データベースシステム　118－5
情　報　学　基　礎　54－5
（１９９９．５．１７）

半構造データの構造表現のための動的スキーマの生成法について

宝珍 輝尚　　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
{ hochin, tsuji }@pear.fuis.fukui-u.ac.jp

半構造データを格納したデータベースに対して問合せを行うには，従来のスキーマ相当の情報が必要である．しかし，半構造データは，あらかじめ構造が決定されない，または，構造が時々刻々変化するという特性を持つことがあるため，従来のスキーマをそのまま適用することはできない．従来のスキーマ相当の情報を表現するものとして，DataGuides やシェイプ（本論文では動的スキーマと呼ぶ）が提案されてきているが，動的スキーマの効率的な生成は大きな課題の一つである．本論文では，動的スキーマの版を使用する，動的スキーマの準動的な生成法を提案する．動的スキーマの版と次の版を生成するのに必要なオブジェクトを保持することにより正しい動的スキーマを迅速に生成することを可能とする．性能評価実験の結果，オブジェクトの更新・削除がないか，または，新しいオブジェクトのみが更新・削除される場合に提案法は有効であるという結果が得られた．

On the Method of Generating Dynamic Schema Representing the Structure of Semistructured Data

Teruhisa HOCHIN　　Tatsuo TSUJI
Dept. of Information Science, Faculty of Eng., Fukui University
3-9-1, Bunkyo, Fukui-shi, Fukui 910-8507
{ hochin, tsuji }@pear.fuis.fukui-u.ac.jp

The structure of semistructured data is strongly required to be obtained in order to query on a large amount of semistructured data, whereas the structure can not be defined a prior. DataGuides and shape have been proposed to represent the structure of semistructured data. As these can behave as schema in the point of view of representing the structure of data, and are changed according to the existence of objects, these are called dynamic schema in this paper. This paper proposes the semi-dynamic construction method of the dynamic schema. The proposed method uses versions of the dynamic schema. Several versions are kept in the system. Each version of the dynamic schema keeps the objects required to construct its next version from it in a list. The correct dynamic schema can easily be constructed by using the versions of dynamic schema and the lists of objects. The performances in deriving the shape, and inserting and deleting objects are experimentally evaluated. Experimental results show that the proposed method is effective in the case that the deletion is occurred only on the recent objects.

1 はじめに

近年，様々なデータ源からの様々なデータの扱いが盛んに研究されている [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]．これらのデータの重要な特性の一つに，あらかじめデータの構造を決定することが困難であるということがある [4, 11]．しかしながら，大量のデータに対して問合せを行うためにはデータの構造が必要になる．半構造データの構造を表現するために，DataGuides[11]，シェイプ [12, 13] やスキーマ生成 [14] が提案されてきている．これらは，データの構造を表現するという観点からは，従来のスキーマに相当するものである．DataGuides やシェイプは，オブジェクトの生成・更新・削除に従って動的に変化する．本論文では，これらをまとめて，動的スキーマと呼ぶ．

動的スキーマを維持管理するのは困難である．オブジェクトが一つしか存在しなくても，対応する動的スキーマは存在しなければならない．また，オブジェクトが削除・更新された場合には，対応する動的スキーマも変化しなければならない．最も困難なのは，動的スキーマに対応する最後のオブジェクトが削除された場合に，対応する動的スキーマ情報を削除することである．このために，DataGuides の実装では，2種類のハッシュ表，ならびに，対象データと DataGuides 間の関係を使用して，全てのオブジェクトをとらえている [11]．しかし，この方法は柔軟な半構造データを管理するにはあまりにも硬い方法であると考えられる．最も柔軟な方法は，動的スキーマを，要求されるたびに，一から生成する方法であるが，これでは大量のオブジェクトを扱うことが困難である．

そこで本論文では，動的スキーマの準動的な生成法を提案する．提案する方法は動的スキーマの版を使用する方法である．動的スキーマの版とともに，次の版を生成するのに必要なオブジェクトへの識別子も保持し，動的スキーマを高速に生成することを可能とする．

以下，2では，本論文で前提とするデータモデル，ならびに，シェイプと呼ぶ動的スキーマについて概説する．3では，シェイプの準動的な生成法を提案し，4で，提案法を実験により評価する．最後に，5でまとめる．

2 動的スキーマとしてのシェイプ

2.1 DREAM モデル

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，バンドルをデータベースエレメントと呼ぶ [12]．

データエレメントは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．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，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．

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

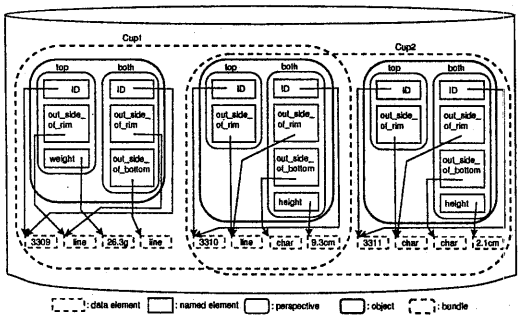

図 1: データベースの例

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．

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．データベース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り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種類の情報を考え，1つのオブジェクトの 2つの視点（“top” と “both”）として格納している．視点 “top” は，少なくとも “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，少なくとも “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．図 1では，さらに，Cup1 という名前のバンドルには，名前付きエレメント “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ている．

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 [12]．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 [12]．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} )
```

また，以下は図1のバンドルのシェイプである．

```
( 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（振舞い１）．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（振舞い２）．

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，データベース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つの振舞いのうち振舞い１のみを満足する，硬シェイプと呼ぶシェイプを導入している [12]．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，従来からのデータベース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　シェイプの生成法

硬シェイプに対しては，データエレメント等の挿入時に対応するシェイプを作成するのみで良く，削除のことを考える必要がない．従って，硬シェイプの実装は容易である．そこで，ここでは，硬シェイプではない通常のシェイプについて検討する．

まず，シェイプエントリの管理方法を述べ，次に，シェイプの生成方法を提案する．

3.1　シェイプエントリの管理

例えば，データ型 dt1 の最後のデータエレメントが削除された場合，dt1 は，対応するシェイプエントリのデータ型名の集合から削除されなければならない．また，シェイプエントリに対応する最後の名前付きエレメントが削除された場合，そのシェイプエントリは削除されなければならない．

このようなシェイプの振舞いを実現するために，カウンタを使用する．すなわち，データ型名ごとにカウンタを設け，データエレメント挿入時に対応するデータ型名のカウンタをインクリメントする．また，データエレメント削除時に対応するデータ型名のカウンタをデクリメントする．カウンタが０になったならば，そのデータ型名をデータ型名集合から削除する．さらに，シェイプエントリのデータ型名集合が空になったならば，そのシェイプエントリを削除する．

3.2　シェイプ導出法

ここでは，シェイプの導出を高速に行うための方法として，シェイプの版を使用する方法を提案する．各版は，シェイプ，ならびに，次の版のシェイプを生成するのに必要なオブジェクトの識別子 (OID) のリストを保持する．このリストをオブジェクトリストと呼ぶ．また，各版は，適正フラグを持つ．適正フラグは，その版のシェイプが適正か否かを示す．また，版は，図2に示すように，ポインタで一つ前の版と関連付けられている．

シェイプは，最新の版の適正フラグが「適正」の場合，最新の版のシェイプにオブジェクトリスト中のオブジェクトを作用させて導出する．こ

こで，最新の版のシェイプをそのまま利用者に返却するのではなく，動的に導出していることに注意されたい．オブジェクトの削除の際には，そのオブジェクトの識別子がオブジェクトリストに存在する版を，最新の版から順次捜してゆく．この際に，適正フラグを「不適正」としてゆく．ただし，求める版の適正フラグは変更しない．求める版がみつかると，OIDをオブジェクトリストから削除する．シェイプ導出にあたっては，最新の版の適正フラグが「不適正」の場合，適正フラグが「適正」である版まで戻ってシェイプを順次導出してゆく．この際，各版に対して導出された正しいシェイプを保持し，その版の適正フラグを「適正」とする．オブジェクトが（削除ではなく）更新された場合は，そのOIDをオブジェクトリストから削除することはないが，適正フラグに関する処理はオブジェクト削除時と全く同じに行う．

図2に，シェイプの版の例を示す．最新の版はver3のものである．ver3の版の適正フラグは「不適正」となっており，その一つ前の版ver2の適正フラグは「適正」となっている．ver2の版のオブジェクトリスト中のオブジェクトが削除または更新されるとこのような状態になる．ここでシェイプ導出の要求があると，最新版（ver3）の適正フラグは「不適正」であるため，一つ前の版（ver2）に対してシェイプ導出を要求する．ver2の適正フラグは「適正」であるので，ver2のシェイプにオブジェクトリスト中のオブジェクトを作用させてシェイプを生成する．生成されたシェイプはver3のシェイプとなり，さらに，ver3の版のオブジェクトリスト中のオブジェクトを作用させてシェイプを生成する．これが求めるシェイプとなる．

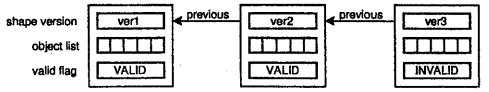

図 2: シェイプの版の例

4 評価

4.1 実験

提案しているシェイプ導出法に関して，バンドルの視点のシェイプの導出の性能，ならびに，オブジェクトの挿入・削除性能を評価する．ここでは，実験を簡単にするために，全てのオブジェクトは視点を一つ持ち，その視点は名前付きエレメントを一つ持ち，その名前付きエレメントは同一データ型のデータエレメントを一つ持つという簡単な構造を持つものとする．性能は，Ultima1 ワークステーション (Axil, Solaris 2.5.1, 256MB メモリ) で測定する．実験に使用するパラメータを表1に示す．ただし，4.1.1(1)

表 1：パラメータ

パラメータ	値
オブジェクトリストの最大オブジェクト数	1024
版数	8
オブジェクト数	8000

と4.1.2(1)におけるオブジェクトの数は，オブジェクトの数が変数であるため8000（固定）ではない．表1の値は，実験結果を明瞭にするために選んであり，他の値でも同様の実験結果が得られる．

4.1.1 シェイプ導出性能評価

提案法を，完全に動的なシェイプ導出法（完全動的シェイプ導出法）と比較する．完全動的シェイプ導出法では，シェイプが要求された場合，バンドル中の全てのオブジェクトをもとにして動的にシェイプを生成する．

シェイプ導出性能を，削除オブジェクトがない場合と，削除オブジェクトがある場合を評価した．

(1) 削除オブジェクトがない場合

バンドルに挿入するオブジェクトの数を変化させてシェイプ導出時間を測定した．測定結果

を図3に示す．完全動的シェイプ導出法では，オ

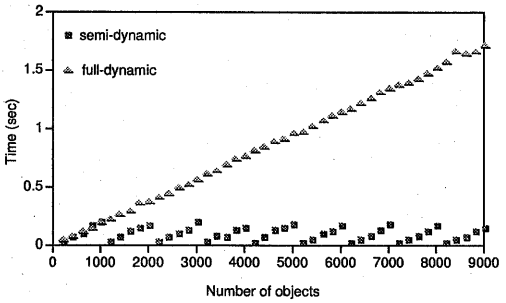

図 3: シェイプ導出時間に対するオブジェクト数の影響

ブジェクトの数が増えるに従ってシェイプ導出時間も単純に増加する．提案法でもこの傾向は見られるが，版のオブジェクトリスト中の最大オブジェクト数ごとに繰り返しており，オブジェクトの数が多くなっても，シェイプ導出時間が無制限に長くなることはない．

(2) 削除オブジェクトがある場合

ここでは，削除オブジェクトの位置を変化させて評価を行う．オブジェクトには通番をつけておき，この番号（オブジェクト番号）でオブジェクトを特定する．オブジェクト番号が小さいほど古いオブジェクトであるということである．このオブジェクト番号をオブジェクトの位置と呼ぶことにする．ここでは，実験を簡単にするため，オブジェクトが一つのみ削除された場合で評価する．削除オブジェクトの位置を変化させてシェイプの導出時間を測定した結果を図4に示す．完全動的シェイプ導出法では，シェイプの導出に削除オブジェクトの位置に寄らずほぼ一定時間必要であるが，提案法では，削除オブジェクトの位置が大きい（すなわち，新しいオブジェクトを削除した）場合にシェイプの導出時間を削減することができている．

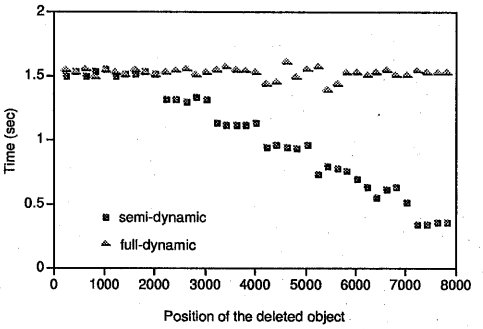

図 4: シェイプ導出時間に対する削除オブジェクト位置の影響

4.1.2 オブジェクトの挿入・削除への影響

次に，提案法がバンドルへのオブジェクトの挿入・削除におよぼす影響について評価する．ここでは，提案法を使用する場合と使用しない場合の，オブジェクトの挿入・削除時間を測定する．

(1) オブジェクト挿入への影響

挿入するオブジェクト数を変化させて性能を測定する．測定結果を，1オブジェクトあたりの挿入に要する時間で図5に示す．提案法を使

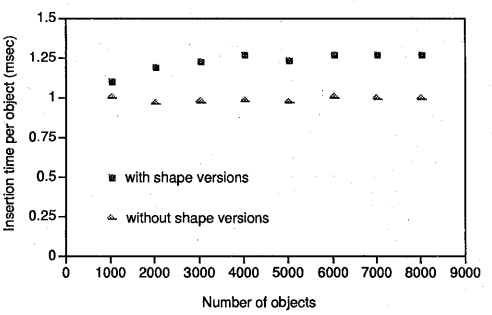

図 5: オブジェクト挿入時間に対するオブジェクト数の影響

用するオーバヘッドは約 0.25msec で，使用しない場合と比べて約 5/4 の時間を要している．

(2) オブジェクト削除への影響

削除オブジェクトの位置と削除オブジェクト

数を変化させて削除に要する時間を測定する．新しいものから古いものへの順番で，連続したオブジェクトを削除する．

(a) オブジェクトの削除位置の影響
削除するオブジェクトの数を1000（固定）とし，削除開始のオブジェクトの位置を変化させて削除に要する時間を測定した．測定結果を，1オブジェクトあたりの削除に要する時間で図6に示す．オブジェクトの削除開始位置を小さくするに従って，削除に要する時間がかかっている．これは，シェイプの版管理に起因する時間である．

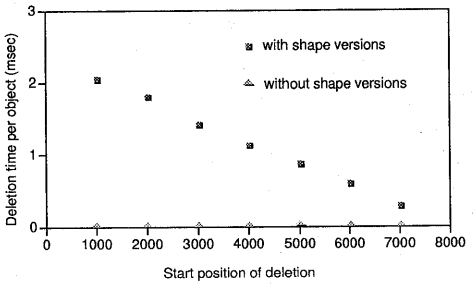

図 6: オブジェクト削除時間に対する削除オブジェクトの位置の影響

(b) オブジェクトの削除数の影響
削除開始オブジェクトの位置を7000（固定）とし，削除オブジェクトの数を変化させて削除時間を測定した．測定結果を，1オブジェクトあたりの削除に要する時間で図7に示す．削除に要する時間は削除するオブジェクト数に依存しない．これは，オブジェクトの削除に際して何も行っていないためと考えられる．また，削除に要する時間の差は，削除開始オブジェクトの位置でほぼ決まってしまっている．

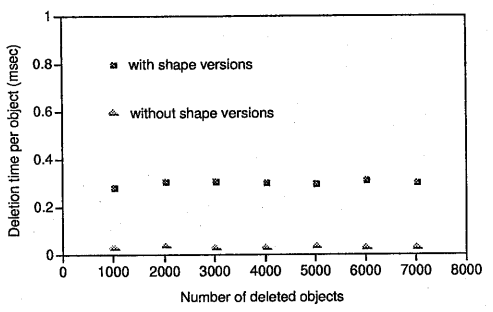

図 7: オブジェクト削除時間に対する削除オブジェクト数の影響

4.2 考察

提案法は，オブジェクトの削除がない場合に，完全動的シェイプ導出法に比べ，良い性能特性を持っている．これは，新しいオブジェクトが削除されている場合にもいえる．これらの場合，版を利用することでシェイプの導出コストを低くすることができるからである．しかし，古いオブジェクトが削除されシェイプの版が残っていない場合は，シェイプを一から生成し直す必要がある．この場合，完全動的シェイプ導出法と同等の時間がかかってしまう．

オブジェクトの挿入・削除へのオーバヘッドは無視できるものではない．特に，削除するオブジェクトが古くなるに従って削除オーバヘッドは大きくなり，また，削除するオブジェクトの位置によって削除性能がほぼ決定してしまう．従って，挿入や削除が頻繁に発生するような分野には提案法は適していない．ただし，挿入・削除直後のシェイプ導出には時間がかかるが，以降のシェイプ導出は高速に行うことができる．

以上より，提案法は，検索の方が更新よりも頻繁に発生する応用分野に適していると考えられる．特に，考古学データのように追加のみで削除が発生しない応用分野や，基本的に削除はない時制データベースには有効であると考えられる．

5 おわりに

本論文では，シェイプの準動的な導出法を提案した．シェイプは，あらかじめ定義を行えないデータに対する動的なスキーマである．提案した方法は，シェイプの版を利用するものであり，正しいシェイプの版にオブジェクトを作用することでシェイプの高速な導出を可能としている．シェイプ導出性能，ならびに，オブジェクトの挿入・削除性能への提案法の影響を実験により評価した結果，提案法は，検索の方が更新よりも頻繁に発生する応用分野に適していると考えられ，特に，オブジェクトが削除されないか，または，新しいオブジェクトが削除される場合に有効であることが分かった．

今後は，実際の応用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 09780258）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] Papakonstantinou, Y. *et al* : "Object Exchange Across Heterogeneous Information Sources," Proc. of 11th Int'l Conf. on Data Eng., pp. 251–260 (1995).

[3] Abiteboul, S.*et al* : "The Lorel query language for semistructured data," Int'l Journal on Digital Libraries, Vol. 1, No. 1, pp. 68–88 (1997).

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

[5] Buneman, P. *et al* : "A Query Language and Optimization Techniques for Unstructured Data," Proc. of ACM SIGMOD 1996, pp. 505–516 (1996).

[6] Buneman, P. *et al* : "Adding Structure for Unstructured Data," Proc. of 6th Int'l Conf. on Database Theory, pp. 336–350 (1997).

[7] Delcambre, L. M. L. *et al*: "Structured Maps: modeling explicit semantics over a universe of information," Int'l Journal on Digital Libraries, Vol. 1, No. 1, pp. 20–35 (1997).

[8] Anjur, V. *et al* : "FROG and TURTLE: Visual Bridges Between Files and Object-Oriented Data," Proc. of 19th VLDB Conf., pp.73–84 (1993).

[9] Shoens, K. *et al* : "The Rufus System: Information Organization for Semi-Structured Data," Proc. of 19th VLDB Conf., pp. 97–107 (1993).

[10] Pfaltz, J. L. and French, J. C. : "Scientific Database Management with ADAMS," Bulletin of the Technical Committee on Data Engineering, Vol. 16, No. 1, pp.14–18 (1993).

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

[12] Nakata, M., Hochin, T., and Tsuji, T. : "Bottom-up Scientific Databases Based on Sets and Their Top-down Usage," Proc. of Int'l Database Engineering & Applications Symposium 97, pp. 171–179 (1997).

[13] Hochin, T., Nakata, M., and Tsuji, T. : "A Flexible Kernel Data Model for Bottom-Up Databases and Management of Relationships, " accepted in Int'l Database Engineering & Applications Symposium 98 (1998).

[14] Soe, D.-Y. *et al* : "Schemaless Representation of Semistructured Data and Schema Construction," Proc. of the 8th Int'l Conf. and Workshop on Database and Expert Systems Applications, pp. 387–396 (1997).