オブジェクト指向データベースの開発

－スキーマ管理方式－

5W－3

土屋 武彦　脇園 竜次　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、産業システムへの組み込み用途を想定し、「言語透過性、高速性、コンパクト性」を目標とするオブジェクト指向データベース管理システム Odb を開発した [1]。

オブジェクトに永続性を持たせるためにはスキーマと呼ばれるオブジェクトの構造を定義したデータモデルが必要である。本システムは C++を基本言語とする OODBMS であり、C++により提供される型を有効に利用できるようなスキーマ構成を実現している [2]。またスキーマ管理を永続オブジェクトの管理と同一の管理体系とすることによりスキーマ操作 (登録、変更、削除) の操作性を高めている。

本稿では、これらの特徴をもつスキーマ管理方式について紹介する。

2 スキーマの仕様

2.1 スキーマ定義

オブジェクトに永続性を与えるには以下の 2 つの方式がある。

(1) DBMS が提供する永続クラスの派生クラスのオブジェクトに永続性を与える。

(2) クラスと無関係に永続性が宣言されたオブジェクトに永続性を与える。

前者の方式ではユーザが定義したクラスの構造に DBMS の継承情報を付加するためユーザが意図するクラスと構造が異なるものになる。そこで本システムは一時オブジェクトと永続オブジェクトとの操作性の統一を図るため後者の方式とした。さらに言語透過性を重視し、DBMS のためにクラス定義の構文を拡張することはしない。

永続性を宣言されたオブジェクトが永続オブジェクトとして機能するためにはスキーマが必要である。本システムのスキーマは以下に示す機能を提供する。

(1) 永続オブジェクト生成時のオブジェクトのサイズを提供する。

(2) 検索やインデックス指定時のメンバのオフセットを提供する。

(3) オブジェクト・マッピング時にポインタ書換えの対象となるポインタのオフセットを提供する。

2.2 スキーマ登録

本システムでは C++で宣言される永続オブジェクトの型を次の 3 タイプに分類した。

(1) int、char などの C++基本型

(2) Collection、Set などの DBMS 既定義クラス

(3) class、union、struct 宣言で定義されるユーザ定義クラス

(1)(2) の型はデータベース生成と共にスキーマが登録されるが、ユーザ定義クラスはデータベース生成後にスキーマが登録される。

ユーザ定義クラスのスキーマ登録には「スキーマ・ローダ」と呼ぶスキーマ解析ツールを用いる。ユーザはスキーマ登録したいクラスが記述されたヘッダファイルのみをスキーマ・ローダに入力する。スキーマ・ローダは入力されたヘッダファイルに定義されているクラスについて解析を行い、データベースへのスキーマ登録を行う。

3 スキーマの論理構造

本システムのスキーマ論理構造は以下に示す 2 つの点を特徴としている (図 1)。

(1) スキーマはクラス定義情報を持つ schema 部と導出型及びコンテナを管理する type 部から構成される。

(2) スキーマ／メタ・スキーマ／メタ・メタ・スキー マの 3 階層スキーマから構成される。

まず、(1) の特徴について説明する。

スキーマは schema 部と type 部によって構成される。schema 部はクラスの種別、メンバ構造、メンバサイズなどのクラス定義情報を保持する。これはスキーマ本来の機能であり、オブジェクトを生成する際に必要な情報となる。type 部には 2 つの管理機能がある。1 つはポインタ構造や配列構造を識別する導出型管理機能である。もう 1 つは schema 部の情報をもとに生成されたオブジェクトを管理するコンテナ管理機能である。

Development of the OODBMS

Takehiko Tsuchiya, Ryuji Wakizono, Toshikazu Kawamura, Tatsuji Tanaka

Heavy Apparatus Engineering Lab., Toshiba Corp.

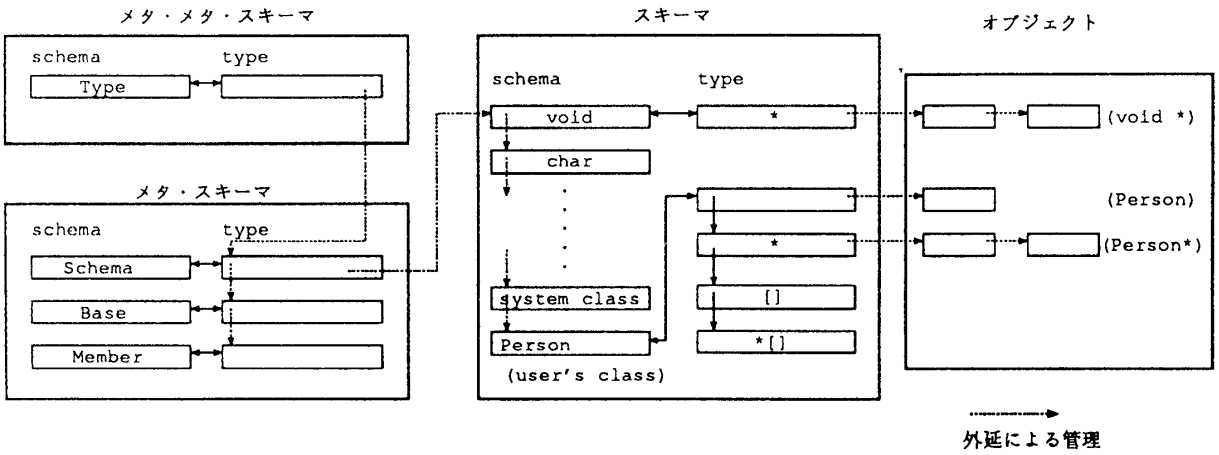

図1　スキーマ構成

次に (2) の特徴について説明する。

本システムでは永続オブジェクトは 3 階層のスキーマ構造により生成、管理される。永続オブジェクトを管理するスキーマ層、スキーマを管理するメタ・スキーマ層、メタ・スキーマを管理するメタ・メタ・スキーマ層の 3 階層である。

スキーマ層では基本型のスキーマとクラスのスキーマという異なる構造のスキーマが管理される。ここで本システムでは基本型のスキーマを「構造的にはメンバを持たないクラスのスキーマと同等である」と考え、クラスのスキーマ情報を統一スキーマ (メタ・スキーマ) として定義する事とした。

メタ・スキーマ層ではクラスの構造を表す情報となる Schema(型の種類)、Base(基底、派生)、Member(メンバ構成) という 3 つのメタ・スキーマが定義されてれる。この情報によりスキーマが定義される。また、このメタ・スキーマにより定義されたスキーマは Schema の type 部によって管理される。

そして、これらのメタ・スキーマは Type というメタ・メタ・スキーマにより定義、管理されている。メタ・メタ・スキーマは全ての永続オブジェクトの起源となる。

以上のようにスキーマ内部にコンテナ管理機能を持つことによりスキーマの管理方式は永続オブジェクトと同一の管理方式とする事が可能となった。つまりスキーマ操作は永続オブジェクト操作と統一的に行う事ができる。

4 スキーマ操作

本システムではスキーマ操作をアプリケーション・インタフェース (API) としてユーザに提供し、アプリケーション・プログラムからのスキーマ登録やスキーマ変更を可能にしている。

図 2 にスキーマ操作に関する API の例を示す。

```
//クラス、メンバ、基底クラスの生成、削除
sc_Class* uClass =    db->createClass ("class_name");
sc_Member* member  = uClass->createMember("member_name");
 uClass->remove();
member->remove();

//スキーマの検索、取得
sc_Class* uClass =    db->getClass("class_name");
sc_Class* uClass =    db->getFirstClass();
sc_Class* uClass = uClass->getNextClass();

//メンバの設定
int flag = member->setSize(short asize);
int flag = member->setOffset(short aOffset);
```

図2　スキーマAPI

5 まとめ

スキーマの登録、変更等の操作をオブジェクト操作と統一的に行うためのスキーマ管理方式について示した。基本言語である C++ で定義できる型を全てサポートする必要があるが、現段階では複雑な導出型やテンプレートのスキーマ解析はサポートしていない。これについては今後開発を進めていく。

また柔軟なスキーマ操作を行える API の実現はスキーマ進化へのステップとして位置付ける事ができるが、スキーマ進化と呼べる機能を実現するにはまだ多くの問題があり、その解決は今後の課題である。

参考文献

[1] 脇園竜次、土屋武彦他：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] M. A. Ellis, B. Stroustrup: The Annotated C++ Reference Manual, Addison-Wesly, 1992.