データベースシステム 94−15
（1993. 7. 22）

オブジェクト指向データベースを用いたアプリケーション構築における論理クラス層の導入

大薗 雅弘，楠見 雄規，宮部 義幸

松下電器産業 情報システム研究所

オブジェクト指向データベースは，文書処理，マルチメディア，CAD などの様々な分野で利用され始めている。しかし，第一世代のオブジェクト指向データベースでは，データベースのスキーマ設計が困難なうえ，データベーススキーマの変更によってそれまで利用してきたデータベースの再構築が必要となるという問題点があった。本稿では，これらの問題点を解決するために，オブジェクト指向データベースが扱う物理スキーマの上に，論理クラス層を設けることで，アプリケーションプログラムからはデータの論理的な意味のみに着目したスキーマ定義を行なう方式について述べる。また，オフィスアプリケーションで扱われるデータに対する論理クラス層と，それを用いたアプリケーション開発事例を紹介する。

Logical Class Layer for Application Development in an Object-oriented Database Management System

Masahiro Oashi, Yuki Kusumi and Yoshiyuki Miyabe

MATSUSHITA ELECTRIC INDUSTRIAL CO., LTD.
Information Systems Research Laboratory
1006 Kadoma, Kadoma, Osaka 571, Japan

Object-oriented Database Management System (OODBMS) has been used in Manipulation of Documents, Multi-media and Computer-aided design (CAD). However, the first generated OODBMS are difficult to design the database schema and needs to rebuild the database when the database schema has been changed.

So, we introduce a logical class layer for Application Development in an Object-oriented Database Management System. And we describe the object-oriented application development software that has the logical class layer for office applications. It also reports that the example of the office application are developed on this software.

1 はじめに

近年，オブジェクト指向データベース管理システム（Object-Oriented Database Management System, 以後 OODBMS）の利用事例が増大しつつある。一般にオブジェクト指向データベースは，

- 扱うことのできるデータ構造の柔軟性，
- データ検索の高速性，および
- バージョン管理やリンクなどの多機能性

を特徴としており，特に，従来の関係データベース管理システム（以後 RDBMS）が適用されていなかった文書処理・マルチメディア・CAD などの分野での活用が著しい。

しかし，OODBMS では従来の DBMS より複雑なデータ構造が扱われるようになったためにスキーマ設計が困難であるという問題がある。また，第一世代の OODBMS ではスキーマ設計の変更にともなってデータベースの再構築が必要となるという問題もある。

本稿では，OODBMS 中のデータの物理的な格納形態を吸収する**論理クラス層** (logical class layer) を導入することで，既存の OODBMS を使いながらも，この問題を解決する手法について述べる。さらに，著者らの開発した[1]マルチメディアソフトウェア開発環境 ActivePage では，文書全般を表すのに十分な汎用性を備えた論理クラス層を導入している。

以下では，2章でオブジェクト指向データベースのスキーマ変更に関連する研究を概観し，次に3章で論理クラス層とその導入の効果に触れる。4章で ActivePage が採用した論理クラス層について解説する。さらに 5章ではアプリケーション開発事例を紹介し，最後に 6章で，本稿のまとめと今後の課題について述べる。

2 過去の研究

OODBMS[2], [3] は，オブジェクト指向プログラミング言語における永続的なオブジェクトを実現するためのシステムとして発展してきた。このため，そのスキーマ定義はオブジェクト指向プログラミングにおけるクラス定義と同程度の困難さを伴い，その方法論の確立が望まれている。また，アプリケーションシステムが稼動してからのスキーマの変更への対応も重要な課題となっている。

従来の OODBMS のスキーマ定義に関する研究はこれらの観点に基づいて為されており，大きくは以下の二つの成果を挙げることができる。

仮想クラス　仮想クラス [1],[6] は，RDBMS のビューを OODBMS に適用した手法であり，定義されたスキーマを柔軟に再利用することを可能とするものである。具体的には，全アプリケーションを包含するように定義された実クラスから，各アプリケーションごとに不要な属性やメソッドを隠蔽した仮想クラスを導出し，実クラスをスーパークラスとしてアクセスする。このような仮想クラスの導入によって，一度クラス定義をおこなっておけば，それを利用した別のクラス定義を容易に行なうことができる。しかし，実クラスの定義の困難さが本質的に解決されたわけではない。

スキーマバージョニング　OODBMS は，物理的にはプログラムの内部のデータ構造を二次記憶上に格納することによって実現されるものであるため，一般にはスキーマ定義の変更によって二次記憶上のデータを再構築する必要がある。スキーマバージョニングとは，複数のバージョンのスキーマを扱うことを可能にすることでこの問題を回避するものである。

スキーマバージョニングには，スキーマ全体を一つの単位としてバージョン管理する方法 [4] と，各クラスの変更をそれぞれバージョンオブジェクトとして管理する方法 [5] がある。前者には，全クラスのオブジェクトのコピーがバージョン毎に存在し，データを多重に持つという問題がある。後者には，クラスの生成・削除に対して一貫性を保証するために，バージョンの異なるオブジェクトを管理するための複雑な処理が必要になるという問題がある。

スキーマバージョニングに関する最近の注目すべき成果としては，ユニバーサルスキーマを導入

[1]米国 Gain Technology 社との共同開発

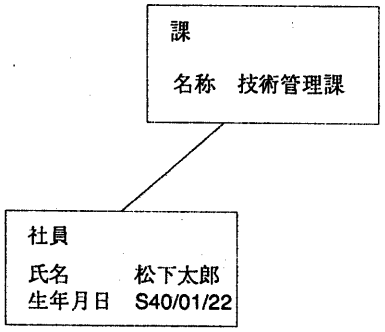

図 1: 組織のデータモデリング

する手法 [7] がある。この手法では，ユニバーサルスキーマを導入することで，異なるバージョンの間のデータ共有を可能としている。

3　オブジェクト指向データベースにおける論理クラス層の導入

前章で述べた OODBMS のスキーマ定義における二つの課題，すなわち，スキーマ定義の困難さとスキーマ変更に伴う諸問題は，アプリケーションプログラム開発における OODBMS の導入の弊害となっている。

OODBMS はオブジェクト指向プログラム言語におけるオブジェクトインスタンスを二次記憶に格納するシステムであるため，一般に OODBMS のスキーマ定義では，データがどのように格納されるかを，ベースとなるオブジェクト指向プログラミング言語と同程度の詳細さで記述する必要がある．

以下では，具体的な例を用いてスキーマ定義の難しさを説明する。

例 1　組織構造を表わすオブジェクトにおけるスキーマ定義を考える。この組織の中の課と社員のデータ構造を図 1 に示す。これらのオブジェクトのスキーマ定義を C++ 風の言語で記述すると次のようになる。

```
class   課 {
char    名称 [256];
link    社員 [] <-> 課;
}

class   社員 {
char    氏名 [256];
char    生年月日 [10];
link    課 <-> 社員 [];
}
```

ただし，link 社員 [] <-> 課 は，オブジェクト間のリンクを表し，課に社員が複数人属していることを意味している。

このように定義された課と社員のスキーマに対して，部というクラスを課の上位組織として追加する場合を考える。この場合，

```
class   部 {
char    名称 [256];
link    課 [] <-> 部;
}
```

のようなデータ構造を持つクラスを導入する必要がある。このようなクラス定義は，階層構造を管理するためのデータ構造とメソッドを二重に定義していることになり，オブジェクト指向プログラミングの利点である再利用性を活かしていない。また，再利用性を活かしたクラス定義にするためには，既存の課と社員のクラス定義を大幅に変更する必要があり，経験のないアプリケーションプログラマにとっては困難である。

また，たとえば，社員クラスに住所データを追加する場合，

```
char    住所 [256];
```

のような定義を追加する必要がある。このような簡単な定義でさえ，256 文字という物理的な記述が含まれるが故に，後になって文字列長を拡張するためのスキーマ変更が必要となる可能性がある。□

上の例から分かるように，OODBMS では，複雑なデータ構造を扱うことが可能であるが故にスキーマ定義の記述も複雑となる点が，アプリケーションプログラマの負担となっている。また，物

理的な記述がスキーマの再定義の要因となりうる場合もある。さらに，このような場当たり的なクラスの追加・変更は，オブジェクト指向プログラミングの本来の利点である再利用性の高さを阻害する要因となりうる。

これらの課題に対する我々のアプローチは，

- 対象とするアプリケーションの問題範囲を限定し，その範囲で扱われるデータに対しては，アプリケーションプログラマにデータの物理的な格納形式を意識させない。たとえば，文字列の長さや，イメージ・音声の形式，リンクを張る場合のリンク先の型に関する詳細をシステム側で隠蔽する。
- スキーマの変更に対しては，システムでバージョニングを行わず，プログラマがメソッドを記述することで対処する。たとえば，クラス中に新たな属性が追加された場合，変更前のデータに対して，正常に動作することを保証することは，プログラマに一任する。

というものである。

具体的には，OODBMS 中でのデータの物理的な格納形態を吸収する論理クラス層を導入する。論理クラス層は，アプリケーションプログラマの為にシステムで予め用意する汎用クラスの集合であり，アプリケーションプログラムにおいて

- データの大きさの変更
- フィールドの追加・削除

を行なっても，OODBMS のスキーマ変更を生じさせない性質を持つ。アプリケーションプログラマは，この汎用クラスに対して論理的な意味のみを与えることで，汎用クラスのサブクラスとしてクラスを定義する。

実際には，すべてのアプリケーションをカバーするような論理クラス層があるわけではない。アプリケーションの対象とする問題領域を設定し，それに応じて論理クラス層として，「階層構造」「イメージ」「音声」などの汎用クラスを定義することになる。たとえば，「階層構造」という汎用クラスを定義すると，アプリケーションプログラマの目的に応じて，会社の組織構造の「部」や「課」

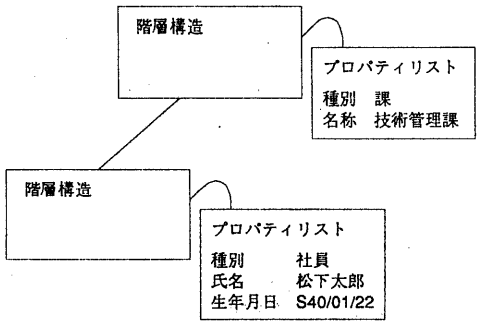

図 2: 論理クラス層を用いたスキーマ定義例

を表現する目的としても，「書類」を束ねる「フォルダ」の目的としても用いることができる。

上記のような論理クラス層の実現の一手法として，可変長のプロパティリスト [3] の応用が考えられる。プロパティリストは，オブジェクトの属性を名前と値の対で表したリストである。ただし，既存の OODBMS では，プロパティリストの中のデータに対してはシステムに備わっている問合せ言語での検索が不可能なものもあり，その場合には検索性能の保証が課題となる．

例 2　例 1 のデータを表現するために十分な論理クラス層として，階層構造を扱う汎用クラスを定義する。そのスキーマ定義は以下のようになる。

```
class 階層構造 {
list    プロパティリスト [];
link    階層構造 [] <-> 階層構造 [];
}
```

この汎用クラスを用いて，例 1 のデータ構造を定義した結果を図 2 に示す。課オブジェクトは，階層構造を扱う汎用クラスのプロパティリストに課の名称を格納する。また，社員オブジェクトは，汎用クラスのプロパティリストに氏名，生年月日を格納する。

さらに，課の上位組織として部というクラスを定義するには，階層構造を扱う汎用クラスのプロパティリストに部の名称を格納するだけで，定義することができる。これは，経験のないアプリケー

ションプログラマでも再利用性を活かしたクラス定義を行なうことができることを意味する。

また，社員オブジェクトに住所データを追加する場合には，プロパティリストに住所を加える。プロパティリストは可変長のデータを扱うことができるので，住所データが最大何文字の文字列であるかを予め知る必要はない。□

論理クラス層に属するクラスが既に定義されている環境においては，アプリケーションプログラマはオブジェクトの物理的な格納形態を意識する必要がない。アプリケーションプログラマは，汎用クラスのサブクラスの論理的な定義だけを行なうことができる。すなわち，OODBMS を利用する上での障害となっている適切なクラス定義が既に定義されているので，経験のないプログラマでも OODBMS を利用したアプリケーションを作成することができる。

また，オブジェクトの内部構造の物理的な定義と，論理的な定義を分離することにより，オブジェクトの論理的な定義を変更してもデータベースを再構築する必要はない。ただし，論理的な定義の変更に伴う一貫性の保証はアプリケーションプログラマに任される。このことは，アプリケーションプログラマが持たせたデータに対する意味付けに応じて柔軟に対処できることを意味する。

4 マルチメディアソフトウェア開発環境 ActivePage

4.1 ActivePage の論理クラス層

著者らは，米国 Gain Technology 社と共同で，オブジェクト指向マルチメディアソフトウェア開発環境 ActivePage を開発した [8]。ActivePage は，テキスト，グラフィックス，オーディオ，アニメーションなどのマルチメディアを取り扱うことのできる，オブジェクト指向に基づいた，アプリケーションプログラムの開発環境である。オブジェクトは，米国 Objectivity 社のオブジェクト指向データベース Objectivity/DB[2] に格納する。

[2]米国 Objectivity 社の登録商標

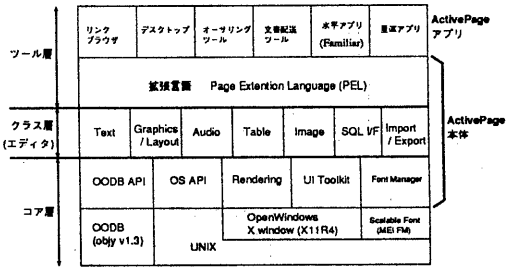

図 3: AvtivePage のソフトウェア構成

ActivePage の構成を，図 3 に示す。

ActivePage には，次の 3 つの特徴がある。

オフィスデータ構造を表す論理クラス層の実装 広義のドキュメントをベースにしたアプリケーション作成に最適な論理クラス層を実装し，アプリケーション開発効率の極めて高い，オブジェクト指向の開発環境である。

ActivePage では，オフィスにおける様々なデータを自然に OODBMS で管理するために，3 章の論理クラス層として，グラフィク，テキスト，オーディオ，ページ，ドキュメント，フォルダというクラスを用意している（図 4）。オフィスに存在する様々なデータの大半は，印刷物である。そして，紙に印刷されたデータをファイルに閉じて書架に入れて管理していた。ActivePage のマルチメディア文書オブジェクトとそれを管理するためのフォルダオブジェクトは，概念的に従来の紙による情報管理と一致するようになっている。

広範囲な機能の実現 ユーザインタフェース，ネットワーク，文書作成，表示·印刷，データ管理，オーディオ，アニメーション，既存 RDBMS とのインターフェース等，広範な機能をカバーする開発環境である。

柔軟な拡張性の実現 操作性，データの互換性が保たれた，特定業務アプリ，汎用アプリ，マルチメディアアプリなど，様々な種類のアプリケーションが容易に構築出来る開発環境である。

任意のオブジェクトはプロパティリストという

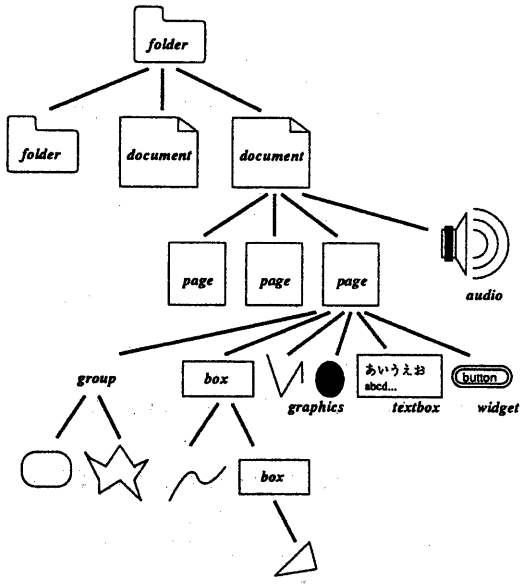

図 4: AvtivePage のオブジェクトの構成

名前と値からなるリストを持っている。各アプリケーションごとに，このプロパティリストにオブジェクトの論理的な属性を設定することによって，ActivePage の提供する汎用クラスのサブクラスを定義することができる。プロパティリストの OODBMS への物理的な格納形態は，ActivePage のシステムが定義するので，アプリケーションプログラマはデータの格納形態を意識する必要はない。また，プロパティリストに対する値の設定や参照は，拡張言語 (PEL) を用いて簡単に行なうことができる。

4.2 アプリケーション開発を支援する機能

検索機能 ActivePage では，プロパティリストを用いてオブジェクトの論理的な属性を定義することができ，SQL 風のデータベース問合せ言語を独自に提供している。この機能により，ユーザないしアプリケーションプログラムによって指示された条件に適合する属性値を持つオブジェクトを，OODBMS から検索することができる。しかし，プロパティリスト中のデータの検索は，OODBMS の機能としてはサポートされていない。また，可変長のプロパティリストを用いてデータを格納すると，直接データ構造を定義する場合に比べて検索効率が劣る。この点を改善するために，ActivePage ではプロパティリスト中の各データに対してインデックスを生成する機能を提供している。

さらに，アプリケーションプログラムにおいて階層的なデータ構造を表現する場合には，ActivePage のフォルダとドキュメントの階層構造を用いることができる。このようなアプリケーションでは，階層構造を探索し，その結果得られたオブジェクトに対する処理を記述することも多い。そのような目的に供するために，論理クラス層中の汎用のクラスに対する，深さ優先と幅優先の探索のメソッドを定義してある。

ユーザインターフェースの開発支援 ActivePage が定義する汎用クラスは，アプリケーションプログラムのユーザインターフェース開発を支援する機能を持っている。たとえば，フォルダは，フォルダの中のドキュメントやフォルダを，ドラッグ&ドロップの操作でリンクを付け換える機能を提供している。この機能を用いれば，フォルダの階層構造を用いて人事データを表現した場合，アプリケーションプログラマが特別なプログラムを作成しなくても，ドラッグ&ドロップの操作で部門間の人事異動を行うことができる。

5 アプリケーション開発事例

本章では，ActivePage 上でのアプリケーション作成事例として，人事管理システムについて説明する。

オフィスアプリケーションにおける階層関係データの一つである組織情報に着目し，OODBMS に情報を格納する人事管理システムの試作を行った。この人事管理システムは，部，課，社員，社員データや顔写真という階層構造を OODBMS に格納し，人の移動や部課の統合といった処理をユーザインターフェースを用いて対話的に実行できるシステムである。

ActivePage が定義している論理クラス層は，フォルダ，ドキュメント，ページ，グラフィックス，

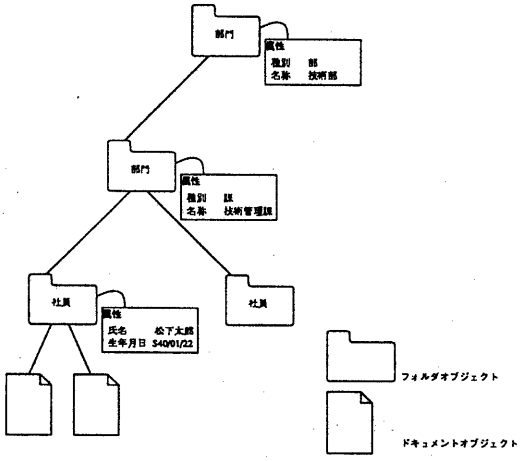

図5: 人事管理システムにおけるデータモデリング

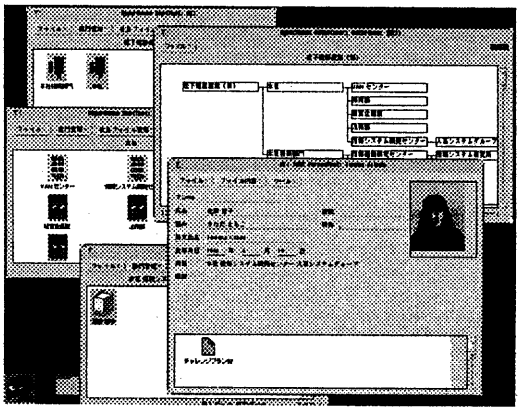

図6: 人事管理システムの画面イメージ

オーディオなど，オフィスワークにおける文書全般を表すのに十分な汎用性を備えている。また，任意のオブジェクトに対してユーザが自由にオブジェクトの属性を設定し，定義済みのクラスのサブクラスとして定義することができる。これらのオブジェクトを用いて現実の世界の様々な事柄を表現することができる。

この人事管理システムでは，組織の中の各部署と社員を，フォルダオブジェクトを用いて表す。そして，

- 組織の中での部門の階層関係
- 社員の部門への所属
- 社員データの中にある様々なドキュメント

の3種類の関係を，フォルダとドキュメントの階層関係に置き換えて表現している。人事管理システムで用いたデータモデリングを図5に示す。このようなデータモデリングによって次の二つの効果が確認された。

実世界の情報を忠実に再現　第一の効果は，実世界の情報を忠実に再現することに成功した点である。たとえば，社員オブジェクトの中には所属部門の名称は記述されていなくとも，その社員が属する部門オブジェクトから取り出すことができる。この場合，ユーザやプログラマは，所属部門に関するデータの更新に関して何も考慮する必要がないのである。この効果はユーザインターフェース面で特に顕著である。ActivePage ではフォルダの中のドキュメントやフォルダをドラッグ&ドロップの操作で移動するための機能を提供している。本システムでもこの機能を適用して，ドラッグ&ドロップの操作で社員オブジェクトを別の部門に異動することが可能である。人事管理システムにおけるフォルダの画面イメージを図6に示す。

情報の電子化の促進　第二の効果は，電子化できるデータの範囲が大幅に広がったことである。社員オブジェクトに対応するフォルダの中には，スキャナから読み込んだイメージデータを含め，任意の ActivePage のマルチメディア文書オブジェクトを格納することができる。これによって，従来電子化されずに紙のまま保存していた文書を既に電子されている文書と共にデータベースで管理することが可能となる。

6　おわりに

ActivePage の提供する論理クラス層により，オブジェクト指向データベースを利用したアプリケーションが簡単に作成できる環境を構築することができた。ActivePage の定義する論理クラス層によって，アプリケーションプログラマは，種々なデータを汎用クラスのサブクラスとして定義する

ことができ，その際にデータの物理的な格納形態に関する詳細を意識する必要がない。しかも，アプリケーションプログラムでのスキーマの変更に対して，データベースの再構築を不要にしている。

オフィスにおけるデータ構造には，関係データモデルで自然に表現できるデータがある。そのようなデータを ActivePage で扱うには，データを既存の RDBMS に置いて SQL I/F 機能を用いてアクセスする方法と，ActivePage 上のフォルダやドキュメントの属性として実現する方法がある。前者は RDBMS が別途必要になるものの，現時点では検索性能において後者よりも勝っている。今後は，関係モデルを表現する論理クラス層のクラス定義を性能とデータ表現の両面から検討する必要がある。また，アプリケーションプログラムから見た RDBMS と OODBMS の違いを無くすことも課題である。

参考文献

[1] S. Abiteboul and A. Bonner: Objects and views, In Proceedings of the ACM SIGMOD, pp. 238-247 (1991).

[2] M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, and S. Zdonik: The Object-Oriented Database System Manifesto, In Proceedings of the 1st International Conference on Deductive and Object-Oriented Databases (DOOD 89), pp. 40-57 (1989).

[3] R.G.G. Cattell: Object Data Management, Addison-Wesley (1991).

[4] W. Kim and H. T. Chou: Versions of Schema for Object-Oriented Database, In Proceedings of International Conference on Very Large Databases, pp. 148-159 (1988).

[5] H. J. Kim and H. F. Korth: Schema Versions and Views in Object-Oriented Databases, In Proceedings of International Conference on IPSJ, pp 277-284 (1990).

[6] K. Tanaka, M. Yoshikawa, and K. Ishihara: Schema Virtualization in Object-oriented Databases, In Proceedings of 4th International Conference on Data Engineering, vol. 4, pp. 23-30 (1988).

[7] 田島敬史，加藤和彦，益田隆司：オブジェクト指向データベースにおけるスキーマ・バージョニングの実現法について，情報処理学会 第 91 回 データベースシステム研究会資料 (1992).

[8] 辻村敏，宮部義幸，楠見雄規，井上信治，南方郁夫:オブジェクト指向マルチメディアソフト開発環境 ActivePage，情報処理学会 第 94 回 データベースシステム研究会資料 (1993).