1R−7

関係データベース上に構築するオブジェクトサーバのための データモデルとC++インターフェイス

福田 剛志,　森本 康彦,　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

ネットワークで結合された異機種間でマルチメディアオブジェクトを共有することを考えた時、新たに出現するメディアに容易に対応できる柔軟性と強力な質問機能を備えたオブジェクトサーバが必要となる。このため我々は関係データベース (RDBMS) の (1) 成熟した技術に基づく頑健性、(2) 普及度、(3) 質問機能、(4) 柔軟性、(5) 既存のデータベースの活用 などに着目し、RDBMS の上にマルチメディアオブジェクト指向環境COSMOS[5] のオブジェクトサーバを構築している。

しかし、オブジェクトを単純な関係データベースのテーブルに対応づけたのでは、パフォーマンスが上がらず[2]、OODBMS に対する根源的要求を満足できない。このため、パフォーマンスを向上させ、しかも関係データベースの長所を損なわないオブジェクトモデルを設計する必要がある。

また、従来データベース言語とホストプログラミング言語との間に、いわゆるインピーダンスミスマッチ[3]の問題が存在している。OODBMS では、これを避ける言語インターフェイスが重要である。

さらに、既存の OODBMS は、オブジェクトの内部状態はデータベースに格納し共有管理するが、メソッドはアプリケーションと直接リンクし管理しない。このため、マルチメディアのような次々に新しいオブジェクトが出現する環境に対応することができない。

これらの問題を考慮して、本稿では COSMOS のオブジェクトサーバのオブジェクトモデル及び、オブジェクトサーバのコンパイル時、実行時の動作を含む、C++ インターフェイスについて述べる。

2 オブジェクトモデル

COSMOS のオブジェクトモデルは、1章で述べた問題を解決するために、次のような特徴を持つ。

永続性 永続オブジェクトは、データベース内に格納され、オブジェクトを作成したプログラムの終了後も存在し利用される。永続オブジェクト以外のオブジェクト (一時オブジェクト) はプログラムの終了と共に消滅する。

COSMOS が定義した、永続基底クラスである `PObj` から導出されるクラスのインスタンスのみが、永続性を持つことができる。この制約によって、永続性を持たないクラスに対する、不必要な制限をなくし、無駄なスキーマをデータベースに格納することを避けることができる。

プログラマは `PObj` を継承することによって、新しい永続クラスを作ることができる。

オブジェクト ID 全ての永続オブジェクトは、その作成時に COSMOS システムが割り振るオブジェクト ID を持つ。このオブジェクト ID は、COSMOS オブジェクトサーバ内で、唯一であることを保証する。

コンテナオブジェクト (container object) 全ての永続オブジェクトは必ず一つのコンテナオブジェクト (COSMOS システムが定義する、コンテナ基底クラス `CObj` (`PObj` のサブクラス) から導出されるクラスのインスタンス) に所有される。コンテナオブジェクトはデータベースとのロード、セーブ及びロックの単位となる。C++ のオブジェクトは小さいことが多く、細粒度のオブジェクトを独立にデータベースに格納すると大幅な性能低下を招く[2]。しかし一般に、小さなオブジェクトは共有の単位にならず、大きなオブジェクトに含まれる形で使用されることが多い。そこで、アプリケーション間で共有される大きなオブジェクトをコンテナオブジェクトとし、コンテナオブジェクト及びそれが主として参照する多数の細粒度のオブジェクトをまとめて一つのクラスタ (cluster) として扱うことにより、性能の向上が期待できる。

プログラマは `CObj` を継承することによって、新しいコンテナクラスを作ることができる。

メタオブジェクト (meta-object) COSMOS システムが定義するクラス `MObj` (`CObj` のサブクラス) のインスタンスは、永続クラスのスキーマやデータベースへのオブジェクトの格納、問い合わせ方法などを保持するオブジェクトである。

コンテナクラスから導出されるのクラスオブジェクトに対して質問を発することにより、クラスに属している条件を満たすインスタンスをデータベースより取り出すことができる。

データベースオブジェクト (database object) オブジェクトサーバへの指示 (トランザクションのコントロール、メタオブジェクトの取得など) は、データベースオブジェクトへのメッセージを通して行なわれる。

3 アプリケーションプログラムの生成

図1に COSMOS のアプリケーションの生成方法を示めす。

商用のオブジェクト指向データベースのほとんどは、アプリケーションインターフェイスに C++ を採用しているが、独自の言語拡張を行なっているものが多い[4, 1]。

An object model for an RDB-based object server and its C++ interface
Takeshi FUKUDA, Yasuhiko MORIMOTO, Qinshian HE, and Kazuya KOSAKA
Tokyo Research Laboratory, IBM Japan Ltd.

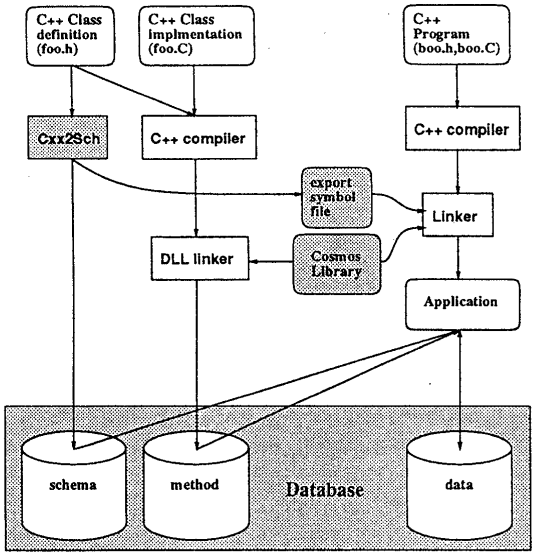

図 1: コンパイル時の処理のながれ

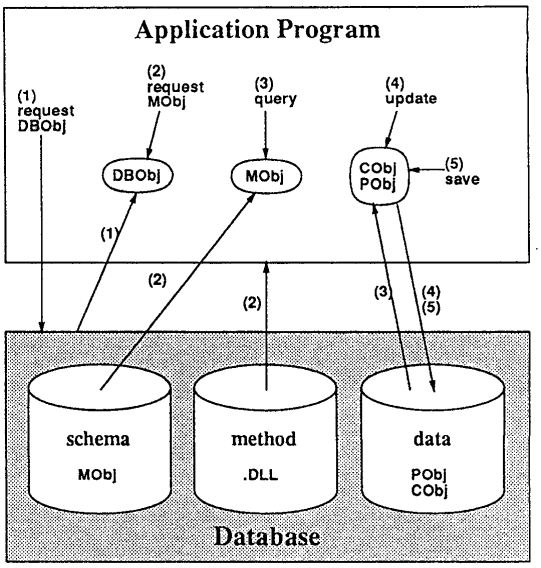

図 2: 実行時の COSMOS システム

COSMOS では 2 章のオブジェクトモデルを、C++ 3.0 を用いて言語拡張なしに実現する。これによって、プログラマは言語の拡張部分を新たに学習する必要がないばかりでなく、アプリケーション開発に特殊なコンパイラを必要としないため、実行環境を変更することが容易である。

また、COSMOS では、従来アプリケーションと静的にリンクされていたオブジェクトのメソッドをダイナミックリンクライブラリとしてオブジェクトサーバが管理し、実行時に必要なモジュールをアプリケーションプログラムとリンクする (4章参照)。これにより、アプリケーションプログラムを作成した後に、それが使用するメソッドの実現を変更したり、新しく作られたサブクラスのオブジェクトを使用することができ、アプリケーションプログラムの安定性と新しいオブジェクトに対する柔軟性が向上する。

4 実行時の動作

図 2 にアプリケーションプログラムとオブジェクトサーバの実行時の関係を示し、その動作の概略を説明する。

1. データベースオブジェクトを要求する。オブジェクトサーバと接続し、データベースオブジェクトができる。
2. データベースオブジェクトに、メタオブジェクトを要求する。クラスの名前または属性を元に、メタオブジェクトを検索し、アプリケーションに返す。この時、そのクラスのメソッドの入ったダイナミックリンクライブラリをデータベースから取りだしアプリケーションとリンクする。
3. メタオブジェクトに、質問を送る。メタオブジェクトが表すクラスに属すオブジェクトの中から、条件を満たすオブジェクトを取りだし、アプリケーションに返す。
4. 永続オブジェクトを変更する。通常の C++ のメッセージパッシングの機構を用いる。
5. 永続オブジェクトをデータベースに格納する。

5 おわりに

現在、COSMOS のプロトタイプ 2 を作成中である。この中で、最初のプロトタイプ [5] では確認できなかった、我々のオブジェクトモデル及び C++ インターフェイスのパフォーマンスとマルチメディアデータに対する表現力、柔軟性などを検証して行きたい。

参考文献

[1] S. Ahmed, A. Wong, D. Sriram, and R. Logcher. A comparison of object-oriented database management systems for engineering applications. Technical report, Massachusetts Institute of Technology, May 1990.

[2] R. G. G. Cattell and J. Skeen. Enginieering database benchmark. Technical report, Sun Microsystems, Apr. 1990.

[3] G. Copeland and D. Maier. Making smalltalk a database system. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 316–325, 1984.

[4] J. V. Joseph, S. M. Thatte, C. W. Thompson, and D. L. Wells. Object-oriented databases: Design and implementation. In *Proceedings of the IEEE*, volume 70, pp. 42–64, Jan. 1991.

[5] K. Kosaka, K. Kajitani, Q. He, Y. Morimoto, and T. Fukuda. オブジェクトサーバとその応用. 情報処理学会研究会報告, volume 92-DBS-89, pp. 19–28, July 1992.