オブジェクト指向DBMS - Odinの版管理方式

2R−2

木村 裕　鶴岡 邦敏

日本電気（株） C&Cシステム研究所

1 はじめに

オブジェクト指向データベースの応用分野であるCAD, CASE等では、幾つかの設計案を作成したり、完成品を流用して次期製品を作成する作業を伴う。このため、これらの分野では版管理機能は必須である。本稿では、現在研究開発中のオブジェクト指向データベース管理システム-Odin [3, 4, 5] における版管理モデルおよびその実現方式の特長について述べる。

2 設計指針

版管理の設計では、次の点を考慮した：

1. 版モデルのないシステムと同じ性能であること。
2. 旧版を通常のオブジェクトと同じ性能で参照できること。
3. 版管理情報のための領域をできるだけ少なくすること。
4. 実現が容易なこと。

この中で、性能に関する項目1,2は重要である。オブジェクト指向DBが使われる理由の1つが高速性にあるからである。特に、CAD/CAEでの要求が強い。

このため下記に述べるように、Odinではversion/getモデルによる版管理方式を考案した。

3 版管理モデル

3.1 オブジェクト定義

Odinでは、版管理可能属性を持つクラスのインスタンスのみが、版オブジェクトを生成できる。この属性はクラス定義時に指定するか、メタクラスのメンバ関数により実行時に動的に付加することができる。

```
class Emp : od_spec versionable {
public:
    persistent char* e_name;
    int   e_age;
    Emp(char*, int);
};
```

この点は、最近のオブジェクト指向DBMSがすべてのオブジェクトを版管理可能にしているのと異なる。Odinの方式では、必要なクラスだけが版管理対象となるので、データベース全体でクラス中の版管理情報の占める割合をかなり小さくすることができる。（上記の例で、変数e_nameはポインタであり、指している先はchar型の永続オブジェクトであることを示している。）

Version Support for
an Object-Oriented DBMS - Odin

Yutaka KIMURA and Kunitoshi TSURUOKA
C&C Systems Research Laboratories,
NEC Corporation

3.2 版の生成

通常の（版でない）インスタンスは、永続オブジェクトの場合、演算子pnewで生成される。この操作によりオブジェクトが生成され、オブジェクトidが割り当てられる。

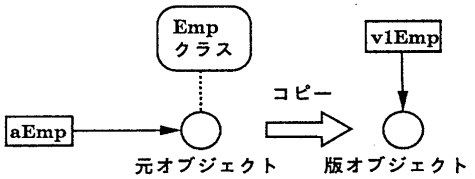
```
Emp*  aEmp;
aEmp = pnew Emp("kimura", 23);
```

pnewにより生成されたオブジェクトは、元オブジェクトと呼び、版オブジェクトと区別される。

版オブジェクトは、メンバ関数versionを呼び出すことによって生成される。この関数の返す値は、版オブジェクトのオブジェクトidである。

```
Emp  *v1Emp, *v2Emp;
v1Emp = aEmp->version();
v2Emp = aEmp->version();
```

Odinでは、O++ [1] と異なり、版オブジェクトidは存在しない。Odin内のすべてのオブジェクトは1種類のオブジェクトidで管理される。このversionにより、現時点での元オブジェクトの内容をコピーした版オブジェクトが生成される。コピーのため、元オブジェクトが指しているポインタおよび指されているポインタは不変である。従って、元オブジェクトから指されているオブジェクトは版からも指されることになる。版生成後の元オブジェクトへのポインタ（aEmp）は、元オブジェクトを指したままなので、このポインタを介した更新は元オブジェクトを更新することになる。従って、元オブジェクトと版オブジェクトの内容はversionを適用した時点では一致しているが、一般的には一致していない。元オブジェクトに対して、versionを繰り返し適用すると、最初の版を先頭ノードとして線形の版の導出リストが形成される。



元オブジェクトに旧版が（3.3節に述べるメンバ関数getにより）コピーされている状態でversionを適用したとき、生成された版は代替版となる。

3.3 版の取り出し

すでに記述したように、元オブジェクトはクラスが管理しているオブジェクトのことである。従って、（版を意識しない）プログラムは、この元オブジェクトの集合に対して操作を行なっていることになる。もし元オブジェクトを古い版に戻したい場合、メンバ関数getを元オブジェクトに対して適用する。

```
aEmp->get();
aEmp->get(v1Emp);
```

引数がない場合、現在版が元オブジェクトにコピーされる。引数がある場合、特定の版オブジェクトが元オブジェクトとなる。

3.4 版の探索

必要な版を見つけるために、版オブジェクトを逐次取り出しす方法と、メンバ関数 select を使った集合検索による方法がある。

版オブジェクトを生成時刻順に逐次取り出すために、メンバ関数 firstT, lastT, nextT, prevT を提供する。導出順に深さ優先で取り出すために nextD が、広さ優先に取り出すために nextB がある。また、代替版の最新版をすべて取り出すには、メンバ関数 leaves を使用する。

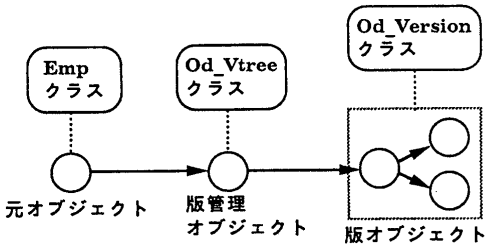
メンバ関数 select を使ってある条件を満足する版オブジェクトを取り出すことができる。この select は、集合オブジェクトに対する select [4] と同じ構文を使う。たとえば、1992 年 10 月 13 日までに作った版で、バグ修正した版を検索する場合、次のような式になる。

```
Od_Version<Emp*>->select([
   where od_srcobj == aEmp
      && od_datetime <= 0t"1992 年 10 月 13 日 "
      && od_vname == "for bug fix";]);
```

この条件式に現れる変数は、4節で述べる版クラスの変数であり、それぞれ版化した日時と備考が格納されている。

4 実現方式

1 つのクラスのインスタンスの版管理は、2 つのクラス - 版クラス Od_Version と版管理クラス Od_Vtree で行なわれる。あるクラスのインスタンスは、1 つの版管理インスタンスを指し、さらに版管理インスタンスは対応する版木のルートの版インスタンスを指す。両クラスとも総称クラス (generic or parameterized class) であり、C++ のテンプレート・クラスとして実現される。



4.1 クラス Od_Version

版管理可能属性が指定されたクラス T をコンパイルすると、その派生クラス（サブクラス）として、総称クラス Od_Version を具現化したクラス Od_Version<T> が自動的に生成される。Od_Version<T> のインスタンス（版オブジェクト）は、元のクラスの構造と手続き、および版管理のための構造と手続き（たとえば、前後の版へのポインタ等）を持つ。

このように版オブジェクトも通常のオブジェクトと同様にオブジェクト id が割り当てられる。また、版を元クラスの派生クラスとして実現しているので、元クラスをドメインとする変数に対応する版のインスタンスを代入することができる。

```
Emp* career[10];
career[0] = aEmp;
career[1] = v1Emp;
```

O++ では、オブジェクト id と版 id を区別しており、版 id を持つ変数の型は、特別な型に変換される。この方式は、実行時に id （ポインタ）の検査をしなければならず、プログラムの実行効率を低下させる。

4.2 クラス Od_Vtree

版管理属性が指定されたクラス T をコンパイルすると、総称クラス Od_Vtree を具現化したクラス Od_Vtree<T> も生成される。クラス T の各インスタンスが最初の版を生成するとき、Od_Vtree<T> のインスタンスがそれぞれ 1 つ生成される。このインスタンスは、版木を管理するためのオブジェクトであり、ルート版、最新版、現在版への各ポインタ等を保持する。

5 他モデルとの比較

O++ [1] ではすべてのオブジェクトは版管理可能であるが、Odin では版管理可能属性を持つクラスのインスタンスのみ版管理可能である。この点は ORION [2] と同じである。ORION では動的な版の選択が可能であり、実行時にオブジェクトの参照の解決が行われる。O++ も基本的にはこの方式を採っている。一方、Odin の版管理は version/get モデルであり、旧版は操作の前に get しておく必要がある。動的な版の選択は、この get の際にのみ生じる。O++ ではポインタとしてオブジェクト id と版 id の 2 種類が存在し、ORION ではオブジェクト id、ジェネリック・オブジェクト id、版番号が存在するが、Odin では 1 種類のポインタ - オブジェクト id のみ存在する。

6 まとめ

本稿では、版管理モデル **version/get モデル**を提案し、その実現方式について述べた。版管理は、クラスに版管理可能属性をクラス定義時または実行時に付加することで可能となる。版管理可能なクラスのインスタンス（元オブジェクト）は、メンバ関数 version で版化され、メンバ関数 get で古い版に置き換わる。アプリケーション・プログラムはこの元オブジェクトを操作する。オブジェクトまたは版は、直接ポインタにより参照されるので、実行時のオーバヘッドは存在しない。また、版の探索のための逐次検索用のメンバ関数だけでなく、版集合の条件検索ができるメンバ関数を提供し、多様な操作を可能にした。

参考文献

[1] R. Agrawal, et.al. “Object Versioning in Ode,” Seventh Int. Conf. on Data Engineering, 1991.

[2] J. Banerjee, et.al. “Data Model Issues for Object-Oriented Applications,” ACM TOOIS Vol.5, No.1, 1987.

[3] Y. Kimura and K. Tsuruoka “A View Class Mechanism for Object-Oriented Database Systems,” Second Conf. on DASFAA, 1991.

[4] Y. Kimura and K. Tsuruoka “A Language and View for Odin, an Object-Oriented Database System,” IEEE Pacific RIM Conf. on Communications, Computers and Signal Processing, 1991.

[5] 鶴岡, 木村 “オブジェクト指向データベース管理システム「Odin」の機能と構成”, 日経 AI 別冊 1991 夏号, 1991.