データベース・システム　72-23
(1989. 7. 21)

並列処理システムＳＭＡＳＨにおける機械系ＣＡＤデータベース実現のための一考察

A study on the implementation of mechanical CAD databases in the parallel processing system SMASH

関　安宏*　　関島章文*　　清木　康**
Yasuhiro SEKI　Akifumi SEKIJIMA　Yasushi KIYOKI

*筑波大学理工学研究科
**筑波大学電子・情報工学系
University of Tsukuba

あらまし　ＣＡＤデータベースの実現においては，複雑なデータ構造およびそれらを対象とする複雑な演算を扱う必要がある．データベースおよび知識ベースを対象とした拡張可能型並列処理システムＳＭＡＳＨ上で，ＣＡＤデータベースを実現するためのデータ構造および基本演算の設計を行っている．ＣＡＤデータベースの実現においては，製品を表現し，操作するためのデータベース・スキーマ，データ構造，基本演算の設計が重要である．本稿では，ＣＡＤデータベースの実現のための３種類の方法について議論する．これらの方法は，それぞれ，関係モデル，関数型データモデル，複合オブジェクトの概念に基づいている．

Abstract　In implementing CAD databases, it is necessary to deal with complex data structures and complex operations for them. We have proposed an extensible parallel processing system SMASH for databases and knowledge bases. We are designing data structures and basic operations for implementing CAD databases on SMASH. In the implementation of CAD databases, it is important to design database schema, data structures and basic operations for representing and manipulating products. In this paper, we discuss three different methods for the implementation of CAD databases. These methods are based on the relational model, the functional data model, and the concept of complex objects, respectively.

１．はじめに

近年，データベースの応用分野が拡大している．たとえば，新しい応用分野の１つである CAD では，設計され，蓄積されたデータに対し，データの一元管理，効率的な検索が必要となっている．これに対処するために，データベース技術の CAD への適用が試みられている．

従来のデータベースの応用分野は，事務処理が中心であり，データベース技術もこれらの分野を対象として発展してきた．この分野において，データベース・システムが扱うデータは，文字，数字が主である．これに対し，CAD は，従来のデータベース・システムが扱ってきたデータに加え，部品の形状情報や製品の構造情報なども扱う必要がある．この結果，データベース・システムが扱うデータの構造は，複雑となり，従来のデータベース・システムでは，CAD において扱われるデータに，十分に対応することができない．このため，新しい応用分野を支援するための新しいデータベース技術の研究が行われている．

我々は，データベースおよび知識ベースを対象とした拡張可能型並列処理システム SMASH の実現を進めている[2]．SMASH の並列処理方式は，関数型計算モデルの概念に基づくものであり，データベースおよび知識ベースの多様な応用分野に適用することができる．SMASH において，データベースの任意の基本演算は，関数として定義され，要求駆動型評価により並列に実行される．また，演算の対象となる大量データは，ストリームとして扱われる．現在，SMASH は，２種類の環境，すなわち，LAN により結合された複数台のワークステーション上，および，バス共有型マルチプロセッサ・マシン上に実現されている．

本稿では，SMASH の拡張性を実証し，また，CAD データベースの構築の指針を得るために設計した CAD データベースの実現方式について述べる．

CAD データベースの研究分野は，トランザクション，版管理，データモデルなど，多方面に及んでいる．ここでは，CAD として機械系 CAD を対象とし，設計されたデータを蓄積しているデータベースに対する問い合わせの処理方式に関する検討を行う．

本研究では，機械系 CAD システムにおいて設計され，データベースに蓄積されたデータに対し，検索を行うためのデータベース管理システムの機能を，関数として SMASH に組み込み，データベースの問い合わせ処理系の実現を行う．

本稿では，CAD において，製品を表現するために必要となる情報を表すデータ構造，それらをデータ

ベースに格納するためのデータベース・スキーマの設定，また，データベースの問い合わせ処理に必要となる機能を実現するための３種類の異なる方法について検討および考察を行う．ここで対象とする方法は，関係データベースを用いる方法，関数型データモデルを用いる方法，および，複合オブジェクトの概念を用いる方法である．

２．ＳＭＡＳＨの概要

SMASH システムの特徴を以下に示す．

- 関数型計算モデルは，関数間の並列性を抽出することが容易である．SMASH システムでは，関数の引数評価を独立に行うことにより抽出される並列性，および，引数の適用側と生成側の間で抽出されるストリーム型並列性を抽出することができる．
- 要求駆動型評価とストリームの概念を組み合わせて関数計算を行うことにより，限られた計算機資源の中で大量データを扱うことができる．
- 関数として記述されたデータベース演算は，SMASH が提供する基本プリミティブを含むオブジェクト・コードに変換された後，システム内に組み込まれる．基本プリミティブは，データベースの個々の演算に依存しないレベルに設定されているので，任意の応用分野における演算を支援することが可能になる[2]．

データベース操作のための基本機能を SMASH において実現する場合，それらの機能を実現するためのデータベースの基本演算を，関数として SMASH に組み込む必要がある．データベースの基本演算は，関数型の言語を用いて定義される．定義された各基本演算は，言語処理系により関数を単位として，SMASH システムの基本プリミティブを含むオブジェクト・コードに変換され，基本関数として SMASH システムに組み込まれる．組み込まれた各関数は，SMASH システムが提供する機能として，システムの応用側から利用することができる．たとえば，関係データベースに対する操作を SMASH において実現する場合，関係演算をデータベースの基本演算として定義し，SMASH に組み込む．データベースに対する操作は，組み込まれたこれらの基本演算を用いて行われる．データベースの問い合わせ処理時には，組み込まれた基本演算間で並列性が抽出される．

３．ＣＡＤデータベース

本稿では，CAD として，３次元形状モデルを用いて３次元立体を表現する機械系 CAD を対象にする．本章では，機械系 CAD において，設計対象となる製品を表現するために必要となる情報について示す．また，設計されたデータを格納しているデータベースに対するデータベース操作を示す．

３．１　ＣＡＤで扱うべき情報

設計作業とは，ある与えられた要求仕様に対し，その仕様を満足しながら，具体的な製品の形状に変換することである．したがって，CAD システムは，製品の形状を表現するための情報を扱う．

一般に，３次元形状は，位相情報および幾何情報により表現される．３次元形状を表現するモデルに，ソリッドモデルの代表的なモデルである CSG(Constructive Solid Geometry)およびB-Rep (Boundary Representation, 境界表現) がある[1]．CSG は，集合演算を用いて複数の基本形状を組み合わせることにより，３次元形状を表現する．図形を表現するためのデータ構造は，木構造である．B-Rep は，立体を構成する面，稜線，頂点の各々の関係を記述することにより，３次元形状を表現する．このモデルは，ワイヤフレームモデルに，面，立体の情報を付け加えたものである．図形を表現するためのデータ構造は，木構造，あるいは，ネットワーク構造などである．

CAD において，形状データだけでは製品を表現することができないので，一般に，製品情報が必要となる．製品情報に含まれるべき情報は，部品情報，構造情報，属性情報である．部品情報は，部品の形状を表現するための形状情報，および，部品の材質など，部品固有の情報である属性情報からなる．構造情報は，製品を構成する各部品の配置のための情報である配置情報，各部品間の接続状態などを表現する接続情報からなる．属性情報は，生産，組立に関する生産情報や管理に関する生産管理情報からなる．

これらの情報は，互いに関連をもっている．そのような関連を表現する方法として，確立されたものはない．本稿では，これらの関連を表現するために，木構造を用いることを前提として議論を行う．

３．２　データベース操作

CAD システムの中心的な機能を実現する CAD データベースは，CAD システムにおいて設計されたデータの一元管理を行い，検索の効率化を図るものである．CAD システムのアーキテクチャは，public データベースを管理する public システムと，複数の private システムから構成される[3]．public データベースは，設計が完了している製品データを管理する．データベースが扱うデータは，製品モデルを表現した複雑なデータ構造を持つ．

CAD のトランザクションは長時間におよぶので，事務処理におけるデータベースのトランザクションの管理とは異なる．そこで，CAD トランザクションのためのトランザクション・モデルが研究されている[3]．１つのトランザクションにおいて，private データベース，および，public データベースに対する操作が行われる．

private データベースに対する操作は，おもに，設計作業中に行われる．設計作業は，形状を定義する形状モデリング，定義した形状に対し性能解析や強度解析などを行う解析やシミュレーションなどで

ある．これらの設計作業のうち，形状定義に関しては，データベース操作として，検索，創成，更新，削除が挙げられる．解析，シミュレーションにおけるデータベース操作は，検索が主である．

public データベースに対する操作は，おもに，設計作業の開始および終了時に行われる．データベース操作として，検索，創成，更新，削除が挙げられる．創成は，設計作業において，新たに設計された部品，あるいは，製品をデータベースに新規登録する際などに行われる．更新，削除は，すでに設計された製品データに対し，設計作業において変更が施された場合に行われる．検索は，設計作業において，すでに設計されている設計物の再利用を図る際，登録されている製品，あるいは；部品を参照する場合などに行われる．

次章以降において，一つのトランザクション内に実行される，public なデータベースに対する問い合わせに関する処理方式を検討する．

4　ＳＭＡＳＨ上での実現

本章では，製品を表現するために必要となる情報とデータ構造を設定する．また，それらのデータを格納したデータベースに対し，問い合わせを行う際に必要となる基本機能を設定する．

4．1　扱うデータとその構造

ここでは，製品を表現するための情報として，部品の形状，および，部品間の構成関係を表現する基本的な情報に限定する．したがって，製品情報は，部品情報である形状情報，構造情報である配置情報，および，接続情報から構成される．

製品を表すために必要となるこれらの製品情報を互いに関連づけ，製品を表現するために，データ構造として木構造を用いる．つまり，一つの製品を，一つの木構造で構成される製品情報により表現する．

製品情報は，部品情報である形状情報，および，構造情報である配置情報と接続情報から構成されるので，これらの情報を木構造に表現するために，木の節の種類として，product, group, parts, solids を設定する（表１）．

各節において，製品情報を格納し，製品における部品間の構成関係を，節と節を結ぶ枝で表現する．製品は，対応した節 product により表現される．節product は，各節に製品情報を持つ木の根に対応する．節 solids においては形状情報，節 parts においては配置情報，節 group においては接続情報が格納される．部品情報である形状情報については，形状を表現するための形状モデルとして境界表現を採用し，同一形状ごとにファイルに格納する．このファイルが，節 solids に対応する．一般に，製品情報は，深さが不定の木構造として表現される．

たとえば，製品”椅子”の部品構成が図１のような場合，製品”椅子”の製品情報は，図２に示す木構造で表現される．製品”椅子”は，product ノードにおいて，表現される．部品”車輪”の形状は同一であるので，これらの部品の形状データは，solids ノードにおいて共有される．しかし，各部品”車輪”の配置位置はそれぞれ異なるので，各部品に対応した parts ノードにおいて，配置情報が管理される．部品”車輪”と部品”足”の関連は，この場合，回転に対応するが，これらの部品の接続情報は group ノードにおいて管理される．

表１　木構造の節の種類と節に含まれる情報

節の種類	節に含まれる情報
product	製品名
group	グループ名，接続情報
parts	部品名，部品の位置， 形状データへのリンク
solids	形状データ

4．2　ＣＡＤデータベース操作のための基本機能

ＣＡＤデータベースを操作する場合に必要となる基本機能について検討を行った．基本機能は，木構造で表現される製品情報に対する操作として定義される．SMASH において，問い合わせ処理を行う際，ここに挙げる基本機能は，SMASH に組み込まれたデータベース演算を組み合わせることにより実現される．

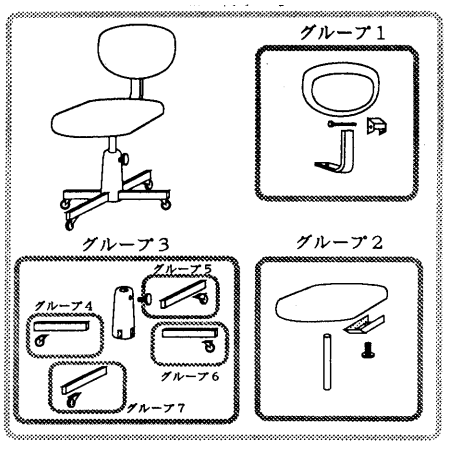

図１　製品”椅子”の部品構成

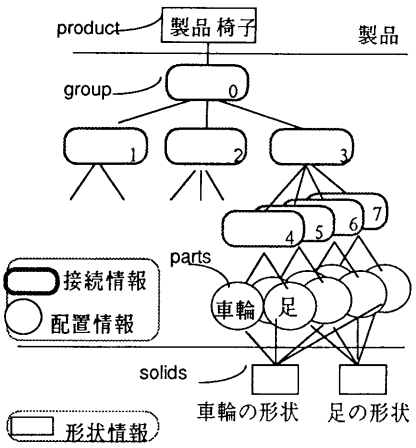

図２　製品”椅子”の製品情報

表２　　基本機能

創成	make-node(node-type, value) -> node-id link-node(parentNode, childNode, link-type)
更新	update-node(node-id, attribute, new-value) update-link(childNode-id, old-parentNode-id, new-parentNode-id, link-type)
削除	delete-node(node-id) delete-tree(rootNode-id)
検索	select-node(node-type, condition) -> node-id get-attribute(node-id, node-type, attribute) -> attribute-value get-child(node-id) -> childNode-id get-all-childs(node-id) -> set of childNode-id get-parent(node-id) -> parentNode-id get-all-parents(node-id) -> set of parentNode-id

データベース操作のための基本機能を，表２に，創成，更新，削除，検索の順に示す．

創成に関する基本機能には，ノードの種類とノードに格納される情報を引数として，ノードを生成し，生成したノードの識別子を返す make-node と，接続する親のノードと子のノード，および，接続の種類を引数とし，ノード間の接続を行う link-node がある．

更新に関する基本機能には，ノードと更新する属性，新しい値を引数とし，ノードに含まれる属性を更新する update-node と，子ノード，古い親ノード，新しい親ノード，新しいリンクの種類を引数とし，ノード間の接続を更新する update-link がある．

削除に関する基本機能には，parts ノードを引数としノードの削除を行う delete-node と，product あるいは group ノードを引数とし，そのノードを根とする木を削除する delete-tree がある．

検索に関する基本機能は，木構造内の特定のノードを選択する機能，ノードに含まれる属性情報を取り出す機能，製品情報を表現している木を探索する機能に分類される．木構造のノードを選択する機能には，ノードの種類と選択する条件を引数とし，選択されたノードの識別子を返す select-node がある．ノードに含まれる情報を取り出す機能には，ノード，そのノードの種類，取り出す情報の属性を引数とし，対応する属性の値を取り出す get-attribute がある．木探索を行う機能には，あるノードの子ノードを求める get-child，あるノードの子孫であるノードを，再帰的に木構造を走査することによりすべて求める get-all-childs，あるノードの親ノードを求める get-parent，あるノードの祖先であるノードを，再帰的に木構造を走査することによりすべて求める get-all-parents がある．

基本機能の例として，検索について示す．製品情報が図３のように表現される場合について，次の２つの問い合わせ例を考える．

(a) 部品７を含む製品名を求める，

(b) 製品２を構成しているグループ名を求める，

問い合わせ(a)の場合，まず，基本機能 select-node を用いて，parts ノードの中から部品７を選択する．次に，基本機能 get-all-parents を用いて，部品７のノードの祖先のノードをすべて求める．求めたものから，基本機能 select-node を用いて，product ノード２を得る．さらに，基本機能 get-attribute を用いて，product ノードが含んでいる情報のうち，属性製品名に対応する値を得ることができる．問い合わせ(b)の場合，まず，基本機能 select-node を用いて，product ノードから製品２の product ノードを選択する．次に，基本機能 get-all-childs を用いて，製品２のノードの子孫のノードをすべて求める．求めたものから，基本機能 select-node を用いて，group ノードを得る．さら

に，基本機能 get-attribute を用いて，各 group ノードが含んでいる情報であるグループ名を求めることができる．

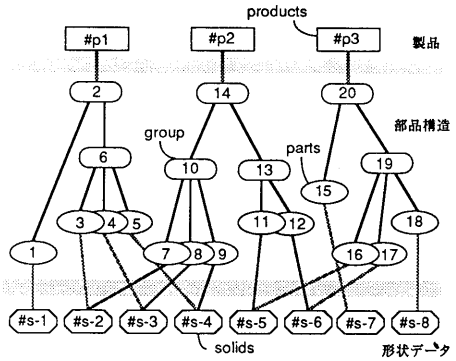

図３　製品ｐ１，ｐ２，ｐ３の製品情報

４．３　データベース演算の実現

データベース操作の処理を SMASH において行う場合，4.2 で示した基本機能を実現するためのデータベース演算を定義し，SMASH に組み込む必要がある．SMASH は，データベースを対象とした任意の演算を柔軟に組み込むことができる．CAD データベースの実現のために，２種類の異なる方法を実現している．一方は，関係モデルの基本演算の実現であり，他方は，関数型データモデルの基本演算の実現である．これらの概念スキーマは，関係モデルにより表現する．また，これらの２種類の方法と異なる方法として，複合オブジェクトに基づくデータの定義と，それらに対する演算の設計を行った．

これらの方法を実現するデータベース演算群は，互いに異なる．SMASH では，それぞれの方法において，各データベース演算を単位とした並列性が抽出される．

本節において，各方法におけるデータ表現と，データベース演算による基本機能の実現方法について示す．

４．３．１　関係モデルに基づく実現

関係モデルによる実現方法では，木構造がフラットな表により表現される．検索は，表に対し関係演算を適用することにより行われる．具体的には，関係演算を SMASH の関数として組み込み，関係の組をストリームの要素として処理することにより実現する．

（１）データ表現

木構造で表わされる製品情報を関係モデルで表現するために，図４に示す関係を用意する．図４において，各関係に格納されている製品情報は，図３で示した製品情報に対応する．これらのうち，node, link は木構造を表すための関係であり，product, group, parts は，各節に含まれる製品情報を表すための関係である．

node

id	type	products
1	parts	#p1
2	group	#p1
3	parts	#p1
4	parts	#p1
5	parts	#p1
6	group	#p1
7	parts	#p2

product

p-id	name	node-id	生産情報
#p1	製品１	2	
#p2	製品２	14	

group

id	name	接続情報
2	グループ１	
6	グループ２	
10	グループ３	

link

id	child-id
2	1
2	6
6	3
6	4
6	5
10	7

parts

id	name	solid	位置情報
1	部品１	#s-1	
3	部品２	#s-2	
4	部品３	#s-3	
5	部品４	#s-4	
7	部品５	#s-2	

図４　関係モデルによるデータ表現

（２）基本機能の実現

４．２に示した基本機能を実現するデータベース演算として，関係演算 join, selection, projection を用いる．

たとえば，基本機能 get-child は，関係演算 selection と projection を用いて以下のように実現できる．

```
(projection child-id
        (selection 'parent-id = X' <link>))
```

基本機能 get-all-childs は，再帰処理を必要とするので，関係演算の枠組だけでは実現できない．したがって，関係演算に再帰処理を行う演算を導入する必要がある．この演算を (transitive {...}) ({}の中の演算を再帰的に実行する．) と表すことにする．基本機能 get-all-childs は，次の演算により実現される．

```
(transitive child-id 'parent-id = X' <link>)
```

４．３．２　関数型データモデルに基づく実現

関数型データモデル[4]による実現方法では，データベース演算として，節と節の関連を表す関数，節を選択する関数，節の属性値を得る関数を設定する．

（１）データ表現

関数型データモデルでは，たとえば，節 group に関する概念スキーマの定義の一部は，DAPLEX[4] の記述を用いると，

```
DECLARE groups() ==>> ENTITY
DECLARE group-id( groups ) ==> INTEGER
DECLARE name( groups ) ==> STRING
DECLARE product( groups ) ==> products
DECLARE child-group( groups ) ==>> groups
```

のようになる．

ここでは，内部スキーマは，関係により表現する．関係として，図５に示す関係を用意する．図５において，各関係に格納されている製品情報は，図３で示した製品情報に対応する．関係 product, group, parts は，製品情報を表現する木構造における各ノードを表すための関係である．これらのノードは，関数型データモデルのエンティティに対応する．関係　products-root group,　group-products, group-child group, group-parts は，各ノード間の関連を表現するための関係である．

products - root group

p-id	group-id
#p1	2
#p2	14
#p3	20

group - products

group-id	p-id
2	#p1
6	#p1
10	#p2
13	#p2
14	#p2

product

p-id	name	生産情報
#p1	製品１	
#p2	製品２	

group

id	name	接続情報
2	グループ１	
6	グループ２	
10	グループ３	

group - child group

parent-id	child-id
2	6
14	10
14	13

parts

id	name	solid	位置情報
1	部品１	#s-1	
3	部品２	#s-2	
4	部品３	#s-3	
5	部品４	#s-4	
7	部品５	#s-2	

group - parts

group-id	parts-id
2	1
6	3
6	4
6	5

図５　関数型データモデルによるデータ表現

（２）基本機能の実現

この方法を実現するデータベース演算は，４．２で示した基本機能を実現する関数として定義される．

たとえば，基本機能 get-child は，DAPLEX では次のように記述される．

```
FOR EACH groups SUCH THAT
                    group-id( groups ) = 'X'
  PRINT group-id( child-group( groups ))
```

本方法では，この基本機能は，次のような関数として定義される．

```
(group-id ( child-group
 (filter groups (= (group-id groups) 'X))))
```

また，基本機能 get-all-childs は，DAPLEX では次のように記述される．

```
DEFINE all-child-groups( groups ) ==>>
      TRANSITIVE OF child-group( groups )

FOR EACH groups  SUCH THAT
                    group-id( groups ) = 'X'
PRINT group-id( all-child-groups( groups ))
```

本方法では，この基本機能は，次のような関数として定義される．

```
(group-id (transitive (child-group
 (filter groups (= (group-id groups) 'X)))))
```

この場合，transitive は，関数 child-group の返り値がなくなるまで，child-group を再帰的に適用することを示す．

４．３．３　複合オブジェクトに基づく実現

この実現方法では，製品情報を表現するための木構造を，複合オブジェクトとして論理的にまとまった単位として扱う．前述の２つの方法が，木構造を，いくつかの平坦な表を用いて表現したのに対し，複合オブジェクトによる実現方法は，一つの木構造をまとまった形で管理する．また，複合オブジェクトの集合を対象とする演算を関数として定義することにより，並列処理を実現する．

現在，複合オブジェクトに基づく実現方法として，以下に示す２種類の方法を検討している．

Ａ．リスト表現に基づく方法
Ｂ．ネットワーク表現に基づく方法

これらの方法におけるデータ表現と，データベース演算による基本機能の実現方法について示す．

A．リスト表現に基づく方法

この方法は，木構造をリストの形で表現する．

（１）データ表現

たとえば，製品情報が，図３のように表現される場合，データ表現は，リストの形で以下のように表現する．

```
(product1 attribute-values
   (g2 attribute-values
       (p1 attribute-values)
       (g6 attribute-values
               (p3 attribute-values)
               (p4 attribute-values)
               (p5 attribute-values))))

(product2 attribute-values
   (g14 attribute-values
       (g10 attribute-values
              (p7 attribute-values)
              (p8 attribute-values)
              (p9 attribute-values))
       (g13 attribute-values
              (p11 attribute-values)
              (p12 attribute-values))))

(product3 attribute-values
   (g20 attribute-values
       (p15 attribute-values)
       (g19 attribute-values
               (p16 attribute-values)
               (p17 attribute-values)
               (p18 attribute-values))))
```

ただし，attribute-values は，各節に含まれる属性値の列である．

（２）基本機能の実現

たとえば，基本機能 get-child は，与えられたノードに対応したリストに対し，入れ子となっているリストのうち，入れ子の深さが１であるものを取り出す演算により実現される．

基本演算 get-all-childs は，与えられたノードに対応したリストに対し，入れ子となっているすべてのリストを取り出す演算により実現される．

B．ネットワーク表現に基づく方法

この方法は，木構造の各ノード間をポインタで接続したデータ構造と，ノードと属性情報を結合したデータ構造として表現する．このデータ構造に対する検索は，ノードの各属性の抽出と，それをもとにした選択，ポインタの走査による木探索によって実現できる．

（１）データ表現

具体的なデータ構造を図６に示す．このように，木構造を一つのまとまりとして配置することにより，検索などの効率化が図られる．また属性情報を他所にまとめて配置することにより，各ノードの種類によるクラスタリングを実現できる．

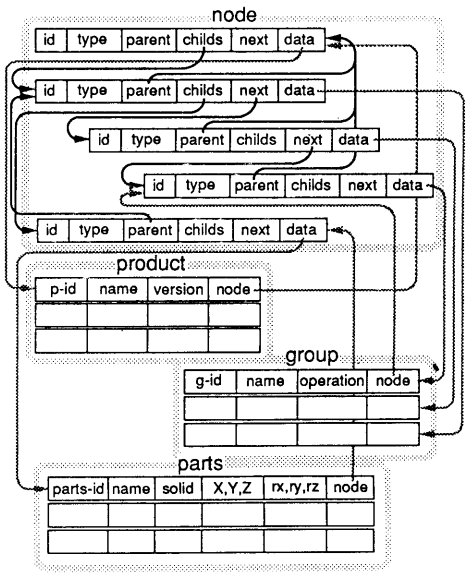

図６　ネットワーク表現によるデータ表現

（２）基本機能の実現

たとえば，与えられた製品に属する全ての部品名を求めるための処理は，木探索をする演算，与えられた種類のノードを取り出す演算，および，ノードの属性情報を取り出す演算の組み合わせにより実現できる．

この方法においては，基本機能は，ポインタ操作を中心とする関数により実現される．　基本機能 get-child については，子ノードのポインタを返す演算 child-node を設定する．

```
child-node(parent-node-pointer,
                              complex-object)
                  -> set of child-node-pointer
```

また，基本機能 get-all-childs については，ポインタをたどり再帰的に木探索をする演算　all-child-node を設定する．

```
all-child-node(root-node-pointer,
                              complex-object)
                    -> set of child-node-pointer
```

これらの演算を実際に SMASH 上で実現する場合，演算処理における並列性を考慮して，その粒度を決定する必要がある．

5．おわりに

本稿では，CAD データベースの SMASH 上への実現に向けて，データ構造および基本機能に関する検討を行った．今後，さらに次の項目に関する具体的な検討が必要である．

(1) CAD データベースを構成するデータの物理表現の設定．
(2) 基本機能を実現する各データベース演算の詳細設計，および，それらを定義する言語とその処理系の設計．
(3) 形状情報のためのデータ構造の設定，および，形状に関する問い合わせ処理の検討．
(4) 複合オブジェクトを支援するための SMASH の基本プリミティブの拡張．
(5) 本稿に示した３種類の実現方法の比較検討．

謝辞

本研究の中で，多くの助言を頂いた本学電子・情報工学系，大保信夫博士，山口和紀博士に深く感謝します．

参考文献：

[1] Kemper,A. and Wallrath,M., " An Analysis of Geometric Modeling in Database Systems", ACM Computing Surveys, Vol.19,No.1,pp47-91, March 1987.

[2] Kiyoki,Y., Kato,K. and Masuda,T.,"A relational database machine based on functional programming concepts", Proc. 1986 ACM-IEEE Computer Society Fall Joint Computer Conf., pp.969-978, November 1986.

[3] 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，pp.73-82, December 1988.

[4] Shipman,D., "The Functional Data Model and the Language DAPLEX", ACM TODS, Vol.6, No.1, pp.140-173, March 1981.

盛光印刷所