# 関数的プログラミング・パラダイムによるデータベース検索言語の設計*

6P−5

種茂 文之† 渡辺 豊英† 杉江 昇†

名古屋大学 工学部 情報工学科‡

## 1 はじめに

近年、次世代のデータベースの一つとして、オブジェクト指向データベース(OODB)への関心が高まってきている。OODBは、データベース操作言語とプログラム言語を融合させ、インピーダンス・ミスマッチ問題を解決している。しかし、多くのOODBが言語として操作的プログラミングの仕様を採用しているため、質問式作成の容易さや、質問式の最適化などに理論的な根拠を欠いている。これに対して、直観性に優れ、標準化や遅延評価などの効果的な手法を持つ関数型言語を、OODBの検索言語として採用することは、一つの解決策である。

我々が考察した関数型言語は、FPとそれを基礎にしたデータベース(関数型データベース)言語FQLである。FPは、(1)データ構造としてリストを導入し、(2)リストを意味のあるまとまりとして扱えるような関数や関数の結合方式を提供している。また、FQLはデータベース検索の基本となる5つの結合方式、すなわち合成、組演算、拡張(全適用)、制限と生成を採用している。我々は、構成子としてリスト以外にタプルを導入し、関数や結合方式を定義することで、OODBモデルの複合オブジェクトを明瞭に扱えるようにした。

我々のOODBモデルは$O_2$モデルに大きな影響を受けている。$O_2$モデルは、まずセットとタプルの構成子に基づき、値空間を定義している。オブジェクトは、その識別子と値の対で表され、それが所属するクラス上で定義されたメソッドによってカプセル化される。また、継承はクラス解釈の包含関係に対応するタプルの順序関係で定義される。$O_2$モデルと我々のモデルの異なる点は、値空間がセットではなく、リストに基づくことと、1引数のメソッドに限定されていることである。

本稿では、関数については概念的な内容にとどめ、データ領域やクラスなど、モデルに関連する要素を詳しく説明する。

## 2 データ領域

我々の検索言語は、この章で定義するデータ領域において閉じている。データ領域$DT$は、オブジェクト領域$O$と値空間$V$に分けられる。まず、最初に次の3つが与えられる。

1. 原定義域 *int*、*real*、*bool* および *str*: それぞれ自然数、実数、真偽値、文字列の領域を表す。すべての原定義域の和集合を$D$で表す。
2. 可算集合$A$: タプルにおける属性を表す。
3. 全順序可算集合$ID$: オブジェクト識別子を表す。

集合$D$,$A$,$ID$は、お互いに重なりを持たない。

**オブジェクト集合** オブジェクトは、$ID$の要素である識別子$o$と以下で定義される値$v \in V$の組$(o,v)$で表す。オブジェクト集合$O$の各要素は、それぞれ異なる識別子を持つ。データベース上でオブジェクトに、値の定義がされていないとき、それはデフォルト値$\perp$を持つ。

**値空間** 値空間$V$は$D$,$A$と$ID$,およびタプルとリストで構成される。$O$がすでに定義されていると、$V$は次のように再帰的に定義される。

1. 特殊記号$\perp$は値である。
2. $D$の要素は値である。これを原値と呼ぶ。
3. $A$の相異なる要素$a_i$と、$O$もしくは$V$の要素$c_i$ $(n>0, i=1,\ldots,n)$において、$[a_1:c_1,\ldots,a_n:c_n]$は値である。これをタプル値と呼ぶ。
4. 空リスト<>は値である。また、$ID$または$V$の要素$c_i$ $(n>0, i=1,\ldots,n)$において、$<c_1,\ldots,c_n>$は値である。これをリスト値という。

$O$と$V$は定義より互いに重なりをもたない。$DT$は$O$と$V$の和集合として定義される。

検索やメソッドなど、関数を定義するにあたって、属性を省略し、単純なタプルを用いた方が便利なことが多い。このタプルとして、順タプルを簡単に定義する。

**順タプル** タプル値のうち$n$個$(n>0)$の要素を持ち、各属性名が$\sharp 1,\ldots,\sharp n$であるものを順タプルと呼ぶ。通常、順タプルは属性が上の順番になるよう並べて、属性名を省略する。

## 3 クラス

クラスは、最初から存在する基礎クラスと、ユーザが定義するユーザ・クラスと、すでに存在するクラスから派生する構成クラスに分類される。ユーザ・クラス以外のクラスは、すべて大文字で記述される。

1. 基礎クラス
   (a) $BOT$と$ANY$:
   $BOT$の外延は$\{\perp\}$、$ANY$はすべての$DT$
   (b) $V$の各要素$x$に対する$'x$:
   外延は$\{\perp, x\}$
   (c) $INT$,$REAL$,$BOOL$,$STR$:
   $INT$の外延は$int \cup \{\perp\}$など。また、$ATOM$はすべての$D$に対応する。
2. ユーザ・クラスはユーザが定義した名前と外延である。
3. 構成クラス
   (a) $a_1,\ldots,a_n \in A, T_1,\ldots,T_n \in Class$に対して$[a_1:T_1,\ldots,a_n:T_n]$:
   外延はタプル値で、少なくとも属性$a_1,\ldots,a_n$を含み、それに対応する要素$c_1,\ldots,c_n$が$c_i \in T_i$である値の集合。順タプルのクラスは$[T_1,\ldots,T_n]$、または$[T]^n$

*The design of a query language on the basis of functional programming paradigm
†Fumiyuki Tanemo, Toyohide Watanabe, and Noboru Sugie
‡Nagoya University

($\sharp 1, \ldots, \sharp n$ に対応する要素がすべてクラス $T$ に属す集合) と書ける。また、タプル $TUPLE$、順タプル $STUPLE$、少なくとも $n$ 個の要素をもつ順タプル $STUPLE^n$ が定義される。

(b) $T \in Class$ に対して $LIST(T)$:
外延は $T$ を要素としてもつリストの集合。また、$LIST$ はすべてのリスト値を示すクラスである。

(c) $T_1, T_2 \in Class$ に対して $T_1 + +T_2$:
$T_1, T_2$ の和集合を示す。

クラスは、その外延の包含関係に応じて、順序が定められる。例えば、$INT \prec ATOM$ や $LIST(INT) \prec LIST$ などは自明な順序関係である。この順序関係はタプル型構成クラスにおいて重要になる。例えば、$[a_1 : T_1]$ と $[a_1 : T_1, a_2 : T_2]$ には後者に属す値はすべて前者のクラスにも属すから、$[a_1 : T_1] \succ [a_1 : T_1, a_2 : T_2]$ となる。

## 4 ユーザ・クラス

クラスのうち、基礎クラスと構成クラスはシステムで定める外延と、クラス階層における位置 (順序関係で決まる) およびメソッド (関数) をもつ。データベースの構築者は、ユーザ・クラスを既存のクラス階層に付け加えて、新しい機能を得ることができる。ユーザ・クラスは名前と構造と、いくつかのメソッドをもつ。構造は、そのクラスに属すオブジェクトを制限し、他のクラスで定義される。ユーザ・クラス $S$ を構造 $T \in Class$ で定義する ($S \Leftarrow T$ と表す) と、$S$ に属すオブジェクト $s$ は $(o, v), o \in ID, v \in ext(T)$ である ($ext$ はクラスの外延を表す)。$S$ は、$T$ によって基礎ユーザ・クラス、構成ユーザ・クラス、多重ユーザ・クラス (ユーザ・クラスのユーザ・クラス) に分類できる。また、$T$ におけるクラス順序関係を引き継ぐこともできるが、$T$ の順序関係を乱すような階層を、$S$ 上で定義できない。例えば、二つのユーザ・クラス $S_1 \Leftarrow [a_1 : T_1]$ と $S_2 \Leftarrow [a_1 : T_1, a_2 : T_2]$ で、$S_1 \succ S_2$ なる順序を定義してもよいが、$S_1 \prec S_2$ は定義できない。

**外延リスト** ユーザ・クラスにおいて、ユーザが外延を決定すると、システムは自動的に、それに対応する外延リストを作成する。外延リストは、クラスに属すオブジェクトを、その識別子の順番に従って[1]、並べたリスト値である。これを求める演算は、基礎関数として定義される。

## 5 関数

関数型言語では、すべての演算が関数を通じて行なわれる。我々の関数は定義域 (クラス) をもち、定義域外の要素が与えられると、文法上の誤りになるが、言語仕様が構造化されているため、誤りの検出は容易である。関数には基礎関数と定義関数、そしてユーザ・クラスで定義されるメソッドがある。

### 5.1 基礎関数

基礎関数は基礎クラス、構成クラスに対するメソッドに相当し、その定義域と機能がすでに定められている関数である。重要な基礎関数としては、自己関数 $id$(定義域 $DT$), オブジェクトの値を取る $self$(定義域 $O$), タプルからその属性値を取り出す選択関数、いくつかの算術、論理演算や、$tl$ や $hd$ などリスト処理の関数がある。また、クラスの外延リストを求める関数も、基礎関数として定義される。

### 5.2 関数結合

基礎関数を組み合わせて、より複雑な定義関数を構成するのが、関数結合である。関数結合は、引数として関数 (または値) をとり、新しい関数を作り出す。作り出された関数の定義域は、関数結合の定義と引数関数の定義域にしたがって定められる。関数結合には、合成 $(F \circ G)$、組演算 $([F, G, \ldots])$、拡張 $(*F)$、制限 $(|P)$ という FQL の形式のほかに、定数化 $(\%a)$、リスト関数化 $(/F)$、条件 $(P \rightarrow G; H)$ がある。

### 5.3 関数定義

基礎関数と関数結合を用いて、新しい関数を定義できる。定義宣言の形式は、(定義関数名) ≡ (定義文) である。定義関数の定義域、値域は、その定義文で自動的に決定する。定義文の中に、その定義関数自身があってもよい。次は定義域 $INT$ で定義された階乗の例である。

$$fact \equiv [self, \%0] \circ lt \rightarrow \%\bot; ([self, \%0] \circ eq \rightarrow \%1; [self, pred \circ fact] \times)$$

定義関数 $fact$ は二重の関数結合 (条件) で組み立てられている。機能を簡単に説明すると、まず、関数引数と 0 を比較し ($[self, \%0] \circ lt$)、小さければ定数関数 $\%\bot$ を ($\bot$ を返す)、そうでなければ第 2 の条件関数を適用する。同様に、第 2 の条件関数は引数が 0 と等しければ 1 を返し、そうでなければ $[self, pred \circ fact] \times$ を適用する。最後の関数は、引数自身と、それから 1 を引き $fact$ を適用した結果の積を返す。

クラスにおけるメソッド定義は、普通の関数定義とほぼ同様であるが、その定義域に条件が必要になる。定義域は

1. メソッドを定義するクラスの順序関係で下位のクラス
2. 属性 $obj$ を持ち、その要素が 1 のクラスであるタプル型の構成クラス

に限られる。

## 6 概略

本稿では、我々のデータモデルと検索言語仕様について述べた。言語は、関数型プログラミング・パラダイムに基づいており、データモデルは言語とデータ領域が一貫性をもつよう定義されている。検索言語の関数は、基礎関数と、関数結合によって構築される定義関数が存在する。

[1] $ID$ は全順序集合として定義されている。