データベースシステム 114－14
（１９９８．１．２０）

半構造データを柔軟に管理可能なデータモデルの実現

中田 充† 岩井信輔†† 宝珍輝尚†† 都司達夫††
†福井大学大学院工学研究科 ††福井大学工学部情報工学科

本論文では，半構造データを管理するためのデータモデルである名前付き集合モデルの実現について述べる．名前付き集合モデルでは，データは３つ組(識別子，名前，集合)である名前付き集合で表現される．集合をモデルの基盤とすることでデータの表現に柔軟性を持つ．さらに，集合の要素についての比較演算を持つことで，集合同士の比較を容易に行うことができる．名前付き集合モデルではデータ型の異なる要素も一つの集合に属する．そのため，集合をデータ型毎にまとめられた要素の集合の和集合として表現し，それぞれの型の比較演算を実現することで名前付き集合同士の比較演算を実現している．現在は同じ型同士の要素のみの比較を行っているが，異なるデータ型同士の比較をユーザが独自に定義することも可能である．

Implementation of a Data Model for Semi-structured Data

Mitsuru NAKATA† Shinsuke IWAI†† Teruhisa HOCHIN†† Tatsuo TSUJI††
†Graduate School of Engineering, Fukui University
††Department of Information Science, Fukui University

This paper describes the implementation of *named set model*. Named Set model is a suitable data model for managing semi-structured data. A *named set* is a triplet (*id, name, S*), where *S* is a set of named sets and/or data values. Named set model has high flexibility because it is based on the set theory. In named set model, elements of various data types can belong to a set. A set is treated as a union of the sets of the elements having the same data type, and the operations comparing the elements in each set are provided to realize the operations comparing named sets. Although the only operations comparing the elements of the same data types is currently, those comparing the elements of different data types can be defined by users.

1 はじめに

データベースの応用分野のうち，データベース化が求められているにもかかわらず，従来のデータベース管理システムではデータベース化が困難な応用分野がある．実験データや測定データなどのサイエンティフィックデータを扱う応用分野はその一つである．サイエンティフィックデータの管理が困難である理由の一つに，それが厳密な構造を持たない半構造データ[1]であることが挙げられる．上記の応用分野において，DBMSを利用する目的は，データを整理分類して管理し，そこから新しいデータを作成したりデータを参照することを容易にすることである．したがって，大量のデータから意中のデータを検索したり，データの操作が容易でなければならない．従来のDBMSは，データ格納の前に定義したスキーマを用いたデータ管理を行っているが，半構造データベースでは，データ格納に先立ってあらかじめデータベースのスキーマを決定することが困難である[4]．そのため，徐々に構造を定義しながらデータベースを構築できる必要がある．また，半構造データは複数の側面を持ちうるため，データ同士の関係や意味のある情報単位となるデータの集まり（以降，これをオブジェクトと呼ぶ）においてこれらを表現できることが必要となる．さらに，半構造データの整理分類は，それを行うユーザの視点や考え方に沿って行われるが，それらの間でのデータ共有が可能であることが求められる．これらの理由により，半構造データベースでは，従来のデータモデルよりも柔軟なデータモデルが必要とされている．

筆者らは，これらの要求を満たすサイエンティフィックデータのためのデータモデルを提案し，それを用いたデータベース管理システムDREAMを設計制作してきている[5-7]．DREAMのモデルには，データを格納するデータエレメント，データエレメントの集合に名前を付ける名前付きエレメント，対象物の一つの側面を表す視点，一つの対象物を表すオブジェクト，オブジェクトの集合を表すバンドルという要素がある．データエレメント以外の要素は，（識別子，名前，集合）の3つ組という非常によく似た構造であるにもかかわらず若干の違いがあるため，DREAMを実現する場合には，それぞれの要素に対して同様のプログラムを複数作成しなければならず，煩雑になるという問題があった．これに対して筆者らは，DREAMモデルの殻となるより抽象度の高いデータモデルを明確化し，これを用いてDREAMモデルを再定義することでこの問題を解決した[8]．殻となるデータモデルは，3つ組（id，name，S）を基礎としている．この3つ組を“名前付き集合”と呼ぶ．ここで，idは名前付き集合の識別子であり，nameは名前，Sは名前付き集合を構成する集合である．また，Sはデータ以外に既存の名前付き集合を含むことが可能である．名前付き集合を用いたデータモデルを名前付き集合モデルと呼ぶ．名前付き集合モデルでは，要素の基盤に集合を採用したことにより，柔軟なデータ表現が可能である．また，名前付き集合モデルは，集合演算を用いた新しい演算を持つことが特徴である．たとえば，名前付き集合同士の比較では，名前付き集合に含まれる集合を構成する各要素ごとに比較を行いその結果により，名前付き集合の比較を行う．また，名前付き集合同士の和演算は，両方の名前付き集合の第3項Sに含まれる要素を持つ新たな名前付き集合を作成する．これにより，複数のオブジェクトに含まれるすべての属性を持つオブジェクトを作成するといった演算が容易に表現できる．このように，名前付き集合モデルの演算は，集合で表されたデータの操作性が高い．このように，名前付き集合モデルは半構造データに適したデータモデルである．

本論文では，名前付き集合モデルを用いたデータ管理機構の実現について述べる．以降，2では本論文の前提として，名前付き集合モデルの対象となる半構造データの定義を行い，そのあと，名前付き集合モデルについて演算を中心に説明する．3では名前付き集合モデルの実現について述

べる．4はまとめである．

2 前提

2.1 半構造データ

半構造データとは以下の特徴を持つデータである．

1) 統一的な構造やフォーマットを持たないデータであるが，特定のアプリケーションからは構造を持つと見なすことが可能である．
2) 複数のデータを統一的に扱える構造を考えることが困難である．
3) 複数の側面を持つことが多く，整理分類が困難である．

1)は，基本的には構造を持たない文字列やビット列，バイト列などのデータであるが，キーワードやタグ情報を持ち，それらを理解できる特定のアプリケーションからは構造を持つデータと見なすことが可能であることを指す．たとえば，電子メールのテキストは，それをメールとしてみた場合は，宛先，サブジェクト，本文などの項目からなる構造を持つデータであると見なせるが，それ以外の場合は，単なる文字列データである．2)は，それぞれのデータは構造を持っているが，データの集合を考えた場合には，それらを統一的に扱える構造を考えることが困難であることを指す．つまり，個々のデータの項目名やデータ型などのフォーマットが一様ではなく，データの増加によりデータの集合を扱うための構造が頻繁に変化し決定することが困難であることを指す．3)は，半構造データは，それが一つのオブジェクトを表すデータであっても，解釈や見方が複数存在し，データを整理分類することが困難であることを指す．これにより，半構造データの多くが，未整理，整理途中のデータであるといえる．未整理・整理途中のデータであるということは，矛盾や冗長な情報を含むかもしれないことを指す．

まとめると，半構造データとは，複数の側面を持ち，整理されておらず，統一的にそれらの集合を扱う構造を定義することが困難なデータであるといえる．

2.2 名前付き集合モデル

ここでは，名前付き集合モデルについて述べる．名前付き集合は，識別子(id)，名前(name)，集合(S)からなる3つ組である．名前付き集合の定義を以下に示す．

[定義 1] 名前付き集合

名前付き集合は，3つ組(id, name, S)である．idは名前付き集合の識別子，nameは名前，Sは名前付き集合または値からなる集合である． □

それぞれの名前付き集合は独立して存在する．これにより，他の名前付き集合とデータを共有する名前付き集合を定義できる．また，名前付き集合の第3項Sは値または名前付き集合の集合であるため，集合S中に同じ値や同じ名前付き集合が複数個存在することはない．

図 1は，ある学生を現す名前付き集合の例である．名前付き集合モデルでは，オブジェクトのの属性も一つの名前付き集合として表され，オブジェクトはそれらを含む名前付き集合として表される．

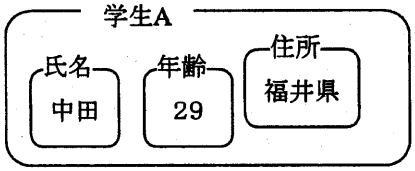

図 1 学生オブジェクトを表す名前付き集合

つぎに，名前付き集合の演算のうち特徴的なものを具体的な例を挙げて説明する．

名前付き集合のある要素を指す航行演算としてピリオド"."がある．名前付き集合 ns_x : (id_x, "$name_x$", S_x)があるとき，ns_x.id は id_x を指し，ns_x.name は $name_x$ を指す．ns_x.S は S_x を指し，ns_x.DATA は S_x の部分集合であるようなデータの集合である．ns_x.NS も同様の名前付き集合の集合である．.

図 2のような陶磁器の破片のオブジェクトを表現する名前付き集合を考える．破片の形状が異なるため，それぞれの破片オブジェクトの持つ属性も異なる．破片A, Bを表現する名前付き集合

を，$ns_a : (id_a, \text{“A”}, S_a)$, $ns_b : (id_b, \text{“B”}, S_b)$とすると，$S_a$と$S_b$は以下のような名前付き集合の集合である．

S_a={(id1,“種別”,{碗}), (id2,“重さ”,{80}),
(id3,“紋様”,{唐草紋}), (id4,“口の形状”,{端反り})}
S_b={(id5,“種別”,{碗}), (id6,“重さ”,{120}),
(id7,“紋様”,{芭蕉葉紋}), (id8,“底の半径”,{3.5})}

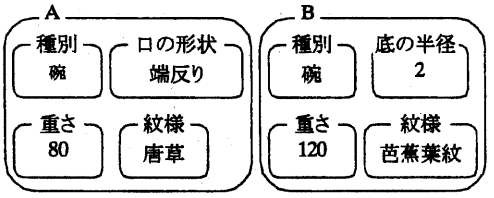

図 2 学生オブジェクトを表す名前付き集合

ここで，破片A,Bが同じ陶器の上半分と下半分の破片であることが判明した場合，名前付き集合モデルでは，2つの名前付き集合の和をとることで，もとの陶器を表す名前付き集合を作成できる．名前付き集合の和ns_a+ns_bは，新たな名前付き集合を作成する．作成される新しい名前付き集合を$ns_x : (id_x, name_x, S_x)$とすると，$ns_x.name = \text{“A”}$, $ns_x.S = S_a \cup S_b$である．

しかし，「紋様」や「重さ」の属性を現す名前付き集合のように，一つのオブジェクトを表す名前付き集合の中に，同じ名前を持つ名前付き集合が属することは，データへのアクセスを考えると好ましくない．そこで，名前付き集合の集合NSに対して，名前が同じ名前付き集合をまとめる演算un(NS)を考えている．un(NS)は集合NS_Aと集合NS_Bの和集合で定義される．ここで，NS_Aは，集合NSの要素のうち，他に同じ名前を持つものがない名前付き集合からなる集合であり，NS_Bは，同じ名前を持つもの同士の名前付き集合の和演算により作成された名前付き集合の集合である．以降では，この演算を名前付き集合の統合と呼ぶ．また，統合を利用した名前付き集合の和演算を強和と呼ぶ．

強和$ns_a(+)ns_b$は，両方の名前付き集合の性質を持つ新たな名前付き集合を作成する．名前付き集合の和の演算との違いは，第3項の集合Sに名前が同じ複数の名前付き集合が属する場合，それらが一つにまとめられる点である．すなわち，$ns_x.S = ns_a.DATA \cup ns_b.DATA \cup un(ns_b.NS \cup ns_b.NS)$である．ただし，$ns_a.DATA$は$ns_a$の第3項に含まれる値の集合であり，$ns_a.NS$は$ns_a$の第3項に含まれる名前付き集合の集合である．したがって，個々での例では，$ns_x.S$ = {(id9,“種別”,{碗}), (id10,“重さ”,{80, 120}), (id11,“紋様”,{唐草紋, 芭蕉葉紋}), (id12,“口の形状”,{端反り}), (id13,“底の半径”, {3.5})}となる．

2つの名前付き集合ns_a, ns_bの比較には，1)名前付き集合の識別子の比較$ns_a \theta_{id} ns_b$と2)名前付き集合の第3項の集合同士の比較$ns_a \theta ns_b$，3)第3項の集合のうち名前が同じもの同士の比較，がある．ここで，θは比較演算子であり，$<, \leq, =, \geq, >, \neq$のいづれかである．それぞれの比較は以下のように定義される．

$$ns_1 \theta_{id} ns_2 \equiv ns_1.id\ \theta\ ns_2.id \quad \cdots (1)$$

$$ns_1 \theta\ ns_2 \equiv ns_1.S\ {}^{\forall}\theta^{\exists}\ ns_2.S \wedge ns_2.S\ {}^{\forall}\theta'^{\exists}\ ns_1.S \quad \cdots (2)$$

$$ns_1 \theta_{name} ns_2 \equiv ns_1.name\ \theta\ ns_2.name \wedge ns_1.NS\ {}^{\forall}\theta^{\exists}_{name}\ ns_2.NS \wedge ns_2.NS\ {}^{\forall}\theta^{\exists}_{name}\ ns_1.NS \wedge ns_1.DATA\ {}^{\forall}\theta^{\exists}\ ns_2.DATA \wedge ns_2.DATA\ {}^{\forall}\theta'^{\exists}\ ns_1.DATA \quad \cdots (3)$$

$ns_1.S\ {}^{\forall}\theta^{\exists}\ ns_2.S$は，$ns_1.S$のすべての要素に対してθが成立するような要素が$ns_2.S$中に少なくとも一つ持つ場合に真となる．つまり，$(\forall x \in ns_1.S)((\exists y \in ns_2.S)(x\ \theta\ y))$である．θ'はθが=または≠のときはθ'はθと同じであり，不等号の場合はθの逆（θが“<”のときは“>”）である．以降では，(1)〜(3)の比較を$ns_1 \theta_x ns_1$と表し，${}^{\forall}\theta^{\exists}$や${}^{\exists}\theta^{\exists}_{name}$などの集合の要素に対する比較を$\Theta_x$と表す．$\Theta_x$には上記以外に${}^{\forall}\theta^{\forall}$などが存在するがここでは省略する．

たとえば，名前付き集合ns_1 : (id1, name1, {1,2,3}), ns_2 : (id2, name2, {2,3,4}), ns_3 : (id3, name3, {1,2,3})について$ns_1 < ns_2$と$ns_1 = ns_3$が成立する．またns_1を(id1, “A”, {(id4,“重さ”,{20}), (id5,“長さ”,{5})})，ns_2を(id2, “A”, {(id6,“重さ”,{30}), (id7,“長さ”,{6})})とすると，$ns_1 <_{name} ns_2$が成立する．これにより，複数の属性を持つオブジェクトの比較が単純に表

記できる．

名前付き集合の集合NSから意中の名前付き集合を選択する演算は，NS[φ]である．ここで，φは選択の条件を表す比較演算である．たとえば，名前付き集合の集合NSから「重さが25より大きいオブジェクトを表す名前付き集合」を選択する演算は，NS[S[name=重さ][DATA$^{\forall}>^{3}$ {25}] ≠ φ]である．この演算は，厳密には名前付き集合の集合NSから「名前が重さであり，そのデータの集合のすべての要素が2よりも大きい」という名前付き集合を含む名前付き集合を選択する演算である．したがって，名前が重さであり，25以下の要素が一つでも存在する名前付き集合を含むものはこの演算では選択されない．

3 名前付き集合モデルの実現

この章では，名前付き集合モデルのインプリメントについて述べる．

3.1 システム設計

名前付き集合モデルを採用したデータベース管理システムのおおまかなモジュール構成を図3に示す．

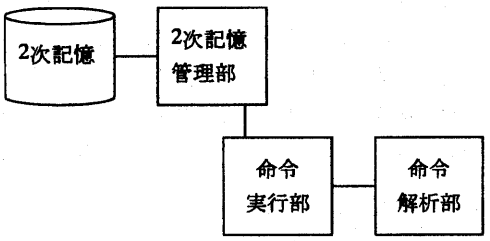

図3 管理システムのモジュール構成

命令解析部

命令解析部は，ユーザとのインターフェースである．具体的には，ユーザが入力した命令を解析して名前付き集合モデルの演算に変換し，それを命令実行部に渡し結果を表示する．

命令実行部

命令解析部から渡された演算を実行し結果を返す．命令解析部とのインターフェースのため，命令実行部は名前付き集合を操作するライブラリ関数群を提供する．強和をはじめとする名前付き集合の演算や検索は，中間結果を大量に作成する．そのため，命令実行部はメモリ中に中間結果である名前付き集合のイメージを作成し，それに対してつぎの演算を行う．演算の結果，既存の名前付き集合に副作用が生じた場合は，2次記憶管理部に名前付き集合の更新・削除のメッセージを発行する．

2次記憶管理部

2次記憶管理部は，実際に名前付き集合を2次記憶中に格納したり，検索することが役割である．命令実行部からのメッセージを受取，2次記憶中の名前付き集合と命令実行部のメモリ中の名前付き集合との整合を取る．

現在は，命令実行部のプロトタイプの実現を行っている．開発環境はAxil Ultima1(256Mbyte)，OSはSoralis2.5.1であり，開発用の言語はC++を用いている．作成が容易であることと，命令実行部では中間結果を示すメモリ中に格納された名前付き集合に対する演算の適応が必要になるといった理由から，プロトタイプではすべての名前付き集合がメモリ中に格納されている．

命令実行部を設計する際に，以下の点を考慮した．

データの扱い

データ型が異なるデータを一つの集合として扱える必要がある．さらに，集合同士の比較を考える場合，異なるデータ型同士の要素の比較をどのように行うかを定義しなければならない．現状では，データ型が異なる要素同士の比較は必ず偽となる．しかし，ユーザによっては，int型とstring型の数字とを比較したいといったことがあるかもしれないし，int型とdobule型は両方ともdouble型として比較したいという要求があるかもしれない．このように，異なるデータ型同士の比較を行う場合の基準は，一般的に決められるものではなく，データを利用するユーザによって変わってくる．そのため，これらの関係を記述できるような機構が今後必要とされる．

また，名前付き集合は集合を基にしているため，比較などにおいて集合の要素同士の総当たりによる比較などを行わなければならず，効率の良い比較方式を実現する必要がある．現在は，集合を単なるリストを用いて実現している．

名前付き集合の入れ子

名前付き集合の第3項は，データと名前付き集合を含むため，名前付き集合が入れ子の状態になる．これにより，名前付き集合の構造が階層的になり比較などの演算がより複雑になる．

値と識別子による比較の違い

名前付き集合を比較する場合に，識別子による比較と値による比較の両方を考える必要がある．

3.2 クラスの設計

名前付き集合を実現するために，以下のようなクラスを定義する．図 5はそれらのクラスのオブジェクトモデル図である．図のように，名前付き集合は，第3項を名前付き集合の識別子の集合とデータの集合に分けて管理している．データはさらにデータ型毎に分けて管理されており，データの集合はそれらの和集合として表現されている．以下では，図 5中のクラスについて述べる．

Sorted_listクラス

C++のテンプレートクラス Sorted_list<class Type>は Type 型のデータをソートされたリストとして格納するデータ構造である．挿入時に Type クラスで定義された比較演算（$<, \leq, =, \geq, >, \neq$）を用いて入力データが昇順に並ぶようにする．このクラスには，挿入，削除，反復子を返す操作，そして検索がある．ここでの検索とは，リスト中に指定された値が含まれているか否かを返す．反復子とは，ポインタの働きをするもので，リストの要素をポイントして読み出すことができる．また，リストの終端を検出することもできる．これにより，繰り返しの操作の実現を図る．

Set_by_Sorted_listクラス

Set_by_Sorted_list <class Type>は Sorted_list クラスを用いて実現した一般的な集合クラスである．要素の挿入，削除に関しては，Sorted_list クラスにそのまま委託する．反復子を用いた繰り返し処理により，集合和や差などの集合演算を実装している．

Ns_data_setsクラス

様々な型のデータの集合クラスである．内部的には，前述のテンプレート Set_by_Sorted_list クラスを用いてそれぞれの型の集合クラスを作り，そのすべてを集約している．航行演算 ns.DATA は，名前付き集合 ns の第3項に含まれる要素のうちこのクラスに属するデータの集合を返す．

NS_idクラス

NS_id 型（名前付き集合の識別子）は，名前付き集合を一意に識別するためのデータ型である．

NS_link_setsクラス

名前付き集合の識別子の集合を表すクラスである．名前付き集合はそれぞれ独立しており同時に複数の集合に同時に属することが可能である．そのため，名前付き集合の集合を識別子の集合として表現している．このクラスはまた，名前付き集合モデルにおける検索結果を表す型でもある．名前付き集合の集合から，条件を満たす名前付き集合を検索した結果も同じく集合であるということである．

NS_third_memberクラス

名前付き集合の第3項S に相当するクラスである．二つの部分集合 NS_data_sets と Ns_link_sets を内部に持つ．

Named_setクラス

名前付き集合を表すクラスである．id（Ns_id）と名前（string）と集合（Ns_third_member）をメンバーに持つ．このクラスには，名前付き集合を作成するコンストラクタ，和や強和などの演算のためのメンバー関数が定義されている．

ここで，名前付き集合モデルの演算であるΘ_x演算の実現についてみてみる．Θ_x演算はその対

象である2つの集合のそれぞれを直和分割した集合間のΘ_x演算に分割できる．たとえば，$S = S_1 \cup S_2 \cup \ldots \cup S_n$（それぞれの集合は互いに素），$R = R_1 \cup R_2 \cup \ldots \cup R_n$（それぞれの集合は互いに素）であるとき以下の式が成立する．

$$
\begin{aligned}
S^{\forall}\theta^{\exists}R = {} & (((S_1{}^{\forall}\theta^{\exists}R_1) \vee (S_1{}^{\forall}\theta^{\exists}R_2) \vee \ldots \vee (S_1{}^{\forall}\theta^{\exists}R_n)) \\
\wedge {} & ((S_2{}^{\forall}\theta^{\exists}R_1) \vee (S_2{}^{\forall}\theta^{\exists}R_2) \vee \ldots \vee (S_2{}^{\forall}\theta^{\exists}R_n)) \\
\wedge {} & \\
\wedge {} & \\
\wedge {} & ((S_n{}^{\forall}\theta^{\exists}R_1) \vee (S_n{}^{\forall}\theta^{\exists}R_2) \vee \ldots \vee (S_n{}^{\forall}\theta^{\exists}R_n)))
\end{aligned}
$$

また，Ns_data_setsはモデル的には，“あらゆるデータ型のデータからなる集合”であるが，実際には，データ型毎の集合の和集合で実現されている．そこで，データ型が異なるデータ間の比較が常に偽であるとすると，Ns_data_setsは各データ型の集合で直和分割できる．つまり，クラスNs_data_setsのインスタンスの集合S_{data}は，$S_{bool} \cup S_{int} \cup S_{char} \cup S_{double} \cup S_{string}$である．ここで，$S_{int}$はint型のデータの集合である．これらの表記を用いると，演算$S_{data}{}^{\forall}\theta^{\exists}R_{data}$は以下のように変形できる．

$$
\begin{aligned}
& S_{data}{}^{\forall}\theta^{\exists}R_{data} = \\
& (((S_{bool}{}^{\forall}\theta^{\exists}R_{bool}) \vee (S_{bool}{}^{\forall}\theta^{\exists}R_{char}) \vee \ldots \vee (S_{bool}{}^{\forall}\theta^{\exists}R_{string})) \\
& \wedge ((S_{char}{}^{\forall}\theta^{\exists}R_{boul}) \vee (S_{char}{}^{\forall}\theta^{\exists}R_{char}) \vee \ldots \vee (S_{char}{}^{\forall}\theta^{\exists}R_{string})) \\
& \wedge \\
& \wedge((S_{string}{}^{\forall}\theta^{\exists}R_{bool}) \vee (S_{string}{}^{\forall}\theta^{\exists}R_{char}) \vee \ldots \vee (S_{string}{}^{\forall}\theta^{\exists}R_{string})))
\end{aligned}
$$

ところで，現在は$S_{int}{}^{\forall}\theta^{\exists}R_{int}$などで利用される同じデータ型同士の比較演算は定義済みであるが，異なるデータ型同士の比較演算は必ず偽となる．これは，Θ_x演算で利用される異なるデータ型同士の比較演算θが未定義であるということが理由である．表 1は現在の演算の定義状況である．

表 1 演算の定義状況

	bool	char	int	double	stirng
bool	○	×	×	×	×
char	×	○	×	×	×
int	×	×	○	×	×
dobule	×	×	×	○	×
string	×	×	×	×	○

しかし，異なるデータ型の要素間における比較は一般的に定義できるものではないので，この部分の定義をユーザに開放する．異なるデータ型のθ演算をユーザに定義させるために，公開クラスns_opを作成する．そこには，すでにインライン形式で演算が定義されており，ユーザはその形式（返値，関数名，引数）のまま，その内容を書き換えることで演算を定義する（図 4）．

```
class ns_op {
   const   def_flag[5][5];←表1のマトリックス
   bool    equal(bool bool_val, char char_val){return flase;}
   bool    equal(bool bool_val, char char_val){return flase;}
      ・・・・・・・・・・・・・・・・・・・・・・・・
      以下同様に，すべての比較演算に関して，すべてのデータ型の組み合わせについて関数が定義されている．
}
```

図 4 演算の定義

実際の演算の実行時には，多数の要素を含む集合同士の比較において負荷を軽減するため，def_flagを参照して，定義されていない（必ず偽となる）演算については比較を行わないようにしている．したがって，演算を定義したユーザは，def_flagで表現されるマトリックスのうち，定義した演算に対応する要素も書き換えなければならない．

4 おわりに

本論文では，半構造データを管理するためのデータモデルである名前付き集合モデルのインプリメントについて述べた．名前付き集合モデルは，識別子，名前，集合からなる3つ組であり，集合を基盤とすることで，データの表現に柔軟性を持つ．しかしながら，データ型の異なる要素も一つの集合に属するため，その実現においては，集合をデータ型毎に構成する要素を分けた部分集合の和集合として表現し，それぞれの要素の比較演算を実現することで，名前付き集合同士の比較演算を実現している．現在は同じ型同士の要素のみの比較を行っており，異なるデータ型同士の比較は必ず偽となる．異なるデータ型同士の比較はシステム側で一意に定義することが困難であるため，公開された比較演算のクラスを変更することでユーザが独自の比較演算を定義できる．

しかしながら，現在の方式では，データ型が異なる要素同士の演算の定義を行った段階で，システムの再コンパイルが必要となる．このため，一つのシステムでユーザ毎に一つの演算に対する複数の定義を表現することができない．再コンパイルすることなく，一つのシステムで演算の複数の定義を管理し演算を実現することは今後の課題である．

謝辞

本研究は，一部，文部省科学研究費重点領域研究(1)(課題番号09204113)による．

参考文献

1) 田島敬史，半構造データに関する研究動向，Proc. of ADBS97, pp.101-110 (1997).
2) Obase Consortium : Obase プロジェクト第三期研究報告書，千里国際情報事業財団, pp.205-219 (1994).
3) Lieberman, H. : Using Prototypical Objects to Implement Shared Behavior in Object Oriented Systems, Proc. of OOPSLA'86, pp. 214-223 (1986).
4) 銭晴 他：ハイパーテキストデータベースシステム TextLink/Gem におけるオブジェクトとスキーマの動的，段階的な構築機能，信学技報，DE92-39, pp. 1-8 (1993).
5) 中田充 他：サイエンティフィックデータベースのためのデータモデルの一提案，情処研報 95-DBS-101, pp. 65-72 (1995).
6) 中田充 他：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, pp. 113-120 (1995).
7) 中田充 他：考古学データの柔軟な管理をめざしたデータベースシステムの設計と実装，情報考古学 Vol.1 (1), pp46-54 (1995).
8) 中田充 他：名前付き集合モデルを用いたDREAMモデルの定義，情処研報 97-DBS-112, pp. 1-8 (1997).
9) Mitsuru NAKATA et al., Bottom-Up Scientific Databases Based on Sets and Their Top-Down Usage, Proc. of International Database Engineering and applications Symposium, pp171-179 (1997).

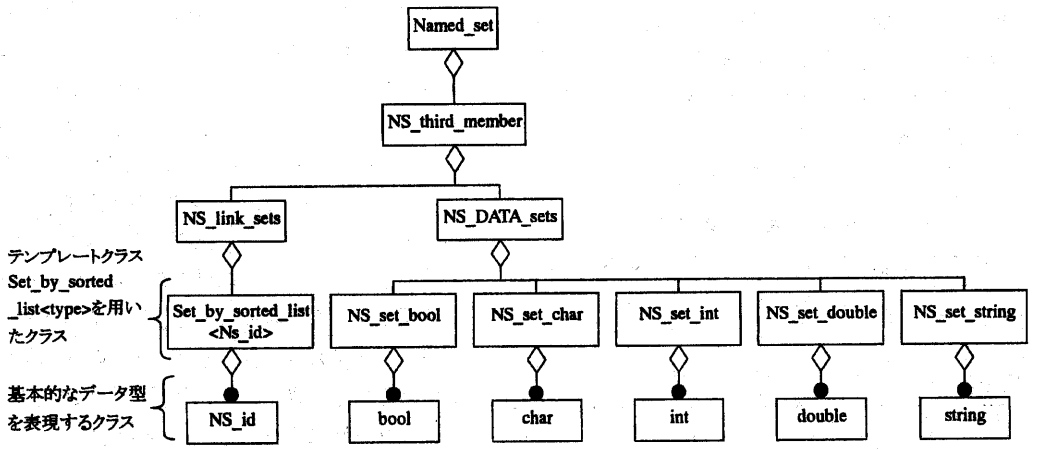

図 5 名前付き集合のオブジェクトモデル図