データベースシステム 91－2
（1992. 11. 5）

インスタンス間継承に基づくオブジェクトモデルとその物理的記憶構造

奥田 太郎
神戸大学大学院工学研究科

下條 真司
大阪大学大型計算機センター

田中 克己
神戸大学工学部

オブジェクト指向データベース管理システム (OODBMS) の特徴は，オブジェクト概念に基づくモデリング機能，クラス階層に基づく属性定義とメソッドの継承機能，複合オブジェクトのサポートなどであるが，研究の進展と応用範囲の拡大につれて，(1) 1つのオブジェクトがそのオブジェクト識別性を保ったまま多様な側面を持つことが困難，(2) インスタンス間で直接，属性や属性値などを継承する機構が有用であるにもかかわらずサポートされていないことなどが指摘されている．

そこで，本稿では，(1) 集合型のオブジェクトのオブジェクト識別子を構造・階層化することとにより，レイヤの概念を導入したモデルを提示し，(2) 集合型オブジェクトの性質のその要素への継承，および，2つのレイヤの各要素間での性質の継承という2種のインスタンス間継承を効率良く実現するための具体的な物理的記憶構造を提案する．レイヤによって同一の側面を持つオブジェクトをまとめることで，実世界の実体の持つ多面性を，直観的に表現でき，また，2つ以上のレイヤを重ね合わせることで，様々な側面の持つ情報を重畳させることができる．

An Object Model with an Instance-Based Inheritance Mechanism and Its Physical Storage Structure

Taro Okuda
Graduate School of Engineering,
Kobe University

Shinji Shimojo
Computation Center,
Osaka University

Katsumi Tanaka
Faculty of Engineering,
Kobe University

Object-oriented Database Management Systems (OODBMSs) offer several valuable features such as its rich modelling capability, inheritance capability of attribute structures and methods based on class hierarchies, and supporting complex objects. However, conventional OODBMSs support neither (1) multiple aspects of objects with the same identities, nor (2) inheritance among instance objects, both of which are considered important to enhance the modelling capability of conventional OODBMSs.

In this paper, we introduce an object model with *layer* concepts, in which each layer is constructed by a set-type object and each object identifier incorporates a layer hierarchy. We will describe two kinds of instance-based inheritances of this model: (1) inheritance of attribute/values of a set-type object to each of its elements, and (2) inheritance of attributes/values of objects in a layer to the corresponding objects in another layer. We also propose a physical storage structure to implement our instance-based instance of our obejct model with layer concepts.

1 はじめに

現在，次世代データベース管理システムの1つの候補として，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, OODBMS) が注目されており，製品開発や研究が活発に行なわれている．その特徴としては，

- 対象とする実世界の実体を1つのオブジェクトとして捉え，オブジェクト間の関連を is-a 関連や part-of 関連を用いてモデル化することで，対象世界のモデル化がより自然な形で行なえると同時に，応用プログラムにおいても，構築したモデルをそのままの形でプログラムコードに反映できる．

- 同じ性質 (属性構造，メソッド) を持つ実体 = オブジェクトをまとめる概念として，クラスが用意されており，さらに，より詳細化／具体化されたオブジェクトをまとめる概念として，サブクラスが用意されている．

- クラスとサブクラスは，意味的には is-a 関連を表現するリンクで接続されクラス階層を形成する．サブクラスは自分の上位に位置するクラスから，その属性定義情報とメソッドを継承する．

- オブジェクトは，その内容とは独立に，オブジェクト識別子 (OID) によって一意に識別可能である．

などが挙げられる．しかし，OODBMS の研究の進展と応用範囲の拡大につれて，現状の OODBMS では，

- 1つのオブジェクトに，そのオブジェクト識別性を保たせたまま，多様な側面を持たせる[1]ことが困難．

- 単なる is-a 関連のみならず種々の関連で結ばれたインスタンスの間で直接，属性や属性値などを継承する機構が有用であるにもかかわらずサポートされていない．

ことなどが問題であると考えられる．

そこで，本稿では，

1. 集合型のオブジェクトのオブジェクト識別子を構造・階層化することとにより，レイヤの概念を導入したモデルを提示し，
2. 集合型オブジェクトの性質のその要素への継承，および，2つのレイヤの各要素間での性質の継承という2種のインスタンス間継承を効率良く実現するための具体的な物理的記憶構造を提案する．

レイヤによって同一の側面を持つオブジェクトをまとめることで，実世界の実体の持つ多面性を，直観的に表現できる．さらに，インスタンス間継承の機構を用いて，2つ以上のレイヤを重ね合わせることで，様々な側面の持つ情報を重畳させることができる．また，レイヤ自身も属性／属性値やメソッドを有し，これも必要に応じて，そのレイヤの各構成要素のオブジェクト群に継承させることが可能となっている．

従来の OODBMS にも多重継承という概念が実現されており，複数の側面をあわせ持つオブジェクトの表現は可能であるが，(1) 複数のクラスをスーパークラスとする形の多重継承を多用すると，スキーマが複雑になる，(2) 多重継承を用いても，1つのオブジェクトが1つのクラスに所属するという制限を緩和することにはならず，時間や状況に応じて動的にオブジェクトの見え方（例えばその属性値など）を変化させるような機能を実現しにくい，などといった問題がある．

本稿で提案するモデルは，基本的に，Obase オブジェクトモデル [1] (後述) に準拠したものであり，本稿ではさらに，文献 [1] の内容に基づき，そのレイヤ操作に関して検討を行なっている．

2節では関連する研究として，オブジェクトの多面性を表現するための他の研究を幾つか紹介する．そして，3節で我々の提案するレイヤモデルの概念を形式化し，オブジェクト間の情報の受け渡しを効率良く実現するための物理的な記憶構造の構築について4節で述べる．5節は結論である．

2 関連する研究

1つのオブジェクトが，そのオブジェクト識別性 (object identity) を保ったまま，多様な様相 (*multiple aspects*) を持てるようにするために，従来から多くの研究がなされている．

2.1 仮想クラスと仮想クラス階層

従来の関係データベースの分野で議論されたビュー (*view*)[2] の概念を OODBMS の枠組みに拡張して適用しようという立場の研究である．OODBMS における

[1] ここで，「多様な側面を持たせる」とは，1つのオブジェクトが，状況や役割などに応じて，異なる属性構造や属性値やメソッドなどを持ったり，所属する型やクラスが異なったりすることを意味する．

[2] 関係データベースにおけるビューとは，仮想的な関係表定義を質問言語を用いて行なうもので，この意味では，データベーススキーマの仮想化を行なうものである．仮想的な関係表に現われる組は，通常，物理的には格納されず，ビューへの問合せの処理を行なう際に動的に生成される．

データベーススキーマは通常，クラス階層で表現されるので，質問言語を用いて仮想的なクラス定義を行なったり，また，仮想的なクラス群や元々存在するクラス群をあわせて，仮想的なクラス階層を実現する．

例えば，田中ら [2] は，Smalltalk-80 ベースの質問言語を用いて仮想クラスを定義する方法を提案し，また，これらの仮想クラス群と実クラス群とが混在した仮想的なクラス階層を構築する問題について考察している．このアプローチの特色としては，仮想クラスの外延 (extent) は動的に計算されること，仮想クラスのインスタンスは元のどれかの実クラスのインスタンスであることであり，これにより，1 つのインスタンスが直接所属できるクラスが複数になっているという点である．さらに，この文献 [2] では，元の実クラスにはない，仮想クラスに固有のメソッド[3]を追加して定義できるということも大きな特色となっており，これにより，1 つのオブジェクトは，状況に応じて異なるメソッド群を適用することが可能となっている．

最近，Rundensteiner は，田中らの文献 [2] で提起された，仮想クラス階層の構築の際の consistency の問題や閉包性 (closure property) の問題を解決するためのアルゴリズム等を発表している [3]．

Abiteboul ら [4] は，より一般的な立場から，OODB のビュー機能として持つべき機能を提案している．例えば，仮想属性の定義機能[4]，属性を明示的に隠蔽する機能，仮想クラスの定義機能，仮想クラスのクラス階層中での配置の自動化機能などである．同様の方向の研究として，Bertino[5] の研究がある．文献 [5] では，主に，ビュー定義言語に焦点をあてて，ビュー固有の属性やメソッドの扱いやビューインスタンスのオブジェクト識別子の問題について論じている．

Kim と Korth は，Composite オブジェクトの アグリゲーション (Part-of) 階層を仮想化してビューを構成する方法を提案している [6]．また，これに関連して，この文献では，アグリゲーション階層が異なるスキーマのバージョニングについても考察が行なわれている．

Scholl[7] らは，質問言語を用いて定義された仮想クラス (ビュー) を通じての更新可能性について論じている．型とクラスを区別し，オブジェクトは複数の型やクラスのメンバーになり得るような COCOON と呼ぶデータモデルを提案し，このモデルに基づきビュー更新の問題を考察している．このアプローチの特色は，質問によって生成されるオブジェクトは，データベース中のオブジェクトとアイデンティティは同一であるが，外延として所属するクラスと，メソッドインタフェースなどの型とが異なるものと見做していることで，これによりビューを通じての更新を可能にしている．

[3]仮想クラスに固有の属性をもたせることについては文献 [2] では論じられていない．

[4]単なる computed attribute だけではなく，複合オブジェクトの属性構造を仮想的に異なる形で見せる機能も含まれている．

2.2 多重型オブジェクト

2.1 節のような「スキーマの仮想化」という立場をとらずに，オブジェクトが複数の型を有するようにするために，最近，以下のような研究がなされている．

Iris は，オブジェクトに動的な型変換を許した，最初の OODBMS である．オブジェクトは，add-type オペレーションなどによって，自由に型を拡張・収縮させることができる．さらに，オブジェクト o は，T(o) の返値を計算することによって，型 T に属するかどうかが確認できる [8]．

Wilkes[9] は，is-a 継承と呼ばれるインスタンス間継承機構を提案しており，ある下位クラスのインスタンスは，その上位クラスのインスタンスのより詳細化された面を表現するものとなっており，上位クラスのインスタンスの属性値が自動的に継承される．

Sciore は，実世界の 1 つの実体を，複数のオブジェクトで表現するアプローチを提案している [10]．型階層とは別に，これと対応する形の，オブジェクトの is-a 階層 (オブジェクト階層) を考え，1 つのオブジェクト階層中の各オブジェクトは，実世界の 1 つの実体の種々の側面を表しているとするもので，オブジェクトは，自分の親オブジェクトから属性やメソッドを継承することができる．

Richardson ら [11] は，comformity という概念に基づいて，型とその実現部を分離しこれらの間の明示的な関連をなくしたモデルの上に *aspect* という概念を実現している．これによって，1 つの型が複数の実現部を持ったり，1 つのオブジェクトが複数の型の実現部をもてるようになっており，ある型のオブジェクトに新たな aspect を追加する extends 演算子を導入している．

Moerkotte ら [12] は，ある複合オブジェクト o 自身がその一部の構成要素オブジェクト o_i の型情報を有することができるようにするために *fashion* という概念を導入している．例えば，車オブジェクトには本来「ガソリンをいれる」というメソッドはなく，このメソッドは，むしろその構成要素であるガソリンタンクオブジェクトのメソッドであるが，fashion という概念により，車オブジェクトに対してこのメソッドが直接適用可能になる．

渡ら [13] の Morphe においては，型とインスタンス

の区別を行なわず，すべてのオブジェクト定義を制約と見て，同じオブジェクトでもその状況に応じて異なる構造や属性値を持たせることが可能になっており，状況はパスという概念にもとづいて実現されている．

本稿で述べるモデルは，オブジェクトが状況／役割／時間などに応じて異なる属性構造や属性値を持つようにするためにレイヤ概念を用いている．さらに，1つのオブジェクトの種々の側面を表したオブジェクト群の間で情報の継承を行なったり，レイヤ固有の情報をレイヤの構成要素に継承させるためのインスタンス間継承を仮想的に実現している点で上記の研究と異なっている．

3 マルチレイヤモデル

3.1 概要

マルチレイヤモデルでは，1つのレイヤに1つの実体の側面を対応させる．すなわち，1つのレイヤ上には，そのレイヤの表す側面から見た複数のオブジェクトが付随している．

そして，複数のレイヤを重ねることにより，実体の持つ複数の側面を直観的に表現することを目的とする．

3.2 レイヤの種類

レイヤを構成する概念としては幾つか考えられる．

- 状況によるレイヤ構成

  人間は，状況によってその役割が学生，先生，父親などに変化する．このとき，それぞれの役割を1つのレイヤと見なすことができる．

- 時間によるレイヤ構成

  実体は，時間とともにその状態を変化させることが考えられる．例えば，ある人は，１９９０年と１９９１年でその職場が違うかも知れない．これらの状態変化は，１９９０年レイヤと１９９１年レイヤを形成することで処理できる．

- 包含関係によるレイヤ構成

  車オブジェクトを考える．全体としては1つの車であるが，細かく見てみると，エンジン，タイヤ，ボディなどから構成されている．このように，実体を見るレベルを1つのレイヤとして表現可能である．

3.3 レイヤの重ね合わせ

マルチレイヤモデルでは，様々な実世界の情報をレイヤ上のオブジェクトとして表現する．このとき，個々のレイヤは独立ではなく何らかの関連性があり，複数のレイヤを重ね合わせることに意味があるものと考える．また，レイヤの重ね合わせによって情報を受け渡しすることにより，オブジェクトの多重性を柔軟に扱うことができる．以下に，重ね合わせによる情報の受け渡しの典型的な例を2，3挙げることにする．

- 地理情報における重ね合わせ

  実世界の地理情報を適当な単位で分割し，複数のレイヤ構成にすることは，検索範囲を限定するうえでも有用である．例えば，土地の利用状況（工業用地，農業用地など）を1つのレイヤに格納し，別のレイヤに県単位の情報を格納したとする．この2つのレイヤを重ね合わせることにより，県単位の土地利用状況ができあがる（図1）．

- 実体の多面性における重ね合わせ

  人間の多面性を表すための親レイヤと教師レイヤが存在するとする．教師レイヤにはその共通する属性として，“担当科目”，“収入”等があり，親レイヤには“子供の数”等があげられる．2つのレイヤを重ねることで，親オブジェクトは教師レイヤ上のオブジェクトからの属性情報を継承する[5]．

- 実体の包含継承における重ね合わせ

  再び車オブジェクトを考える．ある車オブジェクトの属性としては，“ナンバー”，“車種”，“車の色”などが考えられる．また，もう1段詳細なレベルで見た場合，その車はエンジン，タイヤ，ボディなどのオブジェクトから構成されている．このとき，車の属性である“車の色”が赤であれば，当然その車のボディの色属性も赤となるべきである．このような実体とその構成要素レイヤ間の重ね合わせによる属性・属性値の継承も考えられる．

- グループウェアにおける重ね合わせ

  コンピュータ上で協同作業やコミュニケーションをする環境を整える支援ツールが開発されている．グループで使用可能な予定表作成ツールを利用することにより，共同研究を進める場合を考える．個人の予定表をそれぞれ重ね合わせて1つの予定表にすることで，日程の調整を行ない，研究をより円滑に行うことが可能となる．

[5]教師レイヤ上のオブジェクトは，ある実体の1側面を表している．継承する対象は，親レイヤ上で，その実体の別の側面を表しているオブジェクトである．

3.4 定義

この節では，レイヤ及びその操作を扱うために，オブジェクトやレイヤなどの定義を行なう．

[定義 1] オブジェクト

本稿では，レイヤ上のオブジェクトを形式的に表現する方法として，Obaseオブジェクトモデル(OOM)[1]を使用する．OOMは，様々なオブジェクトを柔軟に扱うための枠組として，Obaseプロジェクトにおいて提案しているもので，オブジェクトを以下の3組で表す．

$$Object : OID(contents, property)$$

OIDはオブジェクト識別子であり，contentsはそのオブジェクトが表現しているオブジェクトの集合である．各オブジェクトとそのcontentsの各要素を，それぞれスーパーオブジェクト，サブオブジェクトと呼ぶ．ここでは，全てのオブジェクトの集合を$\mathcal{O}$で表す．propertyは，そのオブジェクトの性質を表現したもので，属性やメソッド，制約などを記述する．各々のpropertyの先頭に↓を付加することで，そのpropertyの値をcontentsへ継承させることが可能である．これによって，複数のオブジェクトが，特定の情報を共有できることになる．

例：
山田 ({ 山田 [1990]，山田 [1991]}，<↓ Name = 'Yamada'，Age = 43>)
は，オブジェクトである．属性Nameは，↓を先頭に付加することにより，山田 [1990]，山田 [1991]へその値を継承させる．

さらに，全ての属性名の集合を$\mathcal{A}$，全ての演算子の集合を$\mathcal{S}$として，propertyを属性名，演算子，属性値の3組で表す．

$$Property : (p\_name,\ sign,\ p\_value)$$

ここで，$p\_name \in \mathcal{A}$，$sign \in \mathcal{S}$，$p_value \in \mathcal{O}$である[6]．

例：
前述の例では，

属性名 (p_name)	演算子 (sign)	属性値 (p_value)
Name	=	'Yamada'
Age	=	43

[6]属性値もまたオブジェクトとして扱う．

となる．

[定義 2] オブジェクト識別子 (OID)

実世界の実体を表すために，異なる側面を表す複数のオブジェクトを用いるが，このとき，それらのオブジェクト間に関連を付けることが重要である．ここではOIDをeidとlidの2種類の識別子で表すことで関連付けを行うことにする[7]．eidは実体を識別するための実体識別子 (Entity Identity) であり，lidは実体の側面を表すためのレイヤ識別子 (Layered Identity) である．ここで，lidは集合$\mathcal{L}$の要素であるとし，並べるlidの数を側面数と呼ぶことにする．具体的には，

$$eid[lid_1][lid_2]\ldots[lid_n], \quad (\{lid_1, lid_2, \ldots, lid_n\} \subseteq \mathcal{L})$$

のように記述する．このとき，側面数はnとなる．

例：

- 鈴木 [Student]
- 鈴木 [Teacher][1990]

は，それぞれオブジェクトを一意に識別するOIDである．両オブジェクトは，eid項が等しいので，同じ実体の違う側面を表現している．鈴木 [Teacher][1990]の側面数は2である．

オブジェクトは，様々な側面を持つ可能性がある．あるオブジェクトOのOIDに，あるlid$L$が存在する時，OはLの側面を持つといい，

$$O \sqsupseteq L$$

で表す．

[定義 3] レイヤ

レイヤは，Obaseオブジェクトモデルで表す．言い換えると，全てのオブジェクトはオブジェクトであると同時に，一つのレイヤとなる可能性がある．オブジェクトをレイヤとしてみた時，$OID(contents, property)$において，OIDはある特定の側面を表すレイヤ名，contentsはそのレイヤ（側面）に属するオブジェクトの集合，そしてpropertyはそのレイヤに共通する属性や属性値，その他の情報を表すことになる．

lidは，3.2で述べたレイヤの種類を基にして，非巡回の有向枝グラフを形成する．これをレイヤ階層と呼ぶことにする．図2に実体“兵庫県”におけるレイヤ

[7]オブジェクト識別子を構造化するというアイデアは，すでに横田の*Quixote*[14]によって提案されているが，*Quixote*では，オブジェクト識別子と多重型オブジェクト／オブジェクトの複数の側面の扱いとの関連に関する議論はあまり行なわれていない．

階層を示す．階層中の各ノードはそれぞれレイヤを表し，有効枝は，サブレイヤへ向かうポインタを意味する．レイヤ階層中の同レベルのノードは，互いに同等レベルであるといい，記号•で表す．また，ノードの上下関係については，記号≻を用いて，上レベルのノードがその下レベルのノードを包含することを表すことにする．このとき，任意のlid $A,B,C \in \mathcal{L}$ に対して，

1. $A \bullet B \equiv B \bullet A$
2. $A \succeq A$
3. $A \succeq B$かつ $B \succeq A$ ならば $A = B$
4. $A \succeq B$かつ $B \succeq C$ならば $A \succeq C$

が成り立つ．

例：

- 兵庫県 [1991][上半期][3 月]
- 兵庫県 [1992][下半期][12 月]

は，それぞれ側面数 3 のオブジェクトのオブジェクト識別子であり，[上半期] • [下半期]，[1992] ≻ [1992][下半期] ≻ [1992][下半期][12 月] である．

4 マルチレイヤモデルにおける重ね合わせ操作

Obase オブジェクトモデルでは，オブジェクトを外延的側面 (contents) と内包的側面 (property) からなるものとして認識している．そして，オブジェクトとその外延オブジェクト間の継承が可能である．これに対して，実体の多面性を表現する時には，個々のオブジェクト間の重ね合わせが必要になってくる．この節では，レイヤ構成に特に必要な重ね合わせ操作について，考察する．

4.1 定義

重ね合わせ操作において重要な点は，どのレベルのレイヤで重ね合わせるのか，という事である．そこで，重ね合わせが可能なオブジェクト同士であるための条件を以下に定義する．

[定義 4] 重ね合わせ可能な条件

オブジェクト O_1 と O_2 は，以下の条件を満たす時，重ね合わせが可能である．

- O_1 と O_2 が，同一の cid を持つ．
- O_1 と O_2 の対応する lid が，レイヤ階層中で同等レベルであるか，あるいは包含関係にある．

例：
2 つのオブジェクト

- 兵庫県 [1991][上半期]({...}, < ... >)
- 兵庫県 [1992][下半期]({...}, < ... >)

は，1991•1992 であるので，レイヤ 1991, 1992 に関して重ね合わせが可能である．

4.2 重ね合わせ操作の比較

レイヤ上のオブジェクトを重ね合わせた結果，どのようなオブジェクトができるのかは，重ね合わせる方法と具体的な重ね合わせ操作の種類 (AND や OR など) によって違いが生じる．そこでここでは，この違いを比較することにする．方法は大きく分けて以下の 2 通りがあると思われる（図 3）．

1. それぞれのレイヤに付属する属性 (property) を外延 (contents) に継承させた後，その contents 同士を比較し，重ね合わせ操作の種類によって，新たなオブジェクトを生成する．
2. それぞれのレイヤの contents を重ね合わせの種類によって操作し，集合 $\mathcal{T}$ を得る．他方，それぞれのレイヤの属性同士の和集合 $\mathcal{S}$ をとる．$\mathcal{T}$ に $\mathcal{S}$ を継承させて新たなオブジェクトを生成する．

上記の 2 つの方法では，属性名が衝突した場合，いずれも上方の属性値を優先させることにする．また，2 つのレイヤに存在するオブジェクトの集合を重ね合わせる操作の種類として，以下の 3 通りを考える．

AND: 両方に含まれるオブジェクトの集合をとる．

OR: 和集合をとる．

DIFF: 上方のレイヤにのみ存在するオブジェクトの集合をとる．

以下に簡単な例を示して，上に挙げたそれぞれの重ね合わせ操作の結果を比較検討してみる．まず，あるレイヤを表現するオブジェクトとして，$L_1(\{O_1, O_3\}, < A = a_1 >)$，$L_2(\{O_1, O_2\}, < A = a_2 >)$ の 2 つを用意する．L_1 上にはオブジェクト $O_1(\{\}, <>)$，$O_3(\{\}, <>)$ が存在し，L_2 上にはオブジェクト $O_1(\{\}, <>)$，$O_2(\{\}, <>)$ があるものとする（図 4）．L_1 を L_2 に重ねた場合の結果を以下に示す．

- オブジェクトの重ね合わせ → プロパティ継承の場合

AND	$O_1[L_1 over L_2](\{\}, < A = a_1 >)$
OR	$O_1[L_1 over L_2](\{\}, < A = a_1 >)$ $O_2[L_1 over L_2](\{\}, < A = a_1 >)$ $O_3[L_1 over L_2](\{\}, < A = a_1 >)$
DIFF	$O_3[L_1 over L_2](\{\}, < A = a_1 >)$

- プロパティ継承 → オブジェクトの重ね合わせの場合

AND	$O_1[L_1 over L_2](\{\}, < A = a_1 >)$
OR	$O_1[L_1 over L_2](\{\}, < A = a_1 >)$ $O_2[L_1 over L_2](\{\}, < A = a_2 >)$ $O_3[L_1 over L_2](\{\}, < A = a_1 >)$
DIFF	$O_3[L_1 over L_2](\{\}, < A = a_1 >)$

この結果を比較すると，1．演算操作としてAND, DIFFを選択した場合，重ね合わせの結果は，外延オブジェクトを重ね合わせてからプロパティを継承させても，あるいはその逆でも，結果は同一のものとなる，2．OR操作において，外延オブジェクトを重ね合わせてからプロパティを継承させると，本来は無いはずの属性や属性値がプロパティとして付加される場合が生じる，ということが分かる．演算操作として，ANDをとるか，あるいはORをとるかという問題は，実現するアプリケーションに依存するので，注意が必要となる．

5 インスタンス間継承を考慮した物理的記憶構造

ここでは，上記のモデルを実現する場合の物理的な記憶構造について，議論する．特に注意しなければならないのは，質問処理をいかに効率良く実現できるか，という点である．良く行なわれると思われる質問は，以下の2通りが考えられる．

1. あるオブジェクトのcontentsを全て求めよ

   これは，ある会社の従業員レイヤがあり，その上に存在する全ての従業員を求めよ，といった集合的な検索質問である．このような質問の検索効率を向上させるためには，スーパーオブジェクト・サブオブジェクト間の関係を何らかの形で保持しなければいけない．

2. 属性Aの値がaであるようなオブジェクトを求めよ

   オブジェクトは直接Aという値を持っていなくても，自分のスーパーオブジェクトからその属性情報を継承している場合も考えられる．このようなインスタンス間の属性継承を効率良く実現する索引が必要である．

以降の節では，上記のそれぞれの項目に対応して，オブジェクトの記憶構造，そして効率の良い属性継承のための索引構成について順に述べる．

5.1 オブジェクトの格納構造

前述した点を考慮した上で，オブジェクトの格納法としては，1ページで1オブジェクトの情報と仮定すると，以下のようになると思われる．

ページの構成：

- OID格納部： OID(cid, lid)，側面数(lidの数)
- contents格納部： contentsの総数，各要素の物理的な格納アドレスのリスト
- 属性情報格納部： propertyの総数，オブジェクトが持つ属性とその値（スーパーオブジェクトから継承される属性情報は除く）

このような構成によって，あるオブジェクトのcontentsを容易に参照することが可能となる．しかし，1つのオブジェクトに対して，膨大な量のcontentsがある場合は，単なる物理的なアドレスのリストでは対応しきれない．この問題に関しては，

1. 1ページに収まりきらない場合，新たにページを設け，追加分としてそこに格納していく．
2. contents格納部に全てのcontentsのアドレスを格納せずに，次の要素へのOIDのみを格納しておく．さらに，あらかじめOIDによる索引を構成しておく．contentsが必要な時は，その索引を利用して，再帰的に検索する．

などの解決策が考えられる．図5に，2番目の方法を採用した場合の格納構造の概要図を示す．

5.2 分割ハッシングによる静的な索引

以降の節では，効率良くレイヤ（オブジェクト）の情報を検索するための索引構成について述べる．基本的な考え方としては，ハッシュ法を用いる．すなわち，自分自身の圧縮された情報を基にして，検索効率を向上させる方法である．具体的には，

- 各オブジェクトのスーパーオブジェクト，サブオブジェクト，プロパティについて，それぞれ個別にハッシュする．この時，プロパティは実際に持っているもの名前のみをハッシュする．
- できたハッシュ値を順につなげ，それをそのオブジェクトのハッシュ値とする．
- 該当するハッシュ値を持つバケットへそのオブジェクトのOIDを格納する．

とする（図6）．

例：

オブジェクト $O(\{O_1, O_2\}, < A = a, B = b >)$ において，そのスーパーオブジェクトを $\{O_{s1}, O_{s2}\}$ とし[8]，スーパーオブジェクト，サブオブジェクト，プロパティのハッシュ関数をそれぞれ $H_{super}(), H_{sub}(), H_{prop}()$ としたとき，

$$H_{super}(O_{s1}) = 1, \quad H_{super}(O_{s2}) = 3$$
$$H_{sub}(O_1) = 2, \quad H_{sub}(O_2) = 2$$
$$H_{prop}(A) = 2, \quad H_{prop}(B) = 1$$

となったとする．スーパーオブジェクトのために4bit，サブオブジェクトのために5bit，プロパティのために3bitを割り当てているとすると，スーパーに関していえば，1bit目と3bit目にフラグを立てて0101となり，サブ，プロパティに関しても同様にして，それぞれ00100, 011となる．従って最終的にこのオブジェクトが格納されるバケットの番地は，010100100011となる．

このような索引を使用する利点としては，スーパーオブジェクト，サブオブジェクト，プロパティの部分に分割してハッシュ値を決定しているため，スーパー・サブの関係も効率良く検索可能であるし，属性情報による検索も可能であることや，複数の条件による検索にも適していることなどが挙げられる．

5.3 多値ビット構成のハッシングによる動的な索引

この節では，オブジェクトの情報をハッシュで圧縮し，各々のオブジェクト自身に付加することで，ダイナミックな質問検索に適応させる方法を考える（図7）．基本的戦略を以下に示す．

- 各オブジェクトのスーパーオブジェクト，サブオブジェクト，プロパティについて，それぞれ個別にハッシュする（プロパティは実際に持っている属性の名前のみ）．
- できたハッシュ値は，静的な情報としてオブジェクト自身に付加しておく．
- 自分のスーパーあるいはサブから継承している情報を全て調べ，ハッシュする．
- 得られたハッシュ値を基にして，全節と同様な方法で全体的なハッシュ値を構成し，オブジェクトに動的な情報として付加する．

例：

前節の例におけるオブジェクト $O(\{O_1, O_2\}, < A = a, B = b >)$ をスーパーオブジェクト，サブオブジェクト，プロパティについてハッシュし，Oの圧縮された情報としてあらかじめオブジェクトに付加しておく．さらに，スーパーオブジェクト $\{O_{s1}, O_{s2}\}$ から属性Cを，サブオブジェクト O_1から，属性Dの情報を継承していたとする[9]．継承属性に関してハッシュした結果が，

$$H_{prop}(C) = 3, \quad H_{prop}(D) = 0$$

であり，5bitの割り当てを仮定すると，動的なハッシュ値は10020となる．この値も O の付加的情報とする．ここで，各bitには，0,1の2進数ではなく，多値を許すことにする．これは，オブジェクトのcontentsが実行時に更新，挿入，削除される可能性があるからである．

例：

前述の例において，Oを O_{s1}のcontentsエントリから削除したとする．このとき，Oの静的ハッシュのスーパーエントリと，動的ハッシュのエントリをOが含まれないように，また継承属性 A が含まれないように，更新しなければならない．しかし，継承属性 A をハッシュして，単純にハッシュ値の該当する箇所に0フラグを立てることはできない．他のスーパーオブジェクトから同名の属性を継承している可能性があるからである（図8：この例の場合，O_{s1}の他に，O_{s2}からも属性 A を継承している）．各bitに多値を許し，カウンタとして使用することで，この問題点は解決できる．

[8] Obase object model の定義より，自分のスーパーオブジェクトは複数存在しても構わない．

[9] このとき，オブジェクト Oは O_{s1}から見た側面と O_{s2}から見た側面とで，属性Cに関して異なる値を持つような，多重オブジェクトになっている．

この多値bitハッシュ法の特徴としては，動的にハッシュ値を計算するので，ダイナミックな質問に適していることや，オブジェクトの更新時にハッシュ値も計算して更新しなければならないので，その分コストがかかること，そしてハッシュ値を各オブジェクト毎に格納するので，ディスクスペースを余分に使用しなければならないことなどが挙げられる．

6 今後の課題

ここでは，実体の持つ多面性を表現することを目的とした，マルチレイヤモデルを提案し，OOMを利用して特に必要と思われる重ね合わせ操作の議論を試みた．また，インスタンス間の情報継承を効率良く実現するための物理的な記憶構造を提案した．しかし，まだまだ不十分な点があると思われる．今後の課題としては，

- モデルの具体化
- オブジェクトの格納構造の詳細化とその実現
- 格納構造の評価

などが挙げられる．

参考文献

[1] Obase Consortium, *Obase* システムの基本概念, 第2回 Obase シンポジウム (オブジェクト指向データベース研究プロジェクト公開成果発表会資料集, 1992年10月.

[2] Tanaka,K., Yoshikawa,M., and Ishihara,K., *Schema Virtualization in Object-Oriented Databases*, Proc. of the 4th Int'l Conf. on Data Engineering, pp.23-30, Feb. 1988.

[3] Rundensteiner,E.A., *Multiview: A Methodology for Supporting Multiple Views in Object-oriented Databases*, Proc. of the 18th VLDB Conf., pp.187-198, Aug. 1992.

[4] Abiteboul,S. and Bonner,A., *Objects and Views*, Proc. of the 1991 ACM SIGMOD Int'l Conf. on Management of Data, pp.238-247, May 1991.

[5] Bertino,E., *A View Mechanism for Object-Oriented Databases*, Advances in Database technology - EDBT'92, Lecture Notes in Computer Science 580, Springer-Verlag, pp.136-151, March 1992.

[6] Kim,H.-J. and Korth,H.F., *Schema Versions and Views in Object-Oriented Databases*, Proc. of InfoJapan'90, Tokyo, October 1990.

[7] Scholl,M.H., Laasch,C., and Tresch,M., *Updatable Views in Object-Oriented Databases*, Proc. of the 2nd Int'l Conf. on Deductive and Object-Orienetd Databases (DOOD'91), pp.189-207, Dec. 1991.

[8] Fishman, D., et. al., *Iris: An Object-Oriented Database Management System*, ACM Trans. on Office Information Systems, 5(1), pp. 48-69, January 1987.

[9] Wilkes,W., *Instance Inheritance Mechanisms for Object Oriented Databases*, Proc. of the 2nd Int'l Workshop on Object-Oriented Database Systems, pp.274-279, Sept. 1988.

[10] Sciore, E., *Object Specialization*, ACM Trans. on Office Information Systems, 7(2), pp. 103-122, April 1989.

[11] Richardson, J., et. al., *Aspects: Extending Objects to Support Multiple, Independent Roles*, ACM SIGMOD Int'l Conf. on Management of Data, pp. 298-307, May 1991.

[12] Moerkotte,G. and Zachmann,A., *Multiple Substitutability Without Affecting the Taxonomy*, Advances in Database technology - EDBT'92, Lecture Notes in Computer Science 580, Springer-Verlag, pp.120-135, March 1992.

[13] Watari,S., Honda,Y. and Tokoro,M., *Morphe: A Constraint-Based Object-Oriented Language Supporting Situated Knowledge*, Proc. of the Int'l Conf.Fifth Generation Computer Systems, Tokyo, June 1992.

[14] 横田 一正, *QUIXOTE* による知識情報処理, 電子情報通信学会技術研究報告, Vol.92, No.51, pp. 27-34, May, 1992.

[15] 増永 良文, 田中 克己, 次世代データベースシステムの展望, 情報処理, Vol.32, No.5, pp.602-613, May 1991.

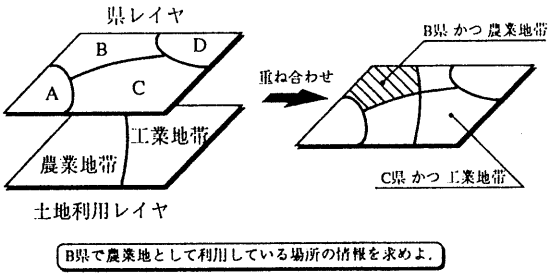

図１：地図におけるレイヤの重ね合わせの例

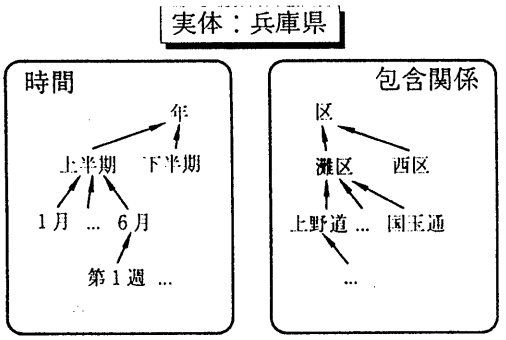

図２：レイヤ階層の１例

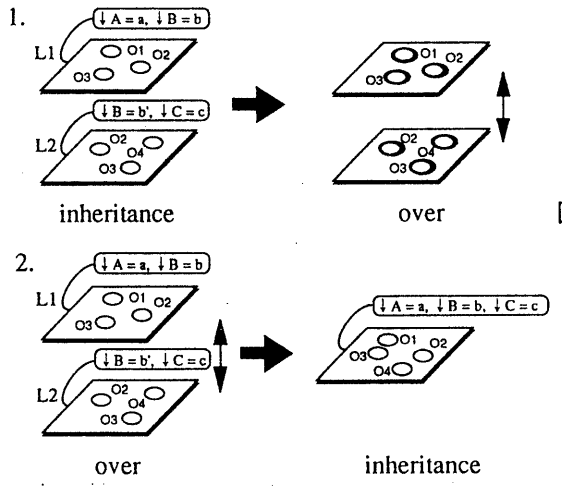

図３：レイヤのおける重ね合わせの方法

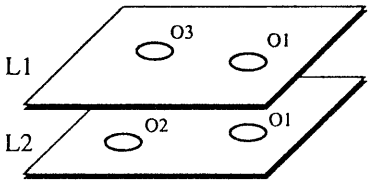

図４：２つのレイヤにおける重ね合わせ

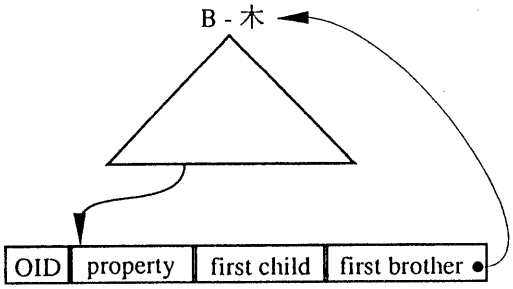

図５：物理的記憶構造の概要図

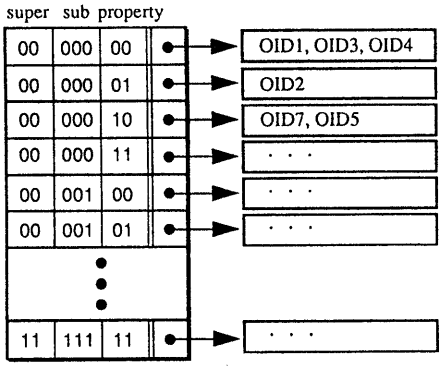

図６：分割ハッシュを用いた索引

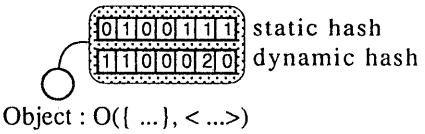

図７：多値ビットの動的ハッシュによる索引

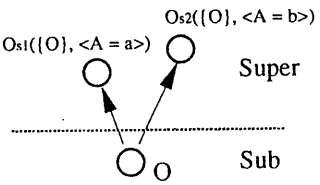

図８：多重オブジェクトの属性継承