CAD 支援を指向した複合対象抽象データ型の提案†

——ソリッド・データベースへの応用——

姜 世杰†† 大保信夫††† 山口和紀†††
北川博之††† 鈴木 功†††

近年，CAD データベースの重要性が幅広く認識されてきている．CAD 応用では，複雑な構造を持った対象，すなわち複合対象を扱う必要がある．これまで事務データ処理分野で広く利用されている関係データベースは，複合対象の格納・操作に必ずしも適していない．本論文では，関係データベースに複合対象抽象データ型の概念を導入し，複合対象の内部構造および挙動を統一的に記述するための枠組みを提案する．一般に，複合対象の木構造は複数の関係にまたがったタプルにより表現される．複合対象抽象データ型の下では，このような内部構造をユーザは直接意識することなく，応用目的に合わせて用意された関数／手続きを通じて，一貫性の保証された複合対象の操作を行うことができる．本論文では，各種基本概念を導入した後，CAD 応用例として3次元ソリッド・データベースを用いてその有用性を示す．また，本提案に基づく拡張関係データベースシステムを開発し現在評価中であるが，その実現方式についても述べる．

1. はじめに

近年，設計・生産の自動化を目指す CAD (Computer Aided Design) の研究が著しく進展するにつれて，製品設計・開発の効率化を高めることを目的としたCADデータベースの役割の重要性が幅広く認識されてきている[16),17)]．しかし，これまで定型的な事務データ処理分野において大きな成功をおさめた関係データベースシステムは，各属性のデータ型として単純なデータ型しか支援していないため，多様なデータ型を扱う CAD 応用に十分適用できていない[8),13)]．またデータの格納，操作単位はタプル，および関係であるため，設計対象を自然な形式でモデル化することが困難である．つまり，応用世界の論理単位としての設計オブジェクトは複数の関係の間にまたがるタプルの集まりにより表現されるが，このようなオブジェクトに関する検索・操作は応用プログラムの責任となるため，ユーザは極めて煩わしいデータ管理を行わなければならないことになる．

この問題の解決のために，我々は関係データベースに抽象データ型の概念を導入し，Dittrich[4)] の指摘にある構造的オブジェクト指向 (structural object oriented) と挙動的オブジェクト指向 (behavioral object oriented) を，統一的枠組みで支援するシステムの開発を行っている．本アプローチの特徴は以下の二点である．

（1） 既存の関係型の汎用 DBMS の拡張によるオブジェクト指向の支援

CAD データベースに対し，Smalltalk 80[6)]，CLOS[2)] 等オブジェクト指向言語に永続的データ管理能力を持たせたオブジェクト指向データベースの試み[1),3),15)]が，近年有力であるとみなされている．しかし，この種のシステムは，管理用データも含めた大量データに対する統合 DBMS としては，まだ理論的に解決すべき問題点が多い．一方，関係データベースを拡張することによりオブジェクト指向を支援する試みは，物理的構成に対する制御能力等に強い制限があるものの，既存の技術を十分に生かし，安定したデータ管理が可能である等の面で，特に管理情報を含めた統合データベースの実現にとっては有力な方法と思われる．

（2） 列抽象データ型，関係抽象データ型，複合対象抽象データ型の導入

CAD 応用に現れる多様なデータ型を支援する上で，抽象データ型の利用が有効であることは広く知られている．関係データベースの拡張としては，関係を抽象データ型として定義する関係抽象データ型 (Relation ADT)[19)]の提案がある．それに対し，INGRES を拡張した ADT-INGRES[10),20)] では，列抽象データ型 (Column ADT) の概念を導入した．すなわち，各属性に対する定義域として抽象データ型の定義を許

† Complex Object Abstract Data Type in CAD Database Environments —Its Application to Solid Databases— by SHI-JIE JIANG (Program in Engineering Sciences, Graduate School, University of Tsukuba), NOBUO OHBO, KAZUNORI YAMAGUCHI, HIROYUKI KITAGAWA and ISAO SUZUKI (Institute of Information Sciences and Electronics, University of Tsukuba).

†† 筑波大学大学院工学研究科
††† 筑波大学電子・情報工学系

し，汎用プログラミング言語による抽象データ型の実現を可能とした．筆者らはこの二つの提案を統合し，関係データベースの枠組みの中でより一般的に抽象データを扱う提案を行った[11]．これらの提案は挙動的オブジェクト指向の要求を満たすアプローチであると言えるが，構造的オブジェクト指向の要求である複合対象の支援は必ずしも十分ではなかった．一方，構造的オブジェクト指向を支援するアプローチとしてQUEL as a Datatype[21]がある．この提案は属性のデータ型として問い合わせ式 (query) を許し，各属性に書かれた QUEL 文で複数の関係にまたがるタプルを参照することにより，設計対象の構造表現に関してより簡明な参照方式を提供したが，データの生成，挿入が繁雑になること，参照一貫性の保持がユーザの責任になること，複合対象の中で柔軟に航行できないなどの問題点がある．本論文では，列抽象データ型，関係抽象データ型および複合対象抽象データ型 (Complex Object ADT) を統一的に扱う枠組みを持った関係データベースの拡張を提案する．列抽象データ型，および関係抽象データ型に関しては，問い合わせ言語により，そのデータ型を検索・操作するための関数を定義する．複数の関係からなる複合対象抽象データ型に関しては，各関係抽象データ型に付随する関数を使った，プログラミング言語記述を用いて複合対象の操作に必要な関数を定義する．また，列のデータ型として複合対象抽象データ型を許すことにより，複数の関係にまたがるタプルの集まりが表す複合対象を列の値として取れるようにした．このアプローチが QUEL as a Datatype と基本的に異なる点は，構造的オブジェクト指向のみならず挙動的オブジェクト指向の支援機能を持ち，CAD の設計対象である複合対象を抽象データ型として定義できる点である．特に，応用分野に特有な操作を複合対象抽象データ型ごとに付加することができるのは，QUEL as a Datatype に見られない特徴である．

本アプローチは現在 UNIX 上の 商用 DBMS である G-BASE☆ 上で設計・開発し，CSG (Constructive Solid Geometry) ソリッド・モデル[18]に基づく CAD 応用システムで実験・評価を行っている．

本論文の2章において，実際の CAD の例を用いて，本システムの概略を述べる．3章では，抽象データ型の導入について述べる．4章では，複合対象の記述モデルおよび抽象データ型の適用を説明する．また5章において，CSG ソリッドデータベースを実例として，関係データベースへの抽象データ型と複合対象の導入が CAD 統合データベース開発の一つの方向であることを示す．6章では，プロトタイプの実現方法について議論する．7章は，全体のまとめである．

☆ G-BASE は(株)リコーの登録商標である．

2. 拡張関係データベースシステム

この章では，本拡張関係データベースシステムの概要を述べる．図1のような機械部品を例として考えて見る．一般に機械や LSI などの設計対象となるものは，複数の部品の入れ子構造になった複合対象が多い．関係データベースにおいて，このような複合対象を単一のフィールドあるいは単一のタプルだけで表すことは困難であり，一般には，複数の関係にまたがる

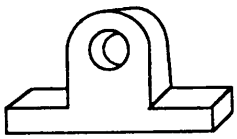


図 1 対象ソリッドオブジェクト例
Fig. 1 Solid object example.

mechanical-part

NO	DESIGNER	DATE	DESC
1001	Yamada	1/4/1989	10

geometric

ID	X	Y	Z	AX	AY	AZ	OBJECT
4	0	0	0	0	0	0	1
5	5	10	0	0	90	0	3
8	0	0	0	0	0	0	2
9	5	10	0	0	90	0	7
12	10	0	0	0	0	0	6
13	0	0	0	0	0	0	11

combined

ID	L_NO	OP	R_NO
2	4	or	5
6	8	dif	9
10	12	or	13

box

ID	X_LEN	Y_LEN	Z_LEN
1	10	5	10
11	30	5	3

cylinder

ID	XY_RADIUS	Z_LEN
3	5	5
7	2.5	5

cone

ID	XY_RADIUS	Z_LEN

図 2 関係データベースにおける表現
Fig. 2 Representation in relational database.

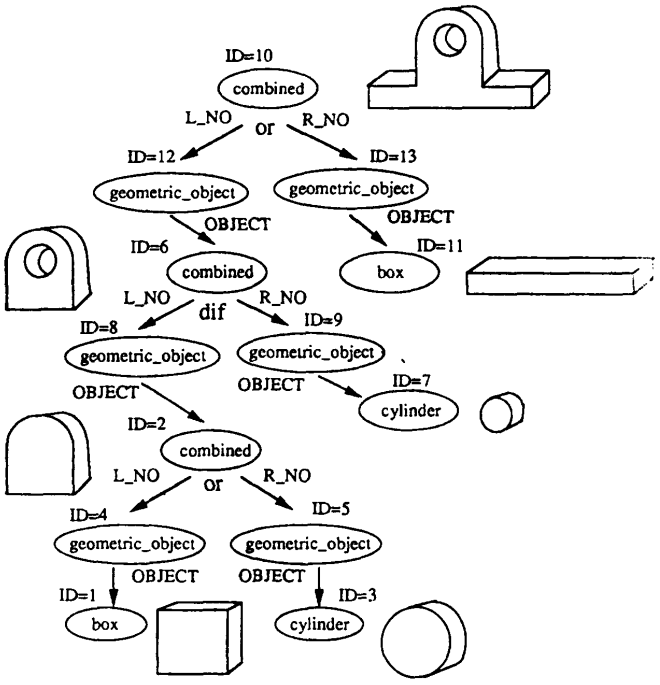


図 3 CSG 木による表現
Fig. 3 CSG tree representation.

タプルにより記述する．もちろん，表現の仕方は一意には決まらないが，図1に対する CSG 表現は，たとえば関係データベースでは**図2**のように表すことができる．図2のタプル間の関係をわかりやすく表すと**図3**のようになる．

この例でわかるように，CSG を表現するタプルは関係データベースの複数の表に散らばっており，CAD 応用ではこの例のように互いに参照関係にあるタプルの集合をまとまった処理単位として扱うことが要求されている．

現在我々が開発している拡張関係データベース管理システムは，これを次のようにして実現している．

（a） タプルで表現されたオブジェクトを抽象データ型として定義する機能

```
define ADT Cylinder :
(     supertype : Primitive
      instance-variable
       (   ID :              integer,
           XY_RADIUS :  real,
           Z_LENGTH :   real     )
      procedure create-cylinder(id xy-radius z-length)
          append to Cylinder( ID = id,
                              XY_RADIUS = xy-radius,
                              Z_LENGTH = z-length )
)
```

図 4 オブジェクト抽象データ型の定義例
Fig. 4 Example of an object ADT definition.

たとえば，図2に現れた関係 Cylinder の一つのタプルをある円柱の記述と見ることにより，関係 Cylinder を抽象データ型と考えることができる．そこで，この抽象データ型にインスタンスに対する操作を行う関数／手続きを付加する．このような関数／手続きは通常の問い合わせ言語を用いて，一般に複数の問い合わせ式により構成する．このように定義した抽象データ型を，オブジェクト抽象データ型と呼ぶ．**図4**は，抽象データ型 Cylinder の定義である．このような定義は他の抽象データ型の定義と同様システムに登録される．

（b） オブジェクト抽象データ型の集まりで構成される複合対象を抽象データ型として定義する機能

上の例からわかるように，CAD 応用における複合対象は，(a)のオブジェクト抽象データ型のインスタンスであるオブジェクトの集まりとみなすことができる．本システムでは，それを新しい複合対象抽象データ型と定義し，それに関する演算を定義する機能を提供する．たとえば，図8に CSG という複合対象抽象データ型が定義されている．その詳細は5.1節で説明するが，この複合対象抽象データ型は基本データ型 (integer, real, character-string) とオブジェクト抽象データ型 (Cylinder, Box, Cone, Object, Geometric_Object, Combined, Primitive) を用いて定義されている．

（c） 複合対象抽象データ型インスタンスを関係の属性値として利用可能にする機能

これは，データベースの値として通常の値だけでなく，複合対象抽象データ型のインスタンスも格納できるようにするもので，我々のデータモデルを特徴付ける機能である．たとえば，図2に現れる関係 mechanical-part の属性 DESC をスキーマ宣言において CSG 複合対象抽象データ型と定義したとする．すると，NO=1001 のタプルの DESC の欄は，関係 geometric, combined, box, cylinder, cone に属する ID=1 から ID=13 のタプルの集まりで表現される CSG 型の一つのインスタンスを表している．

（d） 複合対象指向操作定義を用いた対象指向アクセスインタフェース機能

複合対象抽象データ型のインスタンスに対する操作

関数／手続きは汎用の高級言語で記述する．このとき，複合対象を構成する各オブジェクトに関する操作は，問い合わせで定義されるオブジェクト抽象データ型の関数／手続きの呼び出しにより行う．このようにして定義された複合対象抽象データ型の関数／手続きは，通常の問い合わせ式の中で使うことが可能である．たとえば，番号 1001 の機械部品の設計に関与しているすべてのプリミティブを求めるには，複合対象抽象データ型 CSG に付随する関数 find-primitive を使用して，

```
range of t is mechanical-part
retrieve (find-primitive t.DESC)
where t.NO=1001
```

とすればよい．

3. オブジェクト抽象データ型

一般に，抽象データ型とは，データ構造とそのデータ構造を操作する関数／手続き群を一体化したものである．本論文では，CAD 世界の設計対象を記述するために，複合対象抽象データ型を導入するが，その定義はオブジェクト抽象データ型に基づき行う．この章では，まずオブジェクト抽象データ型について説明する．

3.1 ユーザ定義の抽象データ型

基本データ型は，整数，実数，文字列のようなものである．これに対し，オブジェクト抽象データ型を名前，内部データ構造，手続き／関数群の三つ組として定義する．オブジェクト抽象データ型の内部データ構造は，インスタンス変数の組 $(a_1, \cdots, a_n)$ により記述され，各インスタンス変数を属性に対応させることにより，通常の関係データベースの関係により表される．この内部データ構造としての関係はユーザからは直接見えない．各インスタンス変数 a_i には，その値の取りうるデータ型 T_i が付随する．T_i は基本データ型とオブジェクト抽象データ型の二種類に大別される．なお，外部入力・表示とこのような関係の間の変換ルーチンをその抽象データ型の定義者が明示的に書く必要はなく，システムが抽象データ型の定義処理モジュールを用いて自動的に変換を行う．関数は返り値として基本データ型の値，あるいはその集合を返すものとする．それに対して，手続きは返り値のない操作の系列である．関数も手続きと同様に副作用を持つことがある．関数／手続きの宣言は，引き数の型名，返り値の型名，および通常の問い合わせ式による本体の記述からなる．関数宣言では返り値を持つ問い合わせ式はたかだか一個しか許さない．オブジェクト抽象データ型に付随する関数／手続きの定義には QUEL 言語を用いる．ただし，通常の QUEL 言語と異なり，range 句のタプル変数の範囲を示す関係名を直接に書くかわりに，引き数として与えられた抽象データ型インスタンスを表すタプル識別子に型検査関数（システムが定義する．3.2 節で詳しく述べる）を適用する式を書くことができる．実行時には，この式がそのタプル識別子の代表するインスタンスの型に対応する関係名に置き変わる．図 4 に Cylinder というオブジェクト抽象データ型の定義が示されている．Cylinderの定義においては，すべてのインスタンス変数は基本データ型を持っている．この定義の中に現れる supertype 項はオプショナルな項目であり，抽象データ型間の汎化関係を記述する．この例においては，Primitive が Cylinder の親として指定されている．インスタンス変数の型として他の抽象データ型が現れる例としては，**図 5** に Combined の定義が示されている．そのインスタンス変数 L_OBJECT と R_OBJECT は，Geometric_Object という他のオブジェクト抽象データ型を持つインスタンスの識別子をその値として取る．以上のような定義を行うと，このデータ構造の定義と手続き・関数群全体が一つのブラックボックスとなり，外からは次のものだけを見ることができる．

- オブジェクト抽象データ型の手続き／関数の名前
- 手続き／関数が受け付ける引数の数と型名
- 関数の返り値の型

3.2 Generic 関数／手続き

前節で述べたが，ユーザ定義のオブジェクト抽象データ型の関数／手続きは QUEL 言語で定義されている．オブジェクト抽象データ型のインスタンス変数は型ごとに異なるため，その内部構造としての関係データベース中の関係も型に依存する．したがって，

```
define ADT Combined:
(   supertype : Object
    instance-variable
      ( ID :            integer,
        L_OBJECT :      Geometric_Object,
        OP :            Set_Op,
        R_OBJECT :      Geometric_Object )
    procedure creat-combined(id left op right)
        append to Combined( ID = id,
                            L_OBJECT = left,
                            OP = op,
                            R_OBJECT = right )
)
```

図 5 属性の型に抽象データ型を持つオブジェクト抽象データ型の定義例

Fig. 5 Example of the definition of an object ADT with ADTs as its attributes.

```
find-attribute(tuple-id attribute)
  range of t is type-of(tuple-id)
  retrieve t.attribute
  where t.ID = tuple-id
```

図 6 ユーザ定義抽象データ型の Generic 関数／手続きの例
Fig. 6 Example of a generic function/procedure of user-defined ADT.

類似した処理を行う関数／手続きも関係ごとに定義する必要があるが，しばしば問い合わせ式の相異はその関係の名前と属性名による差異にすぎない．このような場合，関係名と属性名を引き数として扱えるようにすることにより，複数のオブジェクト抽象データ型に適用できる関数／手続きを定義することが可能である．これらを Generic 関数／手続きと呼ぶ．**図 6** に示しているのは，このような Generic 関数／手続きの例である．Generic 関数／手続きは，通常のユーザ定義オブジェクト抽象データ型の関数／手続きと同様に扱われる．

3.3 抽象データ型識別子

オブジェクト抽象データ型の内部構造は関係で実現され，そのインスタンスはそれぞれタプルで表される．各インスタンスを識別するためには，識別子が必要になる．我々は内部識別子の管理機構を導入し，識別子の生成，削除およびそれによる型検査を自動的にシステムが行えるようにした．すべてのオブジェクト抽象データ型のインスタンスが生成されるときに，システム内で唯一な識別子が割り当てられる．一度使われたことのある識別子はそのインスタンスが消されても，二度と使われることがない．インスタンスをコピーするときも新しく生成するインスタンスには新しい識別子を割り当てる．しかし，これはあくまでもシステム内部のことであり，一般ユーザは特に意識する必要はない．

4. 複合対象とその記述モデル

本章では，3章で導入したオブジェクト抽象データ型に基づいて複合対象を導入する．

4.1 複合対象

2章の例で述べたように，一般に機械設計などの対象となるものは，全体と部品の関係 (part of) によって階層的に表現できる．おのおののオブジェクトをグラフのノードで表現すると，上位階層にあるノードはその下位階層の幾つかのノードとアークで結ばれ，このような関係が何重にも繰り返された結果，構造的に木 (Tree) となる．現実には，共有される部品もあるので DAG (Directed Acyclic Graph) となることもあるが，本論文では簡単化のため木に限定して話を進める．我々はこのような木構造で表現できる形のオブジェクトの集まりを複合対象と呼ぶ．たとえば，図1の中に現れたソリッドオブジェクトを複合対象として考えると，**図 7** に書いたようにオブジェクトA，B，c，D，e，f，gが複合対象を構成している．オブジェクトAはB，cより構成され，オブジェクトBはD，e，オブジェクトDはf，gから構成されている．

複合対象を構成する対象を記述するために，我々は3章でオブジェクト抽象データ型の概念を導入した．このようなオブジェクト抽象データ型の集合を T_o，基本データ型の集合を T_p とする．各 $t \in T_o$ に対し，t の構造は $\Pi(T_o \cup T_p)$ の要素である．ただし，集合 A に対し，ΠA は A の要素からなる任意の組全体の集合を表す．すなわち，$\Pi A = \{(a_1, \cdots, a_n) | n \geq 0 \wedge a_1 \in A \wedge \cdots \wedge a_n \in A\}$，たとえば，Combined$\in T_o$ に対しその構造は (integer, Geometric_Object, Set_Op, Geometric_Object) である．複合対象は，ある $S \in T_o$ を木構造のルート型として定義し，この複合対象を構成するオブジェクトがインスタンス変数の値として参照するオブジェクトの構造をたどることにより全体の構造を表現する．この展開は，次のようにとらえることができる．すなわち，ある複合対象を構成するオブジェクト抽象データ型 $t \in T_o$ の構造が $(T_1, \cdots, T_n) \in \Pi(T_o \cup T_p)$ のとき，$t_i \in T_i$ $(1 \leqq i \leqq n)$ のすべてに対し，$t \rightarrow t_1 t_2 \cdots t_n$ を生成規則Pの要素，T_o を変数の集合，T_p を終端記号の集合，S を開始記号と見ることにより，自然に CFG (文脈自由文法)[9]を構成していると見ることができる．ここでは導出木の各ノードは複合対象の構成要素であるオブジェクト抽象データ型のインスタンスに対応することとする．これにより複合対

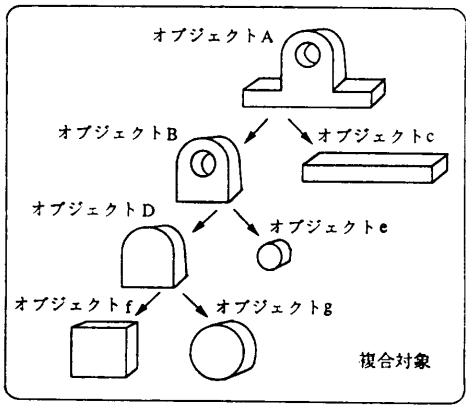


図 7 複合対象の例
Fig. 7 Example of a complex object.

象のインスタンスの構造は，この CFG の導出木に対応する木構造となる．ある変数から終端記号列が生成できない場合，その変数は“不用”というが，不用な変数を決定する手続きが知られているので，それを複合対象を構成するオブジェクト抽象データ型に適用することにより，オブジェクト抽象データ型間の整合性を調べることができる．このように見ることにより，複合対象がオブジェクトにより正しく定義されているかを CFG の理論を活用し検討することができる．3.1 節のオブジェクト抽象データ型で述べた supertype による汎化関係に従い，あるデータ型 A が現れるところにそのサブデータ型 $B_1, B_2, \cdots$ が現れても良い．これを CFG による解釈にあてはめると，A が $B_1, B_2, \cdots$ の supertype であるという汎化関係を定義するということは，生成規則Pに $A \rightarrow B_1, A \rightarrow B_2, \cdots$（略記として，$A \rightarrow B_1 | B_2 \cdots$）を追加することに相当する．

4.2 複合対象抽象データ型の定義

ある複合対象を構成する対象に対する操作は，その対象を記述するオブジェクト抽象データ型に付随した関数／手続きを呼び出すことにより実行できる．たとえば，オブジェクト抽象データ型 A が構造（$T_1, T_2, \cdots, T_n$）および関数／手続きの組（$f_1, f_2, \cdots, p_1, p_2, \cdots$）により定義されているとする．$A$ の関数／手続き f_i/p_i は，$T_1, T_2, \cdots, T_n$ の中から必要なオブジェクト抽象データ型の関数／手続きを実行する．しかし，このようなオブジェクト抽象データ型の関数／手続きは，問い合わせ言語で記述するためその記述が限定されており，複合対象に対する複雑な操作をそれら関数／手続きの組み合わせのみで実現するのは不可能である．複合対象の全体を操作の単位として扱うために，我々は複合対象を抽象データ型として定義し，それに対する操作関数／手続きを複合対象抽象データ型に付随させる機構を提案する．

複合対象抽象データ型は次の形式により定義する．

```
define Complex-ADT Name(
    structure-spec={
      T_p=(T_p1, T_p2, ..., T_pu)
      T_o=(T_t1, T_t2, ..., T_tv)
      S=T_r},
    function f_1(root a_1a_2...a_i) return r_1
      ......
    procedure p_1(root b_1b_2...b_j)
      ......)
```

ここで，Name は複合対象抽象データ型の名前であり，$T_{px}(x=1, \cdots, u)$，$T_{ty}(y=1, \cdots, v)$ はそれぞれ基本データ型名，オブジェクト抽象データ型名である．また，$T_r \in T_o$ はこの複合対象抽象データ型のインスタンスのルートとなるオブジェクト抽象データ型を示す．ここで，注意してほしいのは抽象データ型の定義の中に再帰的な部分と選択（alternative）の部分がありうるのでインスタンスの構造が異なることがある．たとえば，ある複合対象抽象データ型の定義が $\{A \rightarrow B|a,\ B \rightarrow Cb,\ C \rightarrow A\}$ のような生成規則からなるならば，**図 8** に現れた導出木はすべてこの抽象データ型のインスタンスだと考えられる．複合対象抽象データ型の各関数および手続きの定義は本論文では Lisp 言語により記述する．それぞれの関数／手続きの先頭の引き数のデータ型は，その複合対象抽象データ型のルートに当たっているオブジェクト抽象データ型と一致しなければならない．この制約に基づき，関数／手続きの型検査を行う．複合対象抽象データ型の関数／手続きの定義においては，その複合対象を構成するオブジェクト抽象データ型の関数／手続きを使用して検索や操作を行う．このように，データベースへのアクセス部分，つまり，各オブジェクト抽象データ型のインスタンスに対する操作を既存の問い合わせ言語によって効率的に行い，オブジェクト抽象データ型から構成される複合対象に関する操作（比較的複雑な構造を対象とする）を Lisp 言語で処理することにより，情報隠蔽と柔軟性の実現が図られる．

上で述べたように，複合対象に対する関数／手続きを定義するときに，複合対象を構成する各オブジェクト抽象データ型に付随している関数／手続きを使うことができ，結果として問い合わせ言語を呼び出すことが可能となっている．逆にオブジェクト抽象データ型に付随している関数／手続きに対しては Lisp 言語での記述は許していない．また，問い合わせ言語と Lisp 言語を一つの関数／手続き中に混在することは許していない．これらの実現は不可能ではないが，本論文で検討した例では必要性がなかったので，効率上

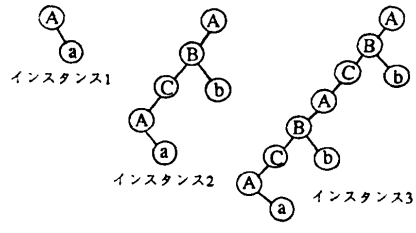


図 8 同一複合対象抽象データ型のインスタンスの例
Fig. 8 Example of instances of the complex-ADT.

の問題などから許さないことにした.

4.3 複合対象定義域の導入

4.2 節で定義した複合対象をデータベースの値として扱うために，システムに用意された基本データ型以外に，複合対象抽象データ型を定義し，関係の属性定義域として使用できるようにする．つまり，一般に，関係 R は次のように定義される.

$$R \subseteq G_1 \times \cdots \times G_m$$

ここで，$G_i = D_i$ or $A_i (i = 1, \cdots, m)$ であり，D_i は単純定義域であり，A_i は複合対象抽象データ型定義域である．たとえば，機械の設計データを記述するための関係は次のように宣言される.

```
create mechanical-part(
    NO=integer,
    DESIGNER=string,
    DATE=string,
    DESC=Complex-ADT(CSG))
```

この関係においては，属性 NO, DESIGNER, DATE はそれぞれ基本データ型を持っているので，通常のデータ操作を行える．属性 DESC は CSG という複合対象抽象データ型を持ち，そのインスタンスは普通の値のように直接参照されることはない．これらのインスタンスに対する操作は，複合対象抽象データ型 CSG に属す各手続きおよび関数を介してのみ可能である.

5. 複合対象のデータ操作

本章では，複合対象のデータ操作について述べる．実例としては，機械系 CAD モデルの重要な構成要素である幾何モデルを表現することを考える．幾何モデルには多くのものが提案されているが，ここでは，三次元形状の構成方式を Boolean 木で表現する CSG 法を例にとる.

5.1 複合対象抽象データ型定義の例—CSG

ここでは，CAD の重要なモデリング技法の一つである CSG を例として，その記述について考えてみることにする．実世界のソリッドオブジェクトを，基本オブジェクトの組み合わせによって表現するのが，CSG の考え方である．このような組み合わせは正準集合演算を用いて木構造で表現される．典型的な例を図 3 に示す．そのスキーマ表現は CFG で表すことができる.

```
⟨Geometric_Object⟩→⟨Transf⟩⟨Object⟩
⟨Object⟩→⟨Combined⟩|⟨Primitive⟩
⟨Combined⟩→⟨Geometric_Object⟩⟨Set_Op⟩
        ⟨Geometric_Object⟩
⟨Primitive⟩→Box|Cylinder|Cone|…
⟨Transf⟩→Real Real Real Real Real Real
⟨Set_Op⟩→or|and|dif
```

図 5 に Combined の定義の例を示した．CSG の構成オブジェクト抽象データ型の間に現れる汎化 (is a) 関係と統合 (part of) 関係は**図 9** に示されている.

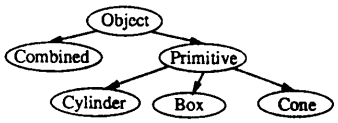
Lee と Fu[14] は CSG 複合対象を統一的に記述する関係データベーススキーマを提案した．しかし，そのスキーマでは複数の関係に属するタプルの集合で一つの対象を表現するため，CSG に対する操作をユーザが行おうとしたとき，データ操作記述が繁雑となり，また誤りを生じやすい．CSG 複合対象を上で示したように抽象データ型として定義すれば，ユーザは複合対象がいかに複数の関係に格納されるかを考慮することなしに，抽象データ型の関数／手続きを用いて CSG に対するアクセスを行うことができる．CSG 複合対象抽象データ型の定義例を，**図 10** に示す．各関数／手続きについては 5.2 節の操作例の中で説明する.

5.2 複合対象のデータ操作例

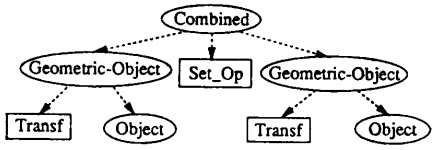
以下では，CSG 複合対象の操作例を示す．本システムにおいては，データ操作の記述には，QUEL 言語を拡張した ADT-QUEL を用いる．6 章で説明するように，本システムでは QUEL 言語は Lisp の S 式に展開されるので，クオートが必要になっている.

5.2.1 検　　索

まず簡単な複合対象に対する検索操作の例を説明する．ある機械部品の設計に用いられるすべてのプリミ



CSGオブジェクトの汎化階層(is a)



CSGオブジェクトの統合(part of)

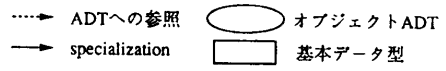


図 9 抽象データ型の階層
Fig. 9 ADT hierarchy.

```
define Complex-ADT CSG(
        structure-spec = (
            Tp = ( integer, real, character-string )
            To = ( Cylinder, Box, Cone,
                   Object, Geometric_Object, Combined,
                   Primitive )
            S = Object ),
        function find-primitive(root) return set-value
            (defun find-primitive(root)
              (if (member (type-of root) '(Cylinder Box Cone))
                  (find-all-attributes root)
                  (find-combined root)))
            (defun find-combined(root)
              (union (find-primitive (find-attribute
                                      (find-attribute root 'L_OBJECT)
                                      'Object))
                     (find-primitive (find-attribute
                                      (find-attribute root 'R_OBJECT)
                                      'Object))))
        function initial-insert(primitive-name xx yy zz &aux new-id) return CSG
            (defun initial-insert(primitive-name xx yy zz)
              (initialize)
              (setq new-id (new-id primitive-name))
              (case primitive-name
                (Cylinder (create-cylinder new-id xx zz))
                (Box (create-box new-id xx yy zz))
                (Cone (create-cone new-id xx zz)))
              new-id)
        function copy(root-id) return CSG
                .....
        procedure delete(root-id)  # delete a CSG instance with root-id as its root id #
                .....
        procedure modify(tree1-root-id typename1 attr-name1 attr-value1
                         tree2-root-id typename2 attr-name2 attr-value2)
                .....
        function add(tree1-root-id geo-para1 op tree2-root-id geo-para2) return CSG
          (defun add(tree1 geo-para1 op tree2 geo-para2 &aux id1 id2 id3)
            (setq id1 (new-id 'Combined))
            (setq id2 (new-id 'Geometric_Object))
            (setq id3 (new-id 'Geometric_Object))
            (create-combined id1 (create-geometric id2 geo-para1 tree1)
                             op (create-geometric id3 geo-para2 tree2))
            id1)
        function add-primitive(tree-root-id geo-para1 op prim-name x y z geo-para2)
                  # add primitive (as constant) to a tree #
          (defun add-primitive(tree-root-id geo-para1 op prim-name x y z geo-para2)
            (setq id1 (new-id 'Combined) id2 (new-id prim-name)
                  id3 (new-id 'Geometric_Object) id4 (new-id 'Geometric_Object))
            (create-combined id1 (create-geometric id3 geo-para1 tree-root-id)
                             op (create-geometric id4 geo-para2
                                  (case prim-name
                                    (Cylinder (create-cylinder id2 x z))
                                    (Box (create-box id2 x y z))
                                    (Cone (create-cone id2 x z))))))  )
```

図 10 複合対象抽象データ型 CSG の定義
Fig. 10 Definition of the complex object ADT "CSG."

ティブオブジェクトを検索することは，もっとも代表的な CSG に対する操作の一つである．この操作は図10で定義した複合対象抽象データ型の関数 find-primitive を用いて，次のように表せる．

```
range of t is mechanical-part
  retrieve (find-primitive t.DESC)
  where t.NO=1001
```

その検索結果としては，複合対象のすべてのプリミティブ構成要素が得られる．この例で示されたように，複合対象定義域を持つ属性に対しては，その複合対象抽象データ型の関数を適用することができる．

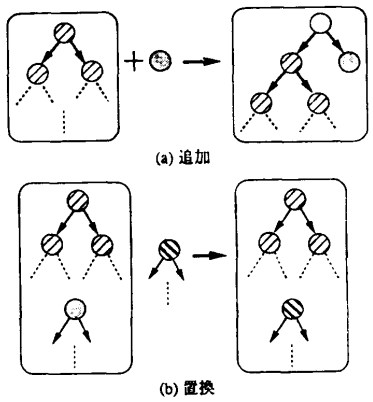


図 11 CSG 木の更新
Fig. 11 Update of the CSG tree.

5.2.2 更　　新

ここで，CSG 複合対象に対して，二つの基本更新操作例を示す．まず一つは，複合対象に対して，あるプリミティブなオブジェクトを追加する操作である．**図 11**（a）に示したように，この操作は CSG の木構造の階層の深さを一つ増加し，CSG を表す関係にいくつかのタプルを追加することを伴う．また，それと同時に，ルートを変えなければならない．この一連の操作を一つの手続きで実現するため，抽象データ型 CSG の手続き add-primitive がある．add-primitive を用いると，次の問い合わせ式によって，この操作をわかりやすく記述できる．

```
range of t is mechanical-part
retrieve (add-primitive
  t.DESC '(0 0 0 0 0 0) 'and
  'Cylinder 10 10 5 '(1 1 1 0 0 0))
where t.NO=1001
```

もう一つの例は，ある複合対象のある部分木をほかの複合対象の部分木で置き換える操作である．複合対象抽象データ型の手続き modify を用いて，次の式でこの操作を記述できる．

```
range of t1 is mechanical-part
range of t2 is mechanical-part
retrieve (modify t1.DESC 'Geometric_Object
                         'Transf '(0 0 0 0 0 0)
                 t2.DESC 'Geometric_Object
                         'Transf '(0 0 0 0 0 0))
where t1.NO=1001
and t2.NO=1002
```

modify の中では，まず複合対象 1001 に対し，型が Geometric_Object であり，なおかつインスタンス変数 Transf の値が (000000) であるようなオブジェクトを検索する．この条件に合致したものを，型が Geometric_Object であり，インスタンス変数 Transf の値が (000000) である複合対象 1002 のオブジェクトで置き換える (図 11(b))．

5.2.3 生成と削除

一つの CSG 複合対象はいくつかのオブジェクトの集まりで構成される．このような複合対象は，逐次にその複合対象を構成するオブジェクト抽象データ型のインスタンスを追加していくことにより構築できる．まず最初に，次のような式で CSG 複合対象を生成する．

```
append to mechanical-part(
    NO=1001,
    DESIGNER=Yamada,
    DATE=10/6/1989,
    DESC=(initial-insert 'Cylinder 10 10 5))
```

すでに存在している CSG 複合対象に対し，add-primitive あるいは add を用いることにより，さらにオブジェクト抽象データ型のインスタンスを付け加えることができる．

```
range of t1 is mechanical-part
range of t2 is mechanical-part
retrieve (Add t1.DESC '(000000) 'and
              t2.DESC '(111000))
where t1.NO=1001
and t2.NO=1003
```

CSG 複合対象のインスタンスは，上に述べたような方式でそれを生成したタプルの属性値として存在する．そのタプルが消されるときには，複合対象抽象データ型の delete 手続きが実行され，それに伴いその複合対象を構成するオブジェクト抽象データ型のインスタンスを表すタプルの集まりが削除される．

6. システムの実現

現在，我々は SUN 3/260 上の商用関係データベース管理システム G-BASE のもとで，本論文で述べた拡張関係データベースを開発中である．G-BASE は UNIX の上で稼働し，データベースにアクセスするための関数を含む LISP 処理系 DM-LISP[12]を提供している．インタプリタ方式の実現言語を採用することにより，応用分野から頻繁に出てくる新しい要求に対応できる柔軟性とエラー処理の容易さを実現している．

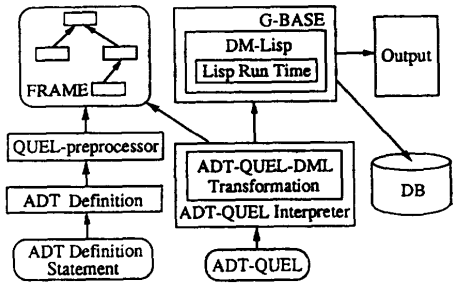


図 12 プロトタイプの構成
Fig. 12 Prototype system architecture.

図 12 に本システムのプロトタイプの構成を示す．QUEL 言語で書かれたオブジェクト抽象データ型の関数／手続きは QUEL-Preprocessor により対応する Lisp の形式に変換される．複合対象抽象データ型の関数/手続きも Lisp の形式で格納し，必要なときに DBMS にロードし実行する．ADT-QUEL の問い合わせ文の構文解析を行うために，"ADT-QUEL Interpreter" モジュールを導入した．このモジュールにより，通常の検索文はそれに対応する LISP の S 式表現に変換される．検索文の S 式を DML の S 式に変換する "ADT-QUEL-DML Transformation" では，抽象データ型の関数定義を参照し，型検査を行い，そのソースコードをロードする．LRT (Lisp Run Time) は，従来の G-BASE のデータ操作言語 DML の S 式表現を DM-LISP の形式に変換する DM-LISP で記述された関数群である．LRT が抽象データ型の関数をも解釈・実行することができるよう，LRT にも必要な機能拡張を行った．変換された DM-LISP 形式を評価することにより，データベースへのアクセスが実行され，さらに抽象データ型の関数がその結果に適用される．

7. おわりに

本論文では，CAD の設計オブジェクトとして現れる複合対象に関する格納・操作を支援するために，抽象データ型を導入した関係データベースの拡張を提案した．ここでの複合対象は，複数の関係の間に広がったタプルの集まりからなる木構造である．このような構成関係の表現のため，オブジェクト抽象データ型と複合対象抽象データ型を導入した．一つの複合対象は一般に複数のオブジェクトから構成される．オブジェクト抽象データ型は各対象の単位で抽象化の機能を提供するのに対して，複合対象抽象データ型は対象の集

まりを論理単位として，抽象化の枠組みを提供する．このように，二段階の抽象化により，従来の関係データベースの操作単位と CAD の分野における論理単位とのギャップを埋めることが可能となった．オブジェクトの抽象化は，問い合わせ言語による比較的単純な機構で実現でき，最適化などの可能性を残している[22]．一方，複合対象抽象化では，さらに一般的なプログラミング言語の制御機構である判定，代入および循環ループを用いて，複雑な制御を実現することができる．複合対象を定義する言語を変更する必要がある場合，従来は DBMS と問い合わせ言語が分かちがたく結合していたため，DBMS の再設計が必要であった．これに対し，我々は拡張可能 DBMS を構築する方法論 MODUS[5] を開発中である．これにより複合対象の定義言語そのものを応用の要求により変更することが可能となる．

CAD による設計の進み方は決して直線的ではなく，多くの試行錯誤が繰り返される．そのため，データベース設計時に恒久的と見なされ，スキーマ情報として表現されたデータベースの意味構造の枠を越える変更の要求も起こりやすい．その場合でも，我々のアプローチを用いれば，抽象データ型の関数／手続きの追加だけで変更要求に対処でき，計算コストの非常に高いデータベースの再構成を避けることができる可能性を高めることができる．この場合の方法論については，適用例を増やしながら，検討を重ねる予定である．

残された問題として検索効率などがある．特に，抽象データ型を関係データベースに導入することに伴って，新しいユーザ定義関数に対応できる効率的なアクセス方法が要求される．多次元空間検索では，その効率的アクセスを支援するスペイシャルインデクシングに関する提案[7]が最近活発である．このようなインデクスは応用領域に依存するので，具体的対象を設定して研究することが必要である．今後，我々はグラフィクスを応用分野として，一層深く研究する予定である．

謝辞 本稿を書くにあたり，多くの御指導，助言をしてくださった筑波大学電子・情報工学系教授藤原譲先生をはじめとする，協力をしていただいたデータベース研究室の諸氏に感謝の意を表する．

参考文献

1) Banerjee, J., Chou, H. T., Garza, J. E., Kim, W., Woelk, D. and Ballow, N.: Data Model Issues for Object-Oriented Applications, *ACM Trans. Office Inf. Syst.*, Vol. 5, No. 1, pp. 3-26 (1987).
2) Bobrow, D., Kiczales, G. et al.: Common Lisp Object System, ANSI X3J13 88-001 (1988).
3) Copeland, G. and Maier, D.: Making Smalltalk a Database System, *Proc. ACM SIGMOD*, pp. 316-325 (1984).
4) Dittrich, K. R.: Object-Oriented Database Systems: The Notion and Issues, *Proc. Int. Work. Object-Oriented Database Systems*, pp. 2-6 (1986).
5) 古瀬一隆，于　旭，山口和紀，北川博之，大保信夫，藤原　譲：拡張可能 DBMS MODUS のアーキテクチャ，第 39 回情報処理全国大会論文集，pp. 947-948 (1989).
6) Goldberg, A. and Robson, D.: *SMALLTALK-80, The Language and Its Implementation*, Addison-Wesley, Reading, Massachusetts (1983).
7) Guttman, A.: R-trees: A Dynamic Index Structure for Spatial Searching, *Proc. ACM SIGMOD*, pp. 47-57 (1984).
8) Hardwick, M. and Spooner, D. L.: Comparison of Some Data Models for Engineering Objects, *IEEE Comput. Gr. Appl.*, Vol. 7, No. 3, pp. 56-66 (1987).
9) Hopcroft, E. and Ullman, J. D.: *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Reading, Massachusetts (1979).
10) James, J., Fogg, D. and Stonebraker, M.: Implementation of Data Abstraction in the Relational Database System INGRES, *ACM SIGMOD Record*, Vol. 14, No. 1, pp. 1-14 (1984).
11) Jiang, S. J., Kitagawa, H., Ohbo, N., Suzuki, I. and Fujiwara, Y.: Abstract Data Types in Graphics Database, *Proc. IFIP TC-2 Working Conference on Visual Databases*, Tokyo, pp. 239-255 (Apr. 1989).
12) (株)リコー編：DM-Lisp マニュアル，東京 (Sep. 1988).
13) Kemper, A. and Wallrath, M.: An Analysis of Geometric Modeling in Database Systems, *ACM Comput. Surv.*, Vol. 19, No. 1, pp. 46-91 (1987).
14) Lee, Y. C. and Fu, K. S.: Integration of Solid Modeling and Database Management for CAD/CAM, *Proc. 20th Design Automation Conf.*, pp. 367-373 (1983).
15) Maier, D., Otic, A. and Purdy, A.: Development of an Object-Oriented DBMS, *Proc. ACM Conf. OOPSLA*, pp. 472-482 (1986).

16) 大保信夫：CAD/CAM のデータベース，CAD/CAM 技術 (bit 別冊)，国井利泰編，pp. 46-61，共立出版，東京 (1985).
17) 大保信夫：CAD データベースの動向，アドバンストデータベースシステムシンポジウム論文集，pp. 73-82 (1988).
18) Requicha, A. A. G. and Voelcker, H. B.: Constructive Solid Geometry, Tech. Memo. 25, Production Automation Project, Univ. Rochester, Rochester, N. Y. (Nov. 1977).
19) Rowe, L. and Shoens, K.: Data Abstraction, Views and Updates in RIGEL, *Proc. ACM SIGMOD*, pp. 71-81 (June 1979).
20) Stonebraker, M., Rubenstein, B. and Guttman, A.: Application of Abstract Data Types and Abstract Indices to CAD Data, *Proc. Engineering Design Applications of ACM-IEEE Database Week*, San Jose, pp. 107-114 (May 1983).
21) Stonebraker, M., Anderson, E., Hanson, E. and Rubenstein, B.: QUEL as a Data Type, *ACM SIGMOD Record*, Vol. 14, No. 2, pp. 208-214 (1984).
22) Yamaguchi, K. and Kunii, T. L.: PICCOLO Logic for a Picture Database Computer and Its Implementation, *IEEE Trans. Comput.*, Vol. C-31, No. 10, pp. 983-996 (1982).

(平成元年8月15日受付)
(平成2年5月 8 日採録)

姜　世杰 (正会員)

1961 年生. 1984 年中国科学技術大学計算機科学技術系卒業. 1988 年筑波大学大学院修士課程修了. 現在，同大学大学院博士課程工学研究科に在学中. データベースシステム，CAD に興味を持つ. ACM 学生会員.

大保　信夫 (正会員)

昭和 20 年 6 月 21 日生. 東京大学理学部卒業. 理学博士. 筑波大学電子・情報工学系勤務. 研究テーマ：データベースシステム.

山口　和紀 (正会員)

1956 年生. 1979 年東京大学数学科卒業. 1981 年東京大学理学部助手. 1985 年東京大学より理学博士の学位取得. 1989 年筑波大学電子情報工学系講師. データベース，コンピュータグラフィックスに興味を持つ. ACM, IEEE 各会員.

北川　博之 (正会員)

1955 年生. 1978 年東京大学理学部物理学科卒業. 1981 年同大学院理学系研究科博士課程退学. 日本電気(株)勤務を経て，1988 年筑波大学電子・情報工学系講師. 現在，助教授. 理学博士. データベースシステム構成法，エンジニアリングデータ管理，ソフトウェア開発支援システムなどに興味を持つ. 著書「The Unnormalized Relational Data Model」(Springer-Verlag). ACM, IEEE-CS, コンピュータグラフィックス学会各会員.

鈴木　功 (正会員)

昭和 8 年生. 昭和 32 年東京大学理学部化学科卒業. 昭和 37 年同大学院化学系研究科博士課程修了. 理学博士. 同年ミネソタ大学博士研究員. 昭和 39 年東京大学理学部助手. 以後，同大学理学部講師，教育用計算機センター講師，助教授を経て，昭和 53 年から筑波大学電子・情報工学系教授. データベースシステム，情報検索等の研究に従事. 人工知能学会，日本教育工学会，日本化学会，日本科学教育学会各会員.