オブジェクト指向データベースの開発

1E－3

－機能仕様－

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

プラント監視制御等のオンライン・リアルタイムシステムにおいてはその高速応答性を要求される事から，既存のデータベース管理システムを使わず専用のファイル管理システムを用いる事が多かった．

これに対してオブジェクト指向データベース(OODB)の出現により，以下の機能を実現することが可能になった．

(1) 複雑，不規則な構造を持つデータベースの構築
(2) データに付随した操作・処理・アルゴリズムのデータベース化
(3) データベース言語とプログラミング言語の統一
(4) 高速処理

その高速性と言語透過性から，CAD，CASE等のエンジニアリング分野への適用だけでなく，従来適用の難しかったプラント監視制御システムへの適用を考慮する事が可能になった．

発電，電力系統および各種産業におけるプラント監視制御システムにおいては，リアルタイム制御プログラムあるいはリアルタイムデータおよび操作・監視画面などのヒューマンインタフェースのデータベース化およびこれらの統合化が必要とされている．OODBの出現によりこれらの性格の異なる情報およびそれらに関連する処理を統合化する情報処理体系の構築の可能性がでてきた．

そこで我々は，「言語透過性，高速性，コンパクト」を目指したOODBMSとしてOdbを開発した．Odbはオンライン・リアルタイムの用途だけでなく，一般の応用システムに組み込んで使う事もできる．

本稿では，この様な組み込み型データベースの機能と「データベース言語」の体系と仕様について紹介する．

2　機能の特徴

我々の開発したOdbは先に述べた様に，リアルタイム用途に使える様に機能を絞り込んだ簡潔な機能仕様とし高性能化を図った．

OdbはOODBとして要求される必須機能を提供しているが[2]，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．

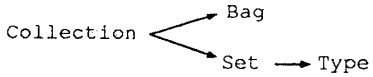
(1) C++を基本言語としたデータベース言語[1]

- 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．

Odbでは外延は以下の通り集合クラスの導出クラス(subtype)のTypeクラスとして位置づけ，集合クラス階層を体系化した．



Typeクラスとしてクラスに属する全オブジェクトの操作・管理機能を提供する．TypeはSetと同様その要素がユニークである集合であり，集合への登録，削除はオブジェクトの生成・削除に伴い自動的に行われる．

これに対して集合(Set, Bag)は，ユーザが指定したオブジェクトを集めて集合化するものである．要素の重複も有り(Bag)，集合への登録・削除はユーザが責任を持って行う．

この様に外延を管理するクラスTypeは集合Setと概念として同じであり，集合と同じ演算・操作体系（登録，削除を除く）を提供する．

Odbの集合機能の体系を図1に示す．

Odbでは集合演算・操作に関して閉じた体系を提供する．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できる様にした．

Development of the OODBMS
Ryuji Wakizono, Takehiko Tsuchiya, Toshikazu Kawamura, Tatsuji Tanaka
Heavy Apparatus Engineering Lab., Toshiba Corp.

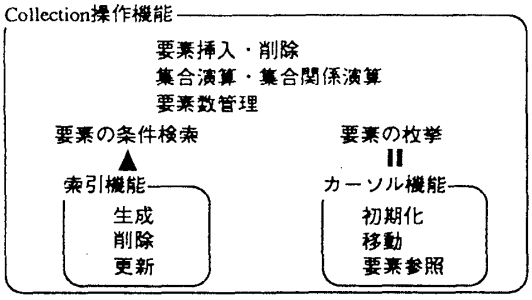


図 1: Collection 機能体系

データベース内の永続オブジェクトをアクセスする基本的手段としては，1) 条件検索，2) 逐次取り出し，が用意されている．これらは何れも集合に属するオブジェクトの検索として行われる．検索されたオブジェクトから関係する他のオブジェクトへのナビゲーションはポインタを用いて行う．

3 データベース言語仕様

図2に Type クラスと集合の例を示す．

図 2・(a) が従業員クラスのユーザ定義集合であり，図 2・(b) が従業員クラスの外延である．

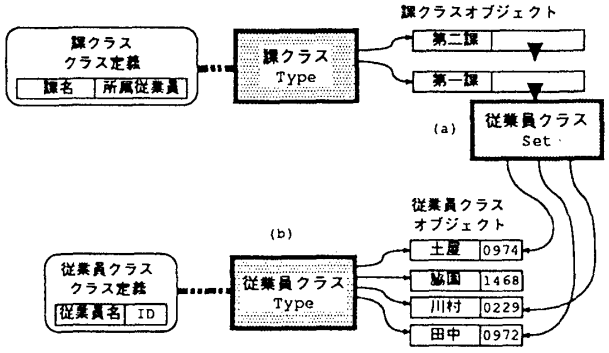


図 2: ユーザ定義集合と外延

この例におけるクラス定義とデータ操作の記述例を図3に示す．

図3の 1 行目〜7 行目・8 行目〜13 行目がそれぞれ課クラス・従業員クラスのクラス定義である．課クラスは属性として集合を持っている (3 行目)．クラス定義と永続性は独立であるので，特別な記述は必要としない．

18 行目の *typeEmployee*・19 行目の *typeSection* がそれぞれ課クラス・従業員クラスの外延である．21 行目〜24 行目において **new(Type*)** を用いて作成した 2 つの永続オブジェクト *WAKIZONO* と *SECT1* はそれぞれの外延で自動的に追加される．これに対し，26 行目の集合 *SECT1STAFF* はユーザの定義によるものであり，管理はユーザによって行なわれる．例えば 27 行目ではこの集合に新しい要素を追加している．29・30 行目は，条件検索による集合要素へのアクセスである．31 行目以降は **for** を用いた集合要素へのアクセスである．このように Set と Type は全く同じに操作できる．

```
 1  class Section {
 2    char    name[20]; // 課名
 3    Set*    staff;    // 所属従業員 (ユーザ定義集合)
 4  public:
 5    Section(char* aSectionName);
 6    Set*  getStaff();       // 所属従業員集合を返す
 7  };
 8  class Employee {
 9    char    name[20];       // 従業員名
10    int     employeeID;     // 従業員番号 (ID)
11  public:
12    Employee(char* aName, int aID);
13  };
14  //              main.cxx
15  int main()
16  {
17    db = Database::open("Employee_DB");
18    Type* typeEmployee = db->type("Employee");
19    Type* typeSection  = db->type("Section");
20    Transaction::start(WRITE_LOCK);
21    Employee* WAKIZONO =
22      new(typeEmployee) Employee("脇園", 1468);
23    Section*  SECT1 =
24      new(typeSection)  Section("第一課");
25    ......................
26    Set*      SECT1STAFF = SECT1->getStaff();
27    SECT1STAFF->insert(WAKIZONO);
28    ......................
29    Employee* aEmployee =
30      SECT1STAFF->find1("employeeID ==",1468);
31    for (SECT1STAFF->first();SECT1STAFF->more();
32         (*SECT1STAFF)++ ) {
33      aEmployee = (Employee*)(*SECT1STAFF)();
34      ......................
35    }
36    for (typeEmployee->first();
37        typeEmployee->more();(*typeEmployee)++){
38      aEmployee= (Employee*)(*typeEmployee)();
39      ......................
```

図 3: クラス定義とデータ操作

4 まとめ

C++の言語構文を最大限に利用する事により，永続オブジェクト宣言・操作を C++言語体系と違和感の無いものとして埋め込んだ．さらに永続オブジェクトと一時オブジェクトを統一的に扱うことにより言語透過性の高いデータベース言語を実現した．

今後は，C++言語以外のプログラミング言語 (C など) のデータベース言語の開発，検索機能の高度化，標準化への対応などを行なっていく予定である．

参考文献

[1] M. A. Ellis, B. Stroustrup: The Annotated C++ Reference Manual, Addison-Wesly, 1992.

[2] M.Atkinson, etal.: The Object-Oriented Database system Manifesto, Proc. of DOOD89, pp.40-57,1989.

[3] W.Kim: Introduction to Object-Oriented Databases, MIT Press, 1990.