、D_S, D_L がマッチングする新しいインスタンス (図 5の破線で囲んだ部分) が新しく生成される。生成されたインスタンス I' は、学生 '斉藤' が履修する講義と同一の講義を履修する。また、担当教官も '斉藤' と同一である。 □

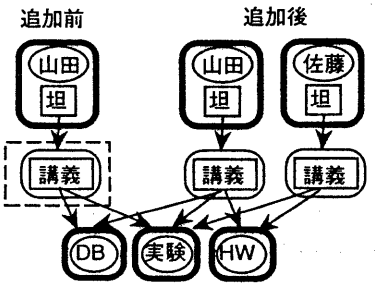

図 4: インスタンスの追加

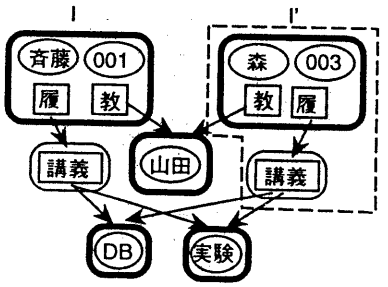

図 5: インスタンスのコピー

2.6 生成

生成操作は、任意のインスタンスを生成するために利用される。ビュー V上で生成操作を実行すると、$DV(V)$ の各領域変数 D_Nに対して新しいインスタンス I_Nが生成される。I_Nの属性/リンク Xの値は $SC(V)$ を満足するように設定される。$SC(V_d)$ で指定されない属性値には $NULL$、リンク値には NIL (リンク値が集合の場合はϕ) が設定される。この時、$SC(V)$ により $I_N.X$が唯一に決定されない場合には、生成操作を拒否する。

2.7 削除

削除操作は任意のインスタンスを削除するために利用される。ビュー V_sと削除操作のための特別なビュー V_Oを定義する。V_Oは領域変数のみで定義される。V_sに属するインスタンス Iを V_Oに移動する。この時、$D_N \in DV(V_s) \cap DV(V_O)$ を満たす領域変数 D_Nがマッチングするインスタンス $I_N(\in I)$ が削除の対象となる。削除操作により、I_Nは削除される。これにより、選択したインスタンスの任意の部分インスタンスが削除できる。削除されるインスタンス I_Nを参照するインスタンスが存在する時、削除操作を拒否する。以下に削除操作の例を示す。

例 6 図 2のスキーマとインスタンスを考える。ビュー V_{11}と削除のためのビュー V_Oを定義する。V_{11}に属するインスタンスを選択し、V_Oに移動する。

$DV(V_{11}) = \{D_S$: 学生, D_L: 履修, D_C: 講義 $\}$
$SC(V_{11}) = (D_S$.履修=$D_L) \wedge (D_L$.講義 $\ni D_C)$
$\wedge (D_C$.講義名='HW')
$DV(V_O) = \{D_S$: 学生, D_L: 履修 $\}$

削除操作により、D_S, D_Lがマッチングするインスタンス (図 6において破線で囲まれた部分) が削除される。 □

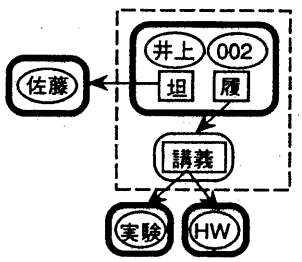

図 6: インスタンスの削除

3 E-R モデルデータベースの操作

3.1 E-R モデルデータベース

E-R モデル [6] は以下に示す 3 つの長所を持つ。(1) 実体や関連を含む実世界の情報を自然な形で表現できる、(2) 実世界の意味的情報を表現できる、(3) データ独立性を持つ。E-R 図はデータベース設計用ツールとしてしばしば用いられる。E-R モデルの実体及び関連は次のように定義される。

実体は人、会社など正確に識別できるものである。実体は実体集合を用いてクラス化される。実体集合は互いに素でなくてよい。実体は属性を持つことができ、各属性には単純値が対応する。実体を唯一に決めるための属性集合をキーと呼ぶ。

関連を用いることによって親子などの実体間のつながりを表現できる。関連は関連集合を用いてクラス化される。関連は属性及び役割を持つことができる。役割は役割名を引数とし、実体を返す関数である。関連を用いて実体間の対応関係を定義できる。ここで、各実体に対応できる実体数の上限や下限を指定することもできる。関連を唯一に決めるための属性及び役割の集合をキーと呼ぶ。

E-R 図の例を図 7に示す。実体集合 '学生' は属性 '学籍番号'、'名前'、'学年' を持ち、実体集合 '講義' は属性 '講義名'、'単位' を持つ。また、実体集合 '教官' は属性 '名前'、'職名' を持つ。関連集合 '履修' は属性として '成績' を持ち、役割として '学生' と '講義' を持つ。関連集合 '担当' は役割として '教官' と '講義' を持つ。図中の N, M は学生と講義、教官と講義が多対多の対応関係を持つことを示している。

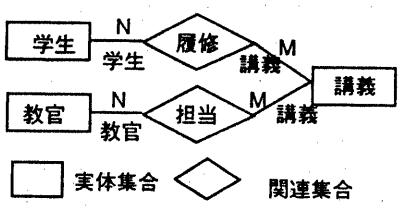

図 7: E-R 図の例

3.2 共通データモデルによる E-R モデルの表現

E-R モデルと共通データモデルの対応を表 1に示す。

E-R モデル	共通データモデル
実体集合	組ノード (リンクを含まない)
関連集合	組ノード
実体/関連	組ノードのインスタンス
属性	属性
役割	リンク

表 1: E-R モデルと共通データモデルの対応

図 7で示したE-R 図は共通データモデルでは図 8 のように表すことができる。

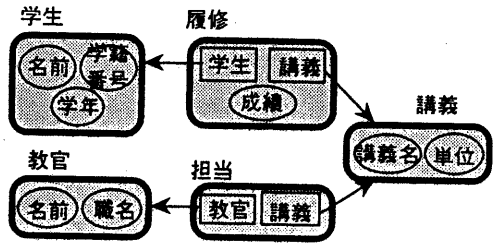

図 8: 共通データモデルによる E-R 図の表現

3.3 検索

E-R モデル DB の検索はビューの定義によって行なう。射影属性でビューに表示する属性を指定し、選択条件で実体/関連を検索するための条件を指定する。

3.4 属性/役割の変更

属性や役割は、共通データモデルの属性やリンクに対応するので、移動操作によって変更できる。移動元のビュー V_sで変更したい実体/関連を検索し、移動先のビュー V_dの選択条件で変更したい属性/役割を指定する。実体/関連を V_sから V_dに移動すると、$SEL(V_d)$ を満たすように属性/役割の値が変更される。なお、E-R モデル DB は、属性値としては単純値のみをとるため追加操作は必要ない。

3.5 実体/関連の生成

実体/関連の生成にはビュー機能の生成操作による方法とコピー操作による方法がある。ビュー V上で生成操作を実行すると、$SEL(V)$ を満足するような属性/役割の値を持つ実体/関連が生成される。$SEL(V)$ で指定されていない属性には $NULL$、役割には NIL が設定される。

コピー操作による生成は次のように行なわれる。ビュー V_sに属する実体/関連を他のビュー V_dにコピーすると、新しい実体/関連が生成される。新しい実体/関連の属性/役割は $SC(V_d)$ を満足するように変更される。

3.6 実体/関連の削除

実体/関連の削除は削除操作によって行なう。ビュー V_sに属する実体/関連を選択し、削除のためのビュー V_Oに移動すると $DV(V_s) \cap DV(V_O)$ に属する領域変数がマッチングする実体/関連が削除される。削除される実体が、いずれかの関連の役割に対応していれば削除操作は拒否される。このような実体を削除する場合には削除操作に先立ち、役割に対応するリンクを切断しなければならない。

4 データー貫性のチェック

E-R モデル DB のデータ一貫性は実体/関連のキーに対するものと、実体間の関連の対応に対するものがある。データの一貫性をチェックするために、ビュー Vに属するインスタンスの数を返す関数を $Count(V)$ と定義する。

4.1 キーチェック

キーが同一の値を持つ実体/関連は 2 つ以上存在しない。変更または生成操作の対象となる属性がキーである時、同一の属性値を持つ実体/関連がすでに存在していれば、その変更や生成は拒否されなければならない。ここで、キーチェックのためのビュー V_Kを定義する。V_Kは移動/コピー先のビュー V_d、または生成操作を行なうビュー Vの選択条件のうち、キーに対応する条件だけを選択条件とするビューである。V_Kに対し、$Count(V_K) > 1$ であれば、変更または生成操作を拒否する。V_Kは、変更または生成操作の時に自動的に定義され、表示されないビューである。

4.2 対応チェック

各実体 Eに対応できる実体数の上限や下限を関連毎に定義できる。この制約を満たさないような役

割の変更や関連の生成は拒否される。ここで、対応チェックのためのビューV_Cを定義する。$SC(V_C)$は、実体EとEを役割rによって参照する関連Rを用いて、以下のように定義される。

$$SC(V_C) = (R.r = E) \wedge (E\text{を指定する条件})$$

$DV(V_C)$は$SC(V_C)$に用いられる領域変数の集合である。このようなV_Cを定義すると、実体Eを参照する関連の数が$Count(V_C)$で与えられる。これを用いて、任意の対応関係がチェックできる。V_Cは、変更または生成操作の時に自動的に定義され、表示されないビューである。

5 おわりに

本稿では共通データモデルとビューの定義を、関係論理の概念を導入して変更した。これにより、連結していないインスタンスや巡回構造のインスタンスなどを検索、変更できるようになった。また、OODBにおける巡航操作も容易に表現できるようになった。本稿で定義したDBの操作は基本的なものであるが、これらの操作の組合せにより、さらに複雑な操作も定義できる[1]。

本稿では、ビュー機能を用いた操作をE-Rモデル DBに適用した。本稿では、インスタンスレベルでの検索・変更操作について議論したが、ノードにマッチングする領域変数を考えることによって、スキーマレベルでの操作にもビュー機能が利用できる。また、ビュー機能を用いて種々のデータ一貫性制約の検査を行なうことが可能である。これらの詳細な議論は今後の課題である。

参考文献

[1] 掛下, "ビュー機能を用いた構造化オブジェクトの段階的検索と編集操作", Proc. ADBS'93, pp.93–102, 1993.

[2] 掛下, 村田, "ビュー機能を用いたRDBとOODBの操作", 情報処理学会 DB研究会 99–3, 1994.

[3] 村田, 掛下, "ビュー機能を用いた入れ子関係データベースの操作", 電気関係学会九州支部連合大会 1227, 1994.

[4] F. Bancilhon and S. Khoshafian, "A calculus for complex object", Proc. ACM PODS, pp.53–59, 1986.

[5] M. Gyssens, et al., "A graph-oriented object model for database end-user interface", Proc. SIGMOD, pp.24–33, 1990.

[6] P. P. Chen, "The Entity-Relationship model— toward a unified view of data", *ACM Trans. on Database Syst.*, Vol.1, No.1, pp.9–36, 1976.

A 変更したビュー機能を用いた他のデータベースの操作

本付録では、本稿で再定義したビュー機能を用いた操作をRDB、OODB、入れ子RDBの操作に適用する。

A.1 RDBの操作

RDBの関係スキーマには属性のみを持つ組ノードが対応する。組には組ノードのインスタンスが対応する。属性には属性が対応する。RDBの検索はビュー定義によって行なう。ビューの選択条件で制約(選択を含む)と結合、射影属性で射影が表現できる。従って、ビュー機能による検索は関係完備性を持つ。属性値の変更は移動操作によって行なう。組の生成は生成操作またはコピー操作によって行なう。組の削除は削除操作によって行なう。キーのチェックは4.1節で示した手法によって行なえる。

A.2 OODBの操作

OODBと共通データモデルの対応を以下に示す。クラスには組/集合ノードの集合(連結グラフ構造)が対応する。オブジェクトにはノード集合のインスタンスが対応する。また、インスタンス変数には属性またはリンクが対応する。複合オブジェクトの検索もビュー定義によって行なう。属性やリンクの変更は移動操作及び追加操作(集合オブジェクトの場合)によって行なう。オブジェクトの生成は生成操作またはコピー操作によって行なう。オブジェクトの削除は削除操作によって行なう。

A.3 入れ子RDBの操作

入れ子RDBの関係スキーマには組/集合ノードからなる集合(木構造)が対応する。組にはノード集合のインスタンスが対応する。属性には属性が対応する。入れ子RDBの組は複合オブジェクトと考えることができるため、A.2節で示した操作によって検索、変更等を行なうことができる。また、キーのチェックは4.1節で示した手法によって行なえる。