オブジェクト指向データベースの開発

－集合と外延－

5W－4

脇園 竜次　土屋 武彦　川村 敏和　田中 立二

(株) 東芝 重電技術研究所

1　はじめに

我々は,「言語透過性，高速性，コンパクト性」を目指したOODBMSとしてOdbを開発した[1]．Odbは OODBとして要求される必須機能を提供しているが，それらの機能およびデータベース言語仕様としての主な特徴は以下に示すとおりである．

(1) C++を基本言語としたデータベース言語

- C++のクラス宣言をデータベースのスキーマ定義として利用する．これにより抽象データ型，クラス階層（継承），複合オブジェクトの機能を提供する．
- 永続性を記憶域クラスの属性とすることにより永続/一時オブジェクトの統一的な扱いが可能となる．オブジェクト相互の関係はC++のポインタを用いて表現する．
- C++言語の拡張機能を利用し，データベース管理，トランザクション管理，集合管理用のクラスライブラリを提供する．

(2) 集合管理機能を一般化したデータモデル

既存のOODBではクラスに属するオブジェクトの集合としての外延 (Extent) とユーザが定義する集合の概念および操作体系の融合が図られていない．Odbでは外延と集合および検索の統合化をはかり，データベースにおけるデータモデルとオブジェクト指向におけるデータモデルを統一した仕様とした．

本稿では，Odbにおける外延と集合の概要について述べる．

2　外延と集合の統合

外延とは，“クラスのインスタンス (オブジェクト) の集合” である[2]．関係データベースにおいては，tableに含まれるtuple(record)の集合が外延に相当するため，この概念は自明である．しかしC++を基本言語とする既存のOODBMSには，外延の概念が明確にされていない．まずOODBMSは，永続オブジェクトの実現方式に関する考え方で次のように大きく二つに分けることができる．

- 永続性はクラス毎の属性
- 永続性はオブジェクト毎の属性

前者の場合，永続クラスを継承することにより，データベース操作と管理のmethodもクラス階層の中に体系として組み込んでいる．これにより外延として永続オブジェクトの管理ができるが，OODBMSが提供する集合とは異なる体系であり，両者の間で統一的な処理を行うことができない．

後者については，集合機能と検索・照合を統合して強力な集合管理体系を持つOODBMSもあるが，システムとして外延の機能を用意していない．すなわちあるクラスのオブジェクトは，外延として管理されないため，作成しただけでは後でアクセスする手段が無くなってしまう．このためユーザは，作成したオブジェクトを明示的に集合へ加えてはじめて集合として管理され，集合の様々な機能を利用できるようになる．

このように外延の概念がないため統一した処理ができない，もしくはユーザが実装・管理しなければならないというのは，ユーザにとって大きな負担である．

3　外延と集合の構成および操作

このようなことからOdbでは，クラスに属する全オブジェクトの管理・操作機能を有するTypeクラスを用意している．以下の図1は，Odbにおける Typeオブジェクトとその他のデータベースオブジェクトとの関係の概念図である．

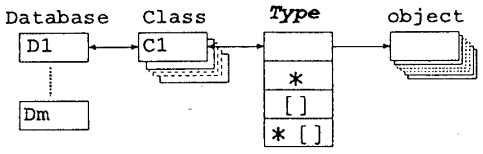

図 1: Typeの DBMSにおける位置付け

図1で Database, Class, Typeの部分は，それぞれデータベース管理オブジェクト，クラス定義情報管理オブジェクト，永続オブジェクト管理オブジェクトである．この図で示すように Typeは，オブジェクトとクラス定義情報を結ぶ役割を果たしている．

Development of the OODBMS

Ryuji Wakizono, Takehiko Tsuchiya, Toshikazu Kawamura, Tatsuji Tanaka

Heavy Apparatus Engineering Lab., Toshiba Corp.

外延を Type の属性として考え，外延の属性が付加された Type は，同じく Odb で提供している集合と同じ操作と演算体系が提供される．Type は Set と同様その要素がユニークである集合であり，Set の subtype として図 2に示すように集合管理体系 (Collection, Set, Bag) の中に組み込まれている．

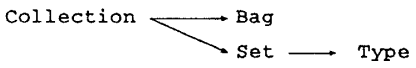

図 2: Type の集合における位置付け

これらの集合機能の体系を図 3に示す．

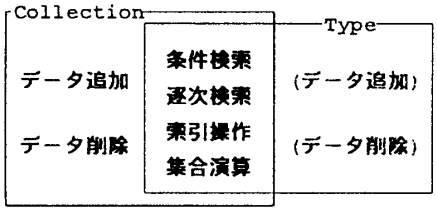

図 3: Collection の機能 (操作と演算)

ただしこの中で，Type におけるデータの追加・削除の操作は直接にはユーザから行えない．これらの操作はオブジェクトの生成・削除に伴い自動的に行われる．それ以外は，基本的に Type と Collection は同一の機能を持つ．この点について，図 1と以下の簡単な例を用いて説明する．

```
Type* typeC1 = DB1->type("C1");      ...(1)
Set*  set1 = new Set("C1");          ...(2)
C1*   aC1 = new(typeC1) C1(2);       ...(3)
set1->insert(aC1);                   ...(4)
Set a1  = typeC1->find("m1 >=",2);   ...(5)
Set a2  = set1->find("m1 <", 2);     ...(6)
set1->remove(aC1);                   ...(7)
delete aC1;                          ...(8)
```

ここで C1 はユーザ定義クラスであり，int 型のメンバ m1 を持ち，コンストラクタで m1 の初期値を設定できるものとする．永続オブジェクトを作成するためには，まずクラス C1 の Type をデータベースから取得する (1)．この Type オブジェクトは，最初に要求されたときに DBMS が作成する．ユーザ定義の集合は，ユーザが管理するクラスを指定して作成する (2)．永続オブジェクトの作成には拡張された new を用いるが，このとき作成されたオブジェクトは，new の引数に指定された Type(ここでは typeC1) の要素として管理される (3)．一方ユーザ定義集合では，insert を用いて明示的に要素を追加する (4)．同様にユーザ定義集合では remove を用いて明示的に要素を集合から削除するが (7)，Type からの削除は delete により行われる (8)．それ以外の要素の検索などの処理には，どちらも同様のインタフェースが使用できる (5, 6)．ここで示したのは，find を用いた C1 のメンバ m1 に関する条件検索の例である．

また Collection(Set, Bag) と Type は，集合演算および操作に関して閉じている．図 4のように，集合と外延の間の集合演算，代入演算，条件検索の結果はすべて集合として得られる．

集合演算	Collection	Type
Collection	Collection	Collection
Type	Collection	Collection

代入演算	Collection	Type
Collection	Collection	Collection

条件検索	find
Collection	Collection
Type	Collection

図 4: 集合演算と検索結果の型

このように，find による検索結果も集合であると考え，検索を集合演算・操作の体系に組み込んでいる．これにより，簡潔で強力な検索・問い合わせ機能を提供している．

4 まとめ

外延を Type の属性として考え，集合と同じインタフェースを与えることにより，自然な形で Odb に外延を導入することができた．現在 Type と Collection(Set, Bag) とでは実装方式が異なっているが，これは今後の研究課題であり，あわせて検索機能の高度化・最適化などを行なっていく予定である．

参考文献

[1] 脇園，土屋，川村，田中: オブジェクト指向データベースの開発，情報処理学会第 48 回全国大会，1994.

[2] R.G.G. Cattell: The Object Database Standard: ODMG-93 Release 1.1, Morgan Kaufmann, 1994.