データベース・システム 84−12
（１９９１．７．１８）

拡張可能データベースのプロトタイプ作成

飯沼 聡　山内 正　(株) 三菱総合研究所

宇田川佳久　岡本基良　富沢研三　三菱電機 (株) 情報電子研究所

iinuma@asdal.mri.co.jp yama@asdal.mri.co.jp

udagawa@isl.melco.co.jp okamoto@isl.melco.co.jp

tomisawa@isl.melco.co.jp

概要

建築ＣＡＤシステムを対象にしたデータベースに対する要件として、建築物の形状データと設計・管理情報としての属性データとを統合管理することや、形状の追加、削除、詳細化に伴いデータベースの構造を頻繁に変更できることがあげられている。形状データと属性データを統合するため、データベース・スキーマ定義言語とデータベース操作言語を設計し、これらの結果をもとに、形状データと属性データの統合管理機能、階層的データベース構造の拡張機能、建築レイアウト図の表示機能等を有する拡張可能データベースプロトタイプシステムの製作を行った。

Prototype system of extensible data base system

Satoshi Iinuma　Tadashi Yamanouchi
Mitubishi Research Institute,Inc.
3-6 otemachi 2-chome Chiyoda-ku , Tokyo　100 , Japan
Yoshihisa Udagawa　Kiyoshi Okamoto
Kenzou Tomisawa
Mitubishi Electric Corporation
5-1-1 Ofuna , Kamakura, 247 , Japan

Abstract

It is said that the Data Base Managing System for CAD,dealing with Architectural Layouts,is necessary to manage form data of Architecture and attribute data as design and management information and to be able to modify the logical structure of database frequently through inserting,deleteing and detailing the from. To integrate the formal data and attribute data,we have designed the database schema definetion language and database management language. In addition,we have implemented a prototype system of Extensible Data Base Management System , which can manage formal data and attribute data integratedly ,extend the hierachical database structure and display the drawing of layout.

1 はじめに

技術情報やノウハウの共有化、標準化、蓄積を目的としているＣＡＤ／ＣＡＭシステムに対する要求の多様化、複雑化に対処するため、柔軟かつ効果的な開発技法や要素技術の研究・開発が待たれている。設計対象のモデリング手法やそれらの内部表現が性能を左右するこうしたシステムにおいては、特にデータベースに対する要求が増加してきている。

建築ＣＡＤシステムを例にした場合、そのデータベースは、建築物の形状データと設計・管理情報としての属性データとを統合管理することや、形状の追加、削除、詳細化に伴いデータベースの構造を頻繁に変更できることが必要となる。更に、図形データをうまく扱うためには、図面に対する処理・管理を中心に部品を表した図面と全体図との関係や、部品図同士の関係、図面と部品との関係などをうまく表現できることが必要である。この要件に対して、従来のＲＤＢ（関係型データベース）を用いた方式だけでは、表現構造に多様性があり、またそのデータの取扱いに関する情報も併せて表現する必要のある図形データを管理するには不十分であった。こうした課題に対して、オブジェクト指向データベースや拡張可能データベースを用いたアプローチの有効性が指摘されている.

本稿では、建築ＣＡＤデータベースに対する適用性を検討する観点から作成された、

- 形状データと属性データの統合管理機能
- 階層的データベース構造の拡張機能
- 建築レイアウト図の表示機能

等を特長とする拡張可能データベースのプロトタイプについて報告する。

2 拡張可能データベースの機能要件

2.1 形状データと属性データの統合管理

従来、主なるデータベースの適用分野であった事務処理において用いられるデータ構造は、文字や数字といった構造を持たないものであった。しかし、ＣＡＤ分野に留まらずＯＡの分野においても、複雑な構造を有するデータの取扱が要求され、形状データと属性データを統合して扱う機能が必須となりつつある。マルチメディア化は、こうした動きをさらに加速すると考えられる。これまでに、形状データと属性データの階層的表現とそれらの統合方法を開発する試みがいくつかなされてきている。

統合の内容・範囲についての設定は、いくつかのレベルが想定されるが、統合、管理機能に対する要件としては、形状（図形データ）から属性（文字データ）を参照でき、さらに、属性から形状を参照できることを目指すものとした。

2.2 階層的データ構造の拡張機能

ＣＡＤやＯＡにおけるデータは、一度に完全な形でデータベースに入力されるわけではない。例えば、建築レイアウト図の場合、一本のエッジ（線分）から始まり、徐々にレイアウト図が作られていく。エッジは“壁”に対応し、エッジによって囲まれた領域は“部屋”に対応する。このように、レイアウト図が作られてゆく過程で、建築オブジェクトの階層構造を形成していかなければならない。そのためには、境界表現（Boundary Representaion）モデルを操作する一連のコマンドを作成する必要がある。具体的には、建築ＣＡＤにおける階層構造の定義に際して、

- 新しいクラスが定義できる
- クラス間の関係がリンクとして定義できる
- クラスの定義変更を動的に、すなわちスキーマ変更後、データベースの再構築を自動的に行える。

等の機能を実現することにより、論理データに対する拡張可能性を実現することとした。

2.3 建築レイアウト図の表示機能

上述の、“建築ＣＡＤ用拡張可能データベース”の内容を確認するために、建築レイアウト図の図形表示機能が必要となる。表示内容は、マンションを対象とした場合、２次元の建築レイアウト図であり、不動産分野のデータベース構造に関する設定の確認が可能なものとした。表示機能の実現にあたり、論理的な視点（ユーザが操作する視点）と物理的な視点（データベース内部構造）から以下の要件を設定した。

1. 論理的な視点
   ・マンション物件全体を表示できる。
   ・物件内の部屋毎に表示ができる。
   ・壁単位に表示ができる。

2. 物理的な視点
   ・クラス名、及び、オブジェクト名で決定されるインスタンス単位に格納されている情報内容を表示できる。
   ・データベースを構成している、インスタンス自身と他のインスタンスをつなぐリンク、及びそのリンクによってつながれた他のインスタンス全体の情報を併せて表示できる。

以上の要件を実現するため、拡張可能型ＤＢＭＳの構成を、データベーススキーマ定義言語とデータベース操作言語の観点から設計し、それらのプロトタイプシステムの試作を行った。

3　スキーマ定義言語の構成

3.1　スキーマ定義言語の検討

データベースとしての仕様、および対象となるマンションデータのモデル化に対する検討結果より、スキーマ定義言語としての機能要件を設定した。

スキーマはデータベースの論理構造を決定づけるものであるが、その設定・指示をどの様な位置づけとするかは、以下の考え方がある。

1. プログラムの中に直接記述する。

2. スキーマをＣの typedef として宣言し、プログラムのコンパイル，リンクによりスキーマのアップデートを行う。

3. スキーマをファイルとして独立させ、スキーマを処理するモジュールを別途用意する。

1. は拡張可能データベースのスキーマとしては拡張性の観点から検討対象外である。

2. はスキーマをＣの構造体として定義する方法であるため、その文法はＣに準拠し、スキーマの解釈および作成はＣの処理系によって行われる。この方法では、解釈系および生成系を作成する必要がないが、スキーマの変更はデータベースシステムそのものの再コンパイル、リンクにつながり、ＣＡＤ用システムのようにスキーマ変更が頻繁に行われるようなシステムには適さない。またＣの複雑な文法規約をスキーマ作成者に要求することとなってしまう。

3. はスキーマ定義のための文法を用意し、その文法に従って記述したスキーマファイルを解釈する処理系を用意する方法である。スキーマ処理系を用意しなければならない分、システムとしては複雑となるが、スキーマ変更に対するシステムの再コンパイルおよびリンクは不要のため、拡張は容易である。また、データベースを使用する側からの要請により文法を拡張する必要が生じた場合でも、スキーマ処理系のみの拡張で済む場合が多い。

以上の検討の結果、本プロイトタイプにおいては 3. のスキーマファイルにより定義する方法を採用した。

スキーマの文法においては、スキーマ処理系の負荷を軽減させる見地から、

- クラスの定義
- 親クラスの定義
- クラスのメンバ（属性）の定義

を行うのみとした。通常のスキーマ定義言語が提供しているような、探索キーの設定や親子集合の定義などはスキーマ定義言語としては特に用意していない。必要があれば各クラスの属性としてキーを定義したり、オブジェクト間に論理的な名前付でリンクを張ることにより関係のあるオブジェクトを集合としてあつかうことができる機能をデータベース操作言語に付与することで、データベースとしての充分な拡張性を確保できると判断した。

3.2　データベースの完全性確保

プログラムとデータ独立の観点から、スキーマのダイナミックな変更に対するデータベースの完全性確保は重要である。前節で触れたように、本プロトタイプにおいてはスキーマファイルの変更によりデータベースの拡張を行うことができる。スキーマファイルが変更された場合、既存のデータ（オブジェクト）に対しても、新スキーマに適合するようにデータ構造を更新する必要が生じる。これをデータベースの再構築と呼ぶが、データベースとしての完全性を保持したうえでデータベースの再構築を実行するのは数多くの問題が存在する。従って、スキーマ変

更に対してある程度の制約を設け、その範囲内であるならばデータベースの完全性を保証して再構築を行えることを目標とした。

ここで、前述の文法を有するスキーマに対する変更要件としては以下のものが考えられる。

1. 新規クラスの追加
2. クラスの削除
3. クラス名称の変更
4. クラスの親クラスの変更
5. クラスのメンバの追加
6. クラスのメンバの削除
7. クラスのメンバの変更

これらの変更に対して、データベースの完全性を保証するため、以下の操作を設定し、一貫性を保持することとした。

1. は既存の他のクラスには影響を与えないため問題はない。

2. は削除対象のクラスに属するオブジェクトが全て削除される。さらに、削除されたオブジェクトと論理的にリンク付けられたオブジェクトについては、そのリンク関係を白紙化する必要がある。また、該当クラスを親クラスとして定義しているクラスに関しては、スーパークラスの削除時に同時に子クラスを全て削除する処理を施すこととした。この方式は、処理としては1つのクラスの削除と同様の処理を繰り返すのみであるので比較的単純であるが、利用者側としては予期せぬクラスが削除されるので注意が必要である。

スーパークラスの定義を無効にする(スーパークラスが存在しない)方式も想定されたが、子クラスは、スーパークラスのメンバ(属性)を全て継承するので、該当クラスの全てのオブジェクトからスーパークラスの属性を削除しなければならず、今回は採用しなかった。

3. のクラス名称の変更はクラスに所属する全てのオブジェクトおよびスーパークラスを指定している全ての子クラスに影響する。さらにクラスの識別を行う唯一のキーであるのでこれを変更するともとのクラスとの同一性を認識できないのでクラス名の変更は全く別のクラスができる(前のクラスのインスタンスは全て消滅する)ことを意味する。

4. のスーパークラスの変更は、基本的には2と同じ問題が発生し、該当オブジェクト全てのスーパークラスから引き継いだ属性を削除し、新たなクラスから引き継いだ属性を設定しなければならない。これは2同様困難である。

5. 及び6. はクラスのメンバの追加・削除は、各オブジェクトに新メンバ用の領域を確保するかデータごと削除するのみであるから、比較的容易である。

7. のクラスのメンバが変更になった場合、特にデータ長が変わるような変更の場合に、格納されているデータそのものをどうするか(新しい型に合わせて変更するか、データそのものを削除するか)が問題となる。

以上の検討から、3. 、4. 、7. を除く変更ならばデータベースの完全性が保証できる仕様とした。

3.3 オブジェクト指向の反映

本プロトタイプにおいて、オブジェクト指向の概念をどこまで反映させるかについて述べる。

オブジェクト指向を特徴づける考え方として、クラス定義や継承などの概念があるがこれらをどの様に反映させたかについて述べる。

(1) データ単位としてのオブジェクト

データの識別単位として、オブジェクトという単位でデータを表現することを検討した。オブジェクトはインスタンス変数の値により一意に識別され、そのインスタンス変数はクラス定義において定義される。このインスタンス変数の定義にデータ型の概念を導入することで、データベースとして、様々なデータタイプを扱うことが可能となる。この考え方はクラス階層や継承の概念を導入することでデータの表現を簡略化することができる。結合、分離などの関係演算もメソッドの記述により実現することは可能である。

(2) クラスと階層

データ型を定義するレコード型に相当するものとしてクラスを考える。クラス定義は上位クラス(親クラス)の指定とインスタンス変数の定義とからなる。レコード型でもレコード型間の関係を定義することは可能であったが、内部のデータ型を共有したり、継承したりすることはできない。クラス定義に

おいては、上位クラスとして指定したクラスのインスタンス変数は自動的にすべて継承されるので、各クラスの定義が簡略化される。

3.4 スキーマ定義言語の文法

クラスの記述はschema命令、super命令、member命令の組み合わせで記述される。各命令の出現順序はこの順序でなければならない。

・schema命令

この命令は、1つのクラス型ごとに最初に与える命令で、この中でクラス型の定義を行う。

・super命令

この命令は、1つのクラスに対する上位クラスを定義するもので、schema命令に引き続いて記述する。上位クラスは必ず定義しなくてはならない。また、super命令は1つのschema命令の後に複数記述することができ、複数のクラスが同一のクラスを親クラスとして宣言することはできる。

・member命令

この命令は、schema命令、super命令に引き続き記述し、schema命令内部で使用される属性（インスタンス変数、つまりオブジェクト内部で使用される関数）を定義する。存在しない場合は定義は不用であり、また1つのschema命令に対して複数のmember命令を記述することができる。

4 データベース操作言語の構成検討

4.1 実現方式の検討

データベース操作言語の利用形式は、C言語等にリンクするライブラリ形式(上位のアプリケーションプログラム（CADなど）より関数形式で呼び出す形態)や、コマンドレベルでユーザがインタラクティブに指示・操作することにより利用するコマンド形式、さらに両者を組み合わせて利用する形式がある。

プロトタイプでは、データベース定義言語およびデータベース操作言語に対する検討を目的としている。したがって、データベース管理システムとしての有為性とアプリケーション（データベース管理システム自体もアプリケーションの1つと考えるならば）に組み込む場合の有為性を考える必要がある。そこで、本プロトタイプにおいてはアプリケーションとしてデータベース管理システムを想定し、ユーザインタフェースを一般のC言語で記述し、データベース操作の為の関数をライブラリとして独立させた。これにより、データベース操作言語としての有為性とデータベースシステムとしての有為性の双方を検証することができる。

実現機能の範囲・レベルに対する検討の結果、以下の機能群を実現することとした。

1. 属性・形状の統合機能
2. 新しいクラスが定義可能
3. インスタンスが動的に作成可能
4. クラスの関係がリンクとして定義可能
5. 動的なリンク付けが可能
6. 動的なクラスの定義変更が可能（スキーマ変更後、データベースの再構築を自動的に行う。）

1. 属性・形状の統合機能

拡張可能データベースの機能要件として、属性(一般データ)と形状(幾何データ)の両方を統合して扱うことがあげられた。属性データと形状データを統合するためには、両者で共通の部分と異なる部分を明確に区分けする必要がある。これについては、スキーマ定義言語の構成検討の節で述べたように、共通の部分を親クラスのインスタンス変数として記述し、属性、形状をその子クラスとして定義することで、統合機能については親クラスの機能を継承し、属性・形状固有の機能については各クラスで定義することで実現できる。

2. 新しいクラスの定義

追加定義に関しては、インスタンスをリンクするための領域をrootクラスのインスタンス変数として用意することで実現できる。このとき、

A. リンクできるインスタンス数の上限

B. 1つのインスタンスがリンクできるリンクの種類の上限

が問題となる。

インスタンスの数に関しては、CODASYL型のデータベースに用いられた様なリンクのメンバ間でリンクポインタを保持する(各メンバが次のメンバへのポインタを持つ)ことで(メモリ等物理的な制約を除けば)上限は存在しない。リンク種類の数に

関しては、同様にリンクの種類ごとにリンクポインタをもつことで上限をなくすことができるがインスタンスとリンクで双方をリンクポインタにより実現するのはかなり複雑な処理となり、かつ必要性はあまりないと考え、リンクの種類には上限を設け、あらかじめそのための領域を確保する方式で実現することとした。変更・削除機能に関しても、上で述べたようなリンクのメカニズムにより実現することができる。検索機能はリンクの論理名を指定することでそのリンクに属するメンバを検索することができる。

3. インスタンスが動的に作成可能

インスタンスの動的な生成は、スキーマファイルによって定義されたクラスのインスタンス変数をもつ領域を動的に確保することで実現することができる。この領域は各インスタンスごとに独立にとられ、かつその大きさは (スキーマの動的な変更が行われない限り) 固定であるのでその領域の先頭ポインタのみ管理していればよい。

動的なデータ変更はこの先頭ポインタから始まる固定長領域に対するデータ変更を行えばよい。削除はこの先頭ポインタで示される領域を解放するのはもちろんであるが、リンクにより関係づけられたインスタンスのリンク情報を修正する必要がある。以下のような処理を行う必要がある。

A. 自分を親とするリンクに属するインスタンスの該当リンクに関するリンク情報の削除

B. 自分を子とするリンクに属するインスタンスの該当リンクに関するリンク情報の修正 (該当インスタンスをバイパスする)

検索はクラスに定義されたインスタンス変数の値により検索する機能のみ実現する。したがって、インスタンス変数を規定するためのクラスを指定しなければならない。これは検索の対象となるクラスの変数のみしか検索条件として指定できないことを意味する。そこで、検索の機能を拡張し検索対象のクラスと、検索条件のインスタンス変数が定義されるクラスとは異なっていても可とすることとした。ただし、検索条件の定義されるクラスのインスタンスと検索対象のクラスのインスタンスとが何らかのリンクにより関係付けられていなければ検索することはできない。この拡張により、「広さ xx m2 以上の台所がある物件」といった検索が可能になった。

4. クラスの関係がリンクとして定義可能

属性・形状インスタンスリンクの追加定義・変更・削除・検索リンクに関しては、属性・形状のクラスでなく、root クラス (全てのクラスに共通の親クラス) において定義された変数を使用する。したがって、属性・形状の区別なくリンク操作を行うことができる。

5. 動的なリンクづけが可能

4. で述べた方式により同様に実現できる。

6. 動的なクラスの定義変更が可能

スキーマの動的な変更 (追加定義、変更、削除) は必然的に既存のデータベースの再構築を伴う。その場合に整合性を保ちつつ再構築する必要があるので若干の制約が生じる。追加定義は新たなクラスが増えるのみであるため、既存のインスタンスには影響を与えないため問題はないが、追加定義か削除かを判定することが困難なため、追加定義の場合もデータベースの再構築を行う。

変更に対しては、特にデータの大きさや型が変更されるため、既存のデータを保持することは困難である。従って、スキーマの動的な変更に対するデータベースの完全性は保証されない。

削除はデータそのものも削除されることを除けば問題がない。

また、スキーマはスキーマファイルにテキストとして記述するため、スキーマの変更はエディタにて行われる。さらに、検索はエディタを開いたうえでのエディタの機能により代替する。

5 プロトタイプシステムの概要

5.1 機能概要

作成したプロトタイプシステムの機能全体構成は、図 1 に示す通りである。各機能について、以下記述する。

1. システム終了機能
   データベース機能の終了

2. データベース初期化
   主記憶上のデータベースを初期化する。

3. データベースオープン
   データベースをオープンする。

4. データベースクローズ
データベースをクローズする

5. オブジェクト生成
指定クラスのインスタンスを生成する。インスタンス変数およびインスタンスのリンク付けは別コマンドで設定する。インスタンスに対しては、ガーベージコレクションは行わない。つまり、生成したインスタンスは陽に削除しない限り消滅しない。

6. オブジェクト格納
指定のインスタンスをセーブする。インスタンスにリンクづけられた関連インスタンスには影響しない。

7. オブジェクト削除
指定インスタンスを削除する。リンクづけられたインスタンスが存在する場合その関係を削除する。リンクづけられたインスタンスからさらに別のインスタンスにリンクしていてもそのインスタンスには影響しない。本コマンドの実行により、表示されているインスタンスにも自動的に再描画がかかる。また、削除できるインスタンスとして正規表現（＊, ？）を利用できる。

8. オブジェクト検索
指定されたクラスに属するインスタンスまたは指定されたリンクに接続するインスタンスの中で、検索条件を満たすものを検索する。検索条件として指定できるのは指定されたクラスのインスタンス変数に対する範囲指定のみである。

検索された結果はインスタンスの名称一覧として表示される。この処理は、指定した条件を満たすインスタンスを検索するものである。

ここでの条件とは、指定されたインスタンスクラスにおけるメンバ変数値によるものであるため、条件の指定としてメンバ変数の所属するインスタンスクラスと、実際の条件式が必要となる。

例　　クラス名 : room
　　条件式　 :　広さ > 15

検索結果出力法としては、条件に一致したインスタンスの名称や、出力インスタンスのクラス名を指定することによって、条件に一致したインスタンスにリンク付けされているインスタンスの名称さらに、出力インスタンスへのリンク名を指定することによって、条件に一致したインスタンスにリンク付けされているインスタンスの名称を出力する。

9. オブジェクト定義
指定インスタンスに対するインスタンス変数の値をセットする。具体的なインスタンスに対する値の設定はウィンドウを用いて行う。

10. オブジェクト属性表示
指定インスタンスの属性を表示する。指定インスタンスが図形データであった場合でも、属性データが表示される。

11. 形状表示指定インスタンスの図形データを表示する。指定インスタンスに子供としてリンクづけられた全ての図形データが表示される。指定インスタンスに図形データインスタンスがリンクされていない場合は何も表示されない。

12. リンク生成
インスタンス同志をリンク付けする。リンク付けはインスタンス１、インスタンス２の双方から行われる。

13. リンク削除
インスタンスをリンク名で指定されるリンクから削除する。リンク名、クラス名、インスタンス名には正規表現を使用できる。

14. リンク表示
指定インスタンスのリンク状態を表示する。

15. スキーマ更新
指定されたスキーマで新しいスキーマを生成し、mode により全インスタンスの再構築を行う。スキーマファイルが変更されていなくてもスキーマの update 並びにインスタンスの再構築が行われる。

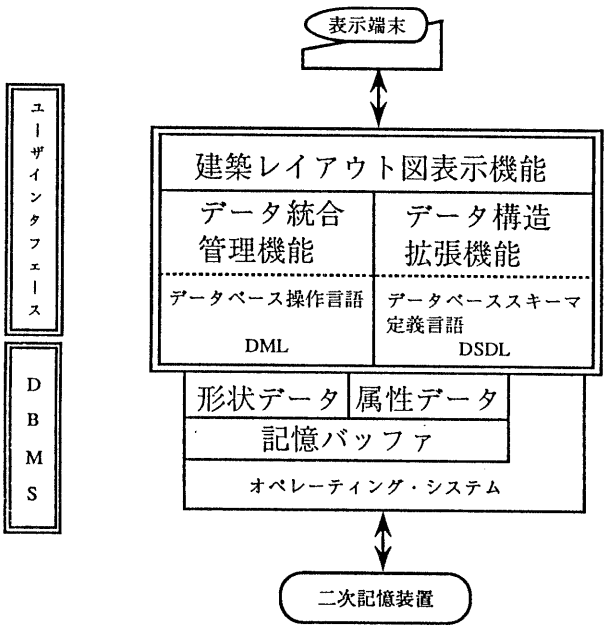

図1 プロトタイプシステムの機能

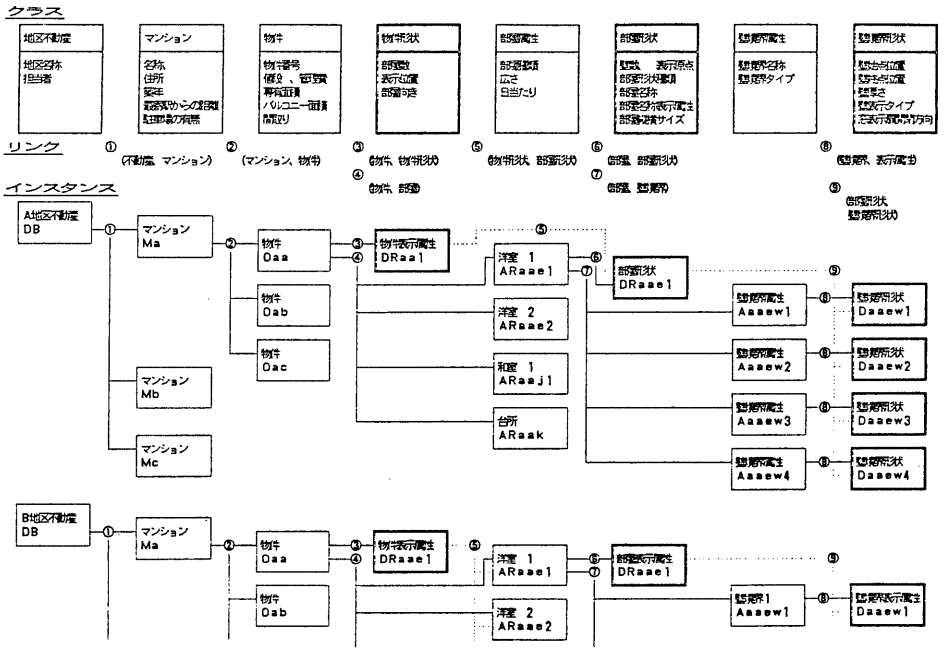

図2 建築CAD用データの論理構成

5.2 登録データ構成

本試作拡張可能データベースにおいて登録された、建築ＣＡＤ用データの論理構成例を図２に示す。登録論理構成は、マンションを対象とした、２次元の建築レイアウト図である。属性データを構成するクラスは、地区不動産クラス、マンションクラス、物件クラス、部屋属性クラス、壁境界属性クラスの５つのクラスである。属性データは、日本語を含む文字列情報と、実数、整数情報が格納されている。形状データを構成するクラスは、物件形状クラス、部屋形状クラス、壁境界形状クラスの３つのクラスである。形状データには、日本語を含む文字列情報と、実数、整数情報に加えて、レイアウト図を表示するための位置・形状情報（直線、円、円弧、文字列）が格納された。

5.3 開発環境

ハードウェア環境として、
SparcStation-1(OS SunOS4.03 + JLE 1.03)
で製作を行った。

ソフトウェア環境として、当初は記述言語として、クラス定義、継承等を持つ開発言語による実現も想定し、Ｃ＋＋やｇ＋＋によるインプリメントが検討されたが、今回は、Ｃ言語による方式を採用した。ユーザインタフェース機能の実現にあたっては、建築レイアウト図の表示や属性情報の表示・確認が行え、かつマウス等を用いてデータ操作言語の利用を画面から容易に直接行えることを前提に、ウィンドウマネージャとしてSunView を用いた。図３は、画面の表示例を示したものであるが、画面上は大別すると、

- コマンド選択ウィンドウ
- 図形表示ウィンドウ
- 属性表示ウィンドウ
- Ｓｈｅｌｌウィンドウ

の４つのウィンドウから構成されている。

6 今後の課題と展望

本試作において目標とした各実現機能毎に今後の課題を示す。

1) 形状データと属性データの統合管理機能
   - モデリング能力の拡充
   - 多彩な検索機能の追加
2) 階層的データベース構造の拡張機能
   - スキーマ言語記述における「入れ子表現」の実現
3) 建築レイアウト図の表示機能
   - 操作性の向上
   - グラフィカルなスキーマ編集操作の実現

その他、処理速度やメモリー効率の向上、スキーマやリンク単位のセキュリティ機能の実現、リンクが容易にたどれるようなブラウザの実現等実用的なＤＢＭＳが具備する機能の追加が今後の共通的な課題として想定される。

本試作においては、建築ＣＡＤを利用分野として想定したが、今回実現した機能群は、建築ＣＡＤ特有のものではなく、広く、エンジニアリングあるいはオフィス・オートメーション用のデータベースの基盤技術となるものであり、また、近年話題になっている拡張可能データベースやオブジェクト指向データベースの開発基盤として利用が期待される。

7 謝辞

本研究開発は、財団法人データベース振興センターより、「データベース構築促進・技術開発」の一環として委託を受け、実施したものである。なお、本開発にご協力いただいた千坂　文男氏および、貴重なご意見をいただいたデータベース振興センターの遠藤　博之課長、三菱電機　情報電子研究所、三菱総合研究所の皆様に深く感謝いたします。

〔参考文献〕

1. 山田、栗原、藤田:「建築設計支援システムＤＥＬＴＡ(全体概要)」、日本建築学会 第９回電子計算機利用シンポジウム、pp199-204、1987
2.「建築設計支援システムＤＥＬＴＡ(データベースの開発思想)」、日本建築学会 第９回電子計算機利用シンポジウム、pp211-216、1987

3. Kurihara,S. and Nishikawa,M.,et al.:MOdel and Database for Integrated Management of Architectual Objects,Proc. 2ns CIB/W78+W74 Seminar in Tokyo,Sept.1990.
4. Yamada,S.,Kurihara,S.,et al,: Architectual Design support System DELTA,Proc. 4th Int. Conf. on Computing in Civil and Building Engineering,July 1991 (in printing).
5. 宇田川、石川、市川、辻、西川「実体ー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」情処データベース・システム研究会研報　Vol.65、No.4 1988.5.19
6. 岡本、宇田川「ＣＡＤ用データベースにおけるレイアウト図について」情処データベース・システム研究会研報　Vol.83、No.2 1991.5.24

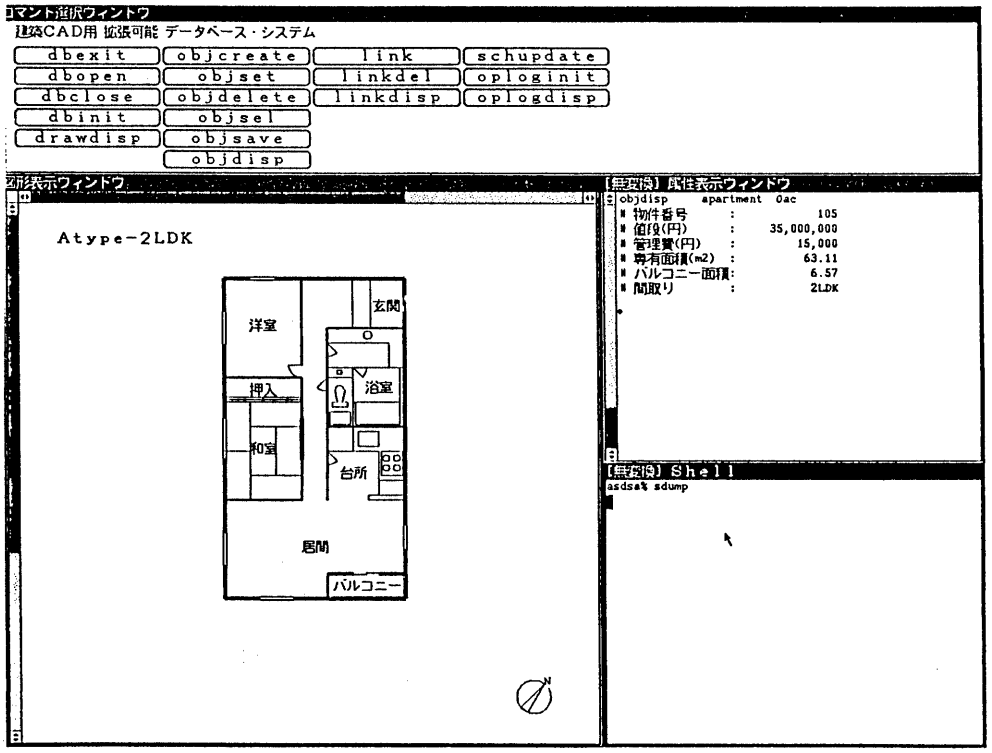

図3 プロトタイプシステムの画面例