

オブジェクト指向データベースにおける参照によるオブジェクトの関連

九州大学大型計算機センター 古川哲也 (Tetsuya Furukawa)

九州大学工学部 上林彌彦 (Yahiko Kambayashi)

1. まえがき

計算機の利用分野の広がりにともない、従来のデータベースの機能の拡張が求められている。オブジェクト指向モデルは、利用者インタフェースとしての使い易さやデータの表現能力の豊かさから、データベースと応用プログラムを統合したオフィスシステムなどに適していると考えられている。本稿では、参照によってオブジェクト間の関連を表すことによるオブジェクト指向データベースの設計に関する問題について議論する。

オブジェクト指向データモデルは、実世界のもの（オブジェクト）をデータベース内のオブジェクトと1対1に対応させてデータを記憶するものである。各オブジェクトには適用可能なメソッドが定義されており、

- (1) 実世界とシステム内のデータの直接対応であるオブジェクト同一性
- (2) 複雑な構造を持つオブジェクト（複合オブジェクト）の表現
- (3) メソッドによるデータと手続きの一体化
- (4) 汎化（IS-A）階層

などの特徴がある。しかし、データベースの基本的な機能は、大量のデータを正しく蓄えることであり、いかにデータ間の関連を表現するかが重要となる。

オブジェクト指向データベースに関する研究では、関係データベースのオブジェクト指向への拡張¹⁾や、Smalltalkなどのオブジェクト指向プログラミング言語によるデータベースシステム²⁾、CADなどの応用のためのシステム³⁾など、システム開発が中心⁴⁾であった。モデルに対する共通の認識がなく、そのため、オブジェクト指向モデルに対する形式的な議論が望まれ⁵⁾、モデルの形式的な定義⁶⁾やメソッドに対する議論⁷⁾なども行われている。

本稿では、参照を用いたオブジェクト間の関連の表現に注目し、オブジェクト

指向モデルの特徴とそれらの参照との関係から、データベースの設計時に考慮すべき問題点について議論する。データベースの設計基準として、冗長性の削減、意味制約の保持、質問処理の効率化がある。オブジェクトに値を持たせるより、その値については他のオブジェクトを参照する方が、冗長性や意味制約の保持に有効である。そのため、参照によるオブジェクトの値の表現について形式的な定義を行い、意味制約として与えられた関数従属性を保持する方法を示す。

2. 基本的事項

オブジェクトの値とオブジェクト間の関連を、参照を用いて次のように定義する。

n 個の定義域を $D_1, D_2, \dots, D_n (n \geq 1)$, 識別子と呼ばれるシンボルの集合を ID とする。 D_i の要素, ID の要素の組 $\langle id_1, id_2, \dots, id_m \rangle$, ID の要素の集合 $\{id_1, id_2, \dots, id_m\}$ を値という。オブジェクト o は, 識別子 id , 値 v , メソッドの集合 m の組 $o(id, v, m)$ である。 (id, v, m) は, 識別子 id のオブジェクトは値 v を持ち, メソッドの集合 m を適用できることを表す。 $id(o)$ をオブジェクト o の識別子, $vl(o)$ を o の値, $mt(o)$ を o のメソッドとする。 $vl(o)$ が定義域の要素 d のとき, $id(o)$ を d で表すこともある。また, $ob(id)$ で識別子が id であるオブジェクトを表す。本稿では, オブジェクトの参照関係に注目するため, $o(id, v, m)$ を単に識別子 id と値 v で表す。

オブジェクトのクラス R は, 構造 S とメソッド集合 M からなり, $R(S, M)$ で表す。 S は, 基本構造 D , 組構造 $\langle A_1:R_1, A_2:R_2, \dots, A_n:R_n \rangle$, 集合構造 $A:\{Q\}$ のいずれかである。 D は定義域名, A_i, A は属性名, R_i, Q はクラス名である。このとき, 属性 $A_i(A)$ はクラス $R_i(Q)$ を参照するといひ, R が参照するクラスの集合を $Ref(R)$ で, 参照の閉包を $Ref^+(R) = \{Q \mid Q=R \text{ または } Q \in Ref^+(P), P \in Ref(R)\}$ で表す。 $\{A_1, A_2, \dots, A_n\}$, A をクラスの属性集合といひ $at(R)$ で, R のメソッド集合 M を $mt(R)$ で表す。また, クラスの集合 \mathbf{R} に対して, $at(\mathbf{R}) = \{A \mid A \in at(R), R \in \mathbf{R}\}$, $mt(\mathbf{R}) = \{m \mid m \in mt(R), R \in \mathbf{R}\}$ とする。

クラス間の汎化階層を, $ISA(Q, R)$ (R はクラス Q を汎化したもの) で表す。汎化には次の性質がある。

(1) メソッドの継承: $mt(Q) \supseteq mt(R)$

QはRの特殊なものであり、Rのオブジェクトに適用可能なメソッドは、Qのオブジェクトにも適用できる。

(2) 属性の継承: $at(Q) \supseteq at(R)$

Rのオブジェクトが属性Aの値を持てば、Qにも属性Aが定義されQのオブジェクトもAの値を持つ。

データベースのスキーマSは、構造が組または集合であるクラスの集合Cとクラス間の汎化階層の集合ISAであり、 $S(C, ISA)$ で表す。属性Aがデータベーススキーマ中ですべて同じクラスRに対応づけられているときは、AをクラスRで表すこともある。

スキーマ $S(C, ISA)$ で、クラス $R(S, M) (\in C)$ の状態は、Rのオブジェクトの集合である。各定義域 D_i に対し、Sが基本構造 D_i であるクラスが存在する。このクラスのオブジェクトの値は、定義域 D_i の要素である。Sが組構造 $\langle A_1:R_1, A_2:R_2, \dots, A_n:R_n \rangle$ であることは、このクラスのオブジェクトの値は組 $\langle id_1, id_2, \dots, id_n \rangle$ (id_i はクラス R_i のオブジェクトの識別子)であることを意味する。Sが集合構造{Q}のとき、このクラスのオブジェクトの値は集合値 $\{id_1, id_2, \dots, id_n\}$ (id_i はクラスQのオブジェクトの識別子)である。

スキーマが $S(C, ISA)$ であるデータベースの状態 ρ は、クラス $R (\in C)$ の状態の和集合、即ちデータベース中のオブジェクトの集合である。

スキーマSと状態 ρ で、オブジェクトoの属性Aの値を $A(o)$ で、属性集合Xの値を $X(o) = \{A(o) \mid A \in X\}$ で表す。また、オブジェクトoのクラスの属性集合を $at(o)$ で、オブジェクト集合 \circ の属性集合を $at(\circ) = \{at(o) \mid o \in \circ\}$ で表す。

クラスRのオブジェクトoに適用可能なメソッドの集合は、 $mt(o) \cup mt(R)$ である。mt(o)とmt(R)に同じ名前のメソッドが存在するときには、名前によるメソッドの実行はmt(o)のメソッドが優先する。

オブジェクトの参照関係を、参照グラフ $G(V, E)$ で表現する。Vはクラスと1対1に対応する節点の集合、Eは節点間の有向枝の集合であり、 R_1 が R_2 を参照する属性を含むとき、 R_1 から R_2 へ向かう枝が存在する。簡単のため、参照グラフは閉路を持たないと仮定する。

節点 R_1, R_2 で、枝 (R_1, R_2) が存在するかまたは R_1 から R_3 、 R_3 から R_2 へ到達可能な節点 R_3 が存在するとき、 R_1 は R_2 へ到達可能であるという。 $Ref^+(R)$ はRおよびRか

ら到達可能な節点のクラスの集合である。 $\text{Ref}^-(R)$ を R へ到達可能な節点の集合 $\{Q \mid R \in \text{Ref}^+(Q)\}$ とする。 クラス R のオブジェクトは、 $\text{Ref}^+(R)$ に含まれるの基本構造クラスのオブジェクトの値で特徴づけられ、 それらの値は $\text{Ref}^-(R)$ のオブジェクトを特徴づける値の一部となっている。

クラス R の基本構造のクラスを参照する属性集合を $\text{at}_b(R)$ 、 クラス集合 \mathbf{R} に対し、 $\text{at}_b(\mathbf{R}) = \{A \mid A \in \text{at}_b(R), R \in \mathbf{R}\}$ とする。

[例1] 図1(a), (b) はそれぞれスキーマ、 状態の例である。 id_{ij} は識別子である。 属性が基本構造を参照している場合は、 識別子ではなく、 その値（野球、 南野、 工など）で表している。 例えば、 識別子 id_{31} のオブジェクトは学生名が“南野”で、 識別子が id_{21} のオブジェクトのクラブに所属しており、 学部名は“工”、 学科名は“情報工”である。 □

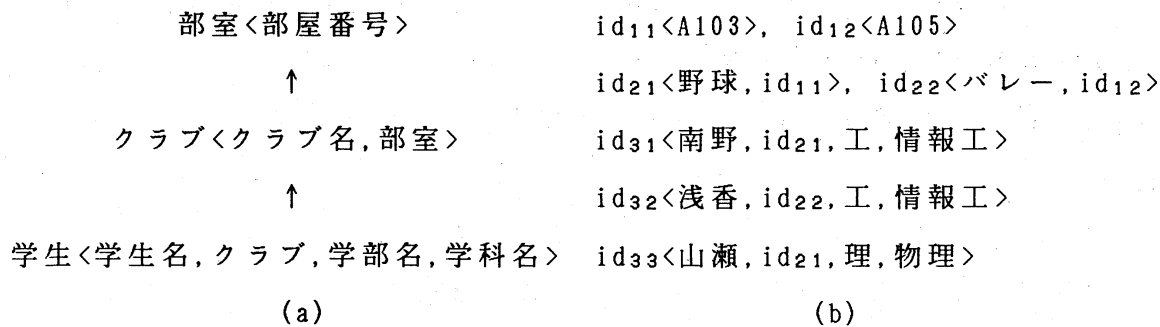


図1 スキーマと状態

3. 参照によるオブジェクトの等価性

データベース内でのオブジェクトの値の表現法として、 すべて基本オブジェクトを参照する方法と、 構造オブジェクトの参照を許す方法がある。 値 a, b, c, d を持つオブジェクト o は、

(1) $\text{id}_1\langle a, b, c, d\rangle$

(2) $\text{id}_1\langle a, b, \text{id}_2\rangle, \text{id}_2\langle c, d\rangle$

などのように表現できる。 値の構造（入れ子など）が本質的でないとき、 (1)、 (2) の id_1 はオブジェクト o の表現については同等である。 (2) で id_2 のオブジェクトの存在は、 データベースのビューとして id_2 を認識する必要があるかどうかの問題である。

値の構造が本質的な違いとなるとき、どの属性値を構造オブジェクトの参照にするかによってオブジェクトが正しく表現されないことがある。オブジェクト o_1 が、(2)のように $\langle c, d \rangle$ を入れ子とする構造のとき、

$$(3) \text{id}_1 \langle a, \text{id}_3, d \rangle, \text{id}_3 \langle b, c \rangle$$

はそのようなオブジェクトを表現できておらず、(1)も表現できていないとすべきである。

(2)と(3)は、(1)のオブジェクトの内部表現とすることができる。次に、参照を用いたオブジェクトの表現を定義し、参照に関するスキーマの性質を示す。

オブジェクトの表現法はスキーマにより記述される。スキーマ S_1 のオブジェクトをスキーマ S_2 のオブジェクトが表現していることを次のように定義する。

[定義1] スキーマ $S_1(C_1, ISA_1)$, $S_2(C_2, ISA_2)$ とその状態 ρ_1, ρ_2 で、 ρ_1, ρ_2 でのオブジェクト o_1, o_2 について、以下の条件を満たすとき、 o_2 は o_1 を表現するといひ、 $o_1 < o_2$ で表す。

(1) o_1, o_2 はともに基本構造オブジェクトであり、その値は等しい

または、

$$(2) \text{at}(o_1) = \{A_i (1 \leq i \leq n), B_{jk} (1 \leq j \leq m, 1 \leq k \leq n_j)\},$$

$$\text{at}(o_2) = \{A_i (1 \leq i \leq n), B_j (1 \leq j \leq m), C_k (1 \leq k \leq l)\} \text{で、}$$

$$(2-1) \text{ob}(A_i(o_1)) < \text{ob}(A_i(o_2)) \quad (1 \leq i \leq n)$$

(2-2) o_1 の B_{jk} の値から新たにオブジェクトを作ると、

$$\langle B_{j1}:B_{j1}(o_1), B_{j2}:B_{j2}(o_1), \dots, B_{jn_j}:B_{jn_j}(o_1) \rangle$$

$$< \text{ob}(B_j(o_2)) \quad (1 \leq j \leq m) \quad \square$$

(2-1)は、 o_1 が参照するオブジェクトを表現しているオブジェクトを o_2 が参照していることを、(2-2)は、 o_1 の属性集合 $\{B_{jk} (1 \leq k \leq n_j)\}$ の値は、 o_2 では B_j の属性値で参照しているオブジェクトで表現されていることを示している。

オブジェクトの表現には次の性質がある。

[補題1] スキーマ $S_1(C_1, ISA_1)$, $S_2(C_2, ISA_2)$ とその状態 ρ_1, ρ_2 で、 ρ_1, ρ_2 のオブジェクト o_1, o_2 と o_1, o_2 のクラス R_1, R_2 について $o_1 < o_2$ ならば、 $\text{at}_b(\text{Ref}^+(R_1)) \subseteq \text{at}_b(\text{Ref}^+(R_2))$. \square

(証明) $o_1 < o_2$ の判定は定義より, $\text{Ref}^+(R_2)$ 以外のクラスには無関係であり, $\text{at}(\text{Ref}^+(R_2))$ 以外の属性を用いない. 属性 $A \subseteq \text{at}_b(\text{Ref}^+(R_1))$ で $A \not\subseteq \text{at}_b(\text{Ref}^+(R_2))$ となるものがあれば, $\text{at}(\text{Ref}^+(R_2)) \subseteq \text{at}_b(\text{Ref}^+(R_2))$ より, A は再帰的な定義のどのステップでも (1)には適用できない. $o_1 < o_2$ であれば $\text{at}_b(\text{Ref}^+(R_1))$ のすべての属性は(1)で用いられるので, $A \subseteq \text{at}_b(\text{Ref}^+(R_1))$ ならば $A \subseteq \text{at}_b(\text{Ref}^+(R_2))$, 即ち, $\text{at}_b(\text{Ref}^+(R_1)) \subseteq \text{at}_b(\text{Ref}^+(R_2))$. (証明終り)

スキーマ S_1 のいかなる状態もスキーマ S_2 のある状態で表現できるかどうかを判定する. 即ち, S_2 は S_1 の他の表現法であることの判定である. そのために, まず S_1 のクラスに対応するクラスが S_2 に存在することを定義し, S_1 の状態が S_2 の状態では表現されていることを定義する.

[定義2] スキーマ $S_1(C_1, ISA_1)$, $S_2(C_2, ISA_2)$ で, C_1 から C_2 への写像 f は, 1対1の全写像, 即ち S_1 のすべてのクラス $R (\in C_1)$ について S_2 のクラス $f(R) (\in C_2)$ が存在し, $R_1 \neq R_2$ ならば $f(R_1) \neq f(R_2)$ のとき, S_1 から S_2 へのクラス写像であるという. \square

[定義3] スキーマ $S_1(C_1, ISA_1)$, $S_2(C_2, ISA_2)$ の状態 ρ_1, ρ_2 で, クラス写像 f が存在し, すべてのクラス $R (\in C_1)$ について R のオブジェクトと $f(R)$ のオブジェクトは1対1に対応し, 対応するオブジェクトについて, $o_1 < o_2$ であれば, ρ_2 は ρ_1 を表現するといひ, $\rho_1 < \rho_2$ で表す. \square

スキーマ S_2 のオブジェクトの表現法で, S_1 の任意の状態についてそれを表現する S_2 の状態が存在しなければならない. また, S_2 の状態で, S_1 のいかなる状態も表現していないものがあれば, そのような状態は S_1 では存在しえないものである. 次にスキーマの表現性を定義し, その性質を示す.

[定義4] スキーマ $S_1(C_1, ISA_1)$, $S_2(C_2, ISA_2)$ とあるクラス写像 f について, S_1 の任意の状態 ρ_1 に対して $\rho_1 < \rho_2$ となる S_2 の状態 ρ_2 が存在し, S_2 の任意の状態 ρ_2 に対して $\rho_1 < \rho_2$ となる S_1 の状態 ρ_1 が存在するとき, S_2 は S_1 を表現するといひ, $S_1 < S_2$ で表す. \square

[定理1] スキーマ $S_1(C_1, ISA_1)$, $S_2(C_2, ISA_2)$ で, $S_1 < S_2$ であ

るための必要十分条件は、 C_1 のすべてのクラス R について

$$\text{at_b}(\text{Ref}^+(R)) \subseteq \text{at_b}(\text{Ref}^+(f(R)))$$

である。□

(証明) $S_1 < S_2$ であれば、 C_1 のすべてのクラス R について、 R のオブジェクトと $f(R)$ のオブジェクトは1対1に対応し、対応するオブジェクト o_1, o_2 は、 $o_1 < o_2$ である。補題1より、 $o_1 < o_2$ であれば、

$$\text{at_b}(\text{Ref}^+(R)) \subseteq \text{at_b}(\text{Ref}^+(f(R))).$$

C_1 のすべてのクラス R について $\text{at_b}(\text{Ref}^+(R)) \subseteq \text{at_b}(\text{Ref}^+(f(R)))$ とする。 S_1 の状態 ρ_1 に対し、 S_2 の状態 ρ_2 を次のように定める。 R と $f(R)$ のオブジェクトを1対1に対応させる。基本構造オブジェクトのみを参照するクラス R について、 R のオブジェクト o_1 に対応する S_2 のオブジェクト o_2 の値を共通属性は同じ値にする。 $f(R)$ に含まれない属性は、参照するクラスに含まれるので、 o_1 と同じ値を持つオブジェクトを参照するようにする。これにより、 $o_1 < o_2$ である。組構造オブジェクトを参照する属性を含むクラスのオブジェクト o_1 で、参照するオブジェクトがすべて S_2 のオブジェクトで表現されていれば、対応するオブジェクト o_2 も o_1 の参照するオブジェクトを参照することにより、 $o_1 < o_2$ とすることができる。従って、 $\rho_1 < \rho_2$ となる。 S_2 の状態 ρ_2 に対しても同様に $\rho_1 < \rho_2$ となる S_1 の状態 ρ_1 を定めることができる。よってこのとき、 $S_1 < S_2$ である。(証明終り)

4. 参照による冗長性の削減

データの冗長性や更新処理を考えると、オブジェクトの値をすべて基本構造オブジェクトの参照にせず、他の組構造オブジェクトを参照する方がよい場合がある。値が基本構造オブジェクトの参照のみからなるときの問題点には次のものがある。

- (1) データの冗長性：属性集合 X の値が等しいオブジェクトが複数あれば、 X のデータの関連が複数のオブジェクトで記憶されていることになり、冗長である。
- (2) 更新時の検査： X の値を変更する際、 X を含むすべてのクラスのオブジェクトを調べなければならない。

(3) 削除による関連の喪失: オブジェクト o を削除する際, $X(o)$ が等しいオブジェクトが他に無ければ, $X(o)$ の関連がデータベースから失われてしまう.

属性集合 X からなるクラスを作り, X の値はそのクラスのオブジェクトを参照するようにすれば, (1)~(3)の問題に対処できる.

[例2.] 図2(a)では, オブジェクトは AB の値を重複して持つ. 図2(b)のように AB をクラス R_1 として独立させ, もとのクラス R に対応するクラス R_2 のオブジェクトは, AB の値については R_1 のオブジェクトを参照するようにすれば, AB についての冗長性を削減できる. \square

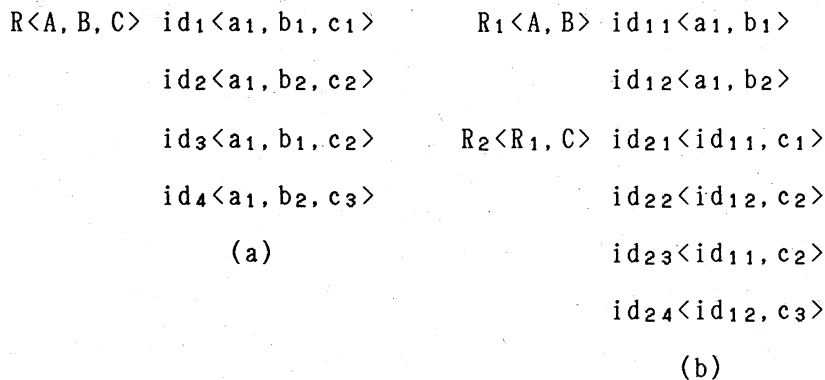


図2 参照による冗長性の削減

複数のクラスに X がある場合も, X をクラスとすることにより, 1ヶ所で記憶できる. クラス R の属性集合 X による分解, および分解後のオブジェクトの対応の決定は次のように行う.

[手続き1] クラス R の属性集合 $X (\subseteq at(R))$ による分解

1. R_1 を $at(R_1) = X$ であるクラスとする.
2. R_2 を $at(R_2) = (at(R) - X) \cup A$ (A はクラス R_1 を参照する属性) とする.
3. R_1 のオブジェクト集合を $\{o \mid \exists R \text{のオブジェクト } o', v_1(o) \subseteq v_1(o'), id(o) \text{は任意の値}\}$ とする.
4. R_2 のオブジェクト集合を $\{o \mid \exists R \text{のオブジェクト } o', at(R_2) - A \text{の値は } o' \text{の値に含まれる, } A \text{の値は } v_1(o'') \subseteq v_1(o') \text{となる } R_1 \text{のオブジェクト } o'' \text{の識別子, } id(o) = id(o')\}$ とする.
5. R を参照しているクラスを R_2 を参照するようにする.

6. スキーマ $\{C, ISA\}$ を $\{C', ISA'\}$ とする.

$$C' = (C - \{R\}) \cup \{R_1, R_2\}$$

$$ISA' = (ISA - \{ISA(R', R'') \mid R' = R \text{ または } R'' = R\})$$

$$\cup \{ISA(R_2, R') \mid ISA(R, R') \in ISA\}$$

$$\cup \{ISA(R', R_2) \mid ISA(R', R) \in ISA\} \quad \square$$

R のオブジェクトは R_2 のオブジェクトと 1 対 1 に対応し、その値は対応する R_2 のオブジェクトとその参照する R_1 のオブジェクトの値として得ることができる。従って R_2 のメソッドは、オブジェクト o で R_2 から除かれた属性 X の値 $X(o)$ を用いる部分を o の参照するオブジェクト $A(o)$ の X の値 $X(A(o))$ を用いるように変えればよい。

[定理 2] スキーマ S_1 と S_1 のクラスを手続き 1 を用いて分解したスキーマ S_2 では、 $S_1 < S_2$. \square

(証明) S_1 の状態 ρ_1 に対し、ステップ 3, 4 のように S_2 の状態 ρ_2 を決めると $\rho_1 < \rho_2$. また、 S_2 の状態 ρ_2 に対して R_2 の参照する X の値を S_2 の R のオブジェクトの値とすることで、 $\rho_1 < \rho_2$ となる S_1 状態 ρ_1 が作れる。(証明終り)

5. 関数従属性を用いた分解

データベースでは、データの持つ制約を保持することが 1 つの重要な要素である。本節では、基本的な制約である関数従属性のオブジェクト指向データベースにおける保持について議論する。

関数従属性は、データ間の対応関係が多対 1 であることを示す基本的な制約である。例えば、学生とその所属する講座の対応は多対 1 であり、各学生に対して所属講座は一意に決まり複数の講座には所属していない場合などである。

オブジェクト指向モデルにおける関数従属性を次のように定義する。

[定義 5] クラス R と属性集合 $X, Y (\subseteq \text{at}(R))$ で、 R の任意のオブジェクト o_1, o_2 に対し、 $X(o_1) = X(o_2)$ ならば $Y(o_1) = Y(o_2)$ のとき、 R のオブジェクト集合は関数従属性: $X \rightarrow Y$ を満足するという。 $X \rightarrow Y$ は、 X の参照するクラスの識別子集合を定めると Y の参照するクラスの識別子集合が一意に定まることを表す。このよ

うな関数従属性をクラス内の関数従属性という。□

[定義6] クラス R のオブジェクトを1つ定めるとそのオブジェクトの値が定まる。即ち、 R のオブジェクトの識別子から参照するクラスのオブジェクトの識別子が定まる。この性質をクラス間の関数従属性といい、 $R \rightarrow at(R)$ で表す。□

データベースで満足すべき関数従属性集合が与えられているとき、関数従属性： $X \rightarrow Y$ についても4節での議論と同様の問題が生じる。

- (1) データの冗長性： X の値が等しいオブジェクトが複数あれば、 X と Y の同じデータの関連が複数のオブジェクトで記憶されていることになり、冗長である。
- (2) 更新時の検査：新しくオブジェクトを追加するときや、オブジェクトの値の変更をする際に、 $X \rightarrow Y$ を満たしているかどうかの検査が必要である。
- (3) 削除による関連の喪失：オブジェクト o を削除する際、 $X(o)$ が等しいオブジェクトが他に無ければ、 $X(o)$ と $Y(o)$ の関連がデータベースから失われてしまう。

属性集合 XY とクラス R ($at(R) \supseteq XY$) について、手続き1を用いてクラスを分解し、 XY からなるクラスを新たに作り、 R から XY を除いて、その値は XY のクラスのオブジェクトを参照するようにすることにより、 R では $X \rightarrow Y$ による問題を考える必要がなくなる。

[例3] 図1のスキーマで、クラス“学生”では、学科は1つの学部にものみ所属するので、関数従属性：学科名 \rightarrow 学部名を満足する。“学生”を手続き1で分解すると、スキーマ、オブジェクトはそれぞれ図3(a),(b)となる。学科名 \rightarrow 学部名はクラス“学科”で保持され、“学生”のオブジェクトは“学科”のオブジェクトを参照する。□

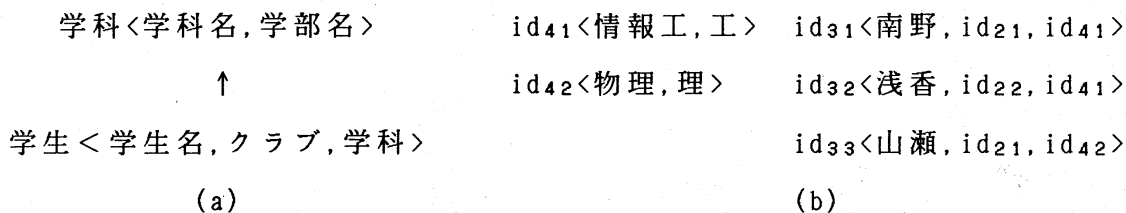


図3 関数従属性による分解

関数従属性集合 F が与えられているとき、手続き 1 を用いて F 中の関数従属性でクラスを分解した結果のスキーマ S_2 では、 F の満足の検査が複雑になる場合がある。

[例 4] 属性集合 ABCD からなるクラス R で、関数従属性集合 $\{A \rightarrow B, BC \rightarrow D\}$ が与えられているとする。 R を $A \rightarrow B$ で分解した結果のスキーマでは、 $BC \rightarrow D$ の満足の検査は、 2 つのクラスにまたがって行わなければならない。 図 4 は、 $BC \rightarrow D$ を満足していない状態の例であるが、参照した値を調べる必要がある。 id_{21} は id_{11} を参照しているので BC の値は b_1c_1 であり、 id_{22} も id_{12} を参照しているので b_1c_1 である。しかし、 D の値は d_1, d_2 であり、 $BC \rightarrow D$ を満足していない。 □

$R_1 \langle A, B \rangle$	$R_2 \langle C, D, R_1 \rangle$
$id_{11} \langle a_1, b_1 \rangle$	$id_{21} \langle c_1, d_1, id_{11} \rangle$
$id_{12} \langle a_2, b_1 \rangle$	$id_{22} \langle c_1, d_2, id_{12} \rangle$

図 4 関数従属性の満足の検査

分解後も他の関数従属性の検査を容易にするため、他の関数従属性の左辺の属性を R_2 に残す。右辺の属性が R_2 に含まれない場合、それはクラス間の関数従属性として保持される。手続き 1 のステップ 2 は以下のようなになる。

2'. R_2 を $at(R_2) = (at(R) - (W - XY)) \cup A$ (W は他の関数従属性の左辺、 A はクラス R_1 を参照する属性) とする。

このとき、 R_2 に残った属性と R_1 の属性に共通のものがあれば、 R_2 の各オブジェクト o でのその値と、 o が参照する R_1 のオブジェクトのその属性の値は同じでなければならない。例 4 の場合、 R_2 に属性 B を残すので、 $B(o) = B(R_1(o))$ である。

[制約 1] 各オブジェクトについて、参照しているオブジェクトが同じ属性を持てば、その値は等しい。 □

6. むすび

オブジェクト指向データベースについて、参照の様々な性質について議論し、データベース設計において考慮すべき問題を示した。データベース設計は、初期

値（データベースのビュー）として与えられたクラスの集合 S と、データの意味制約である関数従属性集合から、 S を表現する範囲で冗長性の少ないスキーマを求めることになる。一般に、関数従属性による分解と、複数のクラスに共通属性があればそれらによる分解で冗長性の少ないスキーマが設計できる。

一方、質問処理効率を考えたとき、値を参照せずオブジェクト内に保持している方がよい場合もあり、これらの関係を明らかにする必要がある。

参考文献

- 1) Stonebraker, M., "Object Management in POSTGRES Using Procedures," Proc. Int. Workshop on Object-Oriented Database Systems, pp.66-72, Sept. 1986.
- 2) Maier, D. and Stein, J., "Development and Implementation of Object-Oriented DBMS," in Research Directions in Object-Oriented Programming, The MIT Press, 355-392, 1987.
- 3) Banerjee, J., Chou, H., Graza, F., Kim, W., Woelk, D., Ballou, N., and Kim, H., "Data Model Issues for Object-Oriented Applications," ACM Trans. on Database Syst., Vol.5., No.1, pp.3-26, Jan. 1987.
- 4) Andrews, T. and Harris, C., "Combining Language and Database Advances in an Object-Oriented Development Environment," Proc. OOPSLA '87, pp.430-440, 1987.
- 5) Bancilhon, F., "Object-Oriented Database Systems," Proc. ACM Symp. on Principles of Database Syst., pp.152-162, March 1988.
- 6) Lecluse, C., Richard, P., and Velez, F., "O₂, an Object-Oriented Data Model," Proc. ACM SIGMOD Int. Conf. on Management of Data, pp.424-433, June 1988.
- 7) Hull, R., Tanaka, K., and Yoshikawa, M., "Behavior Analysis of Object-Oriented Databases: Method Structure, Execution Trees and Reachability," Proc. Int. Conf. on FODO, Lecture Notes in Computer Science 367, Springer-Verlag, pp.372-388, June 1989.
- 8) Kim, W., Nicolas, J.-M., and Nishio, S. (eds.), Proc. Int. Conf. on Deductive and Object-Oriented Databases, Dec. 1989.
- 9) Ullman, J. D., Principles of Database Systems, 2nd ed., Computer Science Press, 1982.