データベースシステム　127－8
情　報　学　基　礎　　67－8
（２００２．５．２１）

柔構造データベースにおける関連制約の検査について

河辺 貞行 †　　宝珍 輝尚 †　　中田 充 ‡　　都司 達夫 †　　樋口 健 †

† 福井大学 工学部 情報・メディア工学科

‡ 山口大学 教育学部

柔構造データベース管理システム DREAM では，あらかじめデータ定義を行うことなく柔軟にデータを格納することが可能である．DREAM では，データを整理・分類した後のデータ間の関連は，利用者が各自で管理するのみであり，関連に関する制約は利用者が管理しなければならなかった．そこで本論文では，関連に存在する一貫性を管理する手法を提案する．まず，関連の制約を表現する 5 つの表現 (基本制約表現) を提案し，この 5 つの基本制約表現によって関連の制約が表現できることを示す．次に，DREAM 上での関連の制約管理の実現法について述べる．

On the Relationship Constraint Checks of Incremental Databases

Sadayuki Kobe†　Teruhisa HOCHIN†　Mitsuru MAKATA‡　Tatsuo TSUJI†　Ken Higuchi†

† Dept. of Information Science, Faculty of Eng., Fukui University

‡ Faculty of Education, Yamaguchi University

In the incremental database management system DREAM, data can be stored without any definition in advance. In DREAM, relationships between data are required to be managed by each user with his/her own responsibility. Users have to manage constraints of relationships. In this paper, a method of managing the consistency in relationships is proposed. First, five expressions representing the constraints of relationships are proposed. Second, it is shown that the constraits of relationships can be represented by using these five expressions. Next, the impementation of managing constraints of relationships on DREAM is described.

1 はじめに

現在，広く一般的に用いられているデータベースは整然と整理されたデータを格納するものであり，データベースを用いてデータを管理する場合は，あらかじめどの様にデータを管理するかという枠組 (スキーマ) を定義する必要がある．しかしながら，実験データや測定データ，さらには文書データなどのいわゆる科学技術データは，その構造が複雑であったり多種多様であるため，実際にデータを格納する前にスキーマを決定することは困難である．

そこで筆者らは，データを柔軟に管理するためのデータベース管理システム DREAM の研究を行ってきた [1, 2, 3, 4]．DREAM では，集合を多用することにより高度の柔軟性を持たせている．すなわち，現実世界の実体を表すオブジェクトは，オブジェクトの側面を表す視点の集合として表現される．また，オブジェクトの集合はバンドルとして扱う．視点は，いわゆる属性名とその値の集合で構成される名前付きエレメントの集合として表現される．これにより，データをとりあえず格納しておいて後でこれらを整理・分類してゆくことが容易となり，ボトムアップにデータベースを構築可能することができる．また，DREAM では，関連の実体も通常のオブジェクトとして扱う [4]．すなわち，関連のためにデータモデル上の特別な要素は設けずに，利用者が個々の関連の実体に対応するオブジェクトを作成する．ただし，関連に関する情報はシステム定義のバンドル中に格納可能とし，オブジェクトを関連の実体として操作したい利用者には便宜を図っている．このように，DREAM では，データを整理・分類した後のデータ間の関連は，利用者が各自で管理するのみであった．すなわち，関連に相当するオブジェクトを利用者が作成し，利用者が管理を行わなければならない．従って，関連に関する制約は利用者が管理しなければならない．個々の応用プログラムでこの管理を行うのは非常に大変で，システムとして制約の検査をする機能の提供が望まれる．

ここで，Fan らは，XML に対する一貫性制約として，キー，参照キー，ならびに，オブジェクト識別子に関する制約について議論している [5, 6]．また，Buneman らは，半構造データと XML に対する制約として，パス内包制約 (Path inclusion constraints) について考察し，Fan らが扱った単なるキーではなく，キーがパスを構成するキーパスについての議論を行っている [7]．これらの研究においては，参照・被参照の関係が議論の中心となっており，本論文で扱うような関連に関する制約については議論されていない．

そこで，本論文では，データ間の関連を柔軟に管理し，また，これを利用して関連に存在する一貫性を管理する手法を提案する．提案する手法は，関連の制約を表現する基本的な表現 (基本制約表現) を設け，この基本制約表現の組み合わせで関連の制約を記述可能とする方法である．そして，この記述を関連に関する情報として格納しておいて関連の制約検査に利用する．これにより，利用者が関連として捉えた情報の持つ制約をうまく利用することができ，情報システムのデータ管理能力を高めることができる．

以下，2章では，柔構造データベース管理システム DREAM と DREAM における関連の表現について述べる．3 章では，関連における制約について述べる．4 章では，関連における制約の表現法を提案する．ここでは，5 種の表現 (基本制約表現) を提案し，これらの表現 (基本制約表現) で関連間の制約が表現できることを示す．5 章では，実装について述べる．ここでは，制約情報の表現とクラス構成について述べる．6 章では，GUI での関連の制約検査の操作例を示す．最後に 7 章で，まとめを行い，今後の課題を示す．

2 柔構造データベース管理システム DREAM

ここでは，DREAM モデルと，DREAM での関連の表現方法について述べる．

2.1 DREAM モデル

DREAM はデータモデルとして集合をもとにした DREAM モデルを採用している．DREAM モデルには，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，バンドルの 5 つの要素がある．

以下，これらについて述べる．

- データエレメント

  データの実体を格納する要素であり，データの取り扱いの最小単位である．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，データベース中またはファイル中のデータの一部分を指すために用いる構造体である．

- 名前付きエレメント

  データエレメントに名前を付けたものである．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．

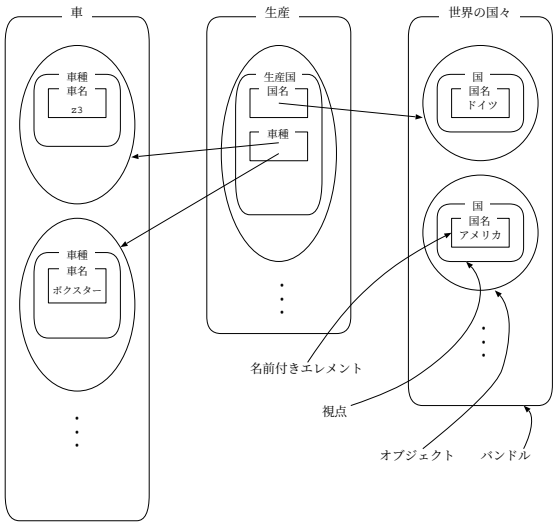

図 1: DREAM での関連の表現例

- 視点

  名前付きエレメントの集合であり，対象物の一つの側面を表す．3 つ組 $(id, name, S)$ で表され，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．

- オブジェクト

  視点の集合である．3 つ組 $(id, name, S)$ で表され，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，名前は空文字列でも構わない．

- バンドル

  オブジェクトの集合である．3 つ組 $(id, name, S)$ で表され，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．

2.2 DREAM モデルでの関連の表現方法

DREAM モデルでは，関連を表現するための特別な要素はなく，バンドルで関連を表現する．例えば，”生産”というバンドルを作って，”車”が”世界の国々”で”生産”されるという関連を表現する様子を図 1に示す．バンドル”生産”の”国名”という名前付きエレメントが，バンドル”世界の国々”のドイツを参照し，バンドル”生産”の”車種”という名前付きエレメントが，バンドル”車”の Z3 とボクスターを参照している．これにより，ドイツが Z3 とボクスターを生産するということが表される．

3 関連における制約

ここでは，関連における制約として，基数制約，完全性制約，排他制約，挿入属性，および，保留属性について述べる [8, 9, 10, 11].

3.1 基数制約

基数制約 (cardinality constraints) は，関連を構成するオブジェクトの数に関する制約を表す．例えば，”国”と”車”との間に”生産 (する)”という関連がある場合，”車”は必ずどこかの”国”で生産されるので，”国”の数は 1 以上で”車”の数は 0 以上という基数制約がある．

3.2 完全性制約

完全性制約 (completeness constraints) は，汎化関連における制約で，スーパータイプのインスタンスが少なくとも一つ以上のサブタイプになければいけないかどうかという制約である．

- スーパータイプのインスタンスが，少なくとも一つ以上のサブタイプになければならないということを，全特化則 (total specialization rule) という．また，被覆制約 (cover constraints) と呼ばれることもある．
- スーパータイプのインスタンスがまた，サブタイプになくてもよいということを，半特化則 (partial specialization rule) という．

3.3 排他制約

排他制約 (disjointness constraints) は，汎化関連における制約で，スーパータイプのインスタンスが同時に 2 つ以上のサブタイプにあってよいかどうかという制約である．

- スーパータイプのインスタンスが，一つのサブタイプにあり，同時に他のサブタイプにあってはならないということを，排他則 (disjoint rule) という．
- スーパータイプのインスタンスが，同時に 2 つ以上のサブタイプにあってよいということを，重複則 (overlap rule) という．

3.4 挿入属性

(1) 自動

データベースに格納する段階で，自動的に関連にも接続しなければならないオブジェクトは，挿入属性が自動 (automatic) であるという．例えば，学校では，新入生は必ずどこかの学科に所属することになっている．すると，”学生”が”学科”に”所属”するという関連がある場合，データベースに新入生を格納する段階で，自動的にこの関連が成立すると考えられる．この場合の”学生”のようなオブジェクトは挿入属性が自動である．

(2) 手動

データベースに格納する段階で，関連にも接続しなければならないかどうかが，格納するオブジェクトごとに決まるような場合，そのオブジェクトは，挿入属性が手動 (manual) であるという．例えば，学校では，入学するとどれかのクラブに所属する人もいれば所属しない人もいる．従って，”学生”が”クラブ”に”所属”するという関連がある場合，データベースに新入生を格納する段階で，この関連が成立するかどうかは，個々の”学生”オブジェクトごとに決まる．この場合の”学生”のようなオブジェクトは挿入属性が手動である．

3.5 保留属性

1. 一時

関連との接続に制限がないオブジェクトは，保留属性が一時 (optional) であるという．学校では，”学生”が”クラブ”に”所属”しているとは限らないし，いったんどこかの”クラブ”に”所属”しても，他の”クラブ”に移ったり，やめることがある．このように別のオブジェクトに接続されたり，切り離されたりする．そして，ある”クラブ”がデータベースから消去された場合，その”クラブ”に”所属”していた”学生”は自動的に切り離される．この場合の”学生”のようなオブジェクトは保留属性が一時である．

2. 永続

関連に必ず接続されていなければならないオブジェクトは，保留属性が永続 (mandatory) であるという． 例えば，会社では，”社員”は必ずどこかの”部課”に”所属”するとする．そして， 人事異動で，”社員”の”所属”する”部課”が変更になることはありうる． しかし，どの”部課”にも”所属”しないということは，絶対ありえない． このように必ずオブジェクトに接続されていて，別のオブジェクトに 接続されることはあるが，切り離されることはない．ある”部課”が データベースから消去される場合には，その”部課”の”社員”は，他の ”部課”に接続されているべきである．この”社員”のようなオブジェクト は保留属性が永続である．

3. 弱実体

一般に，オブジェクトは他と独立に存在してよい．しかし，他のオブジェクトの存在を前提にするとき，そのオブジェクトは弱実体 (weak entity) であるという．例えば，”車”が”ナンバー”を”登録（する）”という関連では，”ナンバー”が弱実体となっている．ある”車”が存在した上で”ナンバー”は存在できるからである．弱実体は独立には存在できないので，”車”も”ナンバー”も関連から切り離してはいけないし，別の関連に接続してはいけない．そして，”車”がデータベースから削除された場合，”ナンバー”も削除しなくてはならない．

4. 固定

関連に結び付いてはじめて意味を持ち，永続や弱実体よりも更に強い結び付きをしているオブジェクトは，保留属性が固定 (fixed) という．”預金口座”が”口座番号”を”持つ”という関連を考えると，”口座番号”は特定の”預金口座”と結び付いてはじめて意味を持つ．”預金口座”も”口座番号” も別の関連に接続されることもなく，切り離されることもない．もし”預金口座”がデータベースから削除される場合には，”口座番号”も一緒に消えてしまわないといけないし，もし”口座番号”がデータベースから削除される場合には，”預金口座”も一緒に消えてしまわないといけない．この”口座番号”のようなオブジェクトの保留属性は永続や弱実体よりも強く，固定である．

4 関連における制約の表現

4.1 制約の表現法

制約を表現するために，次のような表現 (以降，基本制約表現と呼ぶ) を導入する．

1 `a : b = 1 ： 2`

基数制約を：を用いて表現する．上記は，a と b が 1 対 2 の場合の例である．

- 基数の以上を表すのに，＋を，以下を表すのに − を数字の後ろに付ける．例えば，a のオブジェクトが 1 個以上で，b のオブジェクトが 5 個以下の場合，以下のように表す．

  `a : b = 1+ ： 5-`

- 基数の範囲を表すのに，~ を数字の間につける．ある数でなければいけないときには何も付けない．例えば，a のオブジェクトが 1 個以上 5 個以下で，b のオブジェクトが 7 個の場合，以下のように表す．

  `a : b = 1~5 : 7`

- 複数ある場合は，カンマで区切る．つまり論理和を表現する．例えば，a のオブジェクトが 0 個もしくは 2 個以上で，b のオブジェクトが 1 個もしくは 3 個以上 9 個以下の場合，以下のように表す．

  `a : b = 0,2+ : 1,3~9`

2 `b IS CONNECTED`

b のオブジェクトは関連に接続されていなければならないことを表す．従って，b のオブジェクトがデータベースに挿入されたら，関連に接続されなければなないことを表す．また，b のオブジェクトを切り離すなら，b のオブジェクトを他の関連へ移動して接続しなければならない．

3 `b IS NOT MOVED`

他の関連へ b のオブジェクトを移動してはいけないことを表す．従って，b を切り離せない．

4 `IF b IS CONNECTED THEN a IS NOT ERASED`

b のオブジェクトが関連に接続されている場合は，データベースから a のオブジェクトを消去できないし，切り離すこともできないことを表す．

5 `IF a IS ERASED THEN b IS ERASED`

a のオブジェクトを消去するならば，b のオブジェクトも消去しなくてはいけないことを表す．

4.2 制約の表現

ここでは，3 章で述べた制約が **4.1** で示した基本制約表現の組み合わせで表現できることを示す．

4.2.1 基数制約

基数制約は基本制約表現 1 で表現できる．”国”と”車”の間の”生産（する）”という関連における基数制約は以下のように表現できる．

国 : 車 = 1 : 0+

4.2.2 完全性制約

全特化則は，objsuper がスーパータイプのオブジェクトを表し，objsub がサブタイプのオブジェクトを表すとすると，基数制約を用いて以下のように表現できる．

`objsuper : objsub = 1 : 1+`

この制約の記述がない場合は，デフォルトとして半特化則となる．

4.2.3 排他制約

排他則は，上記と同様に，基数制約を用いて以下のように表現できる．

`objsuper : objsub = 1 : 1-`

また，この制約の記述がない場合は，デフォルトとして重複則となる．

4.2.4 挿入属性

1. 自動

   基本制約表現 2 を用いて表現できる．例えば，”学生”が”学科”に”所属（する）”という関連において，”学生”の挿入属性が自動という場合は，以下のように表現できる．

   学生 `IS CONNECTED`

2. 一時

   上記の記述がない場合，デフォルトとして挿入属性が一時となる．

4.2.5 保留属性

1. 永続

   基本制約表現 2 と基本制約表現 4 を用いて表現できる．例えば，”社員”が”部課（する）”という関連において，”社員”の保留属性が永続という場合は，以下のように表現できる．

   - 社員 IS CONNECTED
   - IF 社員 IS CONNECTED THEN
     部課 IS NOT ERASED

(2) 弱実体

基本制約表現 2，基本制約表現 3，ならびに，基本制約表現 5 を用いて表現できる．例えば，”車”が”ナンバー”を”登録 (する)”という関連において，”ナンバー”が弱実体の場合は，以下のように表現できる．

- ナンバー IS CONNECTED
- ナンバー IS NOT MOVED
- 車 IS NOT MOVED
- IF 車 IS ERASED THEN
  ナンバー IS ERASED

(3) 固定

基本制約表現 2，基本制約表現 3，ならびに，基本制約表現 5 を用いて表現できる．例えば，”車”が”車体番号”を”付ける”という関連において，保留属性が固定という場合は，以下のように表現できる．

- 車体番号 IS CONNECTED
- 車 IS CONNECTED
- 車体番号 IS NOT MOVED
- 車 IS NOT MOVED
- IF 車 IS ERASED THEN
  車体番号 IS ERASED
- IF 車体番号 IS ERASED THEN
  車 IS ERASED

5 実装

ここでは，制約情報の表現とクラス構成について述べる．

5.1 制約情報の表現

DREAM では，関連の情報を_relationships と_participants というバンドルに格納する．

バンドル_relationships のオブジェクトの例を図 2に示す．ここでは，”世界の国々”が”車”を”生産（する）”という

```
(o201,"",{
  (o202,"main",{
    (o203,"relname","生産"),
    (o204,"bundle","生産"),
    (o205,"persp","生産国"))
  (o206,"participants",{
    (o207,"国",{o311}),
    (o208,"車",{o321}))
  (o209,"constraints",{
    (o210,"cardinality","国 : 車 = 1+ : 0+"),
    (o211,"connectivity",{"車 IS CONNECTED",
"車 IS NOT MOVED"}),
    (o212,"modification",{"IF 車 IS CONNECTE
D THEN 国 IS NOT ERASED" })})})})
```

図 2: _relationships のオブジェクトの例

関連を例にしている．ここで，例えば，o201 は識別子を表す．視点”main”の名前付きエレメント”relname”は関連の名前である．視点”main”の名前付きエレメント”bundle”は関連を表すオブジェクトが入っているバンドルである．視点”main”の名前付きエレメント”persp”は関連を表すオブジェクトの視点である．視点”participants”の名前付きエレメントの名前は，制約の情報の記述に使用するもので，ユーザー定義の名前である．この名前が示すオブジェクトは，関連しているオブジェクトの情報が入ったオブジェクトである．つまり_participants のオブジェクトである．

視点”constraints”の名前付きエレメント”cardinality”には，基数制約がある場合にはそれを記述する．視点”constraints”の名前付きエレメント”connectivity”には，”b IS CONNECTED”や”b IS NOT MOVED”を記述する．ない場合はなくてよく，複数ある場合は複数記述する．視点”constraints”の名前付きエレメント”modification”には，”IF a IS CONNECTED THEN b IS NOT ERASED”や”IF a IS ERASED THEN b IS ERASED”を記述する．

バンドル_participants のオブジェクトの例を図 3に示す．視点”main”の名前付きエレメント”ne”は関連を表すオブジェクトの名前付きエレメントの名前であり，この名前付きエレメントが関連しているオブジェクトを参照している．視点”main”の名前付きエレメント”relationships”は_relationships のオブジェクトである．視点”main”の名前付きエレメント”bundle”は関連しているオブジェクトが入っているバンドルである．視点”main”の名前付きエレメント”persp”は関連しているオブジェクトの視点である．視点”main”の名前付きエレメント”role”は関連の役

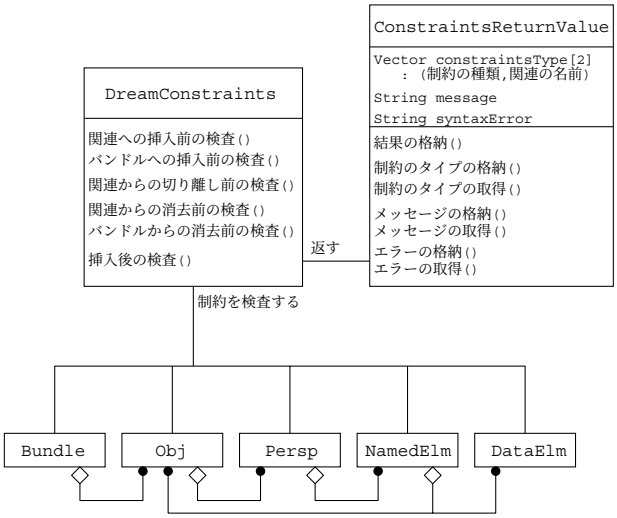

図 4: クラス図

```
(o311,"",{
  (o312,"main",{
    (o313,"ne","国名"),
    (o411,"relationships",o201),
    (o314,"bundle","世界の国々"),
    (o315,"persp","国"),
    (o316,"role","生産する") })})
(o321,"",{
  (o322,"main",{
    (o323,"ne","車種"),
    (o421,"relationships",o201),
    (o324,"bundle","車"),
    (o325,"persp","車種"),
    (o326,"role","生産される")})})
```

図 3: _participants のオブジェクトの例

割である．

5.2 クラス構成

制約の検査を行うための DreamConstraints クラスを作成した．また，ConstraintsReturnValue クラスを作成し，このクラスのインスタンスに，検査した結果を格納するようにした．このクラス構成を図 4 に示す．

関連への挿入前の検査では，基数制約を調べる．バンドルへの挿入前の検査では，"b IS CONNECTED"の制約がないか調べる．関連からの切り離し前の検査では，基数制約，"b IS NOT MOVED"，"b IS CONNECTED"，ならびに，"IF a IS CONNECTED THEN b IS NOT ERASED" の制約を調べる．関連からの消去前の検査では，消去するオブジェクトが入っているバンドルを見つけ，そのバンドルに対して，バンドルからの消去前の検査を行う．バンドルからの消去前の検査では，"IF a IS ERASED THEN b IS ERASED"，"IF a IS CONNECTED THEN b IS NOT ERASED"，ならびに，基数制約を調べる．挿入後の検査では，基数制約と"a IS CONNECTED"の制約を調べる．

図 4に示したクラスは Java で実装している．図 4中，DREAM モデルの構成要素 (Bundle，Obj，Persp, NamedElm, DataElm) のためのクラスでは，C++で実装されたクラスを JNI(Java Native Interface) を利用して使用している．

6 GUI での操作例

GUI(Graphical User Interface) での操作の様子を図 5 に示す．これは，"車"が"車体番号"を"付ける"という関連で，"車：車体番号 = 1：1"の制約がある場合に， 車を切り離す前の検査を行った様子である．「基数制約：切り離すことはできません。」というメッセージダイアログが表示されている．

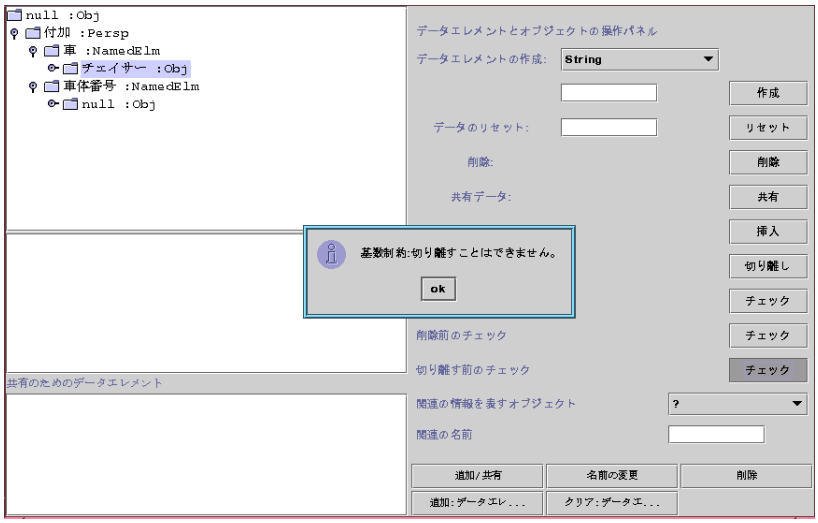

図 5: GUI での操作例

7 おわりに

本論文では，柔構造データベース管理システム DREAM において，データ間の関連を柔軟に管理し，関連に存在する一貫性を管理することを目的として，関連における制約の表現法を提案し，制約の検査を行う機能の実現について述べた．提案する手法は，関連の制約を表現する基本的な表現 (基本制約表現) を設け，この基本制約表現の組み合わせで関連の制約を記述可能とする方法である．この記述を，関連に関する情報として格納しておき関連の制約検査に利用することにより提案手法に基づく機能を実現した．また，GUI での関連の制約検査の操作例を示した．

今後の課題は，キー制約といった一般的な制約の管理である．

参考文献

[1] 中田 充，岩井信輔，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，情報処理学会研究報告 DBS 114-14 (1998).

[2] 宝珍輝尚，中田 充，都司達夫：データの整理・分類支援のためのデータモデル，情報学シンポジウム, pp. 33-40 (1997).

[3] Nakata, M., Hochin, T., and Tsuji, T. : Bottom-up Scientific Databases Based on Sets and Their Top-down Usage, *Proc. of 1997 Int'l Database Eng. & Applications Symposium(IDEAS'98)*, pp. 171–179(1997) .

[4] Hochin, T., and Nakata, M., and Tsuji, T. : A Flexible Kernel Data Model for Bottom-Up Databases and Management of Relationships, *Proc. of 1998 Int'l Database Eng. & Applications Symposium(IDEAS'98)*, pp. 170-177(1998).

[5] Fan, W., and Siméon, J. : Integrity Constraints for XML, *Proc. of 19th ACM PODS(PODS2000)*, pp. 23–34(2000).

[6] Fan, W., and Libkin, L. : On XML Integrity Constraints in the Presence of DTDs, *Proc. of 20th ACM PODS(PODS2001)*, pp. 114-125(2001).

[7] Buneman, P., Fan, W., Siméon, J., and Weinstein, S. : Constraints for Semistructured Data and XML, *SIGMOD Record*, Vol. 30, No. 1, pp. 47-54(2001).

[8] 鈴木健司：データベースがわかる本，オーム社 (1998).

[9] 植村俊亮：データベースシステムの基礎，オーム社 (1979).

[10] 三浦孝夫：データモデルとデータベース，サイエンス社 (1997).

[11] McFadden, F. R., Hoffer, J. A., and Prescott, M. B. : MODERN DATABASE MANAGEMENT Fifth Edittion, ADDISON-WESLEY(1999).