データベースにおける木構造のモデル化の提案

7J-3

-複合オブジェクト支援機構の設計の一貫として-

土屋 直人* 山口 和紀** 大保 信夫** 北川 博之**
(*筑波大学理工学研究科 **筑波大学電子情報工学系)

1.序論

データベースの普及に伴い、CAD/CAM,AI,OIS等の分野を対象としたデータベースを支援する研究が活発に行われている。従来のデータベースシステムは、各属性のデータ型として事務計算で扱うような単純な型のみを支援している。しかし、これらの分野では、扱う対象が複雑な構造を持つことが多く、通常、複数のタプル間に渡る複合オブジェクトとして表現せねばならない。したがってシステム側にもそれに対応する支援機構が望まれる。この問題を解決するために、RDBMSに抽象データ型を導入するアプローチ[1]や、オブジェクト指向データベースを構築するアプローチが研究されている[2]。

これらのアプローチでは、複合オブジェクトの表現は従来より容易になるが、ユーザは個々の応用目的に合わせたデータ型を個別に定義せねばならない。一般に複合オブジェクトの構造のモデリングにおいては、木やDAGが重要な役割をなすことが知られている[3]。したがって、木やDAGといった種々の複合オブジェクトに共通する構造をデータ型として、あらかじめシステムが支援すれば、複合オブジェクトの定義や操作が体系化できると共に、構造操作の一貫性を保証することも可能になると考えられる。

本稿では、順序木をとり上げ、値をノードに対応させる順序木をデータ型として支援する上での基本モデルについて検討した。

2.モデル

ここで、考える木構造が、階層モデルにおける階層やネットワークモデルにおける親子集合と基本的に異なる点は、ノードに対応する値の型が順序木ごとに異なっても構わない点である。

本モデルでは、順序木にcursorという概念を導入したものをC-順序木と定義し、木の値の部分変更を行えるようにした。これにより、木構造の操作をcursorを介してのみ行なうことで1)ノードやエッジのIDを隠蔽することができる、2)操作前と操作後のノードの対応がつけやすい、という利点が得られる。C-順序木は、固有のTID(Tree Identifier)により一意に識別される。またC-順序木をそのインスタンスとして持つC-順序木型を考える。例えば、データベースのモデルとしてリレーショナルモデルを考えた場合、C-順序木型はリレーション内では整数型、文字型等と同様に属性の定義域として扱われる。木検索等の航行はC-順序木型の演算として、そのTIDを介して行うことが可能である。

まず、順序木の定義を行う。定義は文献[4]に従い、以下のように定義する。ただし、[4]では順序関係は、親子関係のみの半順序であったが、ここでは兄弟関係も含めた全順序関係で考える。

定義1 比較演算の定義

N^+ : 正整数

U : N^+によって生成されるモノイド

・: U上のオペレーション

0 : Uの単位元

$a <_2 b$ iff $(\exists x \in U\ (a \cdot x = b)) \vee (\exists c \in U,\ \exists d1,\ \exists d2 \in N^+,\ \exists e1,\ \exists e2 \in U(d1 < d2 \wedge a = c \cdot d1 \cdot e1 \wedge b = c \cdot d2 \cdot e2))$

定義2 順序木Dの定義

Uの有限部分集合Dで、以下の条件を満たす時、順序木と定義する。

a) $b \in D \wedge a < b \Rightarrow a \in D$

b) $a \cdot j \in D \wedge i, j \in N^+ \wedge i < j \Rightarrow a \cdot i \in D$

定義3 関数r: $D \rightarrow N^+$の定義

$r(a) = \max\{i \mid a \cdot i \in D\}$

以下では、C-順序木及びそれに関連する定義を行なう。

定義4 C-順序木tcの定義

$c \in D$, V:値集合, $ts : D \rightarrow V$とした時、

tc = (D,ts,c)とC-順序木を定義する。この時のcがcursorである。さらに、tc全ての集合をTcと定義する。

定義5 データベースDdの定義

TID : Tree identifier集合。無限集合TIDに対して、その有限部分集合をTIΔと定義し、$\varphi : TI\Delta \rightarrow Tc$を定義する。φ(TIΔ)は、ある時点のC-順序木の集合であり、TIΔは変化する。さらに、$\delta = (TI\Delta, \varphi)$と定義し、δ全ての集合をDdと定義する。

定義6 関数succ: $Tc \rightarrow Tc$ の定義

$(D,ts,b) = succ((D,ts,a))$ iff $a <_2 b \wedge \forall b'(a <_2 b' \Rightarrow b \leq_2 b')$

3.オペレーション

以下にオペレーションの定義例を示す。

Move_cursor_forward : TID×Dd → Ddの定義

$x \in TI\Delta$に対して、 $\varphi'(x) \equiv$ if x = i then succ(φ(x)) else φ(x) と定義し、 Move_cursor_forward(i,(TIΔ,φ)) = (TIΔ,φ')と定義する。

Insert_node : V×TID×Dd → Ddの定義

$i \in TI\Delta$, $v \in V$, $\varphi(i) = t$に対して、$c \in D$, $t = (D,ts,c) \in Tc$とする。ここで、$D' = D \cup \{c \cdot (r(c)+1)\}$とDを拡張し、$d \in D'$に対して、$ts'(d) \equiv$ if $d = c \cdot (r(c)+1)$ then v else ts(d) と、tsを拡張する。さらに、$t' = (D',ts',c) \in Tc$であり、$x \in TI\Delta$に対して$\varphi'(x) \equiv$ if x = i then t' else φ(x) とすることより、Insert_node(v,i,(TIΔ,φ)) = (TIΔ,φ')と定義する。

Create_tree : V×Dd → TID×Ddの定義

$(TI\Delta,\varphi) = \delta$, $v \in V$とする。$i \in TID \wedge i \notin TI\Delta$という条件を満たすiを1つ選び、$TI\Delta' = TI\Delta \cup \{i\}$とする。また、新たにts(0) = vと定義して、$x \in TI\Delta$に対して、$\varphi'(x) \equiv$ if x = i then ({0},ts,0) else φ(x), $(TI\Delta',\varphi') = \delta'$として、Create_tree(v,δ) =(i, δ')と定義する。

A Cursor-Tree Model for Complex Object Databases

N.Tsuchiya*, K.Yamaguchi**, N.Ohbo**, H.Kitagawa**
*Division of Scientific Studies, Univ. of Tsukuba
**Institute of Information Sciences and Electronics, Univ. of Tsukuba

以下、同様に定義を行い、オペレーション全体は図1のようにまとめる。これらは5つの種類に分類できる。

2)は各木の内部に存在するcursorを動かす演算である。3)はC-順序木の構造の一部を変化させる演算である。

1)、3)の演算は、C-順序木の値に対して、a)ルートノードは1つである、b)その他のノードは必ず1つの親を持つ、という制約を課しているためにC-順序木のデータ構造の一貫性を満たす。また、a)ノードを作る時には、必ず対応する値を指定し、値に対応しないノードの存在を許さない、b)4)の演算の種類を制限する、という制限により参照一貫性も満たす。

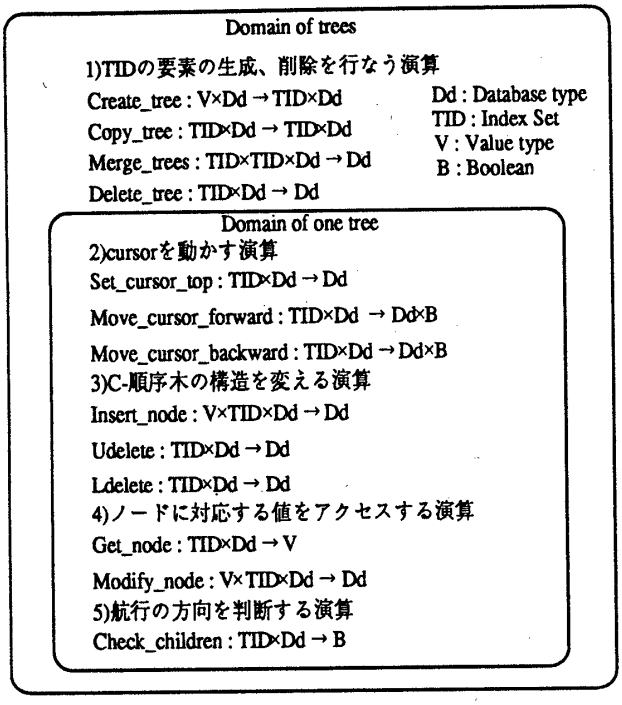

図1 オペレーション定義

4.スキーマ

C-順序木型のデータに対するスキーマ定義は、BNFで定義する。図2にCSGモデルの場合の例を示す。その際、スキーマの整合性条件は一般的に図3の様に示される。図3の式は、トランザクションが終了する直前で、木の構造が、アプリケーションが定めるプロダクションルールに従っているかどうかを調べるためのものである。

```
<Geometric> ::= <Transf> <Object>
<Object> ::= <Combined> | <Primitive>
<Combined> ::= <Geometric> <Set_Op>
                <Geometric>
<Primitive> ::= Box | Cylinder | Cone |....
<Transf> ::= Real Real Real Real Real
              Real
<Set_Op> ::= or | and |dif
```

図2 BNF定義例

ts : D → V (2.モデルで使用した記号を再び用いる。)

s ∈ D ∧ u ∈ U∧ s・u ∈ D ⇒ ts(s) ::=

ts(s・1)...ts(s・u)...ts(s・r(s)) ∈ P(Pは、BNFのルール)

図3 スキーマ整合性条件

5.実現

実際に図1のオペレーションで図2の定義によるCSGモデルを操作するための各種オペレーション[5]を記述できた。(記述例を図4に示す。)これにより、ある種のアプリケーションに対して有用であることを確かめた。

また、テストプログラムとしてC言語で図1のオペレーションに従い、構造はリスト処理として、ノードに対応する値はUNIX上の可変長ファイルに格納するものとして本支援機構の実現を行なった結果、単独なシステムとしては実現可能であることを確かめた。

```
search(tree_name,cmp_attr,get_attr,op,const)
 {
 Set_cursor_top(tree_name);
 while(Move_cursor_forward(tree_name) == True) {
     list = Get_node(tree_name);
     list から比較の対象となる属性値を取り、valueに代入;
        switch(op) {
            case EQUAL:
             /*比較演算が組み込み型の等号の場合*/
                if(const == value)
                   listから返すべき属性値
                   を取り、その値を返す;
                break;
            case *****: .....
            }
    }
 return NULL;
 }
```

図4 CSG演算定義例

6.今後の課題

今後の課題としては、現在定義している図1のオペレーションで、どの程度効率的に、様々なアプリケーションの記述を行なえるかについて、オペレーションの完全性と共に考察する必要がある。

次に、図3のスキーマ整合性条件について、トランザクション中でも生成、削除の際に構造の整合性が判断可能なものに発展させる予定である。

さらには、DAGのモデリングについても考えていく予定である。

参考文献

[1]M.Stonebraker et al., "The Design of POSTGRES,"in Proc. of 1986 SIGMOD Conf., May 1986.

[2]W.Kim et al.,"Integrating an Object-Oriented Programming System with a Database System,"in Proc. of 1988 OOPSLA Conf., Sep. 1988.

[3]W.Kim et al.,"Clustering a DAG for CAD Database," IEEE Trans. Software Eng.,Vol.14, No.11, Nov. 1988.

[4]K.S. Fu et al.,"Tree Systems for Syntactic Pattern Recognition," IEEE Trans. Comput.,Vol.C-22, No.12, Dec. 1973.

[5]S.J. Jiang et al.,"CAD支援を指向した複合対象抽象データ型の提案,"情報処理学会論文誌投稿中.