4L−2　データベースにおける代数記述に基づく木構造モデル
−複合オブジェクト支援の一貫として−

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等の分野でのデータベースの支援の要求が高まっている。従来の多くのデータベースは、事務処理を目的としたものであり、単純なデータ型しか支援していない。しかし、これらの分野では、複合オブジェクト等の複雑なデータ構造を扱うことが多く、それらを支援する機構が、データベース管理システムに要求される。

現在、複雑なデータ構造をデータベースで扱うために、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[2]等の研究がされている。また論理的なデータモデルの研究としては意味モデルを応用したもの等が研究されている。

これらのアプローチでは、複合オブジェクトの表現は、従来よりも容易になるが、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。このために、ユーザはアプリケーションごとにADTを定義したり、データの一貫性制御を行なったりせねばならない。

この問題に対して、本稿では、一般的で複雑な構造の1つとして木に対する航行を可能にするために論理的なカーサーを導入し、一般的な木を扱う支援機構を検討した。

2 モデル

ネットワークモデル等の古典的なデータモデルではデータを航行するためにカーサーが使用されてきたが、このようなカーサーはデータ構造に依存したポインターの性格が強く、扱い難いものであった。そのため、関係モデルなどでは、集合論を用いた演算を導入することにより、カーサーの考え方を除外した。これは物理的実現法と論理モデルの独立を目指す立場としては正しい方向であった。

しかし、CAD/CAM等の分野では、構造情報(木、DAG、有向無向グラフ等)を扱う必要があり、データの性質から以下のような支援が要求される。

1. 単純な構造ではないので、構造中の位置を指定する手段が必要である。
2. グラフ構造のように本来順序のないノードに対して、実際に扱うため には、ノードをたどるための順序が必要になる。

ここでは、以下の目的でカーサーを使用する。

1. 構造の一部に着目する手段を与える。
2. 順序を導入し、ノードを順に探索する事を可能にする。
3. 探索中にデータが操作された時の探索順の受ける影響をフォーマルに決める。

このカーサーを用いて、データの操作や航行を行なう演算の意味を明確に記述する。これにより、構造情報のデータの扱いを明らかにし、データ一貫性制御や、ロックの方式などについてのヒントが得られる。

Modeling Tree Structures in CAD Databases Based on Algebric Specification
N.Tsuchiya,K.Yamaguchi,N.Ohbo,H.Kitagawa
Division of Scientific Studies,Univ. of Tsukuba
Institute of Information Sciences and Electronics,Univ. of Tsukuba

定義 1 カーサーを扱う演算を以下のように定義する。
$\beta, \gamma : T \to T$
T : *tree domain*

公理 1
$\gamma^{-1} = \gamma^{-1}\beta$
$\beta\beta^{-1} = \beta^{-1}\beta = e$
$\gamma^{-1}\gamma = e$
(e は単位元である。従って $\gamma^{-1}e = e\gamma^{-1} = \gamma^{-1}$ 等々)
β, γ 演算を用いて、航行を行なう。

例 1
$\gamma\gamma\beta\gamma$
$\gamma\gamma\beta\beta\beta^{-1}\gamma$

例1から分る通り、あるノードへのたどり方が複数(無限個)存在する。そこで標準型を定義する。標準型の基準となる簡約系を次のように定義する。

定義 2 簡約系の定義
$R = \{\gamma^{-1}\beta \to \gamma^{-1},\ \beta\beta^{-1} \to e,\ \beta^{-1}\beta \to e,\ \gamma^{-1}\gamma \to e,\ \gamma^{-1}\beta^{-1} \to \gamma^{-1},\ \beta e \to \beta,\ \gamma e \to \gamma,\ \beta^{-1}e \to \beta^{-1},\ \gamma^{-1}e \to \gamma^{-1},\ e\beta \to \beta,\ e\gamma \to \gamma,\ e\beta^{-1} \to \beta^{-1},\ e\gamma^{-1} \to \gamma^{-1}\ ee \to e\}$

定義 3 $\beta^{-1}, \gamma^{-1}, \beta, \gamma$ から生成される自由代数系を A とする。$\mu \in A$ に対して、簡約を行なったものを $\bar{\mu}$ とする。$\bar{\mu}$ 全体からなる代数系を $\bar{A}$ とする。

命題 1
μ の簡約によって得られる $\bar{\mu}$ は、一意に決まる。
証明 1[3]

性質 1 $\overline{\mu_1\mu_2} = \overline{\bar{\mu_1}\bar{\mu_2}}$

$\bar{\mu} \in \bar{A}$ が β^{-1}, γ^{-1} を含まない時、$\bar{\mu}$ を real と呼ぶ。real な $\bar{\mu}$ の集合を $\bar{A}_{real}$ と書く 。real でない $\bar{\mu}$ の集合を imaginary と呼び、
$\bar{A}_{imaginary}$ と書く。

real と imaginary を導入する目的は、$\mu \in A$ を簡約する際に、一般的に結合律を成り立たせることである。$\bar{A} = \bar{A}_{real} \cup \bar{A}_{imaginary}$

定義 4 木集合 $T(t \in T)$ の定義
$\bar{A}$ と T の作用 $\otimes$ により T を定義する。

1. $\bar{\mu} \in \bar{A}_{imaginary} \Rightarrow \bar{\mu} \otimes t = \perp$
2. $\bar{\mu} \in \bar{A}_{real} \Rightarrow \bar{\mu} \otimes t \in D \cup \{\perp\}$
3. $\forall \bar{\mu} \in \bar{A}(\bar{\mu} \otimes t_1 = \bar{\mu} \otimes t_2) \Leftrightarrow t_1 =_T t_2$

D : 適当な値域 (木のノードに対応する値)
$\perp$: 特別な値

性質 2 $\forall \bar{\mu} \in \bar{A}_{real}(\bar{\mu} \otimes t_1 = \bar{\mu} \otimes t_2) \Leftrightarrow t_1 =_T t_2$

定義 5 $\bar{A}$ と T の形式的作用 $\odot$ を次のように定義する。結果を $\bar{A} \odot T$ とする。

1. $\bar{\mu_1} \odot t_1, \bar{\mu_2} \odot t_2 \in \bar{A} \odot T$ に対して、$\bar{\mu_1} \odot t_1 =_{\bar{A} \odot T} \bar{\mu_2} \odot t_2 \Leftrightarrow \bar{\mu_1} =_{\bar{A}} \bar{\mu_2}, t_1 =_T t_2$ と定義する。

2. $\bar{\mu}_1 \in \bar{A}, \bar{\mu}_2 \odot t_2 \in \bar{A} \odot T$ に対して、$\bar{\mu}_1 \odot (\bar{\mu}_2 \odot t_2)$ を $(\overline{\bar{\mu}_1 \bar{\mu}_2}) \odot t_2 \in \bar{A} \odot T$ と定義する。

定義 6 α を $\bar{A} \odot T$ に作用する要素として次のように定義する。

$$\alpha(\bar{\mu}_1 \odot t) = \bar{\mu}_1 \otimes t$$

性質 3 $\forall \bar{\mu} \in \bar{A}_{imaginary} \exists \bar{\mu}' \in \bar{A}(\overline{\bar{\mu}' \bar{\mu}} \in \bar{A}_{real}) \wedge$
$\forall \bar{\mu} \in \bar{A}_{imaginary} \exists \bar{\mu}' \in \bar{A}(\overline{\bar{\mu}' \bar{\mu}} = e)$

注意 $\forall \bar{\mu} \in \bar{A}_{imaginary} \exists \bar{\mu}' \in \bar{A}_{real}(\overline{\bar{\mu}' \bar{\mu}} \in \bar{A}_{real})$ は成立しない。

定義 7 $A_B \subseteq \bar{A}_{real}$ が次の条件を満たす時、*boundary* と呼ぶ。 $\exists n \forall t \in T(\bar{A}_{real} A_B \otimes t = \{\perp\} \Rightarrow (\forall \bar{\mu} \in \bar{A}_{real}(|\bar{\mu}|^1 > n \Rightarrow \bar{\mu} \otimes t = \perp))$

定義 8 $boundary A_B$ に対して、$\bar{A}_{real} A_B \otimes t = \{\perp\}$ となる木を $boundary A_B$ に *consistent* であると呼ぶ。

定義 9 *well-defined* の定義
$boundary A_B$ に *consistent* な t か、$(\bar{A}_{real} - \bar{A}_{real} A_B) \otimes t \neq \perp$ の時、t は $boundary A_B$ を持つ *well-defined* な木であると呼ぶ。

定義 10 複数のカーサーの定義
$\beta_i, \beta_i^{-1}, \gamma_i, \gamma_i^{-1}, \otimes_i$ を定義 1 ～ 4 , 6 ～ 9 と同様に定義する。[3]

定義 11 木のノードアドレス表現
有限木 t は、次の性質を満たす $\bar{A}_{real} \times D$ の部分集合 T_t で、一意的に表現される。
$T_t \subseteq \bar{A}_{real} \times D$
$((\bar{\mu}, d) \in T_t \Leftrightarrow \bar{\mu} \times t = d \wedge d \neq \perp)$

定義 12 木の基本操作
$T_{t'} = T_t \cup \{(\bar{\mu}, d)\}$ 又は $T_{t'} = T_t - \{(\bar{\mu}, d)\}$ を木の基本操作と呼ぶ。
但し、t', t は、有限木であるようなものに限る。各々の $T_{t'}$ を
$(\bar{\mu}, d)^+ T_t$,
$(\bar{\mu}, d)^- T_t$ と書く。

定義 13 木の操作列
$(\bar{\mu}_n, d_n)^{\pm} (\bar{\mu}_{n-1}, d_{n-1})^{\pm} \cdots (\bar{\mu}_1, d_1)^{\pm} T_t$
を木の操作列と呼ぶ。特に、$T_t = \{(\bar{\mu}_1, d_1), \cdots, (\bar{\mu}_1, d_1)\}$ の時、$(\bar{\mu}_n, d_n)^+ \cdots (\bar{\mu}_1, d_1)^+$ により T_t を表すこともある。

性質 4 任意の有限木 $T_t, T_{t'}$ に対して、$T_{t'} = (\bar{\mu}_n, d_n)^{\pm}$
$(\bar{\mu}_{n-1}, d_{n-1})^{\pm} \cdots (\bar{\mu}_1, d_1)^{\pm} T_t$ となるような木の操作列が存在する。

$\bar{A}$ と木の操作列の関係

$$\bar{\mu} \bullet T_t = \begin{cases} d, & \text{if } T_t = (\bar{\mu}, d)^+ T_{t'} \\ \perp, & \text{if } T_t = (\bar{\mu}, d)^- T_{t'} \\ \bar{\mu} \bullet T_{t'}, & \text{if } T_t = (\bar{\mu}', d)^{\pm} T_{t'} \wedge \bar{\mu}' \neq \bar{\mu} \\ \perp, & \text{if } T_t = \phi \end{cases}$$

性質 5 $\bar{\mu} \bullet T_t = \bar{\mu} \otimes t$

3 オペレーション

3.1 木に対する操作演算

well-defined な木に対する操作演算を基本操作を用いて以下のように定義する。

[1] || は、記号列の数を表す。

1. merge : $\bar{A}_{real} \times T \times T \rightarrow T$
   $\xi \in \bar{A}_{real}$
   $\text{merge}(\xi, t_1, t_2) = t_3 \Leftrightarrow T_{t_3} = (T_{t_1} - \{(\bar{\mu}\xi, \bar{\mu}\xi \otimes t_1) | \bar{\mu} \in \bar{A}_{real}\}) \cup \{(\bar{\mu}\xi, \bar{\mu} \otimes t_2) | \bar{\mu} \in \bar{A}_{real}, \bar{\mu} \otimes t_2 \neq \perp\}$

2. cut : $\bar{A}_{real} \times T \rightarrow T$
   $\xi \in \bar{A}_{real}$
   $\text{cut}(\xi, t_1) = t_2 \Leftrightarrow T_{t_2} = T_{t_1} - \{(\bar{\mu}\xi, \bar{\mu}\xi \otimes t_1) | \bar{\mu} \in \bar{A}_{real}\}$

3. extract : $\bar{A}_{real} \times T \rightarrow T$
   $\xi \in \bar{A}_{real}$
   $\text{extract}(\xi, t_1) = t_2 \Leftrightarrow T_{t_2} = \{(\bar{\mu}, \bar{\mu}\xi \otimes t_1) | \bar{\mu}\xi \otimes t_1 \neq \perp, \bar{\mu} \in \bar{A}_{real}\}$

3.2 航行演算

逐次的にアクセスする演算を考えてみる。
$t \in T, \bar{\mu} \in \bar{A}_{real}$

$$\tau\bar{\mu}t = \begin{cases} \gamma\bar{\mu}, & \text{if } \gamma\bar{\mu} \otimes t \neq \perp \\ \beta\bar{\mu}, & \text{if } \gamma\bar{\mu} \otimes t = \perp \wedge \beta\bar{\mu} \otimes t \neq \perp \\ \beta(\gamma^{-1})^i\bar{\mu}, & \text{if } \gamma\bar{\mu} \otimes t = \perp \wedge \beta(\gamma^{-1})^{i-1}\bar{\mu} \otimes t = \perp \\ & \wedge \beta(\gamma^{-1})^i\bar{\mu} \otimes t \neq \perp \\ \bar{\mu}, & \text{otherwise} \end{cases}$$

これにより全てのノードを深さ優先探索でたどる演算を定義できた。
ここで大切な点は、航行演算が、木構造とは独立の関係にあることだ。

しかし、上例のように β や γ から全ての航行演算を定義することはできない。例えば、同じ逐次的にアクセスする演算であるが、親ノードをたどり、子ノードを全てたどったあとで、もう一度親ノードを訪れてから次のノードをたどるような航行演算を考えてみる。この場合、既に子ノードを訪れているのか、これから訪れるのか不明なので、何か情報が必要になる。この演算を ϕ とする。
$t \in T, \bar{\mu} \in \bar{A}_{real}$

$$\rho\bar{\mu}t, \phi\rho\bar{\mu}t = \begin{cases} \gamma\bar{\mu}, \downarrow, & \text{if } \gamma\bar{\mu} \otimes t \neq \perp \wedge \phi\bar{\mu}t = \downarrow \\ \beta\bar{\mu}, \downarrow, & \text{if } \gamma\bar{\mu} \otimes t = \perp \wedge \beta\bar{\mu} \otimes t \neq \perp \wedge \phi\bar{\mu}t = \downarrow \\ \gamma^{-1}\bar{\mu}, \uparrow, & \text{if } \gamma\bar{\mu} \otimes t = \perp \wedge \beta\bar{\mu} \otimes t = \perp \wedge \phi\bar{\mu}t = \downarrow \\ \beta\bar{\mu}, \downarrow, & \text{if } \beta\bar{\mu} \otimes t \neq \perp \wedge \phi\bar{\mu}t = \uparrow \\ \gamma^{-1}\bar{\mu}, \uparrow, & \text{if } \beta\bar{\mu} \otimes t = \perp \wedge \phi\bar{\mu}t = \uparrow \\ \bar{\mu}, \phi\bar{\mu}t, & \text{otherwise} \end{cases}$$

3.3 操作例

ユーザは、β, γ 演算等を用いて、操作する木の場所を指定し、操作演算のみで木の操作をすることも可能である。しかし、木の操作点を指定するのが、面倒な場合や、木の構造がわからない場合、操作演算のみでは、対応できない。このような場合、航行演算が役に立つ。以下に操作演算と航行演算を組み合わせた例を示す。これは、t_1 を深さ優先最左検索で値が v_1 となる場所に t_2 をマージする例である。
$t_1, t_2 \in T, \bar{\mu} \in \bar{A}_{real}$
$t_3 = merge(\tau^{\{Min(i)\tau^i\bar{\mu}=v_1\}}\bar{\mu}, t_1, t_2);$

4 まとめ

複合オブジェクトの基本となる木構造のモデルを、アプリケーションに依存せずに、一般的に形式化を行なった。

今後の課題は、スキーマを考え、そのスキーマとオペレーションとの関係を明確し、データベースのモデルとして実用的なものにすることである。

参考文献

[1] M.Stonebraker et al.,"The Implementation of POSTGRES",IEEE Trans.Knowlege and Data Eng.,vol.2,No.1,Mar.1990.

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

[3] 土屋," データベースにおける代数記述に基づく木構造モデル ", 筑波大学理工学研究科修士論文論文,1991 年 2 月

[4] 土屋 他," データベースにおける木構造のモデル化の提案 ", 第 40 回情報処理学会全国大会予稿,1990 年 3 月