データベースシステム 92－8
（1993. 3. 22）

JDMF-92 における クラスメソッド及びクラス変数の取扱いについて

野口 宏, 穂鷹良介
茨城大学, 筑波大学
noguchi@cis.ibaraki.ac.jp, hotaka@shoko.sk.tsukuba.ac.jp

Abstract

現在、盛んに研究されているオブジェクト指向データベースにおけるデータモデル機能はオブジェクト指向言語を基にしているものが殆んどでり、クラスメソッド, クラス変数という概念が存在している。しかし、従来のデータモデル機能ではその様な考えは特に必要としていなく、これらの概念が必要不可欠かどうかが問題となる。一方モデリング概念の最少化という目的を徹底したところクラスメソッド, クラス変数の概念は必ずしも必要ではないという結論に至った。

このためにオブジェクトに対してメタオブジェクト概念を導入し、それを再びインスタンスと見る事により、インスタンスメソッド, インスタンス変数だけの概念でデータモデル機能を構成する事が可能となる。

この考え方を徹底すると、オブジェクトに対するオブジェクトクラスを考える操作が無制限に必要となり、データモデル機能自身基本的な見直しを行う必要に迫られた。

本報告では、その機能の背後にあるメタデータの基本的な構造を一部集合論の助けを借りて明らかにする。

キーワード:JDMF, データモデル, オブジェクト指向, 正則性, クラス変数, クラスメソッド

On the implementation of class methods and class variables in JDMF-92

Hiroshi NOGUCHI, Ryosuke HOTAKA
Ibaraki University, University of Tsukuba

Abstract

Almost all data modelling facilities for object oriented data models are based on object oriented languages where concepts of class methods and class variables are used.

But traditional data modelling facilities did not need these concepts. And there are problems if these concepts are really necessary. On the other hand, by minimizing the number of necessary modelling concepts, we came to the conclusion that the class method and the class variable are unnecessary concepts.

In order to eliminate the class method and the class variable, we have to regard meta data as instances of classes one meta level higher up, which forces us to use infinitely many meta levels.

This paper clarifies the basic structure of an infinitely recursive meta data structure using set theoretic representation.

Keyword:JDMF, data modelling facility, object-oriented, regularity, class method, class variable

1 はじめに

現在、盛んに研究されているオブジェクト指向データベースにおけるデータモデル機能はオブジェクト指向言語を基にしているものが殆んどであると思われる。そこで、Smalltalk/V[4]等のオブジェクト指向言語を見てみるとクラスメソッド, クラス変数という概念が存在している。

しかし、従来のデータモデル機能ではその様な考えは特に必要としていなかった。そこで、標準データモデル機能の設定と云う立場に立ち、これらのクラスメソッド, クラス変数という概念はモデル構成上必要不可欠かということが問題になる。

一方、標準データモデル機能設定の為に、モデリング概念の最少化を行なわねばならないという考えがある。結論としていえる事は、データモデル機能のモデリング概念の数を最少化するという事を徹底すると、クラスメソッド, クラス変数の概念は必ずしも必要ではないという結論に至った。

この結論に至るためにはオブジェクトに対してメタオブジェクト概念を導入し、メタオブジェクトもオブジェクト同様インスタンスと見なした。こうする事により、インスタンスメソッド, インスタンス変数だけの概念でデータモデル機能を構成する事が可能となる。

この考えを徹底するとオブジェクトクラスに対するオブジェクトクラスクラス, オブジェクトクラスクラスに対するオブジェクトクラスクラスクラス…と考える操作が無制限に必要となり、データモデル機能自身基本的な見直しを行う必要に迫られた。

本論文では、この様な考えでJDMF-92データモデル機能(以下JDMFと略)[1]を見たときに、その機能の背後にあるメタデータの基本的な構造を一部集合論の助けを借りて明らかにする。

2 JDMFと集合論

本論文ではオブジェクト指向的に記述されたJDMFを集合論的に論じているため、先ず、その記号系に関して簡単な定義を行う。

JDMFに限らず一般的なオブジェクト指向におけるインスタンス, オブジェクトクラスは、それぞれ集合論でいうところの要素, 集合と類似している。これにより、インスタンス, オブジェクトクラスを次の様に集合論的に記述する事ができる。

> インスタンスxが、オブジェクトクラスyに属している。

$$x \in y$$

ここで、インスタンスに対してオブジェクトクラスを考えるというのは、データに対してメタデータを考えている事と同一のものであると考えられる。

また、データモデル機能及びオブジェクト指向においての汎化, 特化の関係は、集合が含む含まれるという関係になっている。つまり、集合論の記号を用いて表すと、次の様に記述する事ができる。

> 実体型xを汎化したものは実体型yである。(実体型yを特化したものは実体型xである。)

$$x \subset y$$

しかし、集合論と類似しているのはある一時点で見た場合であり、時間を経過して考えるときオブジェクトは集合とは異なるものである。オブジェクトクラスはその中のインスタンスが消滅, 生成してもオブジェクトクラス自身は変わらないし、インスタンスもそのインスタンスメソッド, インスタンス変数が変わったとしてもインスタンス自身が変わる

訳ではない。この様な意味で集合論とは異なるが、ある瞬間で考えた時は集合論に非常に良く似ている事がわかる。

ここで、もう少し詳しく集合論的にインスタンス, オブジェクトクラスを考えてみる。先ず、ある対象世界のある時点の状態を表現するデータモデルでのオブジェクト全体を集合 *Object* とし、オブジェクトクラス全体を *Object** とする。すると、*Object* はオブジェクトクラスと考えられるので、

$$Object \in Object^* \tag{1}$$

となる。しかし、*Object** はオブジェクト全体の集合 *Object* の部分集合とも考えられる。すると、

$$Object^* \subset Object \tag{2}$$

ともいう事ができる。この式(1), 式(2)から

$$Object \in Object \tag{3}$$

となり、*Object* という集合は自分自身を要素として含むことになる。

しかし、通常の集合論では、**正則性 (Axiom of Regularity)**[5][6]

$$\forall a(a \neq \emptyset \rightarrow \exists x \in a(x \cap a = \emptyset))$$

を満たす事を前提としている。これは、式(3)とは矛盾することは明らかである。つまり、JDMF-92 を一時点で集合論的に考える時は、正則性を満たさない集合論を考えねばならない。

この様な集合論を考える事により、(時間を一点に限定する事によって) データモデル機能を形式的に集合論として取り扱う事が可能となる。

3 最小データモデル機能

必要最少のモデル化概念のみを利用する事によりモデルの基本的性格を明らかにする為に、オブジェクトとメタオブジェクトの関係を説明する最小モデルを導入する。そして、それがどの様な自己記述の行なわれているデータモデル機能にも共通の構造として存在する事を明らかにする事が目的である。

ここでは前提として、従来から言われているものとは別の非常に単純なオブジェクトという用語を定義する。モデルの実用に当たってもっと別の概念が必要となった時に必要なものを付け加えていけば良いと考える。

先ずオブジェクトとは、対象世界をデータモデル機能で表現しようとした時、対象世界の個々の対象をデータモデル側で代表するもののことであり、対象世界はこのオブジェクトとオブジェクト間の関係から成っているとする。そして、対象世界を表現するデータモデル機能でのオブジェクト全体を *Object* で表すものとする。前章と同様に、あるオブジェクト x に対して、そのメタオブジェクト (オブジェクトクラス) を考える時は x^* の様に* を付けて表す事とする。また、前章と同様に、理解を助ける為に集合論の記号を用いる。

基本的な前提として、すべてのオブジェクトはそれがインスタンスとして属するクラスを持ち、更にあるオブジェクトを生成する時は必ずそのクラス (メタオブジェクト) を考えねばならず、そのオブジェクトはそのクラスのインスタンスとして生成されるものとする。また、簡略化のために本論文では *Object* の中の任意のクラス x, y を考えた時に、$x \subset y, y \subset x, x \cap y = \emptyset$ のいずれかであることを前提とする。

最小データモデル機能は上記の条件を満たし、更に以下の条件を満たすものとする。

*Object** はそれ自身が何かのクラスのインスタンスでなくてはならない。即ちオブジェクトクラス *Object*** が存在して *Object** は *Object*** のインスタンスとなる。同様の考慮をすべてのオブジェクトクラスに対して成立するように考えると、オブジェクトクラスの

無限系列 $Object, Object^*, Obect^{**}, \cdots$が存在して、次の様になる。第2章 (1) と同様の理由によって

$$Object^{*n} \text{は} Object^{*(n+1)} \text{のインスタンス} \quad (4)$$

である。($n = 0,1,2,\cdots$;但し $Object^{*0} = Object$ とする。) 即ち

$$Object^{*n} \in Object^{*(n+1)}$$

が成立する。

前章 (3) と同様の理由により

$$Object^{*(n+1)} \text{は} Object^{*n} \text{のサブクラス} \quad (5)$$

である。即ち

$$Object^{*(n+1)} \subset Object^{*n}$$

が成立する。

オブジェクトクラスからインスタンスを生成する *new*を考えてみる。例えばオブジェクトクラス *Employee* に対してそのインスタンス *Ohta* を生成するには Smalltalk では

$$Ohta := Employee\ new$$

である。本論文では $Object^{*(n+1)}(n \geq 0)$ の定義から $Object^{*(n+1)}$のインスタンスに *new*を送って生成されるインスタンス x は $Object^{*n}$ の要素であると考えられる。即ち
$y \in Object^{*(n+1)}$の時、

$$x := y\ new$$

としたならば

$$x \in Object^{*n}$$

*new*は polymorphic でレベルにより規則的ではあるが異なった振る舞いをする。そこで、$Object^{*n}$の要素へ送る *new*を、区別する必要があるときは特に new^{*n} と書くことにする。new^{*n} は n の如何に拘らず、基本的には類似な振る舞いをするが、これ以外にもデータモデル機能の特殊化の指定によって別の制約が付くこともある。

また、本論文では簡単化のために次の制約をおく。

- 上述のように生成された x は
  $$y \cap Object^{*(n+1)} = \emptyset$$
  とする。
- $Object^{*n} \notin Object^{*(n+2)}(n = 0,1,2,\cdots)$

Object のインスタンスに対してメソッド *delete* が定義されている: $x \in Object$ のとき、x にメソッド *delete* を送ると x が応用データモデルから消し去られる。ただし x として $Object*n(n = 0,1,2,\cdots)$ を指定することはできない。メソッド *delete* に対してこれ以外にもデータモデル機能の特殊化の指定によって別の制約がつくことがある。最小データモデル機能は次の性質を満たす *Object* とする。

$Object * n \supset Object^{*(n+1)}$
$(n = 0,1,2,\cdots$;但し $Object^{*0} = Object)$
$Object^{*n} \in Object^{*(n+1)}(n = 0,1,2,\cdots)$
$Object^{*n} \in Object^{*(n+2)}(n = 0,1,2,\cdots)$
は成立しない.

4 特殊化データモデル機能

関係データモデル等より多くのモデル概念を持つデータモデル機能は最小モデルの特殊化として考える事が出来るという事を例で示す。新しく定義されたデータモデル機能は最小データモデル機能の特殊化としてみる事が説明でき、世の中の数多く存在するデータモデル機能同士の関係を最小データモデル機能とその特殊化という概念でうまく位置付けできる可能性を示す。

尚、データモデル機能といわれる他を記述する目的の機能と、応用データモデルといわれる対象世界の記述そのものも等しく最小データモデル機能の特殊化として考える事が出来ることが判明した。

従来、データモデルという言葉で、データモデル機能あるいは応用データモデルのいずれかを示すかが、必ずしもクリアに区別されていなかったがどちらも最小モデルの特殊化という見方で統一できるという事が分かり、我々が何気なく使っているデータモデルという概念とうまく合致している事がいえた。

[例] 最小データモデル機能から関係データモデルに類似したデータモデル機能を定義する。

ここで特殊化するデータモデル機能は非常に単純なデータモデル機能であり、オブジェクトクラスの他に属性及び汎化関係のみを表現できるものとし、メソッドに関してはオブジェクトクラスは*new*を持ってはいるがその他のメソッドは持たないものとした。そして、最小データモデル機能のオブジェクトと関係データモデルのエンティティが同一のものとしている。

まず、属性をつける為に、オブジェクトクラス *Object**の中にインスタンス *Attribute* を生成する。次に、汎化関係を示す為に、やはりオブジェクトクラス *Object**の中にインスタンス *SuperSubRelationship* を生成する。

Attribute := *Object* new*
SuperSubRelationship := *Object* new*

ここで、*Attribute*,*SuperSubRelationship* どちらとも *Object**のインスタンスであるから、それぞれクラスと見なすことができる。つまり、それぞれには *Object**で定義されたインスタンスメソッドである*new*を送り、それぞれのインスタンスを生成することができる。そこで、*Attribute* に対して*new*を送り、*Object**, *Attribute*, *SuperSubRelationship* の属性を生成する。

&name := *Attribute new*
&class := *Attribute new*
&primaryRelation := *Attribute new*
&domain := *Attribute new*
&super := *Attribute new*
&sub := *Attribute new*
&class := *Attribute new*

ここでは、*Object**, *Attribute*, *SuperSubRelations* に、それぞれ表2, 3, 4に示す属性の生成を行なう訳だが、

$$Attribute \in Object*$$

$$Attribute \subset Object$$

$$SuperSubRelationship \in Object*$$

$$SuperSubRelationship \subset Object$$

という前提があるので、表2, 3, 4の属性&name, &class は同一のものを示しているとする。

これで、データモデル機能を生成する為の部品ができたが、まだこれらが表 2, 3, 4に示したデータモデル機能を構成しているとは言えない。そこで、その構成を行なう。但し、ここでは表3の一部に関しての構成を記述した。

&name.&name := *&name*
&name.&primaryObjectClass := *$$Object*

&name.&domain := *$$char*
&name.&class := *$$Attribute*

表 1: 応用データモデル例

Part	
P#	PName
101	Bolt
102	Nut

Employee	
E#	EName
401	Ohta
402	Suzuki

この様にして最小データモデル機能を特殊化することにより、データモデル機能のデータ側面の記述枠組を構成することができる。

5 応用データモデル具体例

前章で定義されたデータモデル機能を用いて、表1の応用データモデルを定義を行なう。

$Part := Object^{*}\ new$
$Emplpyee := Object^{*}\ new$
$P\# := Attribute\ new$
$PName := Attribute\ new$
$E\# := Attribute\ new$
$EName := Attribute\ new$

これで部品ができたので、前章で構成した応用データモデルへ登録を行なう。前章同様、次の様にして属性, 汎化関係に関してそれぞれ表3, 4へ登録する。

$P\#.\&name := P\#$
$P\#.\&primaryObjectClass := \$\$Part$
$P\#.\&domain := \$\$integer$
$P\#.\&class := \$\$Attribute$
$\vdots$

この様にして応用データベースをデータモデル機能上に構成することができる。出来上がった表を表5, 6, 7に示す。

6 まとめ

モデリング概念の最少化という事を徹底することにより、クラスメソッド及びクラス変数の概念が必ずしも必要でないことを示した。そして、これらの概念を取り除くことにより構成される最小データモデル機能を導入した。また、データモデル機能と呼ばれているものは、ここで示した最小データモデル機能の特殊化として説明できることを示し、そのデータモデル機能を使って記述される応用データモデルも特殊化されたデータモデル機能の特殊化として得られることを示した。

しかし、一般的にデータモデルがこの最小データモデルの特殊化として取り扱う事ができるかどうかの理論的な裏付けが出来ていない。また、その特殊化に関しても、充分な議論が必要となるであろう。

集合論の公理系から正則性を外すという事により最小データモデル機能が導入できた訳であるが、この2つの事柄に意味的に違いがあると思われる。この違いを埋める事も今後の課題となるであろう。

謝辞

本研究を行なうにあたり、活発な議論を頂いたつくばデータベースシステムセミナの皆様、そして集合論に関して有益な御意見を頂いた茨城大学鈴木信行講師に感謝の意を表します。

参考文献

[1] 日本規格協会: JDMF-92, 1993

[2] 穂鷹良介: データ主導情報モデリング設計支援システム IMDSS について情報処理学会データベース研究会,92-8, 1993.

[3] Björn,Michael, 穂鷹良介: Virtual class-

generation in visual languages, 情報処理学会データベース研究会,92-10, 1993.

[4] Smalltalk/V Tutorial and Programming Handbook, Digitalk inc., 1989

[5] 田中尚夫: 公理的集合論, 培風館, 1982.

[6] Jech,Thomas: Set Theory, Academic Press, 1978

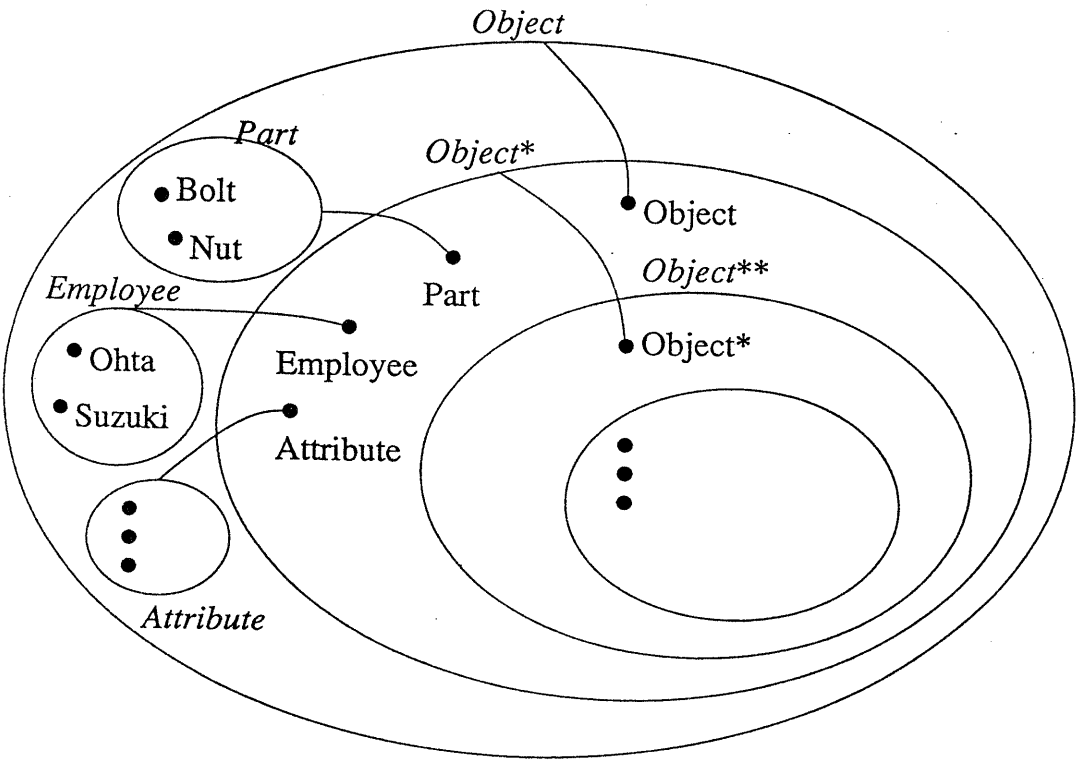

図 1: 無限の階層

表 2: 特殊化の例 (a)

*Object**	
&name	&class
Attribute	$$*Object**
SuperSubRelationship	$$*Object**

表 3: 特殊化の例 (b)

Attribute			
&name	&primaryObjectClass	&domain	&class
&name	$$*Object*	$$*char*	$$*Attribute*
&class	$$*Object*	$$*Object**	$$*Attribute*
&domain	$$*Attribute*	$$*Object**	$$*Attribute*
&primaryObjectClass	$$*Attribute*	$$*Object**	$$*Attribute*
&super	$$*SuperSubRelationship*	$$*Object**	$$*Attribute*
&sub	$$*SuperSubRelationship*	$$*Object**	$$*Attribute*
⋮	⋮	⋮	⋮

表 4: 特殊化の例 (c)

SuperSubRelationship			
&name	&super	&sub	&class
$1	$$*Object*	$$*Attribute*	$$*SuperSubRelationship*
$2	$$*Object*	$$*SuperSubRelationship*	$$*SuperSubRelationship*
$3	$$*Object*	$$*Object**	
$4	$$*Object**	$$*Object***	
$5	$$*Object***	$$*Object****	
⋮	⋮	⋮	

表 5: 応用データモデル登録の例 (a)

*Object**	
&name	&class
Part	$$*Object**
Employee	$$*Object**

表 6: 応用データモデル登録の例 (b)

Attribute			
&name	&primaryObjectClass	&domain	&class
P#	$$*Part*	$$*integer*	$$*Attribute*
PName	$$*Part*	$$*char*	$$*Attribute*
E#	$$*Employee*	$$*integer*	$$*Attribute*
EName	$$*Employee*	$$*char*	$$*Attribute*

表 7: 応用データモデル登録の例 (c)

SuperSubRelationship			
&name	&super	&sub	&class
$5	$$*Object*	$$*Part*	$$*SuperSubRelationship*
$6	$$*Object*	$$*Employee*	$$*SuperSubRelationship*