データベース・システム 65－4
（１９８８．５．１９）

実体－関連モデルのための データ定義言語と従属性の扱いについて

宇田川 佳久 ・ 石川 洋 ・ 市川 照久 ・ 辻 秀一 ・ 西川 正文

（三菱電機㈱ 情報電子研究所）

実体－関連モデルを始めとする従来のデタモデルでは、ＣＡＤなどの新しいデータベース応用で要請される広範なデータの意味をモデル化することがむずかしい。この課題に対する解決案の一つとしてＯＳＭＡＮと称するオブジェクト指向実体－関連モデルを開発している。ＯＳＭＡＮの主な特徴は次のとおり。
(1) オブジェクトを階層的に分類することができる
(2) 多様な関連付けパターンを提供している
(3) 幾何情報を記憶するためのオブジェクトを提供している
(4) 複合オブジェクトを構成することができる
本文では、建築物のレイアウトを例に用いながらＯＳＭＡＮデータモデルを紹介する。また、ＯＳＭＡＮにおけるデータ従属性の扱いや、Smalltalk80 のオブジェクト指向との違いについて論ずる。

Data Definition Language of an Enhanced Entity-Relationship Model and Data Dependency

Yosihisa Udagawa , Hiroshi Ishikawa , Teruhisa Ichikawa
Hidekazu Tsuji and Masafumi Nishikawa

Mitsubishi Electric Corporation
5-1-1 Ofuna, Kamakura City, Kanagawa, 247 Japan

It is widely accepted that conventional data models including the Entity-Relationship model have difficulties to support a wide variety of semantics in new applications. To handle complex semantics of data, we are developing an object-oriented E-R database called OSMAN. Main features of OSMAN are as follows :
(1) hierarchical classification of object types,
(2) a wide variety of relationship patterns,
(3) object types to support geometric data,
(4) recursive construction of complex object types.
In this paper, we show by expmples how architectural layouts are modeled in this framework, give some discussions on data dependency problems and difference of object-oriented concepts between Smalltalk80 and OSMAN.

1．　はじめに

データベースの目的に、アプリケーションプログラムからのデータ独立とデータ共同利用を達成することがある。アプリケーションを完成させるには、多くのユーザやシステム開発者が関与する。従って、データ独立／共同利用を達成するためには対象世界をわかり易くモデル化することが必須の条件となる。実体一関連モデルは、対象世界のあらゆるものを、実体（Entity）と実体同士の関連（Relationship）によって表現するものであり、単純であること、わかり易いこと、表現が自然であるという特徴を持っている［1］。

実体一関連モデルによってモデル化された対象世界は論理モデル、例えばリレーショナル型、ネットワーク型あるいは階層型データモデルに変換され、コンピュータに実装されることが多かった。

このように、実体一関連モデルは、もともと、現実世界の概念モデリングのために考え出されたものであるが、このモデルの普及とともに、論理モデルとしての可能性も研究されている。

例えば、Shoshani［3］のCABLE(ChAin-Based LanguagE),Zaniolo［6］のGEM(General Entity Manipulator)といったデータ操作言語の研究開発がある。これらの言語は、事務分野のアプリケーションを前提に行われてきたが、設計分野への適用も有望であると考えられる。すなわち、設計オブジェクトも認知できる実体と関連によって表現される点では事務分野におけるオブジェクトと共通しており、実体または関連単位のデータ操作が自然であるからである［8,9,10,11］。

一方、従来の実体一関連モデルをそのまま設計分野に適用することが難かしいこともわかっている［4,5,7］。つまり、設計分野のアプリケーションでは、

(1) 意味のあるデータ操作単位が複数の実体または関連(Complex Object)にまたがることが普通である。
(2) 設計オブジェクトは階層的に詳細化されることが多い。
(3) 図形・画像などの多種類のデータ型をサポートする必要がある（マルチメディアデータ）。

Complex Object、階層的詳細化やマルチメディアデータを実体一関連モデルの範囲でモデル化することは難かしく、何らかの拡張が必要である。

実体一関連モデルを建築物のモデル化に適用した結果、次のような拡張が必要であると判断した。

(1) 実体を共通する属性に基づいてタイプ分けし、クラスとして定義できる機能
(2) 実体間の関連付けとして、集合に基づく関連付けに加えて、実体の出現順序をも考慮した関連付けを行う機能
(3) 図形・画像情報を保持し、幾何モデラと情報を受け渡しするための機能
(4) 実体一関連の構造を再帰的に適用し、複合オブジェクトを作る機能

本文は、上記の拡張を行った実体一関連モデルＯＳＭＡＮ(Object-oriented Semantic Model for hAndling Nonmonotonic data)についてデータモデルの観点から論じたものである。第２章では、ＯＳＭＡＮデータモデルの概要を述べ、マンション・レイアウトのモデル化の例を示しながらＯＳＭＡＮの特徴を述べる。第３章では、ＯＳＭＡＮのデータ定義言語を示し、このデータモデルのデータ構造を明らかにする。第４章では、データの従属性の扱いについて述べる。第５章では、ＯＳＭＡＮのオブジェクト指向の意味するところをSmalltalk80 と対比しながら論ずる。

2．ＯＳＭＡＮとマンション・レイアウトのモデル化

2．1　ＯＳＭＡＮの概要

ＯＳＭＡＮデータモデルは、Informationと呼ばれる可変長の記憶単位に基づいて組み立てられている。Informationは図１の様な構造をしている。制御部にはＤＢＭＳが必要とする情報が書き込まれている。また、データ部には各種のデータや、他のInformationへのポインタが格納される。データ部の大きさは任意であり、実行中に変更することもできる。個々のInformationにはシステムにより一意的な識別コード（サロゲート）が割り当てられる。これは制御部に格納される。このInformation は、以下の４種類に分類されている。

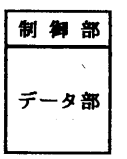


図１．　Information の構造

(1) カーネル(Kernel)と呼ばれるInformationは、属性の値、又は、他のInformation へのポインタを保持している。属性値のみを記憶しているカーネルは、実体一関連モデルにおける実体に対応し、ポインタのみを記憶しているカーネルは、関連に対応する。一般に、１つのカーネルには属性値とポインタが混在していることが多い。ＯＳＭＡＮデータモデルでは、実体と関連との厳密な区別は考えず、両者を合わせた概念である“カーネル”という単位でモデル化する。

(2) クラスカーネル(Class-K) と呼ばれるInformationは、Informationの分類を行うために使われるものである。つまり、いくつかのカーネルをまとめて一つの概念として扱うときに用いるカーネルてある。クラス・カーネルは階層構造を作ることができる。一つのクラス・カーネルに複数の上位クラス・カーネルを定義することができる。従って、クラス・カーネルの階層構造はサイクルを含まない向き付きグラフ(Directed Acyclic Graph)として表現することができる。

(3) 幾何カーネル(Geometric-K) と呼ばれるInformationは、図形、画像といった幾何情報を記憶するためのものである。幾何カーネルのデータ構造は、幾何モデラに依存する。言い換えれば、幾何カーネルは、幾何モデラとの情報を受け渡しするためのものである。

(4) 関連カーネル (Relating-K) と呼ばれるInformationは、上記３種類のカーネルの関連付けを行うときに使われる。関連付けのパターンは、以下の３種類の独立した次元がある。

(1) 関連付けられるカーネルの数、すなわち、１対１、１対多、多対１、多対多の対応。

(2) １対多、多対１、多対多の関連付けを行うとき、関連付けの順序を保存するか否か。

(3) 関連付けられる双方のカーネルにポインタを記憶する対等型による関連付けか、片方のカーネルにのみポインタを記憶する従属型による関連付けか。

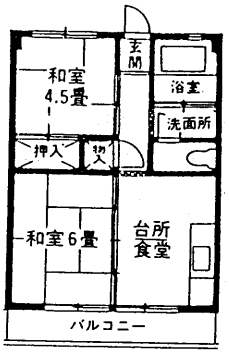


図２．　マンション・レイアウトの例

２．２　マンション・レイアウトのモデル化

本章では、マンションのレイアウトをＯＳＭＡＮによってモデル化した例を示す。モデル化の条件として、マンションのレイアウトを空間として識別し、操作できるようにすることを考える。そのためには、

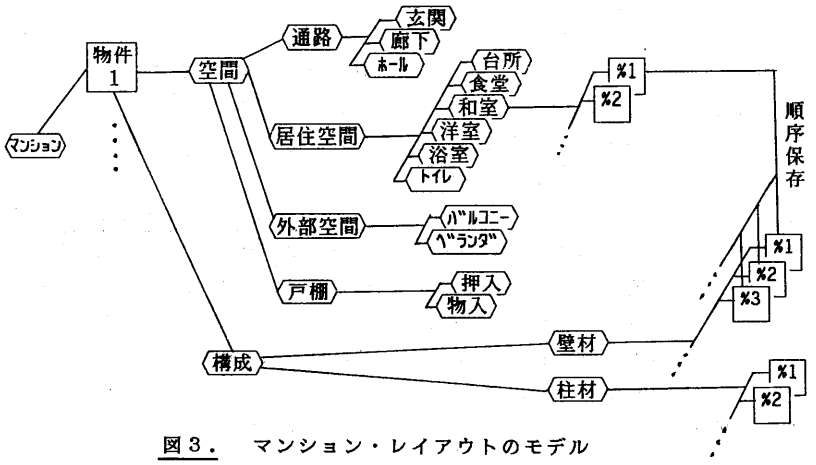


図３．　マンション・レイアウトのモデル

(1) 空間を用途別に分類し、
(2) 空間同士の隣接関係を識別できるようにする
必要がある。

図２はマンション・レイアウトの例、図３は、そのモデル化の例である。以下、図３に示したモデルについて説明する。

(1)をモデル化するには、単に空間を分類すれば良い。すなわち、“空間”を“通路”、“居住空間”、“外部空間”、“戸棚”に分類する。これらの分類はＯＳＭＡＮのクラスカーネルの階層によって表現される。“通路”のサブクラスとして“玄関”、“廊下”、“ホール”などが考えられる。また、“居住空間”のサブクラスとしては“台所”、“和室”、“洋室”などが考えられる。具体的なデータはカーネルまたは幾何カーネルによって記憶される。個々のクラスカーネルに具体的なデータが存在するか否か、また、存在するとしてどのような属性値を持っているかは個々のマンションのレイアウトごとに異なる。

空間同士の隣接関係をモデル化するためには、空間を仕切っている“板材”をモデル化しなければならない。つまり、“板材”を媒介にして空間が結合されているものとしてモデル化するわけである。この部分のモデル化については第３章、第５章でさらに詳しく論ずる。一方、“板材”は“柱材”と結合している。図３に示したマンション・レイアウトのモデルは、おおむね以上の考察に基づいて作られたものである。

３．　データ定義言語

３．１　基本機能

第２章で述べたように，ＯＳＭＡＮデータモデルは
(1) カーネル、クラスカーネル、幾何カーネルという
　３種類の記憶単位を
(2) 対等型，あるいは従属型によって
(3) １対１，１対多，多対１，多対多のいずれかで
関連付けたものである。また、１対多，多対１，多対多の関連付けでは
(4) カーネルの出現順序を保存する場合としない場合がある。以上の関連付けは、関連カーネルと呼ばれる記憶単位によって行われる。ＯＳＭＡＮのデータ定義言語は，上記のカーネルのすべての種類と関連付けパターンを網羅していなければならない。以下，データ定義言語の基本機能を概観する。

(A) クラスカーネルを定義する

クラスカーネルを定義するためには，以下のコマンドを用いる。

```
#DEFC( CKER-NAME( %1,...,%N ));
```

このコマンドは CKER-NAME　で指定された名前をもつクラスカーネルを定義するものである。％印の後の整数値は、定義されたクラスカーネルの識別番号（サロゲート）である。

(B) カーネルを定義する

カーネルを定義するためには，以下のコマンドを用いる。

```
#DEFK(  CKER-ID( %1,...,%N ));
```

このコマンドは CKER-ID　で指定されたクラスカーネルのデータを記憶するためのカーネルを定義するものである。CKER-ID は

　CKER-NAME (% ＜整数値＞)

なる構文であり、一意的に識別されるクラスカーネルの下にカーネルを定義することを指定している。

(C) 幾何カーネルを定義する

幾何カーネルを定義するためには，以下のコマンドを用いる。

```
#DEFG( GKER-NAME( %1,...,%N ));
```

このコマンドは GKER-NAME　で指定された名前をもつ幾何カーネルを定義するものである。％印の後の整数値は、定義された幾何カーネルの識別番号（サロゲート）である。

(D) 対等型あるいは従属型による関連付け

カーネル、クラスカーネル、幾何カーネルを対等型で関連付けるには、以下のコマンドを用いる。

```
#DEFEXL( XKER(#YKER) <=> YKER(#XKER)
```

また，従属型で関連付けるには，以下のコマンドを用

いる。

```
#DEFEXL( XKER(#YKER) ==) YKER
```

または，

```
#DEFEXL( XKER (== YKER(#XKER)
```

(E) 1対1，1対多，多対1，多対多の関連付け

実現値を，1対1，1対多，多対1，多対多で，対等型で、順序を保存せずに関連付けをするためには，以下のコマンドを用いる。

```
%J (=) %M
%J (=) {%M, ... ,%N}
{%J, ... ,%K} (=) %M
{%J, ... ,%K} (=) {%M, ... ,%N}
```

{} は要素の集合を表現している。一方、順序を保存して関連付けをするには、以下のように表現する。

```
%J (=) 〔%M, ... ,%N〕
〔%J, ... ,%K〕 (=) %M
〔%J, ... ,%K〕 (=) 〔%M, ... ,%N〕
```

3．2　マンション・レイアウトの記述例

この節では、第2章で述べたマンション・レイアウトのモデルをＯＳＭＡＮデータ定義言語によってどのように表現するか、また、その結果どのようなデータ構造がつくられるか、といったことについて述べる。

3.2.1　クラスカーネルの階層の定義

図3に示したマンション・レイアウトのモデルには、クラスカーネルの階層が含まれている。例えば、“空間”の下位クラスとして“通路”、“居住空間”、“外部空間、“戸棚”を定義するには、以下のようにする。

```
#DEFC( 空間(%1) );
#DEFC( 通路(%1) );
#DEFC( 居住空間(%1) );
#DEFC( 外部空間(%1) );
#DEFC( 戸棚(%1) );
#DEFEXL( 空間(#通路) (=) 通路(#空間)
         %1 (=) %1 );
#DEFEXL( 空間(#居住空間) (=) 居住空間(#空間)
         %1 (=) %1 );
#DEFEXL( 空間(#外部空間) (=) 外部空間(#空間)
         %1 (=) %1 );
#DEFEXL( 空間(#戸棚) (=) 戸棚(#空間)
         %1 (=) %1 );
```

3.2.2　クラスカーネルとカーネルの定義

例えば、クラスカーネル“和室”の下位に名前が“水仙”、“桔梗”であるカーネルを定義するには、次のコマンドを用いる。

```
#DEFC( 和室(%1) ) ;
#DEFK( 和室(%1)( %1( NAME=“水仙”)));
#DEFK( 和室(%1)( %1( NAME=“桔梗”)));
```

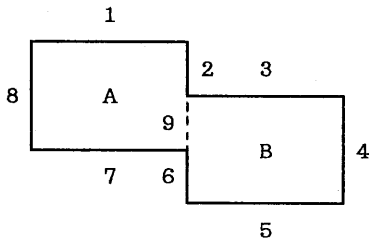


図４．　部屋の例

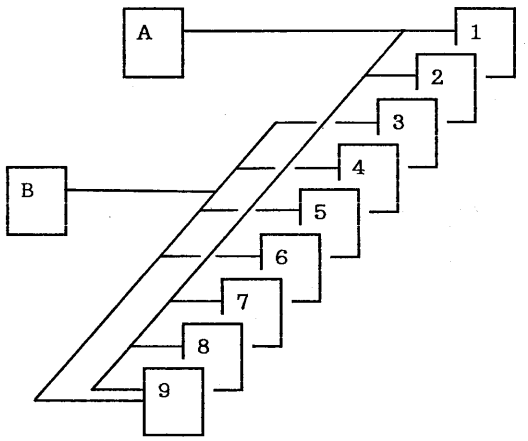


図５．　図４に対応するデータ構造

3.2.3　カーネル同士の関連の定義

現実世界をモデル化するとき、カーネル同士の関連付けが必要になることがある。マンション・レイアウトのモデル化の場合“空間”と“板材”との関連付けがその例である。“和室”Aと“洋室”Bが図4のような配置にあったとする。部屋とそれを囲む板材を時計方向回りに記憶するものとする。すなわち、“和室”Aならば“板材”1－2－9－7－8、“洋室”Bならば“板材”3－4－5－6－9となる。これを、データ定義言語によって表現すると次のようになる。

```
#DEFK(  板材(%1) (%1,%2,%3,%4,%5,%6,%7,%8,%9 ));
#DEFEXL(  和室(%1)(#板材) <=> 板材(%1)(#和室)
          %1 <=> [%1,%2,%9,%7,%8] ) ;
#DEFEXL(  洋室(%1)(#板材) <=> 板材(%1)(#洋室)
          %1 <=> [%3,%4,%5,%6,%9] ) ;
```

図5は、上記のコマンドによって作成されるデータ構造を示している。順序保存の関連付けを行っているので、部屋のトポロジカルな情報を保持することができる。

4．　従属性の扱い

4．1　従属性とデータベース設計

従属性は、対象世界の意味表現の基本となるものである。リレーショナル・データベースの出現とともに、関数従属性（Functional Dependency）をはじめとする従属性の存在が確認された。これらの従属性は、リレーショナル・データベースのスキーマ設計の論拠を与えるだけに留らず、データベース設計方法論の指針を与えるものである〔2〕。

この章では、ＯＳＭＡＮデータモデルにおける関数従属性の扱いについて論ずる。おおむね、従属型で互に関連付けられたカーネルが正規化されたリレーションに相当することを結論する。

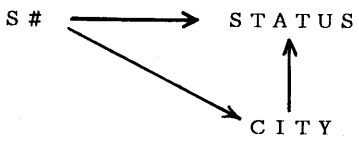
ＳＳＣ

S#	STATUS	CITY
S1	20	London
S2	10	Paris
S3	10	Paris
S4	20	London
S5	30	Athens

図６．　リレーションの例〔2〕

4．2　単純属性から成る関数従属

図６に示したリレーションには、



なる関数従属性がある。関数従属性は推移的（transitine）であることから、上記の関数従属性を、

Ｓ#　⟶　ＣＩＴＹ

と

ＣＩＴＹ　⟶　ＳＴＡＴＵＳ

という２つの関数従属性によって表現することにする〔2〕。リレーショナル・データベースでは、例えば、ＳＣ（Ｓ#，ＣＩＴＹ）とＣＳ（ＣＩＴＹ，ＳＴＡＴＵＳ）というリレーションによって、これらの関数従属性を表現することができる。

一方、ＯＳＭＡＮでは従属型の関連付けによってカーネル同士を関連付けることができる。従って、１つの属性を１つのカーネルに対応させれば、

Ｓ#　⟶　ＣＩＴＹ　⟶　ＳＴＡＴＵＳ

という関数従属性を、図７に示したデータ構造によって表現することができる。

一方、属性ＳＴＡＴＵＳは属性ＣＩＴＹにのみ関数従属しているので、ＣＩＴＹをキーとするカーネル（ＣＩＴＹ，ＳＴＡＴＵＳ）によって表現するのが適当である。この場合、図８に示したデータ構造となる。

4．3　複合属性を含む関数従属

図６に示したリレーションＳＳＣのキー属性Ｓ#をキー属性の１つとするリレーション

ＳＰ（Ｓ#，Ｐ#，ＱＴＹ）

を定義する。すると、リレーションＳＳＣとＳＰの関数従属は図９のようになる。図１０はリレーションＳＰの一例である。

リレーションＳＳＣとＳＰを定義するとき、２つのアプローチが考えられる。

(1) リレーションＳＳＣとＳＰを別々のデータ構造として定義する。ＳＳＣとＳＰを合成して、１つのリレーションとして扱うときは、Ｓ#で２つのリレーションをjoinする。

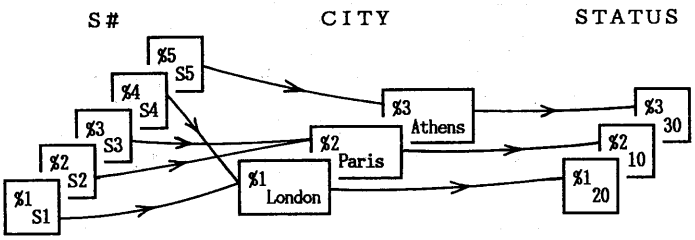


図７．　ＯＳＭＡＮによる関数従属の表現例

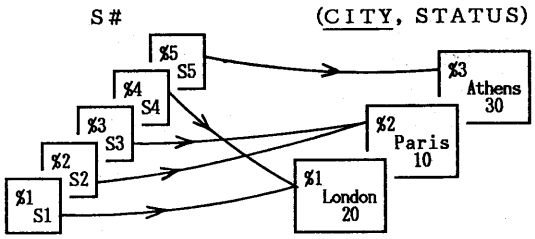


図８．　ＯＳＭＡＮによる関数従属の表現例

SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

図10．　リレーションの例〔2〕

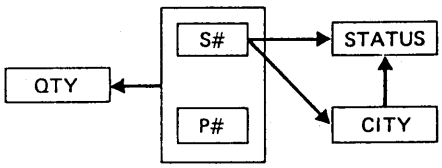


図９．　関数従属の例〔2〕

(2) 図９が示すように、Ｓ＃を記憶する場所は１箇所としつつ、関数従属を満たすように記憶する。

(1)の方法は通常のリレーショナル・データベースと同じである。(2)の方法の特徴は、リレーションＳＳＣとＳＰを合成するときにもjoin演算を必要としないことである。以下、リレーションＳＰを(2)の方法に従って定義する仕方を示す。

リレーションＳＰには

（Ｓ＃，Ｐ＃）　⟶　ＱＴＹ

なる従属性がある。ＳＰそのものはＢＣＮＦであるから、上記の３項目を有するカーネルによって表現することができるのであるが、すでに属性Ｓ＃が実体Ｓ＃として定義されている。

そこで、属性Ｐ＃とＱＴＹを記憶するカーネルＰＱを定義し、カーネルＰＱからカーネルＳ＃への双方向関連（対等型）によって結合する。言い換えると、リレーションＳＰのみならば、属性（Ｓ＃，Ｐ＃，ＱＴＹ）を有するカーネルによって記憶することができる。ところが属性Ｓ＃は他のリレーションのキーになっているので、これを独立したカーネルＳ＃として定義しなければならない。そこで、属性（Ｐ＃，ＱＴＹ）を記憶するカーネルＰＱを用意し、カーネルＳ＃と双方向関連で関連付けるわけである。図９に示した関数従属の構造は、図１１のデータ構造によって表現される。図１２は、実現値レベルでのデータ構造を表現している。

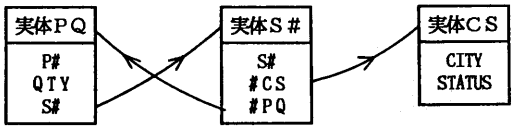


図11.　関数従属を表現するデータ構造

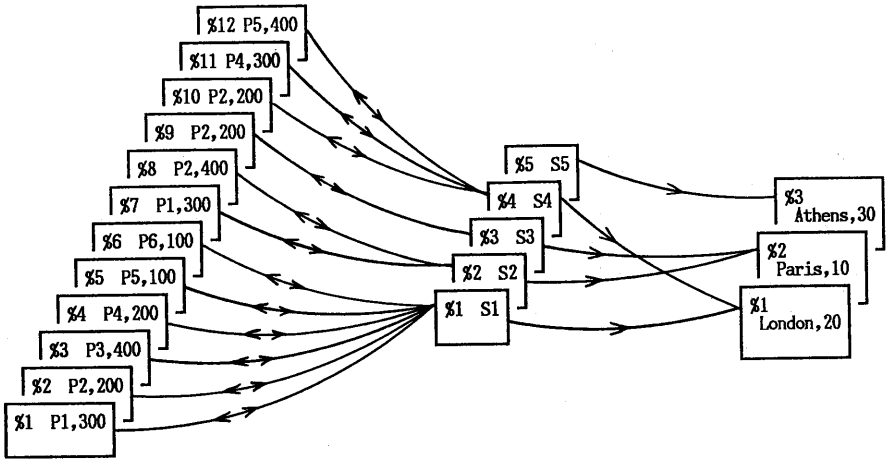


図12.　実現値レベルのデータ構造

5.　OSMANにおけるオブジェクト指向

5.1　オブジェクト指向データモデル

データベースの適用分野の拡大に伴い、データベースが対象とするデータは複雑化、多様化しつつある。近年、複雑かつ多様なデータを扱うためのデータモデルとして、オブジェクト指向アプローチが注目を集めている〔12〕。本章では、マンション・レイアウトのモデル化に伴って“観察”された“複雑なデータ現象”を述べるとともに、この種のデータを扱うためには、データとデータに対する基本操作群を管理の対象とするオブジェクト指向データモデルが好ましいことを論ずる。また、OSMANのオブジェクト指向の特徴をSmalltalk80〔13〕と対比しながら述べる。

5.2　マンション・レイアウトとオブジェクト指向

第2章で述べたように、OSMANでは4種類の記憶単位を用いてマンション・レイアウトを表現している。ここでは、レイアウトに対する操作を考慮しながら、レイアウトを構成するオブジェクトに固有の基本操作について論ずる。

(A) 空間の接続

カーネルに対する基本操作がオブジェクトの基本操作になる例として空間の接続を考える。空間の接続情報は、板材を媒介にして表現している。例として、図4に示した部屋のレイアウトと、図5に示したデータ構造について考えてみる。

部屋AとBが“隣り合っている”かどうかは、部屋AとBが板材を共有しているか否かによって判定することができる。つまり、部屋AとBを構成する板材を集合として考え、それらの集合の共通要素を調べればよい。形式的には次のようになる。

```
X：＝｛%1／和室(#板材)｝∩｛%1／洋室(#板材)｝
ＩＦ　(X＝ＮＵＬＬ)
　　ＴＨＥＮ　“非隣接”
　　ＥＬＳＥ　“隣接”
```

さらに、共通する板材がドア、襖、アコーディオン・カーテンといった属性値をもっていれば、部屋AとBは“続き部屋”ということになる。以上の操作は、カーネルを辿ることと、属性の値を調べるといったOSMANデータモデルの基本演算によって表現できるものである。

(B) 部屋の削除・分割

マンションのレイアウトを決めるときに、部屋の追加、削除、分割といった操作が必要となる。このような操作は、部屋という実体に依存するだけでなく、部屋が置かれている周囲の状況にも依存するものである。例えば、図４に示した部屋から、洋室Ｂを削除したとする。洋室Ｂが外部空間と接している部屋であったなら、板材９は外部空間と接することができるドアや壁に替えなければならない。他方、洋室Ｂそのものが、もっと広い居住空間、通路、戸棚によって囲まれているならば、板材９の変更は必要であるとは限らない。

部屋の分割の場合は、板材と柱材の生成が必要になる。例えば、図１３に示したような部屋の分割をするには、次のような処理が必要となる。

(1) 部屋の分割で使われる板材７と交わる板材を索す。（図１３の場合、板材２と４）

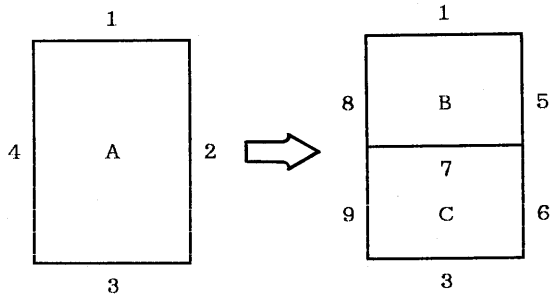


図13. 部屋の分割の例

(2) 上記の板材と交わる板材を索す。（図１３の場合、板材１と３）

(3) (1)で検索した板材をデータベースから削除する。

(4) (2)の板材と部屋の分割で使われる板材を結ぶ板材を生成し、データベースに追加する。（図１３の場合、板材５、６、８、９）

(5) 生成された板材を定義するのに必要となる柱材を生成し、データベスに追加する。

板材や柱材の生成は、ＯＳＭＡＮデータモデルの基本演算の範囲を越えるものであり、アプリケーションからの要求も含めて考えなければならないものであろう。部屋の削除、分割といった操作は、部屋というオブジェクトをどのように捕らえるかといったモデリングの根幹にかかわるものと考えられる。

５．３ ＯＳＭＡＮ v.s. Smalltalk80

ここでは、オブジェクト指向言語Smalltalk80 におけるクラス、インスタンスの概念とＯＳＭＡＮの概念との関連について述べる。

Smalltalk80では、クラスには必ずその親であるスーパクラスが存在し、クラスの階層構造を作ることができる。また、クラスは自分のスーパクラスの性質を継承(Inherit)することができる。この機能によって、必要なことは１箇所にしか書かなくて済ませることができる。必要なことを１箇所にしか書かないことによって、オブジェクトを修正するときの一貫性の維持を容易なものとすることができる。ＯＳＭＡＮで

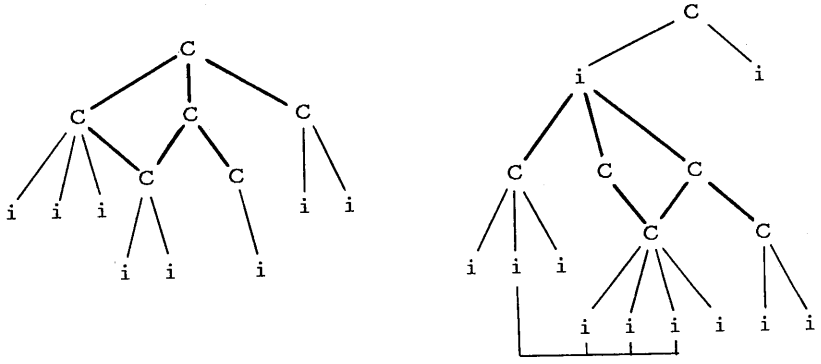


図14. Smalltalk80 とＯＳＭＡＮのオブジェクト指向
C —— クラス
i —— インスタンス

も、クラスの階層構造や属性の継承といった機能を提供している。

Smalltalk80とＯＳＭＡＮとの最大の違いは、クラスとインスタンスの関連付けにある。Smalltalk80におけるインスタンスは原子的である。すなわち、インスタンスは階層構造を成すこともなければ、他のインスタンスとの関連付けを行うこともない。一方、ＯＳＭＡＮにおけるインスタンスは原子的ではない。図３に示したマンション・レイアウトの例では、部屋と板材のインスタンス同士に依存関係が存在している。更に、インスタンスが内部構造を有するとき、クラス－インスタンスの構造を再帰的に使えることも特徴である。例えば、図３では“マンション”というクラスのインスタンスとして“物件”があるが、この物件には空間のレイアウトといった内部構造がある。このような場合、インスタンスの内部記述としてクラス－インスタンスの構造を再帰的に定義することができる。図１４は、Smalltalk80とＯＳＭＡＮにおけるクラス－インスタンス構造の概念的な違いを示している。

６．おわりに

本文では、実体－関連モデルにオブジェクト指向の概念を取り込んだＯＳＭＡＮと称するデータモデルとマンション・レイアウトのモデル化への適用事例を示した。ＣＡＤやＯＡなどのアプリケーションを考えるとき、複雑な対象物を直観的にモデル化することがデータベース構築の鍵となる。複雑さに対処する手段として、複雑なデータ構造を用意することも考えられるが、わかり易さ、直観性に乏しくなる傾向が見られる。単純なデータ構造に対象物固有の手続きを合わせ、これを情報の単位とするオブジェクト指向のアプローチの方が、より魅力的に感じられる。

参考文献

〔1〕 Chen,P.P.: The Entity-Relationship Model Towards a Unified View of Data, ACM Trans. Database Syst. Vol.1,1(1976).

〔2〕 Date,C.L.: An Introduction to Database Systems, Vol.1, 3rd ed. Addison-Wesley,1981.

〔3〕 Shoshani,A.: CABLE, A Language Based on the Entity-Relationship Model, Lawrence Berkeley Lab., 1978.

〔4〕 Su,S.Y.W.: Modeling Integrated Manufacturing Data with SAM*, IEEE COMPUTER, January 1986. pp.34-49.

〔5〕 Wilkins,M.W. et al.: Relational and Entity-Relationship Model Databases and Specialized Design Files in VLSI Design, Proc. 22nd Design Automation Conference, 1985, pp.410-416.

〔6〕 Zaniolo,C: The Database Language GEM, Proc. SIGMOD'83, pp.207-217.

〔7〕 大須賀：知識処理システムKAUSの設計方針と方式 信学技報AI87-35,1988.

〔8〕 石川，宇田川他，’建築ＣＡＤデータベースのスキーマ定義について”情報処理学会DB研究会、64-2(1988 年３月）.

〔9〕 宇田川，石川他，’建築ＣＡＤﾃﾞｰﾀﾍﾞｰｽのデータ定義言語の実現’第36回情報処理全国大会，1E-3,1988.

〔10〕西川，伊藤他，’建築ＣＡＤデータベースの基本思想’第35回情報処理全国大会，７Bb-3,1987.

〔11〕藤代，白井他，’リンク指向データモデルの論理スキーマ設計手法’第36回情報処理全国大会，2F-6,1988.

〔12〕大特集：オブジェクト指向プログラミング、情報処理、Vol.29-4(1988).

〔13〕 The Smalltalk80 Syatem, BYTE, Vol.6-8, (1988) pp.36-47.