情 報 学 基 礎 50－5
（1998．7．17）

名前付き集合モデルを用いたシェイプの実現に関する一考察

宝珍 輝尚　　中田 充　　都司 達夫
福井大学 工学部 情報工学科
〒910-8507 福井市文京3丁目9－1

サイエンティフィックデータベース管理システムDREAMでは，集合を用いてデータを表現している．また，データベースの構造を記述しデータベースへの更新に従って変化するシェイプも一種の集合に基づいている．これまでに，DREAMのデータベースの要素ならびにシェイプを共通的に表現することを目的として，名前付き集合モデルを提案し，そのためのライブラリを実現してきた．本論文では，この名前付き集合モデルのためのライブラリを用いてデータベースの要素ならびにシェイプを実現した結果について報告する．データベースの要素ならびにシェイプのためのクラスを名前付き集合クラスのサブクラスとし，その構造とメソッドを利用することにより，実現が容易であることを示す．

On the Implementation of Shape by Using the Named Set Model

Teruhisa HOCHIN　　Mitsuru Nakata　　Tatsuo TSUJI

Dept. of Information Science, Faculty of Eng., Fukui University
3-9-1, Bunkyo, Fukui-shi, Fukui 910-8507 Japan

The scientific database management system called DREAM manages data by using sets. Shape, which is the information describing a database, and is changed according to the database updates, is also based on a set. The Named Set Model has been proposed to represent both kinds of database elements and shape in DREAM. The software library for this data model has been constructed. This paper describes the implementation of database elements and shape through this software library. The classes of database elements and shape are implemented as the subclasses of the class *Named_set*. Using the structure and the methods of named sets enables us to implement database elements and shape easily.

1 はじめに

近年の計算機の進歩により，CAD，VLSI設計，科学技術データ管理といった分野でもデータベース（ＤＢ）を利用したいという要求が強くなってきている．このような設計や研究開発に関する分野では，データの整理・分類が業務の主要な部分を占める．すなわち，仮説の設定，検証，修正というサイクルを繰り返し，うまくデータの整理・分類を行って何らかの情報を得ることが主な業務である．整理・分類が完了し，抽象度の高い情報が得られれば，業務はほぼ完了したといえる．

このような分野では，あらかじめＤＢ設計を行いスキーマ（データ定義）を完全に決定することは不可能である．なぜならば，スキーマは業務で得るべき情報によって初めて決定できるからであり，決定できたときには業務は終了しているからである．従って，このような分野では，データ挿入に先立ってスキーマを決定しなければならない従来のＤＢを利用するのは困難である．ここでは，データ定義を行わずとりあえずデータを格納し，後から整理を行いＤＢを構築し，その後はそのＤＢを利用してゆくことが要求される [1–5]．筆者らは，そのようなＤＢの実現を目的として，ＤＢをボトムアップに構築可能とし，トップダウンに利用可能とするためのデータモデル（DREAM モデル）を提案してきた [6–10]．DREAM モデルでは，未整理データを格納するＤＢ（基本ＤＢ）と基本ＤＢ中のデータを業務や研究の進展に応じて徐々に整理することを可能とするＤＢ（導出ＤＢ）という２種類のＤＢを考える [6, 7, 10]．基本ＤＢでは，基本的にデータを文字列で格納することで，データ定義を行わずにデータを格納可能とする [8]．導出ＤＢでは，従来，属性のｎ項組で表された事物を属性の集合としてとらえ，これにより，ＤＢの柔軟性を高めている．また，従来のデータ定義に相当しデータ主導で変化するシェイプの概念を導入することで，従来のＤＢが持つデータ定義に基づく問い合わせといったトップダウンなデータ利用を可能としている [9, 10]．さらに，導出ＤＢの構成要素とシェイプに共通に存在する構造に着目し，この構造に基づくデータモデル（名前付き集合モデル）を提案し [11]，その実現方法について報告してきた [12]．名前付き集合モデルを用いることにより，導出ＤＢの構成要素とシェイプを容易に実現できることが期待される．

本論文では，導出ＤＢの構成要素とシェイプを，名前付き集合モデルのためのライブラリを用いて実現した結果について報告する．名前付き集合モデルのライブラリを用いることによって，容易に，しかも，見通し良く，導出ＤＢの構成要素とシェイプを実現することができる．

以下，２で，DREAM モデル，ならびに，シェイプについて述べ，３で，名前付き集合モデルとその実現について述べる．そして，４で，導出ＤＢの構成要素とシェイプを名前付き集合モデルを用いて実現する方法について述べる．最後に５で，まとめを行い，今後の課題を示す．

2 DREAM

ここでは，本論文に関係する導出ＤＢについて概説する．

2.1 導出データベース：DB エレメント

データエレメント，名前付きエレメント，視点，オブジェクト，バンドルを導出ＤＢのＤＢエレメントと呼ぶ．

データエレメントは導出ＤＢにおいてデータ

の実体を格納する要素であり，導出ＤＢにおけるデータの取り扱いの最小単位である．導出ＤＢはデータエレメントの集合である．データエレメントは，３つ組 $(id, "", \{d\})$ で表される．ここで，id は識別子，""は空文字列，d はデータ値または指示エレメントである．d は集合の唯一の要素である．指示エレメントは，基本ＤＢ中の基本データの一部分を指すために用いる構造体である [13].

データエレメントに名前を付けたものが名前付きエレメントである．名前付きエレメントは，３つ組 $(id, name, S)$ で表される．ここで，id は識別子，$name$ は名前付きエレメントの名前，S はデータエレメントまたはオブジェクトの集合である．

名前付きエレメントの集合が視点である．視点は，３つ組 $(id, name, S)$ で表される．ここで，id は識別子，$name$ は視点の名前，Sは名前付きエレメントの集合である．

視点の集合がオブジェクトである．オブジェクトは，３つ組 $(id, name, S)$ で表される．ここで，id は識別子，$name$ はオブジェクトの名前，Sは視点の集合である．ただし，名前は空文字列であっても構わない．

オブジェクトの集合がバンドルである．バンドルは，３つ組 $(id, name, S)$ で表される．ここで，id は識別子，$name$ はバンドルの名前，S はオブジェクトの集合である．

導出データベースの例を図１に示す．これは，遺跡から得られた碗の破片データのデータベースである．基本ＤＢまたはファイル中に格納されたデータからデータ値がデータエレメントとして格納される．これに名前を付けることによって名前付きエレメントが得られる．ここでは，上から見て得られる情報と上下から見て得られる情報という２種類の情報の導出を考える．上から見て得られる情報には，口縁の外面の紋様の情報のみがある．上下から見て得られる情報には，これに加えて，底の外面の情報が含まれる．これらの２種の情報は，１つのオブジェクトの２つの視点（“top” と “both”）として格納される．視点 “top” は，“out_side_of_rim” という名前の名前付きエレメントを持つ．視点 “both’ は，“out_side_of_rim” と “out_side_of_bottom” という名前の名前付きエレメントを持つ．図１には，このような２つの視点を持つ３つのオブジェクトがある．図１の最左のオブジェクトの視点 “top” には，“weight” という名前付きエレメントが存在するが，他のオブジェクトには存在しない．このような不規則性を持つ構造を扱えることに注意されたい．次に，これらのオブジェクトを分類することを考えよう．Cup1 という名前のバンドルには，名前付きエレメント

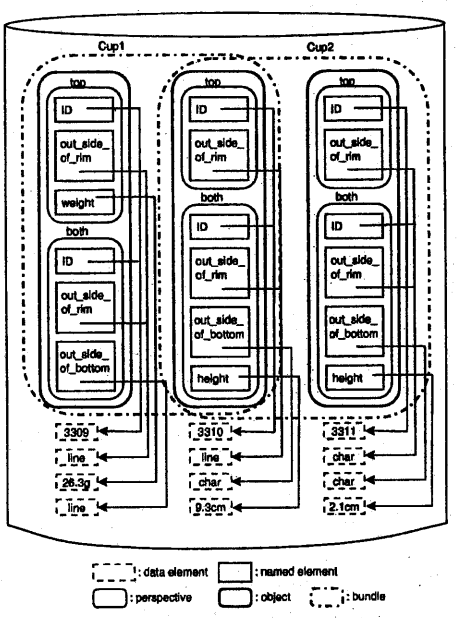

図 1: 導出データベースの例

"out_side_of_rim" のデータとして "line" があるオブジェクトを入れ，Cup2 という名前のバンドルには，名前付きエレメント "out_side_of_bottom" のデータとして "characters" があるオブジェクトを入れるもととする．得られるバンドルを図 1 に示す．

2.2 シェイプ

シェイプとは，データの形を記述する情報である．ここでの情報とは名前とデータ型である．

シェイプの基本単位であるシェイプのエントリは 3 つ組 $(id, name, DT)$ で表される．ここで，id は識別子，$name$ は名前付きエレメントの名前，DTは名前付きエレメントのデータ型またはシェイプの集合である [9, 10]．ある視点のシェイプは 3 つ組 $(id, name, S)$ で表される．ここで，id は識別子，$name$ は視点の名前，S はその視点に含まれる名前付きエレメントに対するシェイプエントリの集合である．あるオブジェクトのシェイプは 3 つ組 $(id, name, S)$ で表される．ここで，id は識別子，$name$ はオブジェクトの（空文字列でありうる）名前，Sはそのオブジェクトに含まれる視点のシェイプの集合である．あるバンドルのシェイプは 3 つ組 $(id, name, S)$ で表される．ここで，id は識別子，$name$ はバンドルの名前，Sはそのバンドルに含まれるオブジェクトの視点のシェイプの融合集合である．ここで，融合集合とは，要素となるシェイプエントリの名前はすべて異なるような集合である．すなわち，名前が同じ 2 つのシェイプエントリ (id_1, nm, DT_1) と (id_2, nm, DT_2) は，$(id_{new}, nm, DT_1 \cup DT_2)$ となる．

図 1の名前付きエレメントに対するシェイプエントリを以下に示す．

```
( o30, "ID", {int} )
( o31, "out_side_of_rim", {string} )
( o32, "weight", {float} )
( o33, "ID", {int} )
( o34, "out_side_of_rim", {string} )
( o35, "out_side_of_bottom",{string})
( o36, "height", {float} )
```

以下は，図 1の視点のシェイプである．

```
( o37, "top", {o30, o31, o32} )
( o38, "both", {o33, o34, o35, o36} )
```

次に，図 1のバンドルのシェイプを以下に示す．

```
( o39, "Cup1", {o37, o38} )
( o40, "Cup2", {o37, o38} )
```

次に，シェイプの振舞いについて述べる．シェイプは，現在のシェイプに存在しない名前やデータ型を持つ名前付きエレメントが挿入されると，その名前やデータ型が含まれるように変化する．この場合，名前付きエレメントが挿入されたオブジェクトのシェイプが変わり，これに伴い，視点やバンドルのシェイプにも変更が必要な場合にはそれらが変化する．また，名前付きエレメントの削除によりその名前付きエレメントの名前やデータ型を持つような名前付きエレメントが存在しなくなった場合に，シェイプからその名前やデータ型は削除されなければならない．

シェイプはデータの挿入・削除・変更によって変化する．これは，柔軟性という点で有効であるが，データの削除によって属性がなくなるという問題や，ＤＢ設計手法を利用できないという問題がある．そこで，前述の 2 つの条件のうち条件 1 のみを満足する，硬シェイプと呼ぶシェイプを導入する．硬シェイプでは，データの削除によってシェイプを変更しなくても構わ

ないし，データ挿入に先立ってシェイプが存在していても構わない．硬シェイプを導入することで，柔軟性を保ちつつ，従来からのＤＢ設計手法を利用でき，また，必要な属性の削除を避けることができる．

3 名前付き集合モデル

名前付き集合は，３つ組 ($id, name, S$) である．ここで，id は識別子，$name$ は名前，S名前付き集合または値の集合である．名前付き集合の例は，($o1$,"$length$",$\{12, 12.3, o2\}$) である．ここで，$o1$ はこの名前付き集合の識別子であり，$o2$ は他の名前付き集合の識別子である．

ここでは，記号θを，値の比較のための二項演算子 $=, \neq, <, >, \leq, \geq$を表すものとし，記号$\sigma$を$\ni, \not\ni$，記号$\lambda$を，集合を比較する演算子$=, \neq, \subset, \supset, \subseteq, \supseteq$ を表すものとする．

演算子 .id, .name, .S は，各々，名前付き集合の識別子，名前，集合を取り出す．また，演算子->NS と->DATA は，各々，名前付き集合の名前付き集合の集合と値の集合を取り出す．これらの演算子は，以下に示すように名前付き集合の集合に拡張することができる．

$NS.\mathtt{id} = \{ns.\mathtt{id} \mid ns \in NS\}$

$NS.\mathtt{name} = \{ns.\mathtt{name} \mid ns \in NS\}$

$NS.\mathtt{S} = \bigcup ns.\mathtt{S}$, where $ns \in NS$

$NS\mathtt{->NS} = \bigcup ns\mathtt{->NS}$, where $ns \in NS$

$NS\mathtt{->DATA} = \bigcup ns\mathtt{->DATA}$, where $ns \in NS$

ここで，NSは名前付き集合の集合である．

名前付き集合の比較には，識別子による比較，名前による比較，ならびに，集合による比較がある．これらを以下に示す．

$ns_1\ \theta_{id}\ ns_2 \equiv ns_1.id\ \theta\ ns_2.id$

$ns_1\ \theta\ ns_2 \equiv {}^{\forall}s \in ns_1.S\ {}^{\exists}r \in ns_2.S\ (s\ \theta\ r)$
$\wedge\ {}^{\exists}s \in ns_1.S\ {}^{\forall}r \in ns_2.S\ (s\ \theta\ r)$

$ns_1\ \theta_{name}\ ns_2 \equiv$
$ns_1.name = ns_2.name\ \wedge$
${}^{\forall}s \in ns_1\mathtt{->NS}\ (\ {}^{\exists}r \in ns_2\mathtt{->NS}\ (s\ \theta_{name}\ r)\ )\wedge$
${}^{\exists}s \in ns_2\mathtt{->NS}\ (\ {}^{\forall}r \in ns_1\mathtt{->NS}\ (s\ \theta_{name}\ r)\ )\wedge$
${}^{\forall}s \in ns_1\mathtt{->DATA}\ (\ {}^{\exists}r \in ns_2\mathtt{->DATA}\ (s\ \theta\ r)\ )\wedge$
${}^{\exists}s \in ns_2\mathtt{->DATA}\ (\ {}^{\forall}r \in ns_1\mathtt{->DATA}\ (s\ \theta\ r)\ )$

名前付き集合の集合を比較する演算を以下に示す．

$S\ {}^{\exists}\theta_x{}^{\exists}\ R \equiv {}^{\exists}s \in S\ (\ {}^{\exists}r \in R\ (s\ \theta_x\ r)\ )$

$S\ {}^{\exists}\theta_x{}^{\forall}\ R \equiv {}^{\exists}s \in S\ (\ {}^{\forall}r \in R\ (s\ \theta_x\ r)\ )$

$S\ {}^{\forall}\theta_x{}^{\exists}\ R \equiv {}^{\forall}s \in S\ (\ {}^{\exists}r \in R\ (s\ \theta_x\ r)\ )$

$S\ {}^{\forall}\theta_x{}^{\forall}\ R \equiv {}^{\forall}s \in S\ (\ {}^{\forall}r \in R\ (s\ \theta_x\ r)\ )$

ここで，Sと Rは名前付き集合の集合であり，x は id, $name$, または $null$のいずれかである．これらの演算子をまとめて $S\Theta R$ と表す．

次に名前付き集合モデルの演算について述べる．演算 $new(name, S)$ は，名前 $name$ と集合 Sを持つ新たな名前付き集合を生成する．新しい名前付き集合には識別子が与えられる．演算 $del(id)$ は，識別子が id の名前付き集合を削除する．

次に，名前付き集合の追加，削除，フィルタについて述べる．名前付き集合 ns_iに対する名前付き集合 ns_jの追加 (ns_i <+ ns_j) は，ns_i を $(id_i, ns_i.\mathtt{name}, ns_i.\mathtt{S} \cup ns_j.\mathtt{S})$ にすることである．ns_iからの ns_jの削除 (ns_i <- ns_j) は，ns_i を $(id_i, ns_i.\mathtt{name}, ns_i.\mathtt{S} - ns_j.\mathtt{S})$ にすることである．ns_jによる ns_iのフィルタ (ns_i <* ns_j) は，ns_iを $(id_i, ns_i.\mathtt{name}, ns_i.\mathtt{S} \cap ns_j.\mathtt{S})$ にすることである．

同一の名前をもつ名前付き集合を一つの名前付き集合にしたいことがある．これを行うのが，融合である．名前付き集合の集合 NSの融合 ($un(NS)$) は，各要素が $nm_i \in NS.\mathtt{name}$ なる名前 nm_iを持ち，その要素は NS中の名前 nm_i

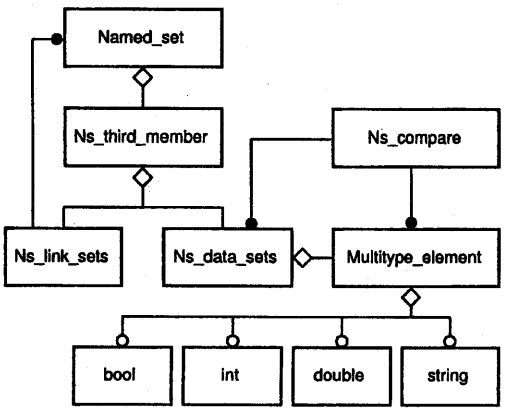

図 2: 名前付き集合モデルのためのオブジェクト図

をもつ名前付き集合中の集合の和を集合として持つような名前付き集合の集合である．本演算は，値を含めるように容易に拡張できる．

次に，選択演算を示す．

$NS[\,\mathtt{id}\ \theta\ x\,] \equiv \{ns \mid ns \in NS \wedge ns.\mathtt{id}\ \theta\ x\}$

$NS[\,\mathtt{name}\ \ \theta\ x\,] \equiv$

$\{ns \mid ns \in NS \wedge ns.\mathtt{name}\theta x\}$

$NS[\,\mathtt{S}\ \sigma\ x\,] \equiv \{ns \mid ns \in NS \wedge ns.\mathtt{S}\ \sigma\ x\}$

$NS[\,\mathtt{S}\ \Theta\ R\,] \equiv \{ns \mid ns \in NS \wedge ns.\mathtt{S}\ \theta\ R\}$

$NS[\,\mathtt{S}\ \lambda\ R\,] \equiv \{ns \mid ns \in NS \wedge ns.\mathtt{S}\ \lambda\ R\}$

ここで，NSは名前付き集合，x は値もしくは識別子，Rは名前付き集合の集合である．

名前付き集合の集合の直積 $NS_1 \times NS_2$は，

$\{\ new(\ ns_1.\mathtt{name},\ ns_1.\mathtt{S} \cup ns_2.\mathtt{S}\ ) \mid ns_1 \in NS_1 \wedge ns_2 \in NS_2\}$ である．また，結合 $NS_1[\Theta]$ NS_2は，$\{\ new(\ ns_1.\mathtt{name}, ns_1.\mathtt{S} \cup ns_2.\mathtt{S}\ ) \mid ns_1 \in NS_1 \wedge ns_2 \in NS_2 \wedge ns_1.\mathtt{S}\Theta ns_2.\mathtt{S}\}$ である．

名前付き集合モデルのためのライブラリをC++を用いて実現している [12]．オブジェクト図を図 2に示す．名前付き集合には値に加えて名前付き集合（の識別子）が入り得る．ここでは，値の集合（"Ns_data_set"）と識別子の集合"Ns_link_set"）に分けてこれらを管理している．また，値として様々なものが入り得るので，"Multitype_element"クラスを導入して値の共通化を図っている．

4 実装

名前付き集合モデルを使用することで，DB エレメントならびにシェイプを容易に実現できる．名前付き集合モデルを使用した DB エレメントならびにシェイプのオブジェクト図を図 3 に示す．ただし，煩雑になるのでメソッドを省略している．"Named_set"が名前付き集合クラスである．

"dbElm"クラスは，今回作成した全てクラスのスーパクラスであり，"Named_set"クラスのサブクラスである．ここでは，単に，要素の種別を格納できるようにしている程度である．"dbElm" クラスを継承して，"shape"クラス，"bundle"クラス，"obj"クラス，"persp"クラス，"namedElm" クラス，"dataElm"クラス，"dbType"クラス，"dbValue"クラスを作成している．

"bundle"クラス，"obj"クラス，"persp"クラス，"namedElm"クラス，"dataElm"クラスは，おのおの，バンドル，オブジェクト，視点，名前付きエレメント，データエレメントのためのクラスである．

"dbType"クラスはデータエレメントのデータ型のためのクラスであり，"dbValue"クラスはデータエレメントの値のためのクラスである．ここでは，"dbType"クラスと"dbValue"クラスも名前付き集合として実現している．すなわち，集合の要素として一つの要素のみを持つ名前付き集合として実現している．従って，"dataElm"

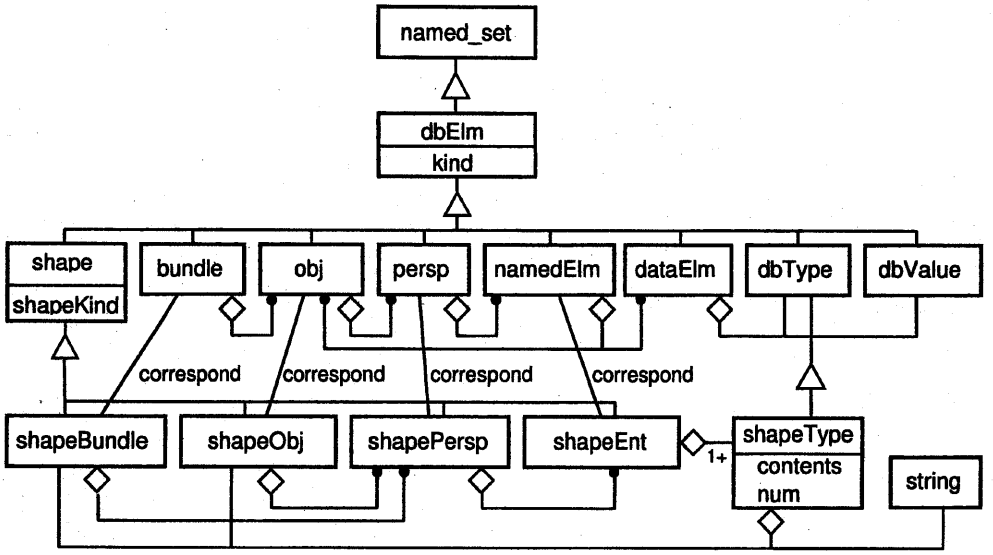

図 3: オブジェクト図

クラスのインスタンスは，"dbType"クラスのインスタンスと"dbValue"クラスのインスタンスを一つずつ持つ．

"shape"クラスはシェイプ関係のクラスのスーパクラスである．本クラスでは，通常のシェイプか硬シェイプかの種別を持っている．

"shapeBundle"クラス，"shapeObj"クラス，"shapePersp"クラス，"shapeEnt"クラスは，おのおの，バンドルのシェイプ，オブジェクトのシェイプ，視点のシェイプ，シェイプエントリのためのクラスである．

"shapeType"クラスは，シェイプエントリにおけるデータ型ならびに他のシェイプの識別子を管理するクラスである．格納されているのがデータ型を表す文字列かシェイプの識別子かを識別する情報を属性 contents として持っている．また，データの挿入・削除によって変化するシェイプを実現するために，被参照カウンタ（属性 num）を利用している．名前付きエレメントが挿入されるとカウンタをインクリメントし，削除されるとデクリメントする．本カウンタが０となるとその"shapeType"クラスのインスタンスは削除される．硬シェイプの場合は，"shapeType"クラスのインスタンスが一旦作成されるとシステムが自動的にこれを削除する必要はないので，このカウンタを使用する必要はない．その代わりに，利用者が明示的に削除するメソッドを用意している．"shapeType"クラスは"dbType"クラスを継承して作成している．

全てのクラスにおいて，名前の設定，ならびに，集合への要素の追加・集合からの要素の削除が必要であるが，"Named_set"クラスのメソッドを利用することで問題なく実現できた．また，シェイプの実現に不可欠である融合集合は，名前付き集合モデルの強和演算を用いることで容易に実現できた．実現が容易であることは，各クラス独自の属性がほとんどないことからも分かる．

5 おわりに

本論文では，名前付き集合モデルのためのライブラリを用いた導出ＤＢの構成要素とシェイプの実現について述べた．名前付き集合モデルのためのライブラリを用いることによって，容易に，しかも，見通し良く，導出ＤＢの構成要素とシェイプを実現することができた．

今後は，検索機構の実現，シェイプの効率の良い管理，導出過程のデータモデルにおける表現に関する検討，ならびに，実際の業務への適用が課題である．

謝辞

本研究は，一部，文部省科学研究費（課題番号 09204113）による．

参考文献

[1] Zdonik, S. B. : Incremental Database Systems: Database from the Ground Up, Proc. of ACM SIGMOD 1993, pp. 408–412 (1993).

[2] Shoens, K., Luniewski, A., Schwarz, P., Stamos, J. and Thomas, J. : "The Rufus System: Information Organization for Semi-Structured Data," Proc. of the 19th VLDB Conf., pp. 97–107 (1993).

[3] Hachem, N. I., Gennert, M. A., and Ward, M. O. : "Managing Derived data in the Gaea Scientific DBMS," Proc. of the 19th VLDB Conf., pp.1–12 (1993).

[4] 田中 克巳，西尾 章治郎，吉川 正俊，下条 真司，上善 恒雄：Obase: 動的継承とアクティブルール機構を有するインスタンス指向オブジェクトＤＢシステム，情処研報 DBS-100-9, pp. 87–96 (1994).

[5] 上島 紳一，大月 一弘，森下 淳也，田中 克己：歴史的資料を対象としたサイエンティフィックＤＢのシステム設計，信学技報，DE93-47 (1993).

[6] 中田 充，宝珍 輝尚，都司 達夫 サイエンティフィックＤＢのためのデータモデルの一提案，情処研報，DBS-101-9, pp. 65-72 (1994).

[7] 中田 充，宝珍輝尚，都司達夫: サイエンティフィックデータモデルの一評価，信学技報，DE95–64, Vol. 95, No. 287, pp. 113–120 (1995).

[8] 中田 充，宝珍 輝尚，都司 達夫: サイエンティフィックＤＢのための一次データの一管理法，情処学第 52 回全大，2Q-2 (1996).

[9] 宝珍 輝尚，中田 充，都司 達夫：インスタンスベースのＤＢにおける柔軟なスキーマについて，情処学第 53 回全大，3R-6 (1996).

[10] Mitsuru Nakata, Teruhisa Hochin, Tatsuo Tsuji : Bottom-up Scientific Databases Based on Sets and Their Top-down Usage, Proc. of International Database Engineering & Applications Symposium, pp. 171–179 (1997.8).

[11] 中田 充，宝珍 輝尚，都司 達夫：名前付き集合モデルを用いた DREAM モデルの定義，情報処理学会研究報告，データベースシステム研究会 112-1, 情処研報，Vol. 97, No. 38, pp. 1–8 (1997-05).

[12] 中田 充, 宝珍 輝尚, 都司 達夫：半構造データを柔軟に管理可能なデータモデルの実現，情報処理学会データベースシステム研究会研究報告，DBS114-14, 97-104 (1998.1)

[13] 原田 正則, 宝珍 輝尚, 中田 充, 都司 達夫：データ型に基づくマルチメディアデータ参照機構の実現とその有効性，情報処理学会論文誌，Vol. 38, No. 8, 1603–1612 (1997.8)

[14] Goldman, R. and Widom, J. : "DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases," Proc. of the 23rd VLDB Conf., pp. 436–445 (1997).

[15] Abiteboul, S. : "Querying Semi-Structured Data," Proc. of the 6th Int'l Conf. on Database Theory, pp. 1–18 (1997).

[16] Peter Buneman, Susan Davidson, Mary Fernandez, and Dan Suciu : "Adding Structure for Unstructured Data," Proc. of the 6th Int'l Conf. on Database Theory, pp. 336–350 (1997).