データベース・システム　72－ 6
（1989. 7. 20）

オブジェクト指向意味データベースＯＳＭＡＮの実装と評価
Implementation and Evaluation of an Object-Oriented Semantic Database OSMAN

石川　洋　・　西川　正文　・　宇田川　佳久
Ishikawa Hiroshi　Nishikawa Masahumi　Udagawa Yosihisa

福地　陽一　・　佐々木　和恵　・　辻　秀一　・　市川　照久
Hukuchi Yhoichi　Sasaki Kazue　Tsuji Hidekazu　Ichikawa Teruhisa

三菱電機㈱
MITSUBISHI Electric Corp.

あらまし　設計作業で発生するデータは、非常に複雑であり、また煩雑に変化する。この様なデータを扱うのは、従来のデータベース・システムでは困難である。

エンジニアリング・データベースＯＳＭＡＮは、オブジェクト指向の概念を導入したデータベース・システムである。これにより、ＯＳＭＡＮは複雑な設計データを扱うことができる。

本文では、ＯＳＭＡＮのインプリメント上の特徴について述べる。また、ＯＳＭＡＮの性能測定の結果について述べる。

Abstract　Design data are very complicated in structure, and we often change these data in every design processes. It is hard for conventional database systems to manage these design data.

We developed a engineering database system on the basis of ER-model, called OSMAN. This system has some good points of object-oriented database which are useful for management of complex design data.

In this paper, we discuss the feature on implementation of OSMAN. And, we describe the performance of OSMAN with results of access speed measures of OSMAN.

1. はじめに

エンジニアリング・データベースは、設計・開発作業において生成・参照されるデータを管理対象としている。これを実現するためには、エンジニアリング・データベースは非常に多様で複雑なデータを扱わなければならない。また、設計データなどは、設計作業が進むにつれ、頻繁に変化し詳細化されるため、エンジニアリング・データベースはこの様な動的な変化をも管理できる必要がある。

上記のような、複雑なデータを扱うためのものとして、オブジェクト指向アプローチ[1,2]が注目されつつある。本文で述べるＯＳＭＡＮ[3,4,5,6] (Object-oriented Semantic Model for hAndling Nonmonotonic data) は、建築ＣＡＤデータベースとして開発されたものであり、ＥＲモデルに対してオブジェクト指向の概念を導入したものとなっている。

本文では、まずＯＳＭＡＮの特徴について簡単に述べる。次に、ＯＳＭＡＮのインプリメンテーション上の特徴を示し、さらにＯＳＭＡＮの性能測定の結果について述べる。

2. OSMANの特徴

本章では、ＯＳＭＡＮのオブジェクト指向の特徴を、マンション・レイアウトを例にして述べる。また、ＯＳＭＡＮとSmalltalk80 [7] との違いについて考察する。

2．1　基本要素

ＯＳＭＡＮは、以下に示す３種類のノードをポインタで結合することで実世界を記述する。これらのノードは、オブジェクトであると考えることが出来る。また、これらのノードは利用者から見ると可変長となっている。

(1) カーネル（Kernel）と呼ばれるノードは、属性の値、または他のノード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、ＥＲ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。一般にカーネルは、この中間の状態をとる。カーネルは、オブジェクト指向におけるインスタンスに対応する。

(2) クラス・カーネル（Class-K）と呼ばれるノードは、ノードの分類を行うために用いられる。つまり、いくつかのノードをまとめて、一つの概念として扱うときに用いられるものである。クラス・カーネルは階層構造を作ることができる。クラス・カーネルは、概念の記述を行うということから、オブジェクト指向におけるクラスに対応する。

(3) 幾何カーネル（Geometric-K）と呼ばれるノードは、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の受け渡しをするためのものである。

これら三つのノードはそれぞれ、図１に示す図形で表わされる。ノード間の結合は、ノードを線で結ぶことにより表わされる。この様な形でデータを表現することで、複雑な設計データにも容易に対処できる。

２．２　ＯＳＭＡＮにおけるオブジェクト指向

ここでは、不動産物件としてのマンションのレイアウトを例として、ＯＳＭＡＮにおけるオブジェクト指向の特徴について述べる。ここでいうマンションとは、図２に示すような物件の集合体と考える。

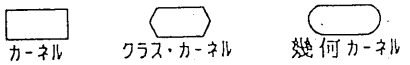

図１　各カーネルのシンボル

モデル化する際の要件として、マンション・レイアウトを、単なる図形情報としてではなく、「部屋」などの空間の集りとして認識し、操作できることを考える。この為には、空間を用途別に分類することと、空間同士の隣接関係を識別できるようにすることが必要となる。この方針に従ってモデル化した例が、図３である。

個々のマンションはクラス・カーネル「マンション」のインスタンスである。また、個々のマンションは物件の集合体となる。各物件はそれぞれ特有の構造を持っており、これを表現するためにインスタンスである物件の下に、更にクラス階層が作られる。

物件の構造は、空間の集りと、壁などの構成要素の集りという２通りの視点から記述されている。クラス・カーネル「空間」は、その下に、より詳細化されたクラス・カーネルを持っている。この様なクラス階層を構成することにより、データの詳細化に対応することができる。

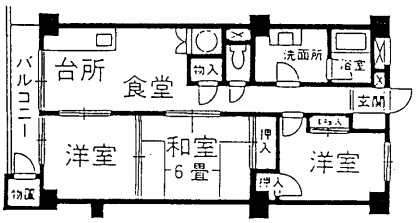

図２　マンション・レイアウトの例

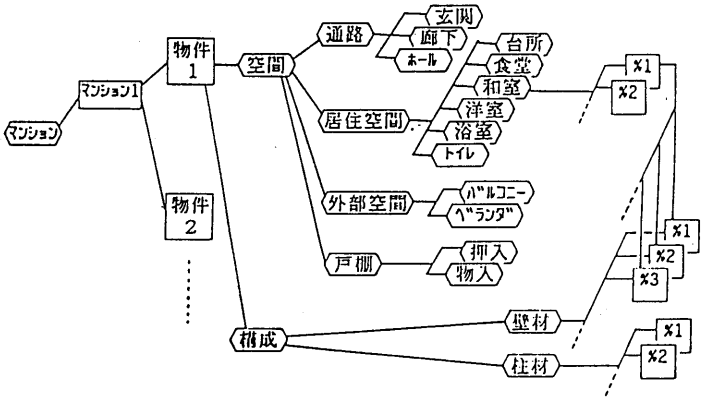

図３　ＯＳＭＡＮにおけるレイアウトの表現例

個々の壁材を表わすカーネルと、部屋を表わすカーネルはそれぞれ関係付けられており、部屋同士が共有する壁を認識することで、部屋の隣接関係を知ることができる。また、壁材や柱材は、部屋などの空間と関係なく扱うことも可能である。この様にして、多数のユーザからの、データに対する異なる要求を統合することが可能である。

以上のように、ＯＳＭＡＮでは、多くの異なる立場で認識された「もの」をオブジェクトとして定義し、それらの間に関係付けを行ない、さらにクラス階層を定義することで、多様で複雑なデータを表現することができる。また、データに対して、各ユーザごとの認識にそった処理を定義することができる。

２．３　Smalltalk80 とＯＳＭＡＮ

Smalltalk80 とＯＳＭＡＮとを比較した場合、もっとも顕著な違いは、クラスとインスタンスの関係にある。前章で示したように、ＯＳＭＡＮではインスタンスであるカーネルの下にさらにクラス・カーネルによるクラス階層を構成することができる。Smalltalk80 の場合には、インスタンス変数を用いて、ある程度までは暗にこの様な構造を作ることはできるが、陽にこれを表現することはできない。

この様な違いは、両者におけるインスタンスの扱われ方の違いに起因していると言える。Smalltalk80 では、クラスは多数のユーザに共用されるが、インスタンスは基本的にアプリケーションの実行中しか存在せず、保存され、共有されることはない。しかし、ＯＳＭＡＮの場合には、インスタンスであるカーネルは、半永久的に存在し、多数のユーザに利用され、変更されていく。また、カーネルの総数は膨大なものとなる。さらに、設計分野などへの適用を考えると、カーネルは、非常に複雑な構造を持つ実体を表わす必要がある。

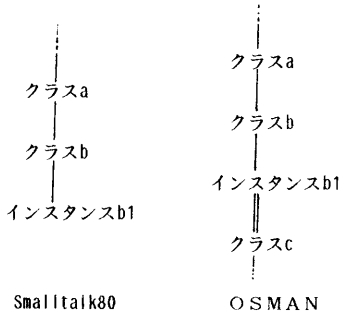

図４．クラス―インスタンス関係の違い

よって、ＯＳＭＡＮではインスタンスの構造を明確に表現しておく必要がある。構造を表現することにより、そのインスタンスが意味するところがユーザに対して明らかにされる。これにより、データ共用を達成することが容易になる。また、クラス階層によりインスタンス構造を表現することで、あるインスタンスの変更などの影響の範囲を限定することができる。これにより、データ独立を保障できる。

３．ＯＳＭＡＮの内部構造

第２章で述べたように、ＯＳＭＡＮは可変長のカーネルをポインタで組み合わせることにより、非常に複雑なデータをも容易に表現することができる。ＯＳＭＡＮのこの様なイメージは仮想記憶空間上に実現されている。ＯＳＭＡＮの仮想記憶管理部は、利用者イメージであるカーネル―ポインタ構造と２次記憶上のデータベース本体との間のマッピングを管理しており、カーネルの実体が２次記憶上のどこにあろうと利用者に意識させない様になっている。

３．１　ＯＳＭＡＮのデータ管理方式

ＯＳＭＡＮが利用者に提供するカーネル―ポインタ構造は、仮想記憶空間の上に構築されている。ここでは、この様な機能を実現するために、ＯＳＭＡＮが行なっているデータ管理の方式の概略について述べる。

ＯＳＭＡＮの仮想記憶管理は、ページングにより行なわれる。通常のインプリメンテーションでは、ページサイズは１６Ｋバイト（１６３８４バイト）となっている。主記憶上には、数～数十ページ分の領域が確保され、データに対する操作はすべてこの主記憶上にあるページに対して行なわれる。主記憶上にないページに対する操作を行なう場合には、現在主記憶上にあるページの内、適当なものをページアウトし、必要なページを２次記憶からページインする。主記憶と２次記憶の対応関係は図５のような形となる。これが、ＯＳＭＡＮにおける最も低レベルな（すなわち、計算機に近い）データ管理方式である。

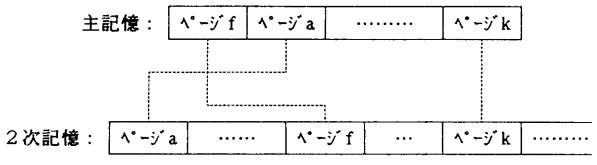

図５　主記憶と２次記憶の間の関係

この仮想記憶の上に動的メモリ管理機能が実現されている。ここでの管理の基本単位は６４バイトの「セル」である。各ページはこのセルが２５６個集まったものであるとして扱われる。また、利用者が扱う基本単位であるカーネルは、セルをいくつかつないだ形で実現されている。すなわち、ある大きさのカーネルを生成しようとすると、そのカーネルを形成するために十分なサイズを満たす様に、未使用セルの集合からセルがいくつか確保され、これによって、カーネルを実現する。また、カーネルのサイズを変更する場合には、必要に応じて新たにセルを確保したり、不要になったセルを未使用セルの集合に戻したりする。ＯＳＭＡＮは、このようにして可変長データの取り扱いを実現している。

３．２　ＭＩＯルーチン群

ＯＳＭＡＮでは、ページ管理の実現手段として、ｋｅｙｅｄファイルを用いている。ページングを行なう場合には、ページを識別するために各ページにページ番号が割り当てられる。ＯＳＭＡＮでは１つのページを１つのレコードとし、ページ番号を基にレコードのｋｅｙを作り出している。こうすることにより、ページイン／アウトは、ＯＳの提供するｋｅｙｅｄアクセス・メソッドを使って容易に実現することが出来る。

この様な、ｋｅｙｅｄファイルによるページのＩ／Ｏを行っているのが、ＭＩＯと呼ばれるルーチン群である。ＭＩＯ部分は、ＯＳＭＡＮの中でほぼ唯一ＯＳに依存した部分であり、ＯＳがどの様なアクセス・メソッドを用意しているかによって、実現方法が異なってくる。逆にいえば、このＭＩＯルーチン群を実現できれば、その計算機にＯＳＭＡＮを移植することが可能ということであり、この様な実現手段を取ることでＯＳＭＡＮの移植性を高く保つことに成功している。事実、ＯＳＭＡＮは現在、汎用機、ミニコン、ＷＳなど様々な計算機上で稼働している。

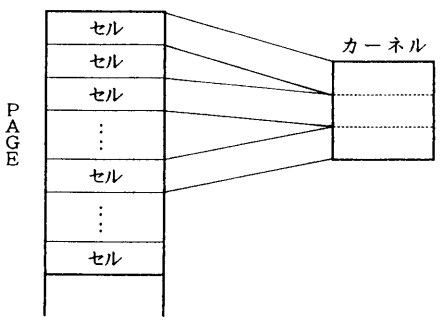

図６　ページ－セル－カーネルの関係

３．２．１　ページの形式

ＯＳＭＡＮで、ページを実現するためのレコード形式は図７に示す通りである。

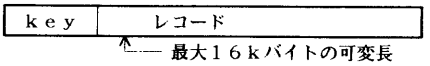

図７　ページの構造

ＭＩＯルーチン群は、上に示したレコード形式のファイルについて、ＯＰＥＮ／ＣＬＯＳＥ、ＲＥＡＤ／ＷＲＩＴＥ、レコード削除の機能を提供する。また、ｋｅｙｅｄファイルのＯＰＥＮ時にファイルのモードとして、ＩＮ（読み出し）、ＯＵＴ（書き込み）、ＩＮＯＵＴ（読み書き可、既存ファイル）、ＯＵＴＩＮ（読み書き可、新規ファイル生成）の４つのモードがあり、このモードをチェックする機能も用意されている。

4. OSMANの性能測定

前節で述べたように、ＯＳＭＡＮは仮想記憶空間上でデータの管理を行っている。ところで、この様な仮想記憶を用いた場合、実際の主記憶上にないデータに対するアクセスを行なおうとすると、主記憶と２次記憶の間でページイン／アウトを行なう必要がある。よって、この場合には主記憶上のデータをアクセスする場合にくらべてページイン／アウト分のオーバーヘッドが生じ、データアクセスの性能が低下することが知られている。これはＯＳＭＡＮに関しても同様で、特に仮想記憶空間に対してランダムなアクセスを行なった場合には、かなりの性能の低下が生じることが確かめられている。。

しかし、実際にＯＳＭＡＮの上でＣＡＤアプリケーションを実現した場合には、データベース全体をランダムにアクセスすることは非常に稀であると考えられる。ＣＡＤアプリケーションで取り扱う対象は、一般に非常に複雑であり、これをデータベース上で表現した場合には、多くの単純なデータの集りからなる、いわゆる複合オブジェクトを単位として扱うことになる。よって、データ定義も複合オブジェクト単位で行なわれ、その結果、関連が密なカーネルは集中して配置される。このことから、データベースに対するアクセスはかなり局所的なものが多くなると思われる。

本章では、アプリケーションレベルで見たＯＳＭＡＮの性能測定について述べる。測定の対象としては、ＵＮＩＸワークステーションＭＥ１３００上のＯＳＭＡＮを用いて試作したマンション物件情報システムを用いて、物件情報の登録と検索について測定を行なった。

４．１　マンション物件情報システム

ここでは、性能測定の対象とした「マンション物件情報システム」について述べる。このシステムはＯＳＭＡＮの評価用として試作したものである。これはマンションに関する不動産物件情報を扱うもので、間取りなどのレイアウト情報も扱うことができるようなデータ構造を持つものである。このシステムは図８に示すような機能を持つ。

このシステムで取り扱うデータ構造は、２章で示した図３の様なものである。この様なデータ構造を用いることにより、、個々の物件を「部屋」などの空間の集合としてみた場合に得られるデータと、「壁」、「柱」などの部品からなる構造物としてとらえた場合のデータを一括して扱えるようになっている。また、両者を関連付けることにより、部屋同士の隣接関係なども扱えるような形になっている。

４．２　性能測定

ＯＳＭＡＮに対してアクセスする際に、３．１節で示した「セル」をランダムに参照／更新するような状況が生じると、図９に示すように、極端な性能低下が生じることが確められている。ただし、これはＯＳＭＡＮの低レベル・アクセスルーチンを用いて測定した結果である。

ここでは、ＯＳＭＡＮの性能評価の一環として、前節で示したマンション物件情報システムの性能を測定する。このとき、測定方法としてさまざまなものが考えられるが、今回は、物件情報のデータベースへの登録と物件情報の検索の２点について測定した。

この際に扱うデータの単位は、図１０に示すような「物件」以下の全データである。マンション物件情報システムの場合、これを単位として扱うことが最も多いと考えられる。この様な大きな単位でデータを扱った場合に、ＯＳＭＡＮがどの様な性能を示すかを調べるのが今回の測定の目的である。

４．２．１　測定方法

性能測定は、測定しようとする機能のコマンドをマンション物件情報システムに与え、その実行時間を測定する、という形で行なった。具体的な方法は、以下の通りである。

１）実際のマンション物件情報システムと、ＯＳＭＡＮへのインタフェースをｃａｌｌしている部分の

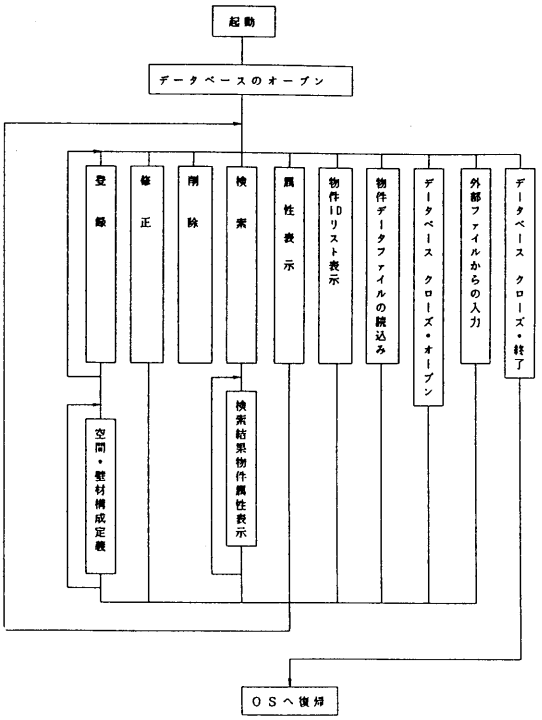

図８　マンション物件情報システムの機能

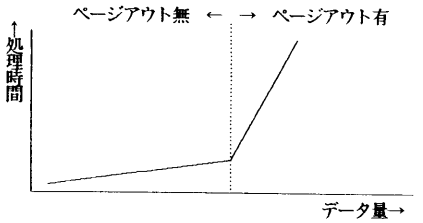

図９　ランダム・アクセス時の性能

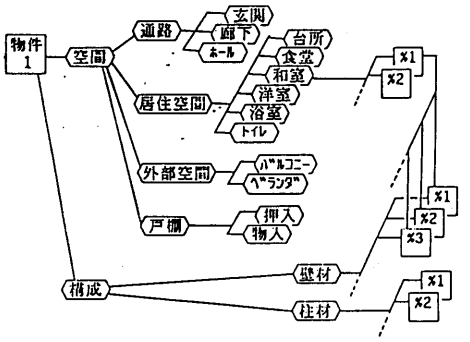

図１０　処理単位

```
KDEF %MB(                    ：物件の登録開始
        1:0002                                  ：物件ID
        2:0001                                  ：不動産屋ID
        3:   3290                               ：価格（万円）
        4:0                                     ：管理費
        5: 74.14                                ：専有面積
        6:  8.18                                ：バルコニー面積
        7:壁式ＰＣ造                            ：構造
        8:3LDK                                  ：間取り
        9:５階建の４階                          ：マンション内での位置
        10:-                                    ：環境
        11:63.02                                ：築年
        12:徒歩１５分　バス停下車徒歩１分       ：交通
        13:63.12.1                              ：データ入力日
)@
KDEF %MB111(1:1)@

          :
          :
  「部屋」「廊下」「壁」などの定義
          :
          :

KDEF %MB142(1:3)
```

図１１　物件情報登録用コマンドの例（：以降はコメント）

みを取り除いたマンション物件情報システムの２つを用意する。

２）両者に同一のコマンドを与え、それぞれの実行時間を、ｔｉｍｅｘ[8] コマンドで計測する。

３）ｔｉｍｅｘコマンドの結果の内、ユーザ時間について
（実際のシステムの実行時間）－
（ＯＳＭＡＮを呼ばないシステムの実行時間）
を計算し、これを測定結果とする。

以上のようにすることで、システム自体の初期化の手間や、マンション物件情報システムのコマンド解析にかかる時間などを除外した実行時間を知ることができる。

また、測定項目としては、

１）マンションを一件登録し、その中の物件以下の情報を５、１０、１５、２０、２５および３０件登録したときのそれぞれの時間。

２）３０件の物件が登録されているデータベースから、物件ＩＤにより５、１０、１５、２０、２５および３０件の物件情報を先頭から順に検索するのにかかるそれぞれの時間。

３）３０件の物件が登録されているデータベースから、物件ＩＤにより５、１０、１５、２０、２５および３０件の物件情報をランダムに検索するのにかかるそれぞれの時間。

の３つについて行なった。

さらに、ページイン／アウトを起こしやすくするために、ＯＳＭＡＮ自体の主記憶上のページ数を１０ページに制限した。今回登録した物件データは、１ページにつき１．６～７件ほど登録できるサイズであるので、１５件を越えたあたりからページイン／アウトが起きることになる。

図１１に、物件情報登録用のコマンドの例を示す。

４．２．２　測定結果

図１２から図１４に、測定結果のグラフを示す。これらは、同一項目について１０回の測定を行ない、それを平均したものをグラフ化したものである。これを見ると、検索に関しては実行時間の変化はほぼ線形であるといえる。また、生成にかかる時間は、１５件を境に増加の度合が大きくなっているが、これもそれほど極端に増加している訳ではない。

これにより、以下のことが推察される。まず、検索のみを行なう場合には主記憶上のページは変化しないため、ページアウトを行なう必要がなく、直接ページインを行なうことができる。従って、この場合のオーバーヘッドはページアウトが必要な場合に比べてかなり小さくて済む。また、登録時には、かなりまとまったデータを一度に生成しているため、それらのデータは局所的に配置され複数のページをまたがってしまうことが少なくなっていると考えられる。これは検索に対しても良い影響をあたえると思われる。

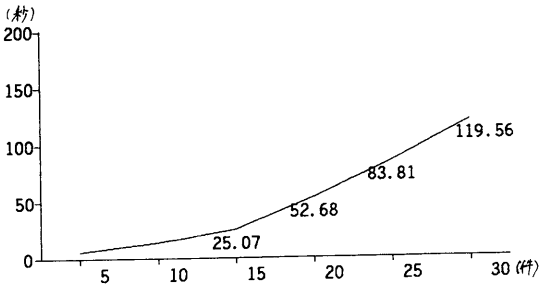

図１２　物件の生成

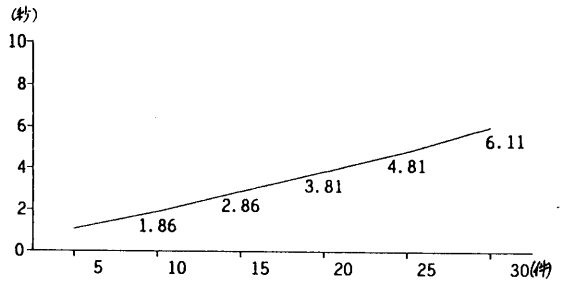

図１３　物件の検索（シーケンシャル）

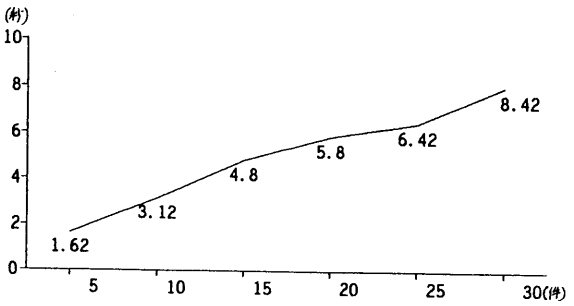

図１４　物件の検索（ランダム）

5. おわりに

今回、アプリケーションレベルでのデータ登録および検索について性能測定を行ない、ＯＳＭＡＮの性能の評価を試みた。今回の結果から、ある程度まとまった量を単位としてデータを取り扱う場合には、ＯＳＭＡＮの性能は最良の場合からそれ程低下しない、と言える。ＣＡＤなどといったかなり複雑なデータを扱うアプリケーションにおいては、この様な場合がほとんどであると思われる。

ただし、これはデータの局所性が比較的高くなるという条件が必要と思われるため、データベース構造を局所性のないような形に構築したり、また局所性を著しく乱す操作を行った場合には、性能がおちるということも十分に考えられる。今後はこの点を考慮し、カーネルの削除などといったデータベース構造自体の変更を含む操作に関する考察が必要であろう。また、この様な変更に対しても局所性を保存できるようなデータベース管理方式に対する検討を行なうことも必要であると思われる。

参考文献

[1] 田中：“オブジェクト指向データベース・システム－その背景と概念”、bit Vol.20 No.6,共立出版、1988.6

[2] “コンピュータ新時代を開くオブジェクト指向の潮流”、日経コンピュータ 1989/5/22号,pp108-121

[3] 西川、伊藤他：“建築ＣＡＤデータベースの基本思想”、第35回情報処理学会全国大会、7Bb-3,1987

[4] 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64-2,1988.3

[5] 宇田川、石川他：“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”、情報処理学会DB研究会65-4,1988.5

[6] 石川、宇田川他：“オブジェクト指向意味ネットワークモデルＯＳＭＡＮについて”、第37回情報処理学会全国大会、1Q-4, 1988.9

[7] アスキー書籍編集部 編：“Smalltalk 入門”、アスキー出版局、1986

[8] “ＵＮＩＸ ＳｙｓｔｅｍＶ ユーザ・リファレンス・マニュアル 第2版 リリース３．０”、共立出版、1986