オブジェクト指向データの関係データベースによる一保存、検索、更新手法

2R−8

森本 康彦, 福田 剛志, 何 千山, 小坂 一也

日本アイ・ビー・エム株式会社 東京基礎研究所

1 はじめに

より高機能なアプリケーションが求められるに従い、データベースに保持すべきデータもより複雑な構造を持つ必要性がでてきた。近年、オブジェクト指向モデルに従って、その操作法とともに構造化されたデータ（オブジェクト）は、複雑で機能的な実体を簡単に管理することが容易なためひろく使われている。

一方、データベース上のデータは多くのアプリケーションまたはユーザに広く利用され得なければならない。この点においてはオブジェクト指向モデルに従って構造化されたデータは、特定のアプリケーションに依存して作られるため、高機能ではあるがデータ自身は広く共有するのは難しい。逆に、従来データベース上のデータとして広く使われている関係モデルのデータ（リレーション）は共有しやすいが表現力にかける。

そこで我々は両者の特徴を生かし、データは関係モデルに従って保存し、それをオブジェクトとして利用するオブジェクトサーバを開発し、それに基づくオブジェクト指向環境COSMOSを構築した[1]。したがって本環境では、データはすべて単純なリレーションの形で保存されるため、ひろくその一部あるいは全部を多くのアプリケーションで利用できる。また同時に必要に応じて、データベース上の関係データをオブジェクト化して利用することも可能である。

本論文では、この両モデル間のデータおよび操作の変換と、その技術を利用して構築したオブジェクトサーバの概要について述べる。

2 COSMOS 環境の概要

我々は異機種のコンピュータがネットワークで結合された環境においてマルチメディアなどの複雑なアプリケーションの開発を支援するオブジェクト指向環境COSMOS(COmmon Service for Multimedia ObjectS) を構築している。

2.1 COSMOS の構成

COSMOS 全体の構成を図 1に示す。COSMOS 環境は大きく分けて5つの構成要素からなっており、それぞれ (1) アプリケーションから直接利用可能なマルチメディア・ライブラリ、(2) それらを実現するクラスを記述するための C++インタフェース、(3) 各クライアントとサーバの間でオブジェクトを転送するためのオブジェクト・マネジャ、(4) オブジェクトをリレーションに変換するためのデータベース・インタフェース、そして、(5) 基本的なデータ管理を行う関係データベース管理システム (RDBMS) である。

現在の構成ではサーバとして IBM RS/6000*上で稼働する

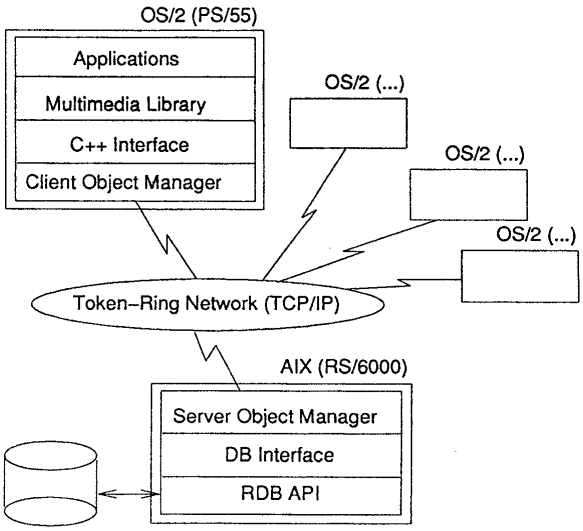

図 1: COSMOS の構成

AIX*を、そしてクライアントとして IBM PS/55*上で稼働する OS/2*を採用している。また、RDBMS には現在米国 IBM で開発中の拡張可能 RDBMS である Starburst[2] を使用している。

2.2 データベース機能

クライアントは COSMOS の C++インタフェースをとうして ObjectManager の管理のもとでサーバのデータベース機能を利用する。各アプリケーションは C++上でデータベース Handle をインスタンシエイトしてデータベースにアクセスする。この際、データベースに書き込まれたり、読み出されたりするオブジェクトはその ID 、検索の対象となるフィールド、検索の対象外のフィールドと分けられた PackedObject として授受される。データベースに送られた PackedObject は次章で述べられるマッピングポリシーに従ってリレーションとして保持される。

このように関係データベース上に書き込まれたデータは、アプリケーション上のデータベース Handle に対し、テーブルをクラスにインスタンス変数をカラム名に置き換えた SQL のストリームを与えることにより、必要なものを読み出すことができる。PackedObject として読み出された場合、そのクラスの PackedObject を引数とする構築子をつかって元の C++のオブジェクトに戻すことができる。

クライアント・サーバ間のオブジェクト転送では、それぞれの ObjectManager がパフォーマンスをあげるためのキャッシュを持つ。また、この ObjectManager が様々なトランザクションの管理を行なう。

A Method of Mapping Data and Operations between Objects and Relations

Yasuhiko MORIMOTO, Takeshi FUKUDA, Qianshan HE, Kazuya KOSAKA

Tokyo Research Laboratory, IBM Japan Ltd.

*RS/6000 は IBM 社の登録商標です。

*AIX は IBM 社の登録商標です。

*PS/55 は IBM 社の登録商標です。

*OS/2 は IBM 社の登録商標です。

ClassID	variables	vtable	constructor	super
#people	name, age,..			

(a) Catalog Table

People Class Table

OID	name	age	...	Long Field

(b) Table

図 2: テーブルとメタテーブル

3 マッピング

C++インターフェース上で定義された永続クラス、及び永続オブジェクトはサーバのデータベース・インターフェースを介して、データベースに登録される。

3.1 クラステーブル

永続クラスはデータベースの CatalogTable の 1 タプルとして登録される。CatalogTable は図 2のように ClassID 、変数名の配列である variables 、メソッドなどの関数にかんする情報をもつ DLL を指すポインタ vtable 、各種コンストラクタ関数へのポインタの配列 constructor 、スーパークラス ID の配列である super からなる。

永続クラスを定義した場合、クライアントからそのクラスの定義情報が送られてくる。その情報に基づいてデータベース・インターフェースでは、検索の対象となる変数を抽出し(画像、音声などはデータベースにはしまわれるが、検索の対象にはならない)、それを variables の配列に登録する。それ以外のメタな情報も同様に、それぞれ CatalogTable の所定のカラムに直されて登録される。

永続クラスが登録される際、さらにこの CatalogTable の情報をもちいて、それぞれの永続クラスのためのテーブルが作成される。各クラスのテーブルは図のように、オブジェクト ID 、検索のための条件となる変数のカラム、そしてそれ以外のオブジェクトの値をしまうための LongField からなる。LongField とは Starburst によって提供される長大データフィールドである [3]。この LongField には画像、音声、そのオブジェクトに含まれるオブジェクトのバイナリーコードなどがある決められたフォーマットに従ってしまわれる。各テーブルの変数用カラムは、CatalogTable の variables 配列と super で示されるクラスをたどって得られるすべてのクラス（基底クラス）の variable 配列に登録されている変数名とその型に従って決められる。このとき C++上の型はそれぞれ Starburst 用のデータ型に変換される。また、継承される変数をすべて 1 つのテーブルに持つことにより、検索時の JOIN 操作を少なく押えることができる。

3.2 永続オブジェクトのタプル化

本環境では永続クラスのインスタンスは永続オブジェクトとなり、その属するクラスのテーブルの 1 タプルとしてデータベースにしまわれる。

アプリケーション上に存在する永続オブジェクトは、保存するためのメッセージを受けとると、そのクラスの定義に従い図 3のような、ID 、SIZE 、検索対象のフィールド値の列、その以外の値およびメタ情報からなる PackedObject という

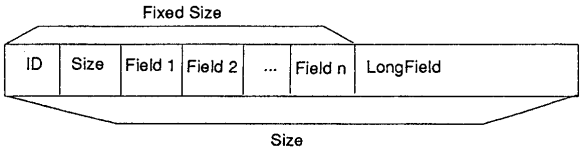

図 3: PackedObject のフォーマット

構造に変換され、サーバに送られる。

サーバのデータベース・インタフェースでは、その PackedObject の ID をもとにそのクラスの CatalogTable を見つけ、対応するクラステーブルにその ID とそれぞれのカラムの値を 1 タプルとして挿入する。検索対象外の値とメタ情報は各テーブルの LongField にそのまま保存される。カラムの型が OID の場合、それは他の永続オブジェクトへの reference を示す。

3.3 オブジェクトの検索

データベース上のオブジェクトをアプリケーション上に読み出す場合、テーブル名をクラス名に、カラム名をインスタンス変数名と置き換えた SQL に準じた検索文をデータベース Handle に送る。サーバはそれを実際のデータベースに登録されているテーブルやカラム名に変換し検索をする。検索されたタプルはそれぞれ必要な PackedObject に変換されクライアントに返される。

4 おわりに

本稿ではオブジェクト指向に基づくアプリケーション上のデータ（オブジェクト）を関係データベースを用いて永続化するシステムと、その際のオブジェクトとリレーション間のマッピング法について述べた。

このようにデータを 1 度単純な構造に落して保持しておくことにより、従来のオブジェクト指向データベースに比べ、永続オブジェクトやその中のデータをあらゆるアプリケーションから共有することができる。さらに、従来の関係データベースにほぼ匹敵する柔軟な（SQL）問い合わせをすることが可能である。

この手法は従来のオブジェクト指向データベースに比べ、タプルとオブジェクト間の変換というオーバーヘットを伴うが、それ以上にデータの共有性、検索の容易さ、問い合わせの柔軟性が増すため、データベースシステムとして評価した場合、優れているといえる。

このシステムをベースとして今後、ヘテロな環境への対応や従来のオブジェクト指向データベースでは難しいとされたビューのサポートを強力にすすめていきたい。

参考文献

[1] 小坂、梶谷、何、森本、福田: オブジェクトサーバとその応用, 情報処理学会データベースシステム研究会報告, 89-3, 1992.

[2] G. Lohman et al, "Extensions to Starburst: Objects, Types, Functions, and Rules," Communications of the ACM, vol. 34, no. 10, pp. 94-109, Oct. 1991.

[3] T. Lehman and B. Lindsay, "The Starburst long field manager," Proc. of 15th VLDB Conference, pp. 375-382, 1989.