4M-8 ソリッドオブジェクト索引用の横型探索８進木

PengChong Koh* 張暁冬* 山口和紀** 北川博之** 大保信夫**
(*筑波大学理工学研究科 **筑波大学電子・情報工学系)

1. はじめに

近年、CADの普及に伴い、それを支援する統合データベース・システムに対する要求が高まっている。しかし、従来の関係型データベース・システムをCADに応用する際、CADで使用する莫大な構造化データをどう扱うかなど様々な問題点が生じる。これらの問題点を解決するため抽象データ型による解決法が提案されており[1]、いくつかの例に適用した結果[2]では有効性が示されている。しかし、属性の値が複雑なオブジェクトを表わす場合、何らかのアクセス手段がなければ検索などの処理は不可能である。我々は抽象データ型などにアクセス法を付ける事を可能とするためのDBMSのアーキテチャMODUS[5]を開発中であるが、本稿では、機械系CADで使用されるソリッドモデラの補助的構造として８進木(octree)を使用する場合に相互干渉による検索を高速化するデータ構造とADTの定義の例について述べる。

2. 背景

機械系CADで使用するソリッドモデラとしてソリッドをプリミティブから正規化論理演算や平行移動などの演算によりどうのように構成されるかをCSG木のようなデータ構造によって記述する方法がある。しかし、このようなソリッドモデルでは干渉チェックなどの処理に時間がかかるので、近似データをデータベースに格納して管理するのが望ましい。しかし、このようなデータの操作をアプリケーションレベルで行う場合はデータベース中の全てのソリッドに対し一つずつソリッドを取り出して干渉しているかどうかの判断を下す事になり効率が悪い。データベースレベルで行う事よりインデックスなどのDBMS機構により与えられたソリッドと干渉するソリッドを検索する事が可能となる。しかし、従来のDBMSではデータベースレベルでの実現は困難である。現在、我々は拡張可能DBMSのアーキテクチャMODUSを開発中であるが、ここではその研究の一環として８進木のような複雑な構造の干渉チェックをADT機構によりDBMSでサポートする方法について検討する。

2.1 LS-Octree

本研究ではソリッドに対するインデックス用の近似表現として８進木を用いる。８進木の線形な表現法としてDF(Depth First)[4]表現がよく知られている。しかしDF表現は縦型探索の構造のため浅いレベルの木のノードをアクセスするにはその前にある深いレベルのノードのデータを読み飛ばさなければならずデータ検索の効率は悪い。これに対しLayered String (ＬＳ)データ構造[3]は横型探索のため不必要に深いレベルのノードを読む必要がなく有効である。LSでは、{ 0, 1, X }の値をとる節ストリング(NodeString)のリストを木の深さ順に並べてある。0と1は対応した８進木の葉の値を示すが、Xは対応した８進木が節であることを示し、更に深いリストにデータがあることを示す。（図1参照）

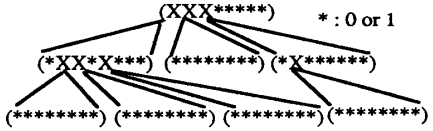

図1 LSの例

2.2 ADT定義

ADTは（N, S, F）の三つ組で定義される。NはADTの名前、Sはそのデータ構造の仕様、FはADTに付随する関数群である。ADTの名前と関数は外部から見えるがデータ構造はADT関数のみアクセスできる。

８進木のLS表現はADTとして定義され、また、ADTを操作するためのADT関数、つまり８進木を操作する関数はADT-QUELインタープリターに設けられる機構によって定義される。

```
ADT-definition
    (ADT-name = LS-OCTREE
        A-function   union
            is       LS-OCTREE x LS-OCTREE
                                --> LS-OCTREE
        B-function   interference
            is       LS-OCTREE x LS-OCTREE
                                --> BOOLEAN
        P-function   depth
            is       LS-OCTREE --> INTEGER
```

Layered String Octree for Solid Object Indexing

PengChong Koh*, XiaoDong Zhang*, Kazunori Yamaguchi**, Hiroyuki Kitagawa**, Nobuo Ohbo**
*Division of Scientific Studies, Univ. of Tsukuba
**Institute of Information Sciences and Electronics, Univ. of Tsukuba

filename = "/usr1/koh/adt/OCTREE")

上に示されたのはLS-OCTREEというADT定義の一部である。functionの左にあるA,B,Pはそれぞれ ADT, BOOLEAN及びPRIMITIVETYPE（論理型を除く）を示す。union関数は2つの8進木の論理和をとって、8進木をリターン値として返す。interferenceは2つの8進木は干渉しているかどうかのチェックである。またdepthは8進木の深さを調べる関数である。filenameは関数の本体が格納されているファイル名を表す。

2.3 データベース・スキーマ

この例の8進木のデータベース・スキーマは以下の2つのリレーションからなる。まず図2に示されたリレーションsolidでは、属性Sidは整数型、属性SnameとDesignerは文字型、属性Dateは日付型、属性LS-OctreeとDisplay-InfoはADTである。図3に示されたリレーションLS-Octreeでは、属性OidとLayer#は整数型で、属性NodeStringはADTである。8進木をこのように層に分けることによりinterference関数では指定されたLSの各層を独立に操作することができ、8進木の葉までの大部分のパスは検索しなくて済む。つまり、LSの各層はある種のフィルタの働きを果たす。

Solid

Sid	Sname	Designer	Date	LS-Octree	Display-Info	...

図2 Solid 関係

LS-Octree

Oid	Layer#	NodeString

図3 Octree 関係

2.4 問い合わせ処理

データベースに対する問い合わせはADT-QUELという拡張QUEL問い合わせ言語によって行う。拡張関係型システムのADT-QUELインタープリターにより、問い合わせ文は低いレベルの操作機能を持つDM-LISPの形式に変換され、データベースのアクセスが行われる。

```
RANGE OF t1 IS Solid
RANGE OF t2 IS Solid
RETRIEVE ( t2.Sname  display ( t2.Octree ))
      WHERE   t1.Sid = " 100 "
            AND    interference ( t1.Octree,  t2.Octree )
```

上の問い合わせの例ではSid = 100のソリッドと干渉する全てのソリッドを画面上に表示する。ADT関数interferenceは以下のように定義することができる。

```
(defun interference (t1 t2 &optional (layer# 0))
  (let ((c1 (fetch-layer t1 layer#))
        (c2 (fetch-layer t2 layer#)))
    (if  (or (null c1) (null c2))
     nil
     (ecase (LS-interference c1 c2);LSの干渉チェック。
          (TRUE t)
          (FALSE nil)
          (UNKNOWN (interference t1 t2 (1+ layer#)))))))
;その層で判断がつかない時のみ次の層をチェックする。
```

3. システム構成

システムの最下層のG-BASEは、UNIX環境のLISPベースの商用関係型DBMSである。その上にADT-QUELインタープリターによりADTを処理できる拡張関係型システムが実現されている。また、8進木データモデルを用いたソリッドモデラの応用CADシステムはその上に構築される。(図4参照)

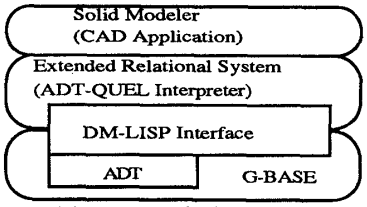

図4 システム構成

4. おわりに

現在、ADT-QUELの拡張により実現する作業を行っており、8進木のシステムのシステムへの組み込み作業を行っているが、2の背景でも述べたように、このようなADT機構を効率よく実現するには、アクセスメソッドやデータマネジャーの支援を必要とする。MODUSではDBMSカーネルにADT機構を組み込み（図4のADTの部分）、これを実現する方法論を研究中である。

5. 参考文献

[1] Ong, J. Fogg, D. and Stonebraker, M., Implementation of Data Abstraction in the Relational Database System INGRES, ACM SIGMOD Record, Apr. 1983, pp. 1-14.

[2] Jiang, S.J. et al., Abstract Data Types in Graphics Databases, Proceedings of the IFIP TC 2/WG 2.6 Working Conference on Visual Database Systems, Tokyo, 3-7 April, 1989, pp.239-255.

[3] Yamaguchi, K. Kunii, T.L., A Layered String Data Structure for an Octree Model, 1983 IEEE Computer Society Workshop on Computer Architecture for Pattern Analysis and Image Database Management.

[4] Samet, H. Webber, R.E., Hierarchical Data Structures and Algorithms for Computer Graphics, IEEE Computer Graphics & Applications, May 1988, pp.548-568.

[5] 古瀬一隆 et al.,拡張可能DBMS MODUS のアーキテクチャ、情報処理学会第39回全国大会予稿集、Oct. 1989.