ObaseLang：柔軟な構文と拡張経路式を持つオブジェクトデータベース言語

吉川正俊[†1] 田中克己[†2] 上善恒雄[†3]
田中康暁[†1,*] 蛭井潤[†4,**] 堀田光治郎[†5]

Obaseオブジェクトデータベースのための言語であるObaseLangの検索機能およびメソッドについて論じる．ObaseLangの検索機能では，Obaseオブジェクトの構造的な特徴である組，集合の両構成子に対応する順方向および逆方向の航行演算子を導入している．また，データモデルの特徴である柔軟な継承指定に基づくオブジェクトの多様なビューの提供を実現するために，組，集合両構造に沿った継承演算子を持っている．さらに，これらの航行および継承演算の正規表現を導入することにより再帰質問を含む強力な演算の簡潔な表現を可能としている．検索の構文は基本的にはSQLに従うが，FROM句は必須ではなく，また経路式をそのまま書くことを許すなど，自由度を高め等価な質問に対する多様な表現を可能としている．メソッドは，格納された経路式やSQL文の遅延評価による導出属性のほかにローカルメソッドを支援している．ローカルメソッドは，経路式をシステムの実装言語であるSmalltalkと組み合わせたものであり任意の計算可能関数の定義が可能となる．

ObaseLang : An Object Database Language with Flexible Syntax and Extended Path Expressions

MASATOSHI YOSHIKAWA,[†1] KATSUMI TANAKA,[†2] TSUNEO JOZEN,[†3]
YASUAKI TANAKA,[†1,*] JUN HIRUI [†4,**] and KOJIRO HOTTA [†5]

We will present the retrieval and method functionalities of ObaseLang, which is the database language of the Obase object database system. One of the structural features of the Obase model is that each object has both of a set and a tuple structures. The retrieval functionality of ObaseLang includes forward and backward navigational operators for these two structures. Inheritance operators are introduced to provide multiple views of objects, which is a prominent feature of the model. The inheritance can be specified along both of the tuple and set structures. Furthermore, regular expressions of these navigational and inheritance operators are supported. It makes users possible to express powerful yet succinct operations including recursive queries. The syntax of the retrieval expressions basically follows SQL. However, more flexible specifications are allowed; for example, FROM clauses of SQL sentences are optional, and also path expressions by themselves are first-class retrieval sentences. In Obase, two types of methods are supported; derived attributes and local methods. Derived attributes are achieved by late evaluation of stored path expressions and SQL sentences. Local methods allows users to define any computable functions by combining path expressions and Smalltalk sentences.

1. まえがき

我々は新しい概念に基づくオブジェクトデータベースObase[1),2)]の設計，開発を行った．Obaseデータベースは，マルチメディアデータベース，科学技術データベース，文書データベースなどを主たる応用分野として想定し，これらのデータベースに必要とされる半構

†1 奈良先端科学技術大学院大学情報科学研究科
Graduate School of Information Science, Nara Institute of Science and Technology
†2 神戸大学工学部情報知能工学科
Faculty of Engineering, Kobe University
†3 千里国際情報事業財団
Senri International Information Institute
†4 神戸大学大学院工学研究科計測工学専攻
Graduate School of Engineering, Kobe University
†5 コベルコシステム株式会社
Kobelco Systems Corporation

* 現在 NTT境界領域研究所
Presently with NTT Interdisciplinary Research Laboratories
** 現在 (株)オージス総研アドバンストシステム本部
Presently with Advanced System Division, OGIS Research Institute Co., Ltd.

造化データベース，段階的なオブジェクト同定，段階的なオブジェクト生成などの要件に答えることを目標としている．そのために，Obase データモデルは，種々の複合データの柔軟でかつ多面的な表現，コンテナ型オブジェクトの概念の拡張，インスタンス間の動的な継承機構の導入などの特徴を持つ．

ObaseLang は，Obase のためのデータベース言語であり，検索機能に加え，更新機能，オブジェクト定義機能，メソッドなどオブジェクトベース操作に必要な基本的機能をすべて装備した言語である．ObaseLang の検索，更新機能は，Obase データベースの操作体系のための理論的基盤である Obase 代数[3] に基づいている．本論文では，ObaseLang のこれら諸機能のうち特に検索機能とメソッドに焦点を絞って設計方針，特徴について述べ，構文の概要を例を用いて説明する．

ObaseLang の検索機能は，従来のオブジェクト指向データベースの質問言語と比較した場合，以下の特徴を持つ：

- 組，集合の 2 種類の構造を航行するための演算子を含む．従来の経路式は組構造のみを対象としてきたが，ObaseLang では集合構造も含む点が異なる．
- オブジェクト間の継承演算を明示的に指定する．
- 質問中に正規表現を用いた閉包機能を含む強力な経路式を書ける．
- 構文は了解性のため基本的には SQL に倣うが，自由度を高くし多様な表現を許す．

また，ObaseLang におけるメソッドは，導出属性とローカルメソッドから成る．導出属性は，オブジェクトのプロパティ値に経路式や SQL 検索文の格納を許し，その遅延評価機構により実現する．ローカルメソッドは，システムの実装言語である Smalltalk のブロック文と Obase 検索文を組み合わせることにより実現する．

実際に Obase システムを利用する場合は，グラフィカルユーザインタフェースである ObaseGUI[4] と ObaseLang を併用することになるが，本論文では，ObaseLang に焦点を絞って議論する．以下，2 章ではまず Obase データモデルの概要を述べる．また，3 章で ObaseLang の検索機能について述べ，4 章ではその意味論の概略を与える．さらに 5 章では，ObaseLang におけるメソッドについて述べる．また，6 章では関連する研究との比較を行う．7 章では考察と今後の研究課題について言及する．

2. Obase データモデル

Obase データモデルは，マルチメディアデータの柔軟な表現を目的として開発され，以下のような特徴を持つ．

1. 各オブジェクトは集合と組の両方の構造的構成子を持つ．
2. クラスとインスタンスオブジェクトを区別せずに統一的に扱う．このため，クラスも一つのオブジェクトとなる．また型の概念は存在しない*．

Obase データベースは，**Obase オブジェクト**と**関連プロパティ**から成る．

Obase オブジェクトは，**オブジェクト識別子**(oid)，**サブオブジェクト集合** (subobjects)，**プロパティ組** (property tuple) の三つの構成子から構成される．この三つ組表現を，

オブジェクト識別子（サブオブジェクト集合，<プロパティ組>）

と表すことにする．

オブジェクト識別子は，オブジェクトを一意に識別するものであり一つのデータベース中では唯一性が保証される．すなわち，一つのデータベース中ではある一つのオブジェクト識別子を持つオブジェクトは高々一つしか存在しない．サブオブジェクト集合は，オブジェクト識別子の集合である．サブオブジェクト集合の要素に対し，それを含むもとのオブジェクトを**スーパオブジェクト**と呼ぶ．スーパ/サブオブジェクト関連が表す意味は，従来の OODBMS でサポートされてきた a-kind-of 関連，instance-of 関連および member-of 関連を包含している．スーパ/サブオブジェクト関連全体は，非巡回有向グラフを構成し，データベース内には推移的に全オブジェクトのスーパオブジェクトとなっている特別のオブジェクト“Objectworld”が存在するものとする．

Obase オブジェクトのプロパティ組は，0 個以上のプロパティから成る組である．各プロパティは属性，制約，あるいはメソッドを表す．各プロパティは，

$Ia=x$

の形で表現される．ここで，I は**継承制御子**並び，a はプロパティ名，x はプロパティ値である．プロパティ値としては，原子値，オブジェクト識別子に加え，経路式，SQL 文や Obase メソッド定義文も格納することができる．Obase データモデルでは，多くのオブジェクト指向データベース同様，文字列，数値，ブール値などの原子値を通常のオブジェクトと区別している．I は継承制御子↓，↑，→，←を 0 個以上並べたものである．オブジェクト x のあるプロパティの値としてオ

* 「クラス」と「型」の用語の意味は，基本的には代表的なオブジェクト指向データベースシステムである O_2[5] に従う．

```
person({student, professor, employee, ... }, <↓ name=string,...>)
student({田中_1, 田中_2, 堀_1, ... }, <...>)
professor({植村_1, 鈴木_1}, <...>)
employee({田中_1, ... }, <manager = ↓ employee, ...>)
田中_1({...}, <name='田中', age=29, supervisor=植村_1, ... >)
田中_2({...}, <name='田中', age=22, supervisor=(田中_1.supervisor),
               college=('奈良先端大'!name), ... >)
堀_1  ({...}, <name='堀', age=24,
               supervisor=(SELECT $x FROM professor $x WHERE area='database'), ... >)
植村_1({...}, <name='植村', age=51, area='database', area='multimedia'>)
鈴木_1({...}, <name='鈴木', age=27, area='network'>)
(student, 田中_1)<↓ evaluation='bad'>
(employee, 田中_1)<↓ evaluation='good'>
```

図1 Obase データベースの例
Fig. 1 An example Obase database.

ブジェクト y が格納されている場合，y を x の**部品オブジェクト**，x を y の**全体オブジェクト**と呼ぶことにする．各プロパティの継承制御子の意味は，↓（↑）が，そのプロパティの値をサブオブジェクト（スーパオブジェクト）に継承可能であることを表し，→（←）が，その値を部品オブジェクト（全体オブジェクト）に継承可能であることを表す．これらの継承制御子が付されたプロパティは，3.3節で述べる継承演算子を実行することにより，実際に継承される．

次に関連プロパティの説明に移る．関連プロパティとは，あるオブジェクト x とそのサブオブジェクト y 間のスーパ/サブオブジェクト関連自身に付加されたプロパティであり，オブジェクト x の影響下でのサブオブジェクト y の性質，言い替えればオブジェクト x の側面から見たサブオブジェクト y の性質を表現する．これは，

（オブジェクト識別子 x，オブジェクト識別子 y）
＜プロパティ組＞

のように表す．ここでオブジェクト識別子 y で指定されるオブジェクトは，オブジェクト識別子 x で指定されるオブジェクトのサブオブジェクトでなければならない．関連プロパティはこのオブジェクト識別子の順序対で識別される．すなわち，オブジェクト識別子の順序対が同じでプロパティ組が異なるような複数個の関連プロパティは存在し得ない．また，オブジェクト識別子の順序対 (x, y) で識別される関連プロパティ中の各プロパティの継承制御子は↑または↓のみとし，それぞれ対応するプロパティがオブジェクト x, y に継承可能であることを表す．

図1に学生，教官，従業員に関する Obase データベースの例を与える．この例において，オブジェクト'田中_1' はプロパティ値すべてに原子値またはオブジェクト識別子が代入されている．それに対し'田中_2' のプロパティ supervisor と college には経路式を含み，'堀_1' のプロパティ supervisor には SQL 文を含む．これらについては5章で述べる．また，田中_1 オブジェクトは二つのオブジェクト student と employee のサブオブジェクトとなっており，これは田中_1 オブジェクトが社会人学生であることを表す．田中_1 オブジェクトとこれら二つのスーパオブジェクトとの間にはそれぞれに関連プロパティが定義されている．これらの関連プロパティは，学生としての田中_1 オブジェクトの評価は 'bad' であるのに対し，従業員としての田中_1 オブジェクトの評価は 'good' であることを表す．このような関連プロパティは3.3節で述べる継承演算を実行することにより検索できる．

3. ObaseLang の検索機能

3.1 Obase 質問言語の設計方針と特徴

Obase データモデルのための質問言語を設計する上では，以下の点を目標とした．

- 関係データベースモデルにおいて確立された「宣言的な集合演算」の導入というデータベース言語設計上の重要な概念を継承すること
- 簡潔な表現で強力な検索機能を実現すること

Obase 質問言語の特徴としては，以下の点を挙げることができる．

1. オブジェクト指向データベースでは一般にポインタを用いてオブジェクト間のリンクが表現される．このリンク構造を航行するために多くの質問言語で経路式（path expression）が導入されている．Obase 質問言語においても質問中に経路式を書くことができる．我々の経路式は，従来提案されてきた経路式の概念を拡張した強力な表現能力を持つ．拡張点としては以下のものを挙げることができる：

(a)2章で述べたようにObaseデータモデルの構造上の特徴の一つに各オブジェクトが集合(すなわちサブオブジェクト集合)と組(すなわちプロパティ組)の両方の構造を有することがある．Obaseデータベース全体は，各オブジェクトを節点，オブジェクトとそのサブオブジェクトとの関連を枝，オブジェクトとそのプロパティ組の要素との関連を別の種類の枝とすると，2種類の枝を持つグラフとして表現できる．従って，データベース航行演算子も組，集合構造に応じて2種類のものを用意し，それに応じて経路式も組と集合の両方を組み込んでいる．従来の経路式(たとえば文献6))は組構造のみを対象としてきたが，Obase質問言語では集合構造も含む点が異なる．

(b)Obaseデータモデルでは，オブジェクト間のプロパティの継承方法は構造に基づいてあらかじめ決められているのではなく，継承演算子を用いて実行時に明示的かつ動的に指定する．経路式中には組，集合構造に基づいた継承演算を含めることができる．

(c)正規表現を用いることにより，上述の航行演算や継承演算の閉包機能を含む強力な経路式を書ける．

2. 言語の基盤はObase代数[3]に基づいている．Obase代数では，各演算子のオペランドおよび結果は必ずObaseオブジェクトの集合である．従って，更新文も含め，Obase質問文の結果は必ずObaseオブジェクト集合である．
3. 構文は了解性のため基本的にはSQLに倣うが，自由度を高くし通常のSQLを拡張した多様な表現法を許す．(たとえば，FROM句が無いようなSQL文も許す．)

このうち，3については，3.2節でさらに詳しく議論する．

3.2 Obase質問言語の構文について

Obase質問言語のうち，検索文については一般的に目的オブジェクト，変数の範囲，検索条件式の指定によって表現される．これらはSQLではそれぞれSELECT句，FROM句，WHERE句で指定される．一般論としてはObase検索文の構文は，これらの検索文の要素がわかりやすく表現できるものであればどのようなものでも良く，必ずしもSQLのようにこれらの要素を別々の句で表現する必然性はない．また，Obaseモデルの場合は，他の多くのオブジェクト指向データベースモデルとは異なり，クラスとインスタンスの明確な区別をしておらず，しかも各オブジェクトが型を持たない．型を持つオブジェクト指向SQLではFROM句において変数の範囲の宣言を行うことにより同時に変数の型宣言も行うことになるが，Obase検索文では型宣言という意味でのFROM句の存在意義はない*．ある変数がある集合の要素であることを表現する述語を導入することにより，変数の範囲はFROM句ではなくWHERE句で指定することもできる．さらに，SQLに馴染みがない利用者の中には，SQLのSELECT, FROM, WHERE句による3要素による表現よりも，一階述語論理の表現，

{目的オブジェクト|
　　変数の範囲指定および検索条件式}

のような2要素による指定の方が自然に質問を表現できる場合があると考えられる．

しかし一方では，データベース質問言語の標準として確立しているSQL[7]は，既に多くの利用者に馴染みがあるため，SQLと全く異なる構文を採用することは現実的ではない．

以上のことを総合的に考慮し，Obaseデータモデルの質問言語は，

「了解性と互換性を考慮し，構文は基本的にSQLに倣うこととするが，大きな特徴としてFROM句を持たないSQL文も許す．」

という選択を行った．

ObaseLangの検索文のうちSQL構文に倣う部分をObaseSQL検索文と呼ぶことにする．ObaseLangの検索文およびObaseSQL検索文の基本構文をそれぞれ**図2**と**図3**に示す．図2の2行めにあるように単独の経路式自身もObase検索文である．これは，Obase-SQL検索文の

SELECT＜経路式＞　　(Q1)

と等価である．もともとSQLのWHERE句は省略可能であり，またすでに述べたようにObaseSQL検索文ではFROM句も省略可能としている．従って，ObaseSQL検索文は最も簡単な場合，SELECT句のみから成る．SELECT句には一つ以上の経路式の列を指定する．ただし，(Q1)のように指定する経路式が一つの場合は，予約語である'SELECT'も結果的に省略でき，経路式をそのまま書けることになる．

* もちろん，Obaseモデルにおいても検索文中に現れる変数をすべてFROM句で宣言した方が質問処理は容易になると考えられる．すなわち，この問題は一般には構文の自由度を高めることによる質問作成の簡便性と，質問処理の容易性とのトレードオフとなる．ここでは，まず質問作成の簡便性を優先して言語設計を行った．

```
〈 Obase 検索文 〉 ::= '{' 〈 オブジェクト識別子 〉 {, 〈 オブジェクト識別子 〉 ... } '}'
                    | 〈 経路式 〉
                    | 〈 ObaseSQL 検索文 〉
                    | 〈 Obase 集合演算文 〉

〈 Obase 集合演算文 〉 ::=  (〈 Obase 検索文 〉) INTERSECT (〈 Obase 検索文 〉)
                          | (〈 Obase 検索文 〉) UNION (〈 Obase 検索文 〉)
                          | (〈 Obase 検索文 〉) EXCEPT (〈 Obase 検索文 〉)

注: '{' と '}' は、BNF 記法のメタ記号ではなく、言語の終端記号としての { と } を表す。
```

図2 ObaseLang の検索文の基本構文
Fig. 2 The core part of the syntax of ObaseLang retrieval sentences.

```
〈 ObaseSQL 検索文 〉 ::= 〈 SELECT 句 〉 [ 〈 FROM 句 〉 ] [ 〈 WHERE 句 〉 ]

〈 SELECT 句 〉 ::= SELECT 〈 経路式 〉 {, 〈 経路式 〉 ... }

〈 FROM 句 〉   ::= FROM 〈 変数宣言 〉 {, 〈 変数宣言 〉 ... }
〈 変数宣言 〉  ::= 〈 副 Obase 検索文 〉␣〈 変数 〉
                 | 〈 最右セレクタ付き経路式 〉
〈 副 Obase 検索文 〉 ::= '{' 〈 オブジェクト識別子 〉 {, 〈 オブジェクト識別子 〉 ... } '}'
                       | 〈 経路式 〉
                       | (〈 ObaseSQL 検索文 〉)
                       | (〈 Obase 集合演算文 〉)

〈 WHERE 句 〉  ::= WHERE 〈 探索条件 〉
〈 探索条件 〉  ::= 〈 ブール項 〉
                 | 〈 探索条件 〉 OR 〈 ブール項 〉
〈 ブール項 〉  ::= 〈 ブール因子 〉
                 | 〈 ブール項 〉 AND 〈 ブール因子 〉
〈 ブール因子 〉 ::= [ NOT ] 〈 ブール一次子 〉
〈 ブール一次子 〉 ::= 〈 述語 〉 | (〈 探索条件 〉)
〈 述語 〉  ::= 〈 最右セレクタ付き経路式 〉
             | 〈 最右セレクタなし経路式 〉 〈 比較演算子 〉 〈 最右セレクタなし経路式 〉
             | 〈 セレクタ 〉 IN 〈 副 Obase 検索文 〉
             | EXISTS 〈 副 Obase 検索文 〉
〈 比較演算子 〉 ::= = | != | < | <= | > | >=
```

図3 ObaseSQL 検索文の基本構文
Fig. 3 The core part of the syntax of ObaseSQL retrieval sentences.

通常の SQL は等価な質問を幾通りにも書くことができる* という意味において冗長な言語であるが，上で述べたように ObaseSQL 検索文も，この意味において非常に冗長である．たとえば，30 歳以上の教員の名前と研究分野を求める場合，標準的な SQL 構文に従った記法では，

```
SELECT $x.name, $x.area
FROM professor $x                    (Q 2)
WHERE $x.age >= 30
```

となる．ここで$x はオブジェクト変数を表す．ObaseSQL 検索文において FROM 句は変数の範囲指定を行う役割を果たし，(Q 2)の場合$x の範囲は professor オブジェクトのサブオブジェクト集合である*．従って，WHERE 句に集合のメンバーシップ(∈)に相当する比較演算子を許すならそれにより FROM 句の代替ができ，FROM 句がない場合でも表現能力としては同じとなる．ObaseSQL 検索文では，“IN”が集合のメンバーシップを表し，またセレクタ(3.3.1 項参

* たとえば一般に結合を含む質問は，入れ子質問（nested query）によって表現することもできるし，FROM 句に結合の対象となる関係名を列挙することにより入れ子質問を使わない「フラットな」SQL 文によっても表現できる．

* 変数の範囲指定についてはより一般的に 4 章で述べる．

```
〈 経路式 〉   ::= 〈 セレクタ 〉[ 〈 セレクタ付経路演算子 〉 ]
〈 最右セレクタ付経路式 〉 ::=
    〈 セレクタ 〉[ 〈 セレクタ付経路演算子 〉〈 最右セレクタ付航行演算子 〉 ]
〈 最右セレクタなし経路式 〉 ::=
    〈 セレクタ 〉[ 〈 セレクタ付経路演算子 〉〈 最右セレクタなし航行演算子 〉 ]

〈 セレクタ 〉 ::= 〈 原子値 〉 | 〈 オブジェクト識別子 〉 | 〈 オブジェクト変数 〉

〈 セレクタ付経路演算子 〉 ::= 〈 セレクタ付航行演算子 〉
                    | 〈 セレクタ付航行演算子 〉〈 セレクタ付経路演算子 〉
                    | '{'〈 セレクタ付経路演算子 〉, ... ,〈 セレクタ付経路演算子 〉'}'
                    | (〈 セレクタ付経路演算子 〉)*
                    | (〈 セレクタ付経路演算子 〉)+

〈 セレクタ付航行演算子 〉 ::= 〈 最右セレクタ付航行演算子 〉
                        | 〈 最右セレクタなし航行演算子 〉
〈 最右セレクタ付航行演算子 〉 ::= 〈 組航行演算子 〉[〈 セレクタ 〉]
                          | 〈 集合航行演算子 〉〈 セレクタ 〉
〈 最右セレクタなし航行演算子 〉 ::= 〈 組航行演算子 〉
                            | 〈 集合航行演算子 〉

〈 組航行演算子 〉 ::= .〈 プロパティ名 〉 | !〈 プロパティ名 〉
                  |..〈 プロパティ名 〉 | !!〈 プロパティ名 〉
                  |.〈 無名プロパティ変数 〉 | !〈 無名プロパティ変数 〉
                  |..〈 無名プロパティ変数 〉 | !!〈 無名プロパティ変数 〉

〈 無名プロパティ変数 〉 ::= _

〈 集合航行演算子 〉 ::= / | \ | >> | <<
```

注: 基本的に通常の BNF 記法に従っているが、[と] は BNF メタ記号であり、これらで囲まれた構文要素は省略可能であることを表し、一方、[と] は、言語の終端記号である。

図4 経路式の構文
Fig. 4 The syntax of path expressions.

照)もその役割を果たす．たとえば ObaseSQL 検索文(Q 2)の FROM 句を WHERE 句の IN によって代替すると，

```
SELECT $x.name, $x.area
WHERE $x IN professor/
AND $x.age >= 30                    (Q 3)
```

のように表現できる．また，セレクタを用いることにより，

```
SELECT $x.name, $x.area
WHERE professor/$x AND $x.age >= 30
                                    (Q 4)
```

のように表現することもできる．ObaseSQL 検索文では，(Q 3)や(Q 4)のように質問中に FROM 句で宣言されていない変数が現れた場合，その変数は暗黙的にデータベース中の全オブジェクトを範囲とするものと解釈する，という意味論を採用している．

さらに，全教員の研究分野は，

```
SELECT $x.area
FROM professor $x
```

のような ObaseSQL 検索文で表現することも，

```
professor/.area
```

のような経路式で表現することもできる．

3.3 経 路 式

ObaseLang の特徴の一つに従来提案されてきた経路式をいくつかの点で拡張したより強力な経路式表現を支援していることを挙げることができる．ObaseLang における経路式の構文を図4に与える．本節では，従来の経路式からの拡張点を中心に ObaseLang の経路式の特徴を例を用いて順に説明する．

3.3.1 組，集合航行演算子

2章で述べたように，Obase データモデルでは各オブジェクトが組と集合の両方の構造的構成要素を持つ．従って質問言語にはこれら両者の航行機能が必要となる．ObaseLang の検索文では Obase 代数に基づき，組の航行はドット(.)の後にプロパティ名を指定し，集合（すなわちサブオブジェクト）の航行はスラッシュ(/)を指定することにより実現する．航行の基

本的な単位は組の場合，

オブジェクト識別子.プロパティ名

となり，オブジェクト識別子で指定されたオブジェクトの対応するプロパティ値を返す．また集合航行の基本的な単位は

オブジェクト識別子/

となり，オブジェクト識別子で指定されたオブジェクトのサブオブジェクト集合を返す．また，これら航行の基本的な単位にはセレクタを付加することもできる．セレクタは，航行結果を特定するための原子値，オブジェクト識別子，または変数である．セレクタが変数の場合は，通常その変数がObase検索文中の他の場所でも使用され，同じオブジェクト識別子（または原子値）を参照することを表す．セレクタを含む場合の組，集合航行の基本的な単位はそれぞれ

オブジェクト識別子.プロパティ名[セレクタ]

オブジェクト識別子/セレクタ

となる．セレクタがオブジェクト識別子または原子値の場合，これらの航行の基本的な単位はいずれも真偽値を返す．また，セレクタが変数の場合，その変数に特定のオブジェクト識別子または原子値を代入した結果は，真偽値を返す．

たとえば，3.2節のObaseSQL検索文(Q 4)の例では，WHERE句にセレクタ付の集合航行演算子 professor/$x ($xがprofessorのサブオブジェクトに含まれるという述語を表す）とセレクタなしの組航行演算子$x.age ($xのageプロパティの値を表す）が現れている．なお，質問(Q 4)を，図1のデータベースに対して実行した場合，植村_1オブジェクトのみが質問の条件を満足し，質問結果は，新規に生成された単一のオブジェクトのみからなる集合

{植村_1' ({...}, <name='植村',
area='database', area='multimedia'>)}

となる*．

また通常のSQL同様，文字列の部分一致機能を持ち，これはセレクタ中でも使用可能とする．たとえば，名前の中に「田」という文字を含む従業員を求める場合は，

```
SELECT $x
FROM employee $x
WHERE $x.name = '%田%'
```

あるいは，

```
SELECT $x
WHERE (employee/$x).name['%田%']
```

のように表現することができる．(ここで%は任意の文字列と一致することを表す特殊記号である．)

また，あるオブジェクトのプロパティの値を求めたりサブオブジェクトの値を求める操作を順方向の航行とみなすと，逆方向の航行に相当する演算も支援している．すなわち，これは組構造については，あるオブジェクトまたは原子値に対しそれをあるプロパティの値として持つオブジェクトを求めることに相当し，集合構造に対しては，あるオブジェクトまたは原子値のスーパオブジェクトを求めることに相当する．これらの航行演算記号はそれぞれ!と\を用いる．従って，セレクタがない場合には，組構造の逆航行の基本的な単位は

オブジェクト識別子!プロパティ名

原子値!プロパティ名

となり，集合構造の逆航行の基本的な単位は

オブジェクト識別子\

原子値\

となる．また，順方向の航行の場合同様，セレクタを付加することができ，その場合は上記4種類の逆航行の基本的な単位はそれぞれ

オブジェクト識別子!プロパティ名[セレクタ]

原子値!プロパティ名[セレクタ]

オブジェクト識別子\セレクタ

原子値\セレクタ

となる．たとえば，'田中'という値をnamcプロパティとして持っているオブジェクトをすべて検索する場合は，

```
SELECT $x
WHERE $x ='田中'!name
```

あるいは，

```
SELECT $x
WHERE '田中'!name[$x]
```

のように書くことができる．

3.3.2 継承演算子

Obaseデータモデルでは継承をデータモデルにあらかじめ組み込まれた機能とは考えずに，オブジェクト同上の操作によって実現し，これにより利用者が継承を行うオブジェクトの範囲を実行時に自由に指定しオブジェクトを多面的に表現できる機構を提供している．継承演算は，組，集合構造に沿って実行することができ，それぞれ..プロパティ名，および>>という記法を用いる．組構造に沿った継承演算の基本的な単位は，

オブジェクト識別子..プロパティ名

である．たとえば，oをオブジェクト識別子，aをoの

* 検索文の意味論に関するより詳しい説明は4章で行う．

プロパティ名とすると，$o..a$ は，o のプロパティのうち継承制御子→が付されたものを $o.a$（一般に集合）の各要素に継承した結果得られる新たな*オブジェクトの集合を表す．また，集合構造に沿った継承演算の基本的な単位は，

オブジェクト識別子＞＞

である．たとえば，o をオブジェクト識別子とすると，o＞＞は，o のプロパティのうち継承制御子↓が付されたものを $o/$ の各要素に継承した結果得られる新たな* オブジェクトの集合を表す．ただし，$o_i \in o/$ に対して o と o_i の間のスーパ/サブオブジェクト関連にもし関連プロパティが定義されていれば，そのプロパティのうち継承制御子↓が付されたものも合わせて o_i に継承する．継承演算子の場合も組，航行演算子同様セレクタを指定してもよい．

たとえば，従業員としての評価が'good'である人物の名前を求める場合を考える．図1のObaseデータベースにあるように，プロパティevaluationは，employeeオブジェクトそのものや個々の従業員オブジェクトそのものではなく，employeeオブジェクトと個々の従業員オブジェクトのスーパ/サブオブジェクト関連の関連プロパティに登録されている．これは，ある人物が従業員かつ学生であった場合，従業員としての評価と学生としての評価は一般に異なることを表す．そのため，ある人物の従業員としての評価を調べるためにはemployeeオブジェクトからその人物への継承演算を実行し，それらの間のスーパ/サブオブジェクト関連に付されている関連プロパティも合わせてその人物へ継承する必要がある．この質問は，

```
SELECT $x.name
FROM employee＞＞ $x
WHERE $x.evaluation＝'good'
```

または，

```
SELECT (employee＞＞$x).name
WHERE (employee＞＞$x).evaluation
['good']
```

によって表現できる．最初のObaseSQL文のemployee＞＞は，employeeオブジェクト（およびemployeeオブジェクトと各サブオブジェクトとの間の関連プロパティ）を各サブオブジェクトに継承して得られる新たなオブジェクトの集合を返す．従って，変数$xは継承後のオブジェクトを表す．また，2番目のObaseSQL文はセレクタ$xを用いた例である．この例において，employee＞＞$xは，$xがemployeeのサブオブジェクトの場合のみ，employeeオブジェクト（およびemployeeオブジェクトと$xとの間の関連プロパティ）を$xに継承して得られる新たなオブジェクトを返す．従って，この場合は変数$xは継承前のオブジェクトを表す．

3.3.3 経路式の正規表現

ObaseLangにおける経路式の大きな特徴として，経路式の正規表現を導入することにより経路の推移的閉包を表現できる点がある．正規表現を用いることにより，利用者はデータ構造をすべて明示的に指定することなく検索を行うことができる．これは，Obaseモデルのように統一的なスキーマ構造を必ずしも持たず構造自身が段階的に進化していく半構造化データベースを前提としたモデルでは特に有効な機能である．

たとえば，自動車のデータベースを例にとる．自動車はそれ自身が巨大な複合オブジェクトであり，多数の部品から成る．ドアもその部品の一つであると考えると，ドアは自動車から見た場合，部品階層のいずれかのレベルに存在することになる．また，一般にドアの色は自動車からドアに至る経路のいずれかのレベルのオブジェクトのプロパティが継承されたものである．**図5**のデータベースでこれを説明する．図中，長方形はオブジェクトを表す．また，長方形の間を結ぶ太い矢印はオブジェクトからサブオブジェクトへの関連を表し，細い矢印はオブジェクトから部品オブジェクトへの関連を表す．さらに各長方形の内部にはプロパティが記されており，プロパティに継承制御子が定義されている場合は，プロパティ名の上の矢印がそれを表す．

自動車オブジェクトalの場合は，プロパティdoor

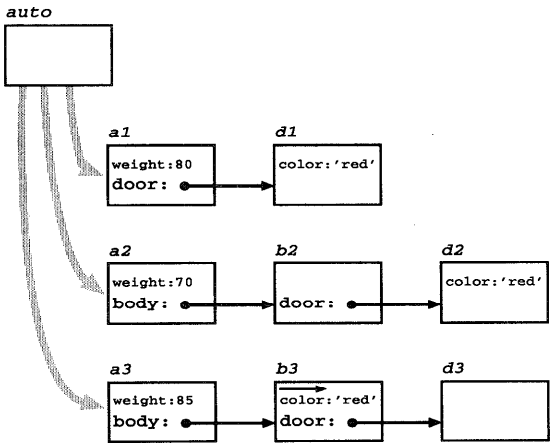


図5 自動車データベース
Fig. 5 An automobile database.

* 継承後のオブジェクトは継承前のオブジェクトとは異なるため，継承演算に伴いオブジェクト識別子の生成が行われる．

として格納されているドアオブジェクト d1 にドアの色が記述されている．それに対し，a2 の場合は，より大きな部品としてまずボディがオブジェクト b2 としてモデル化され，b2 の部品としてドアオブジェクト d2 がモデル化されており，a1 の場合とは部品展開の構造が異なる．さらに a3 の場合は，部品展開構造は a2 と同様であるが，色に関する情報はボディオブジェクトに記述され，それがドアオブジェクトに継承可能であることが指定されている．一方，weight プロパティは a1, a2, a3 いずれのオブジェクトにも定義されているが継承制御子は付されていない．(当然のことながら，自動車の重量はそのままドアの重量ではない．)

このようなデータベースに対して利用者が，赤いドアの自動車を求めたい場合を考える．この場合に，利用者が上記データベースのような複雑な部品階層に対応するデータ構造を理解しなければならないとすると過剰な負担を強いることになる．

そこで正規表現を用いることにより，自動車からドアに至る経路のどこで「色」プロパティが定義されていても，その定義場所を明示的に指定することなくこのような質問を表現することができる．以下に ObaseSQL 検索文によるこの質問の表現例を与える．

```
SELECT $x
FROM auto $x                                (Q 5)
WHERE $x(._) *..door.color = 'red'
```

ここで * は閉包演算子であり，直前の括弧内の経路式を 0 回以上繰り返すことを表す．また括弧内の下線(_)は無名プロパティ変数でありどのようなプロパティ名ともマッチすることを表す．たとえば，図 5 の auto オブジェクトのサブオブジェクトのうち a3 を例にとると，経路式

```
a3..body..door.color
```

が，上記 ObaseSQL 文(Q 5)の WHERE 句の左辺にマッチする．この場合，a3..body の結果はプロパティ組として<→ color＝'red', door＝d3>を持つ．ここで，オブジェクト a3 のプロパティ weight と body は継承制御子→が付されていないため継承されていないことに注意されたい．また，a3..body の結果に対してさらに継承演算..door を実行した結果得られるオブジェクトのプロパティ組は<→ color＝'red'>となる．この場合は，二つのプロパティ color と door のうち，継承制御子→が付されている color のみが継承されている．

また，いわゆる再帰質問（recursive query）も表現可能である．たとえば，田中_1 オブジェクトの（直接および間接の）上司すべては

```
SELECT $x
FROM 田中_1(.manager)+ $x
```

のような ObaseSQL 検索文によって求めることができる．

3.4 SPJ 質問

関係データベースでは，制約，射影，結合演算によって表現されるいわゆる SPJ 質問が典型的な質問であるが，Obase データベースに対しても，同様の質問を表現できる．たとえば，共通の研究分野を持つ教官の名前の対を求める質問は ObaseSQL 検索文では，関係データベース同様

```
SELECT $x.name, $y.name
FROM professor $x, professor $y
WHERE $x.area = $y.area
AND $x.name < $y.name
```

のように書くこともできるが，

```
SELECT $x.name, $y.name
FROM (professor/$x).area $z,
     (professor/$y).area $z
WHERE $x.name < $y.name
```

のように書くことができる．この ObaseSQL 検索文において，オブジェクト変数$z のスコープは FROM 句内全域である．このように FROM 句において同じ変数を二度以上宣言することによりその変数の範囲が対応する複数個のオブジェクト集合の共通集合であることを表現できる．このような表現による結合の実現は従来の SQL では考えられていなかった新しい表現法である．また，この例のような射影，直積演算に対応する質問の意味論は Obase 代数の射影，直積の意味論に準じる．従ってこの場合，質問結果は新たに生成されるオブジェクトの集合となり，結果の各オブジェクトのプロパティは，$x と$y の name プロパティ値である文字列の組である．

4. 検索文の意味論

ObaseLang の検索文の意味論の概略を以下に与える．

1. 検索文が経路式の場合は，先頭に'SELECT' を付加することにより ObaseSQL 検索文に変換する．
2. FROM 句で指定された副 Obase 検索文* を実行した結果得られる集合を求める．ただし，副 Obase 検索文として単一のオブジェクト識別子が指定されている場合は，そのサブオブジェクト

* 図 3 参照．Obase 検索文と基本的には同じだが，構文上のあいまいさをなくすために必要な箇所に括弧が付けられている．

集合を求める*.以上求めた集合を変数の範囲とする．すなわちこの集合の各要素をFROM句においてその副Obase検索文に対応する変数に順に代入する．なお，FROM句で宣言されていない変数$xが，質問文中に現れる場合は，暗黙的に

```
Objectworld(/) * $x
```

という宣言がされているものと解釈する．(この場合，FROM句が無いときは質問文中のすべての変数に対してこのような宣言がされているものと解釈する．)また，FROM句内で同じ変数に対し2つ以上の集合が対応付けられた場合，それらの集合の共通集合が対応付けられたものと見なす．

3. (a) WHERE句が無い場合，またはステップ2の代入結果がWHERE句で指定された探索条件を満足する場合は，その各代入値をSELECT句で与えられる（メソッドを含む）経路式の列中の対応する変数に代入する．
   i. SELECT句に指定された経路式が単数個の場合，代入結果の(既存または新規)オブジェクトを検索結果に追加する．
   ii. SELECT句に指定された経路式が複数個の場合，代入の結果得られる組をプロパティ組として持つ新たなオブジェクトを生成し，それを検索結果に追加する．(組の各要素はSELECT句の各経路式に対応)

   (b) ステップ2の代入結果がWHERE句で指定された探索条件を満足しない場合は，結果に何も追加しない．
4. ステップ2,3をFROM句で指定された集合のすべての要素に対して実行する．

5. メソッド

Obaseにおけるメソッドは，導出属性とローカルメソッドから成る．

導出属性は，オブジェクトのプロパティ値に経路式やObaseSQL検索文の格納を許し，その遅延評価機構により実現する．言語の意味論を単純化するためプロパティ値として更新文を格納することは許していない．たとえば，図1のObaseデータベースでは，オブジェクト'田中_2'のプロパティsupervisorとcollegeには経路式を含み，オブジェクト'堀_1'のプロパティsupervisorにはObaseSQL検索文を含む．これらは括弧で囲まれているため構文的に原子値やoidと区別が可能である．導出属性の値は検索時に評価される．たとえば，田中_2.supervisorという経路式は評価されるとまず経路式田中_1.supervisorが検索され，次にこの経路式を評価することにより最終的に植村_1を返す．

ローカルメソッドは，システムの実装言語であるSmalltalk[8]のブロック文とその引数としてのObase検索文を組み合わせることにより，計算可能な任意の関数を実現するものである．構文は，

```
(
   (<Obase検索文>)(<Obase検索文>)...
   '<Smalltalkブロック文>'
)
```

で与えられ，$n(\geq 0)$個の<Obase検索文>と引数をn個持つSmalltalkブロック文が指定される．システム内部では以下のようにSmalltalkのvalue:メソッドを用いることにより，<Obase検索文>の評価結果を順に引数として利用することによりSmalltalkブロック文を実行する．

```
<Smalltalkブロック文>
     value: <Obase検索文>
     value: <Obase検索文>
     ...
```

たとえば，学生とその指導教官との年齢差を返すローカルメソッドの定義文は以下のようになる．

```
(
   (supervisor.age)(age)
   '[ :i :j | i-j]'
)
```

6. 他のオブジェクト指向データベース言語との比較

表1にObaseLangと他の代表的なオブジェクト指向データベース言語との比較を示す．比較対象とした言語は，ObjectStore[9),10)]，ONTOS[11),12)]，O_2[5),13),14)]，XSQL[6]，OOPC[15]である．

Obaseデータベースはデータモデルの点では各オブジェクトが組と集合の両方の構造を持つ点が特徴的である．それに伴いObaseLangは2種類の航行演算子を持つ．また，オブジェクト間の継承自身を操作言語によって制御することによりオブジェクトの多様な側面の表現を可能としている点が大きな特徴であり，航行演算子は継承を伴うものと伴わないものの2種類が用意されている．また，セレクタを持つ閉包演算子を完全に支援している点および構文面ではFROM句

* これは，通常のオブジェクト指向データベースにおいてFROM句でクラス名を指定すると集合としてそのクラスの外延（extension）が求められることにならっている．

表1 オブジェクトデータベース言語の比較
Table 1 A comparison of object database languages.

			ObjectStore	ONTOS	O_2	XSQL	OOPC	ObaseLang
データモデル	オブジェクト（または値）の構成子（注1）		組	組	組、集合、リストのいずれか	組	組	組と集合の両方
	集合値属性		許す（Collectionクラス）	許す（Aggregateクラス）	許す	許す	許す	許さない
言語	継承	継承可否の指定	×	×	×	×	×	○
		操作言語による継承の制御	×	×	×	×	×	○
	経路式	航行演算子	組	組	組、リスト	組	組	組、集合
		メソッド変数、経路変数	△（注2）	×	×	○	×	△（注3）
		再帰質問	○（foreach loopと集合演算による）	×	×	×	○（再帰演算子ρによる）	○（閉包演算子による）
		条件文（限量子）	経路式同士の比較	経路式同士の比較	経路式同士の比較（for all ... in ..., exists ... in ...）	経路式同士の比較（*some*, *all*）	経路式同士の比較（$\exists, \forall$）	経路式同士の比較
	構文		C++の拡張	SQLの拡張	SQLの拡張	SQLの拡張	SQL様の3つの節から成る	SQLの拡張（ただしFROM句も省略可）
	質問結果		既存オブジェクト集合の部分集合	結果が既存の型に合わない場合はQueryIteratorによってC++に渡す	任意の複合値	新規オブジェクトの生成可能	新規オブジェクトの生成可能	新規オブジェクトの生成可能

注1: オブジェクト（または値）自身の構成子であり、その属性として参照されるオブジェクト（または値）の構成子ではない。

注2: 経路式自身をオブジェクトとして定義可能。

注3: 無名属性と閉包の組合せにより質問条件部に現れるメソッド変数および経路変数の意味論は表現可能。

も省略可能である点が大きな特徴である．

7. 考　　察

ObaseLangの構文はSQLにも準拠しつつ自由度を高くしており，利用者の習熟度や嗜好に応じて使いやすい構文を選択できる．また，ObaseLangはObaseモデルの特徴である組，集合の2種類の構造構成子を用いた柔構造データの多様な航行を表現するため，動的継承や正規表現機能を持つ経路式を支援している．Obaseシステムの利用者は，実際には本論文で述べたObaseLangとグラフィカルユーザインタフェースであるObaseGUI[4]を併用する．ObaseGUIは，あるオブジェクトからの局所的な航行に適しており，ObaseLangは大域的な条件検索に適している．両者を相補的に利用することにより，ObaseGUIによる航行によって得られたオブジェクト識別子を検索の手がかりとして次のObaseLang文を実行できる．このように少数のオブジェクトを手がかりに検索によって既知のオブジェクトを段階的に増やしていくことが可能である．

ObaseLangの実際の使用経験を通じて，これらの機能の有効性を確認できた．しかし，大規模なデータベースに対してより使いやすいシステムとするためには，データモデル，システムに対して改良すべき点が存在することも明らかとなった．これらの点としては以下のものがあり，今後の重要な研究課題である．

1. Obaseモデルは，各オブジェクトが自由な構造を持ち，オブジェクトの柔軟な生成，変更，削除によってボトムアップ的にデータベースを構築するためには優れたモデルである．ただし，大量のオブジェクトを考えた場合，データベース内のすべてのオブジェクトに自由な構造を持たせるのは現実的ではなく，実際には構造化され比較的安定したオブジェクトの部分と非構造化オブジェクトの部分がデータベース内に混在するものと考えられる．このような大量の構造化，非構造化両者のオブジェクトを持つデータベースに対する質問の手がかりを利用者に与えるためには，上述のように段階的に既知オブジェクトを増やしていくだけではなく，データベースの全体像の俯瞰図を利用者に提供し，代表的なオブジェクトやオブジェクトの関連を要約して提示するなど，オブジェクト構造の抽象化機構が必要であろう．これは非構造化オブジェクトを含む点で，トップダウン的に定義される通常のスキーマと同一のものではないと予想される．たとえば，3.3.3項の質問(Q5)を作成するためには，利用者はautoオブジェクトの部品階層の詳細を知る必要はないが，「部品オブジェ

クトを継承を伴ってたどって行くとdoorオブジェクトのcolorプロパティを得ることができる」という知識をシステムから簡潔な形で与えられている必要がある．

このような抽象化機構をデータモデルに装備し，それを前提としたObaseLangとObaseGUIとの統合型利用者インタフェースモデルの構築が重要である．

2. 現システムはプロトタイプであり，質問処理方法は単純なアルゴリズムを用いている．そのため言語の持つ豊富な機能すべてに対して十分な処理性能は得られていない．システムの高速化のためには，ObaseLangの特徴である動的継承や正規表現に適した処理最適化アルゴリズムの開発が重要である．

8. あとがき

Obaseシステム[2]は，ObjectWorks\Smalltalk (R) 4.1を用いてプロトタイプを実装し，SPARCstationおよびMacintosh上で稼働している．Obaseシステムの重要な構成要素としては，ObaseLang以外に，動的ルール機構ObaseECA，グラフィカルユーザインタフェースObaseGUI[4]，並行処理制御部などがある．

本論文では，Obaseシステムのための言語ObaseLangについて述べ，実際の使用経験を通じてObaseモデルとシステムをより洗練されたものにするための新たな研究課題を明らかにした．

謝辞 ご討論頂いたObaseコンソーシアムの諸氏に衷心より感謝の意を表します．また，種々の有益なコメントを頂戴した査読者の方々に深謝いたします．

参考文献

1) Tanaka, K., Nishio, S., Yoshikawa, M., Shimojo, S., Morishita, J. and Jozen, T.: Obase Object Database Model: Towards a More Flexible Object-Oriented Database System, *Proc. of Int. Symp. on Next Generation Database Systems and Their Applications,* pp. 159-166 (1993).

2) Tanaka, K., Nishio, S., Yoshikawa, M., Shimojo, S. and Jozen, T.: Obase: An Instance-Based Object Database System with Dynamic Inheritance and Active Rule Mechanisms, 情報処理学会データベースシステム研究会研究報告，第94-DBS-100巻 (1994).

3) 吉川正俊：構造検索機能と継承演算子を持つオブジェクトベース代数，情報処理学会データベースシステム研究会研究報告，第93-DBS-95巻 (1993).

4) Pradhan, S., Jozen, T. and Tanaka, K.: Navigation of an Instance-based Object Database with Dynamic Inheritance Capability, *Proc. of the Int. Symp. on Advanced Database Technologies and Their Integration,* pp. 23-30 (1994).

5) Bancilhon, F., Delobel, C. and Kanellakis, P. eds.: *Building an Object-Oriented Database System: The Story of O_2,* Morgan Kaufmann (1992).

6) Kifer, M., Kim, W. and Sagiv, Y.: Querying Object-Oriented Databases, *Proc. of ACM SIGMOD Int. Conf. on Management of Data,* pp. 393-402 (1992).

7) 日本規格協会：JIS X 3005-1990 データベース言語SQL (1990).

8) Goldberg, A. and Robson, D.: *Smalltalk-80: The Language and Its Implementation,* Addison-Wesley, Reading, Mass. (1983).

9) Object Design, Inc.: *ObjectStore Reference Manual, Release 2.0* (1992).

10) Object Design, Inc.: *ObjectStore User Guide, Release 2.0* (1992).

11) ONTOS, Inc.: *ONTOS Object SQL Guide for ONTOS DB Release 2.2* (1992).

12) ONTOS, Inc.: *ONTOS DB 2.2 First Time User's Guide* (1992).

13) Lecluse, C. and Richard, P.: The O_2 Database Programming Language, *Proc. of the 15th Int. Conf. on Very Large Data Bases,* pp. 411-422 (1989).

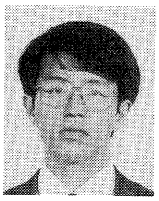
14) Bancilhon, F., Cluet, S. and Delobel, C.: The O_2 Query Language Syntax and Semantics, Technical Report 45-90, Altair (1990).

15) Bertino, E., Negri, M., Pelagatti, G. and Sbattella, L.: Object-Oriented Query Languages: The Notion and the Issues, *IEEE Trans. on Data and Knowledge Engineering,* Vol. 4, No. 3, pp. 223-237 (1992).

(平成6年5月25日受付)
(平成7年1月12日採録)

吉川 正俊（正会員）

1980年京都大学工学部情報工学科卒業．1985年同大学院博士後期課程修了．工学博士．同年京都産業大学計算機科学研究所講師．同大学工学部助教授，南カリフォルニア大学客員研究員を経て1993年より奈良先端科学技術大学院大学情報科学研究科助教授．データベースモデル，操作言語などの研究に従事．電子情報通信学会，ACM，IEEE各会員．



田中 康暁（正会員）

1964年生．1986年大阪大学工学部卒業．1988年同大学院工学研究科博士前期課程修了．同年NTT入社．1994年奈良先端科学技術大学院大学情報科学研究科博士前期課程修了．現在NTT境界領域研究所研究主任．

田中 克己（正会員）

昭和26年生．昭和49年京都大学工学部情報工学科卒業．昭和51年同大学院修士課程修了．京都大学工学博士．神戸大学教養部助手，神戸大学工学部助教授を経て，平成6年神戸大学工学部教授（情報知能工学科），現在に至る．主にデータベース（RDB，履歴DB，OODB）の研究に従事．昭和60年，情報処理学会創立25周年記念論文採択，平成3年度情報処理学会Best Author賞受賞．現在IEEE Computer Society，ACM，日本ソフトウェア科学会，人工知能学会各会員．

蛭井 潤（正会員）

1970年生．1992年神戸大学工学部卒業．1994年同大学大学院工学研究科修士課程修了．1994年オージス総研アドバンストシステム本部オブジェクト技術部．1995年現在，同社新技術研究所に所属し，プロセスコントロール支援システム構築ツールの開発に従事．

上善 恒雄

1962年生．1988年京都大学大学院工学研究科数理工学専攻修了．現在，阪急電鉄（株）文化・技術研究所と（財）千里国際情報事業財団を兼務．

堀田光治郎

1990年3月神戸大学工学部計測工学科卒業．1990年4月コベルコシステム株式会社入社．現在技術本部技術・商品開発部所属．オブジェクト指向技術の研究開発に従事．