オブジェクト指向データモデルにおけるオブジェクトの多重表現[1)]

An Object Oriented Data Model for Multiple Representation of Object Semantics

石丸　知之
（株）東芝　青梅工場
Tomoyuki ISHIMARU
TOSHIBA Corp., Ome Works

植村　俊亮
奈良先端科学技術大学院大学
Shunsuke UEMURA
Advanced Institute of Science and Technology, Nara

あらまし

本論文では、オブジェクト指向データモデルを拡張した「多態オブジェクト」データモデルを提案する。多態オブジェクトモデルの目的は、マルチメディアデータを自然にデータベース内に表現することである。

マルチメディアデータの一つの特徴は、実世界の実体の多様な表現である。多態オブジェクトモデルでは、実体をオブジェクトとして理解し、そのオブジェクトに複数の完結した表現を与えることのできるモデルを提案する。このようなオブジェクトを多態オブジェクトと呼ぶ。

多態オブジェクトモデルでは、一つのオブジェクトが複数のインスタンスを表現としてもちうる。クラスはインスタンスの形式を定義する。同じオブジェクトの複数のインスタンスはクラス名で識別する。複数のインスタンスの属性が同一のスーパークラスで定義されているときには、その属性は同一の意味をもつとみなし、インスタンスの間で値を共有し一貫性を達成する。

多態オブジェクトの実現例として、現在試作中のデータベース管理システム「沙羅」のアーキテクチャについて述べる。

Abstract

This paper proposes "Poly-Aspect Data Model" to represent real world multimedia data naturally.

One of the major characteristics of multimedia data is its multile representation of one real world entity. In this model, each such entity is captured as an object, which may have more then one complete representation. Each representation is modelled as an instance. An object can have multiple instances, each of which correspomds to one representation. In this sense, this model is an extention of object oriented data model.

Class is a definition of instances and used distinguish each instance in an object. When classes of instances in an object have same superclass as its ancestor, attributes defined in the superclass are shared in the object. This mechanism guarantees the consistency among attributes.

A database management system "Sarah" is also described as an experimental implementation of Poly-Aspect Data Model.

---

1) 電子情報通信学会論文誌掲載予定

１．まえがき

本論文は、マルチメディアデータの特徴である多様な表現を自然にデータベース中に表現するデータモデルについて論じる。

マルチメディアデータの一つの特徴は、実世界の実体に対して多様な表現が可能なことである。実世界では一つの実体であっても、認識やモデル化の方法により複数の表現で表すことが可能である。多態オブジェクトモデル（Poly-Aspect Data Model）では、実世界の実体をオブジェクトとして理解し、そのオブジェクトに複数の表現、たとえば、文字列による表現とイメージによる表現とを、それぞれ完結したものとして与えるモデルを提案する。このようなオブジェクトを多態オブジェクトと呼ぶ。

多態オブジェクトモデルでは、実世界の一つの実体が複数の表現をもつことを、一つのオブジェクトが複数のインスタンスをもつことで表す。インスタンスは実世界の表現の一つで、その仕様はクラスで定義される。オブジェクトを呼び出すときには、対象となるインスタンスを、それを定義しているクラス名で指定する。また、複数のオブジェクトに結び付けられたすべてのインスタンスを、動的に一つのオブジェクトに構成し直すことができる。これをオブジェクトの融合という。このときクラスの定義の変更やクラスに対する操作を必要としない。一つのオブジェクトの複数のインスタンスの属性が同一のクラスで定義されているときには、それらのインスタンスの属性は同一の意味を表現しているものとみなし、インスタンスの間で値を共有する。この機構によって表現の値としての属性値の一貫性を保証することができる。

以下、２．では、マルチメディアデータの分析と本モデルの特徴について論じ、３．では関連研究と比較する。４．で多態オブジェクトモデルを提案する。５．では、このデータモデルの実装の提案として、データベース管理システム「沙羅」について述べる。６．では、結論を述べ、今後の課題に触れる。

２．マルチメディアデータの表現と多態オブジェクトモデル

マルチメディアデータは、データの表現が多様であるという特徴をもつ。例えば、世界の人物に対して、写真のようなイメージ（image）の表現と、履歴のような文字データによる表現とが存在する。さらに、一口にイメージデータといっても、ビットマップ、ファクシミリ用圧縮形式、カラー圧縮形式などの形式がすでに存在し、さらにいろいろなものが提案されている。

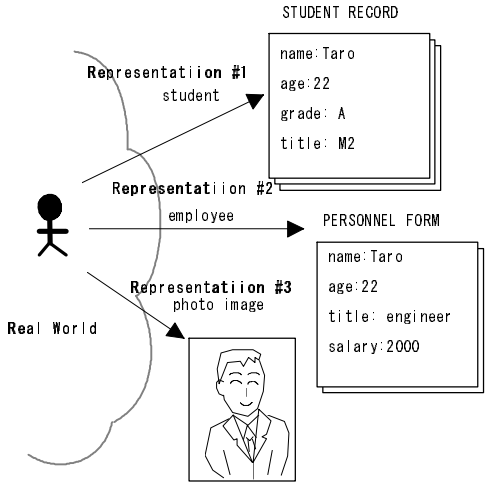

図１実世界の実体の複数の表現
Fig. 1 Multiple representation of an entity in real world

このように多様なデータに対して、従来の CODASYL 型や関係型のデータベース管理システムでは、あらかじめシステムが用意したデータ型しかデータベースに使用できないので、柔軟に対応できないことが知られている。

オブジェクト指向データモデルでは、利用者が新しくデータ型をオブジェクトとして定義することができ、オブジェクトの実現の詳細を隠ぺい（カプセル化）できる。カプセル化は、実現に依存しないインタフェースを用意し、オブジェクトの内部の詳細を利用者に意識させないという意味で、多様なデータを統一的に扱おうとするときに有効である。また、マルチメディアデータは複数の表現が混在し、個別の表現の間で複雑な関連が存在している。オブジェクト指向データモデルでは、オブジェクト内の属性として別のオブジェクトを参照して、実体間の関連を表現できるため、複雑な関連を表現することが容易である。

これらの性質からオブジェクト指向データモデルがマルチメディア管理のためにふさわしいと考えられている[1]。

増永[5]が主張するように、マルチメディアデータの特徴は、実世界の一つの実体に対して、複数の表現が存在し得ることである（図１）。このようなデータをデータベースに取り込むときに、実人物の学業成績（STUDENT RECORD）、従業員記録（PERSONNEL FORM）、そして写真（image）の各々の表現は、従業員記録が写真の表現をもつといった所有の関係や、写真が従業員記録のサブクラスであるといった継承の関係では、自然に把握できない。従業員記録と写真は表現しているものが実世界で関係をもっているだけで、表現として直接関係がないからである。例えば便宜的に、従業員記録というオブジェクトのなかに写真

という別のオブジェクトが含まれていると考えることができる。しかし、これでは、従業員記録オブジェクトの表現が増えたに過ぎない。また、人物というオブジェクトの属性として、従業員記録と写真とを考える立場もあり得るが、これは、人物オブジェクトを実世界の実体の代表として便宜的に想定していることになり、自然ではない。これらの写真や従業員記録といった表現は、実世界の一つの実体の異なる表現であって、一つのオブジェクトの二つの側面であると考えるべきである。

従来のオブジェクト指向データモデルでは、オブジェクトは一つの表現しか許されないことが多く、ただ一つのクラスにだけ属するという制限がしばしば見られた[2][3][4]。

実世界の存在に対する表現をデータベース内に複数もつことは自然である。データベース管理システムも一つの実体に対して複数の表現を合理的にもつ機能が必要である。本論文はそうした機能を実現する多態オブジェクトモデルと、その実装について述べる。

もちろん、異なる表現の間でも、表現の要素となる値の共有が必要である。例えば、人物を会社員と学生の二つの視点から捉える場合を考える。このときにも、実体の氏名や年齢のような基本的な人物の要素は、表現が異なっていても一貫性を保証すべきである。

一方、マルチメディアデータベースは、高速ネットワークや大容量記憶メディアなどを利用して配布されると想定される。この場合受取側でもデータベースを操作し、新たにデータを追加するといった編集操作が可能であることが望ましい。そのためには、既存のデータベースに制約されないでクラスを定義でき、そのインスタンスを追加できなければならない。

多態オブジェクトモデルの定義は、４．で述べるが、そこでは、オブジェクトのインスタンスが実体の表現結果であり、オブジェクトは同時にインスタンスを複数もつことができる。参照時には、クラス名を指定してオブジェクトのインスタンスを選択する。一つのオブジェクトを構成する複数のインスタンスがクラス階層で共通の祖先をもつときには、共通の祖先で定義された属性を共有する。また、クラス定義の変更なしに、オブジェクトのインスタンスを動的に追加できる。

３．関連研究との比較

オブジェクトに対して複数の表現や意味を割り当てるデータモデルの研究は、すでにいくつか発表されている。

Iris[6]は、もっとも早い時期の提案である。Iris のデータモデルは、抽象データ型をもつ意味データモデルによっており、一つのオブジェクトに複数の型を動的に割り当てることができる。しかし、属性の有効範囲を定義する機能が用意されていなかったため、同じ属性名を複数のクラスで定義すると、属性名の競合が発生した。OpenODB[7]もこの流れをくんでいる。

fashion[8]は、クラスの定義によって、オブジェクトの一部を全体としてみせたり、全体を一部としてみせる機構をもつ。しかしこの操作は、クラスとして定義したオブジェクトの属性をもとにしているので、自由度が小さい。

object hiererchies[9]は、委譲 (delegation) に基づいたデータモデルである。このモデルでは、Smalltalk-80[10]などのインスタンスのテンプレートとしてのクラスの概念がない。委譲先は、オブジェクトごとに個別に指定することができ、委譲先は木構造にすることができる。また委譲先を動的に変えることができる。オブジェクトに「複製」のメッセージを送ると、オブジェクトが生成される。このとき、委譲の経路をたどってすべてのオブジェクトを複製する。特定の形式のインスタンスを生成するためには、データベース設計者が決めた特定のオブジェクトをテンプレートとして使用する。

オブジェクトに新たに性質を追加するときに、object hiererchies では上記の委譲の機構を使用して新たに性質を表現するオブジェクトを付加することができる。付加されたオブジェクト間での関連を調べる手段はない。

実際のデータ管理という面からは、object hierarchies のように、オブジェクトごとに委譲先を指定でき、その結果によって振舞いが異なるのは、柔軟すぎて、現実のデータベース応用の構築には向かないと考えられる。

Aspects[11]は、強い型検査機構をもったオブジェクト指向データモデルであり、この検査機構のもとで、オブジェクトに新たに属性や振舞いを拡張する機能を提供している。オブジェクトがほかのオブジェクトと結びつけられるかどうかは、クラス定義で示され、オブジェクト作成時に検査される。そのためクラスの定義の時点で型の定義を見通した完全なクラスを作成する必要がある。また、Aspects では、一つのオブジェクトとみなす属性の共有を明示的に記述する必要がある。

Omega[5]では、データベース内の複数のオブジェクトが、３種類の等値関係で、実世界の一つの実体の表現であることを示すことができる。また、一つのオブジェクトが複数のクラスに属することができる。しかし、各々の表現の間で一貫性を保証する手段は提供されていない。

CLOS[12]では、オブジェクトを定義するクラスを動的に変更することができる。しかし、オブジェクトが同時に複

数の表現をもったり、表現の間で一つの属性を共有したりする機能はない。

「多態オブジェクト」は、こうしたモデルの欠点を補う提案である。Iris と異なり、クラスによって有効範囲を明示的に宣言してオブジェクトを呼ぶので、インスタンスの属性は自動的に識別できる。インスタンスはテンプレートとしてクラスをもち、クラスによって管理される。しかし、オブジェクトはクラス定義と独立に複数のインスタンスをもつことができるので、fashion や Aspect より自由度の高い実体の表現が可能となっている。

多態オブジェクトでは、一つのオブジェクトの複数のインスタンスは互いに完全に独立したものではなく、継承階層に基づいて同じクラスで定義された属性を共有する。つまり、クラスの継承階層によって表される意味概念が同一の属性はインスタンス間で単一の値を使用する。利用者は属性の共有を明示的に記述する必要はない。インスタンスの間で属性の共有という関係が生じる点で、オブジェクトは単なるインスタンスの集合ではない。

また、型検査をオブジェクト定義の時点ではなくて、実際にオブジェクトに対する属性が参照されたり、メッセージを送付する時点で行なうので、他の環境で作成したデータベースに対してクラスを定義し、既にあるオブジェクトに対してインスタンスを追加することが容易である。さらに、このことにより、あらかじめ完全なクラス構造を設計する必要がないため、データベース設計が容易になる。

４．多態オブジェクトモデル

多態オブジェクトモデルは、オブジェクト指向データモデルの拡張である。本節では、多態オブジェクトモデルの定義を与え、例を挙げて説明する。

４．１　多態オブジェクトの定義

多態オブジェクトモデルのの基本要素は、オブジェクト、クラス、インスタンスである。Smalltalk-80[10]の定義と異なり、ここでは、一つのオブジェクトに対して、それぞれクラスの異なる複数のインスタンスが存在し得る。クラスは、インスタンスの枠組みである。

[定義１] インスタンス

インスタンス ｉ の形式は次による。

$$i = \{ n_1=v_1, \ldots, n_n=v_n \}$$

ここで、

a) n_j は属性名で、v_j は属性値、または

b) n_j はメッセージ名で、v_j はメッセージに対応するメソッドの返値とする。(1<=j<=n)

インスタンスは、属性名およびその値の組、またはメッセージ名およびその返値の組の集合からなり、情報の一つの完結した意味をもつ単位である。インスタンスは、実世界の実体の一つの見え方のデータベース内の表現となる。属性名とメッセージ名はインスタンス内で一意である。属性名は値の名札であり、それに対応して属性値をもつ。メッセージ名はメソッドと呼ぶ手続きの名前とみなされて、その実行はいくつかの値を返す。属性名であるかメッセージ名であるかは外部から直接には識別できない。

[定義２] クラス

クラスはインスタンスの形式 (型) を定義する。

クラスは、インスタンスの属性とメソッド、いいかえると、構造と振舞いを定義する。クラスは定義を他のクラスから継承することができる。このとき、定義を参照するクラスをサブクラス、参照されるクラスをスーパークラスと呼ぶ。スーパークラスは複数あってもよい。

[定義３] オブジェクト

オブジェクト ｏ の形式は次による。

$$o = \{ Oid_1, \ldots, Oid_m \} : \{ c_1=i_1, \ldots, c_n=i_n \}$$

ここで Oid_j ($1 <= j <= m$) はオブジェクト識別子、c_k はクラス名、i_k はそれに対応するインスタンスである ($1 <= k <= n$)。

オブジェクトはオブジェクト識別子の集合とインスタンスの集合との集まりである。

オブジェクトは、実世界の実体に対応する。オブジェクトはオブジェクト識別子によって識別する。オブジェクト識別子は、データベース管理システムが生成する。オブジェクトは、複数のインスタンスからなることがあり、それらはクラス名により識別する。一つのオブジェクト中のインスタンスの所属するクラスの間には制約がない。言い替えると、一つのオブジェクトがもち得るインスタンスを、クラスなどでの定義にあらかじめ宣言しておかなくてもよい。

オブジェクトが複数のクラスの表現をもち複数の形態を表すので、この機能を「多態オブジェクト」と呼ぶ。この機能によって、一つのオブジェクトが複数の視点からの表

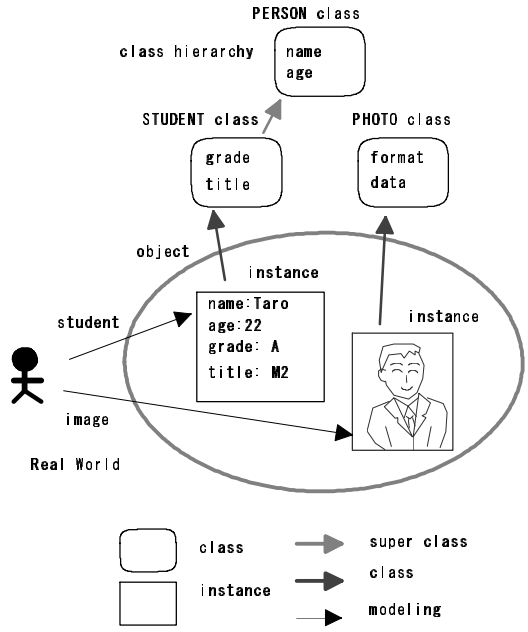

図 2 二つのインスタンスをもつオブジェクトの構造
Fig. 2 Structure of an object with two instances

現をもつことができる。図 2 では、人物を学生とイメージの二つの視点でモデル化したものを一つの多態オブジェクトの二つの表現として取り込んだ例を示している。この例では、STUDENT クラスのインスタンスと PHOTO クラスのインスタンスが一つのオブジェクトに含まれている。

多態オブジェクトモデルでは、一つのオブジェクトに結び付けられたインスタンスを参照する場合に、クラス名を指定する。たとえば、イメージによる表現と文字による表現のそれぞれのインスタンスをもつオブジェクトを参照する場合には、クラス名を指定して、イメージによる表現を示す PHOTO クラスの属性を参照したり、文字による表現を示す STUDENT クラスの属性を参照範囲としたりする。参照範囲は、オブジェクトの中での属性の有効範囲と言い換えることができる。

[定義４] 同一性

オブジェクト識別子 A と B が同じオブジェクトのオブジェクト識別子の集合に含まれるならば、A と B は同一のオブジェクトを示している。

この定義は、オブジェクトが複数のオブジェクト識別子をもつことができ、識別子が異なっていても同一のオブジェクトを指すことを示している。オブジェクト識別子とインスタンスは、n 対 1 の関係にあり、オブジェクト識別子がオブジェクトを一意に識別するという性質は変わっていない。

４．２　多態オブジェクトの参照

多態オブジェクトの参照に関して、以下のように定義する。

[定義５] 継承祖先

クラス A が直接または間接にクラス B を継承するときには、クラス B はクラス A の継承祖先と呼ぶ。

[定義６] 属性値の共有

同一のオブジェクトのインスタンス A とインスタンス B がそれらを定義するクラスで同一の継承祖先で定義された属性値をもつときには、属性値は一つのものを共用する。

多態オブジェクトモデルでは、複数のインスタンスをもつオブジェクトは、それぞれのインスタンスが属するクラスの属性値をすべてもつ。ただし、それらのクラスが同一の継承祖先をもつ場合は、オブジェクトは継承した属性値を共有する。

図 3 の例では、一つのオブジェクトが、STUDENT クラスと EMPLOYEE クラスのインスタンスをもっている。それぞれのインスタンスが PERSON クラスを共通の祖先としてもち、PERSON クラスで定義した属性値を共有している。

Smalltalk-80 などのオブジェクト指向でのクラスの継承は、単に上位のクラスの定義を利用する機構であり、インスタンスの値に関しては、制約を課すものではなかった。

多態オブジェクトでは、同一の継承祖先をもつことはクラスが上位の意味概念を共有できることを示す。この意味概念を表すための機構が属性とメソッドである。一つのオブジェクトが複数の表現をもつことは、複数のインスタンスをもつことで実現する。この複数のインスタンスが同一の継承祖先をもつことは、インスタンスが同一の意味概念をもつことであり、その意味概念は単一のものに一致すべきである。このことから、多態オブジェクトは属性値を共有する。

多態オブジェクトのクラス名によって指定される参照範囲は、関係モデルのビューとは異なる。ビューはスキーマ構造の部分または結合された部分として定義されるの対して、クラス名によって指定されるオブジェクトの参照範囲は一つのオブジェクトの中のインスタンスを選択しているだけである。

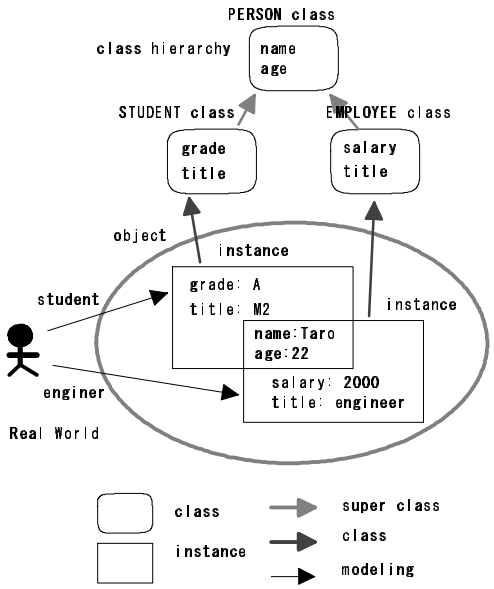

図 3 属性を共有するインスタンスをもつオブジェクトの構造

Fig. 3 Structure of an object with two instances having shared attribute values

４．３　多態オブジェクトモデルの動的な性質

多態オブジェクトモデルの動的な性質を議論する。

すでに存在するオブジェクトが実世界では同一のものの別の表現であると作成後に判明したり、また、配布されたデータベースに対して新たにオブジェクトを追加したり、ほかのデータベースと混合するときがある。このような場合も、同様な操作が可能でなければならない。

この場合には、それぞれのオブジェクトを、それらがもつインスタンスをすべてもつような一つのオブジェクトに構成し直す操作を行う。これをオブジェクトの融合と呼ぶ。このオブジェクトの融合の前後でオブジェクト識別子を付け替えると、そのオブジェクトへのすべての参照を変更しなければならない。参照するオブジェクトはデータベース内に散在するので変更の負荷は重い。そのため多態オブジェクトでは、オブジェクト識別子が異なっていても、実体としてのインスタンスが一つのものを指す場合は、同一のオフジェクトとみなす方式を採用し、融合の操作ではオブジェクト識別子を保存するようにした（定義３参照）。

多態オブジェクトの操作の例を示す。

まず、オブジェクトを生成する。

```
aPerson = STUDENT.new      aPerson を STUDENT クラスとして作成する
aPerson.grade = 'A'        aPerson の成績を設定する
aPhoto = PHOTO.new         aPhoto を PHOTO クラスのインスタンスとする
aPhoto.image = file("photo3902")
                           aPhoto にイメージデータを設定する
```

複数のクラスに属するインスタンスをもつオブジェクトを、一つのオブジェクトにするためには、他方を引数にして一方にメッセージを送付する。

```
aPerson.append( aPhoto )
```

このあとは、aPerson と aPhoto の二つのオブジェクト識別子が、一つのオブジェクトを示す。このオブジェクトには二つのインスタンスが含まれる。

オブジェクトを参照するときには、クラス名を引数として渡す。

```
((PHOTO)aPerson).image     aPerson の PHOTO クラスの部分を参照する
((STUDENT)aPerson).grade   aPerson の STUDENT クラスの部分を参照する
```

メソッドも、属性と同様に、オブジェクトが属した順にクラスを検索する。また、特定のクラスからのオブジェクトを参照するときには、指定したクラスに属しているとして、メソッドを検索する。

参照している属性（または呼出されているメソッド）を定義しているクラスは、オブジェクトを作成するやり方によって動的に決まる。例えば、PHOTO クラスのインスタンスをもつオブジェクトは PERSON クラスのインスタンスをもつオブジェクトと融合されたり、HOUSE クラスのインスタンスをもつオブジェクトに融合されたりする。融合されたインスタンスによってオブジェクトのもつ属性やメソッドは異なる。

多態オブジェクトでは、オブジェクトの融合の時点では検査を行なわず、オブジェクトの参照するときの有効範囲に対して属性やメソッドが有効かどうかを検査する。これを「動的な検査」と呼ぶ。

クラス定義の時点でオブジェクトの融合を許すクラスを決めておいて、実際に融合しようとしたときに、融合が許されるかどうか検査する方法がある。例えば、PERSON クラ

スを定義するときに PHOTO クラスと結びつけられる可能性があることを示しておいて、検査のための情報とする。これを「静的な検査」と呼ぶ。静的な検査では問合せを発行したとき、クラスの定義を参照することで、融合の操作が正しいものかを検査することができる。これは誤りを早期に発見でき、一般的には、バグを少なくすることにつながる。反面、静的な検査では、クラス階層の設計時点で、融合できるクラスを予想したクラス階層を作成しなければならない。そのため、クラス階層を設計するときには将来のクラス階層の定義の拡張方式を考慮する必要があり、そのことは困難である。特にデータベースの設計者とデータベースの利用者や拡張者が異なっている場合は至難なことである。

オブジェクトの融合の逆の操作としてオブジェクトの分離が考えられる。オブジェクトの分離の意味は、問題も多く、これからの課題である。

現在は、オブジェクトの分離は、一つとみなしていたオブジェクトから、インスタンスを分離した新しくオブジェクトを作りだすことにした。これは、オブジェクトが融合されていた期間にそのオブジェクトに対して新たな参照が作られる可能性があるので、単純に融合する前の識別子とインスタンスの組にもどすことはできないためである。分離の操作は、もとのオブジェクトから分離するオブジェクトの属性を取り除き、新たにオブジェクト識別子を発行し別のオブジェクトを生成し直すことになる。

データベースの検索機能として、オブジェクトは、そのオブジェクト内のインスタンスの属するクラスから検索できる。検索では、条件を設定し、その条件を満たすインスタンスだけを選択することができる。検索は、単独クラスの、またはそのクラスをスーパークラスとするクラス階層を対象とすることができる。

４．４　多態オブジェクトと多重継承

多態オブジェクトと多重継承は異なる概念である。

多重継承は、クラス階層の中で複数のクラスの定義を利用して、新たなクラスを定義する機構である。多重継承したクラスでは、スーパークラスで定義した属性やメソッドをすべて参照することができる。そして、必要に応じてそれらの定義を利用したメソッドを定義することもできる。言い替えると、スーパークラスの定義を利用して新たな定義を作成する機構である。定義を作成する機能であるので、実際のオブジェクトは、定義したクラスから生成する。

一方、多態オブジェクトは、複数のクラスにわたったオブジェクトの性質や振舞いを定義するものではない。多態オブジェクトは、一つのオブジェクトに対する複数の有効範囲を、クラス名を使って必要に応じて設定する機構である。複数の有効範囲のうち同時に使用できるのは一つである。しかし、クラス階層による定義の共有により同じ概念を表現している属性は、異なった表現で一つの属性を共有する。

多態オブジェクトではオブジェクトの融合により表現を追加しようとするときに、多重継承のようにクラス階層での定義を必要としない。そのため、多態オブジェクトではある集合に対する独立した分類が入った場合のクラスの数の指数関数的な増加は起こらない。また、オブジェクトの融合でのクラス定義の変更を要しないためスキーマを進化させそれに属するオブジェクトを変更する必要もなく、大容量データベースの実現に有利である。書換えが必要なデータのみを書換可能記憶に移し、それ以外のデータを読出し専用媒体に残すこともできる。これにより読出し専用媒体を有効に利用できる。

５．「沙羅」データモデルの実装例

多態オブジェクトモデルの実装上の課題を検討するために、オブジェクト指向データベース管理システム「沙羅」を試作した[13]。ここでは、その概要とオブジェクトの融合に関する実装を報告する。

「沙羅」データベースのオブジェクトはデータベース操作言語「沙羅双樹」で操作される。沙羅双樹のプログラム（データベース操作手続き）は、データ定義と問合せごとに解釈系によって、仮想機械の命令列に変換される。メソッドも仮想機械の命令列に変換される。データベース操作言語は宣言的であるのに対して、仮想機械の命令は手続き的である。

沙羅双樹は、従来のデータベース言語と異なり、データベース定義とデータベース操作（問合せを含む）が分離していない。データベース操作の効果はデータベースに累積する。

これに対応して、仮想機械もデータベース定義とデータベース操作を実行順にデータベースに累積する。

仮想機械は、データベースの基本機能を提供する。メソッドはこの仮想機械での命令の列となる。メソッドは、同じクラスのインスタンスで同一であるので、クラスに記録する。

メモリ管理部は、仮想機械のために、実際の主メモリを管理し、そしてディスクとの転送を管理する。

オブジェクトは、メモリ管理部で管理される。仮想機械は、オブジェクトを識別子で参照し、必要ならば主メモリ

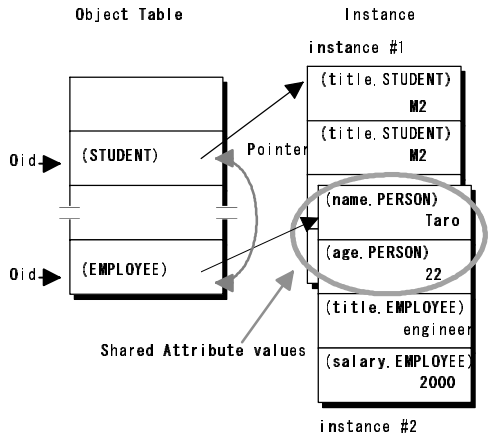

図 4 属性を共有するオブジェクトの内部構造
Fig. 4 Internal structure of shared Attribute values

に割り付ける要求をメモリ管理部に出し、メモリ上で操作する。

属性値は、整数や実数のような実行系で実現されたオブジェクトならばその値自身に、クラス定義されたインスタンスならばそのオブジェクト識別子になる。

図 4 にメモリ管理部の構造の概要を示した。

メモリ管理部では、オブジェクトは、インスタンス単位でオブジェクト表（Object Table）の一つのエントリを使用する。この表のエントリは、クラス名、インスタンスの大きさ、読み書きなどの属性と主メモリまたはディスク位置のアドレス、そして多態オブジェクトのための双方向ポインタからなる。図 4 ではクラス名とインスタンスのアドレスだけを図示している。

インスタンス（instance）は、一つの属性値に対してクラス名と属性名をもち、それらの配列となっている。図 4 では括弧の中に属性名とクラス名を記入している。クラス名は属性を定義しているクラスの名前であり、継承による場合は実際に定義しているクラス名となる。属性を定義したクラス名を属性名とともに格納するのは、属性名がクラス内でしか一意でないためである。複数のクラスで定義した属性を融合したオブジェクト内で識別するために必要である。多態オブジェクトの複数のインスタンスは、複数のエントリを双方向ポインタで連結することで表す。このように実際に属性値を共有することで一貫性を実現している。

クラスには、インスタンスのもつ属性の一覧表を格納し、これをインスタンスの有効範囲として利用している。インスタンスの参照や代入では、一覧表をマスクとして使用し、有効範囲を設定する。クラス階層の参照を少なくするためにこの表にはスーパークラスから継承する属性も記録している。

オブジェクトを融合したときは、もとのインスタンスを一つにまとめる。このときの時間のコストは、インスタンス中の属性の数の合計に比例する。4 バイトのオブジェクト識別子の現在の実装では、属性一つあたり 36 バイトの複写となる。インスタンスの融合の後、オブジェクト識別子の連結をする。これはオブジェクト表で、該当するエントリを双方向リストで連結し、融合したインスタンスを参照するようにする。図 4 では、instance #1 と instance #2 を一つにし、双方向のポインタで連結した状態を示している。（ポインタを 1 本にする前の状態である。）一つにまとめられた属性値を楕円で図示している。

このように融合したオブジェクトの参照は、次のような手順となる。まず、参照で指定されたオブジェクト識別子からオブジェクト表を参照する。このとき表のエントリをつないだ双方向リストで指定したクラスを検索する。指定されたクラス名が見つかったならば、そのクラス定義を検索し、クラス中の属性の一覧表を取り出し、インスタンスを参照する。

上記の仮想機械とメモリ管理部を試作した。仮想機械はスタックを基にした演算方式を採用していて、この仮想機械の上で複数のプロセスが動作する。仮想機械は命令として 152 の基本オペレータをもち、クラス定義、オブジェクト定義、四則演算、変数定義、条件判断、繰り返し処理、手続き定義、ファイル入出力、そしてプロセス管理を行う。メモリ管理部は仮想空間でオブジェクトの定義と配置を行い、オブジェクトは識別子をキーとした B 木で管理している。ファイルの読み書きは開始時と終了時だけである。

試作システムは C 言語で書かれ、規模は仮想機械とメモリ管理部を合わせて約 12000 行である。動作環境は、Unix(SunOS 4.1.3)であるが、OS に依存する部分は少なく、Unix 版とほぼ同じソースコードで、データベースをファイルシステムに格納しない MS-DOS 版もある。

６．むすび

多態オブジェクトモデルについて述べた。多態オブジェクトモデルは、実世界に対する多様な表現をデータベース内に反映するための機能である。この機能によれば、実世界の実体をクラス階層に依存しない複数のインスタンスで表現し、複数のインスタンスを所有するオブジェクトで代表することができるので、マルチメディアデータの表現に有効である。

このデータモデルを元にしたデータベース管理システム「沙羅」を試作した。「沙羅」は多態オブジェクトモデルを実現するだけでなく、さらにメソッドを含んだ可搬なデータベースを作成することをめざしている。現在の実装はデータモデルを直接実現したものであり、実装方式の評価と改良が今後の課題である。

謝辞

本研究を進めるにあたって有意義な意見を提供していただいた情報処理学会データベースシステム研究会のみなさんに感謝します。また、細かい点について議論していただいた久保田和己氏（富士通）と須崎有康氏（電総研）に感謝します。

文献

(1) Woelk D., Kim W., Luther W., "An Object-Oriented Approach to Multimedia Databases", ACM SIGMOD Inter. Conf. on Manage. of Data, pp.311-325 (1986)

(2) Atkinson M., Bancilhon F., DeWitt D., Dittrich K., Maier D. and Zdonik S.:"The Object-Oriented Database Systems Manifesto", in Deductive and Object-Oriented Databases, eds. W.Kim, J.M.Nicolas and S.Nishio, pp.223-240, North-Holland(1990)

(3) Lamb C., Landis G., Orenstein J., Weinreb D., "The ObjectStore Database System", Commun. ACM, 34, 10, pp.50-63 (1991)

(4) Maier D., Stein J., "Development and Implementation of an Object-Oriented DBMS", Research Directions in Object-Oriented Programming, eds. B.Shriver and P.Wegner, pp.355-392, MIT Press (1987)

(5) Masunaga Y.:"Design Issues of OMEGA: An Object-Oriented Multimedia Database Management System", J. Inf. Process., 14, 1, pp.60-73 (1991)

(6) Fishman D.H., Beech D., Cate H.P., Chow E.C., Connors T., Davis J.W., Derrett N., Hoch C.G., Kent W., Lyngbaek P., Mahbod B., Neimat M.A., Tyan T.A. and Shan M.C.:"Iris: An Object-Oriented Database Management System", ACM Trans. on Off. Inf. Syst., 5, 1 (1987)

(7) Ahad R., Dedo D., "OpenODB from Hewlett-Packard: a commercial object-oriented database management system", J. Object-Oriented Program., 4, 9 (1992)

(8) Moerkotte G. and Zachmann A.:"Multiple Substitutability Without Affecting the Taxonomy" Advances in Database Technology - EDBT'92, eds. A.Pirotte, C.Delobel, G.Gottlob, pp.120-135, Springer-Verlag (1992)

(9) Sciore E.:"Object Specialization", ACM Trans. Inf. Syst., 7, 2, pp.103-127 (1989)

(10) Goldberg A. and Robson D.:"Smalltalk-80: The language and its Implementation", Addison-Wesley (1983)

(11) Richardson J. and Schewarz P.:"Aspects: Extending Object to Support Multiple, Independent Roles", Proc. of the 1991 ACM SIGMOD Int. Conf. Manage. Data, pp.298-307 (1991)

(12) Keene S., "Object-Oriented Programming in Common Lisp", Addison-Wesley (1989)

(13) Ishimaru T., Uemura S., "Sarah: An Object-Oriented Database System Capturing Multiple Meanings", Proc. of Advanced Database Systems Symposium, IPSJ SIGDBS, pp.35-43 (1992)