XML 文書の統計量を用いた関係データベースへの格納手法

幕　武佳† 太田　学† 片山　薫† 石川　博†

† 東京都立大学大学院　〒 192–0397 東京都八王子市南大沢 1–1
E-mail: †{maku, ohta, katayama, ishikawa}@hikendbs.eei.metro-u.ac.jp

あらまし　現在までに，XML 文書の関係データベースへの格納手法は多数提案されてきた．しかし，表現する情報により構造が大きく異なる XML 文書に対して，画一的に格納する従来手法ではテーブル数が多くなり，検索コストが高くなる．XML のモデルである DTD の構造を利用した格納手法も存在するが，情報量の少ない DTD では不十分である．本研究では，DTD ではなく XML 文書の統計量を用いて，少ないテーブル数と冗長性の低いテーブルスキーマの生成手法を提案する．まず，テーブル数を削減させるために，テーブル数の増加の原因となっていた文字修飾や列挙等の特徴的構造を適切なデータ型として格納する．さらに，出現回数の少ない要素を同一テーブルに格納することを試みる方式をとる．その際は，XML の統計量から冗長性を示す評価関数を利用してデータベーススキーマを決定する．
キーワード　XML，半構造データ，関係データベース，統計量，ストレージ，XML Schema

XML Storage Method Based on the Relational Database using Statistics of XML Documents

Takeyoshi MAKU†, Manabu OHTA†, Kaoru KATAYAMA†, and Hiroshi ISHIKAWA†

† Tokyo Metropolitan University Graduate School
Minamiohsawa 1–1, Hachiohji-shi, Tokyo, 192–0397 Japan
E-mail: †{maku, ohta, katayama, ishikawa}@hikendbs.eei.metro-u.ac.jp

Abstract　Many XML storage methods based on relational databases have already been proposed. These conventional storage methods, however, increase the number of tables because of a wide variety of structures of XML documents and make the search cost high. Although there are also storage methods using DTD which is a document model, the descriptive power of DTDs is insufficient in some cases. In this paper, an efficient XML storage method without using DTDs is proposed, which generates a schema for few number of tables with little redundancy by some statistics of XML documents. In order to reduce the number of tables, appropriate data types for the characteristic structures are defined. We try to store elements of a few occurrences in the same table. Which elements are stored in the same table is decided by using a function evaluating the redundancy of tables.
Key words　XML, Semi-Structured Data, Relational Database, Storage, XML Schema

1. はじめに

1990 年後半に W3C（World Wide Web Consortium）が XML（Extensible Markup Language）を勧告して以来，XML 文書は広く使用されるようになった．XML は，可読性の高さや作成の容易さから，さらにエンドユーザに普及し，現在 Web の標準言語となっている HTML（HyperText Markup Language）にとって代わると考えられる．

現在までに，XML の格納手法は多数提案されてきた．そのなかでも，関係データベースを利用した格納手法が多い．それは，関係データベースは，データ管理しやすいからと思われる．

1.1 関連研究

関係データベースを利用した XML の格納手法は，概ね次の 2 種類に分類できる．

- 固定されたスキーマに格納する手法
- DTD 等のモデルから生成される
  動的なスキーマに格納する手法 [1], [6]

固定されたスキーマに格納する手法には，XML 内の位置情報と要素や文字データ（PCDATA：2.1 章参照）を同一テーブルに格納する手法 [2]，文字データ，要素，パスに分割して格納する手法 [3], [5] などがある．

固定されたスキーマに格納する手法は，どのような XML に対しても，同一のスキーマに格納する．多様な構造の XML を

要素の順序を考慮して格納することができる．また，子孫関係を含んだ問い合わせは，容易に SQL に変換できる．しかし，固定されたスキーマに格納する手法は，数値や固定長文字列等のデータ型の定義できず，データ管理という点でふさわしくないと思われる．データの集計における合計や平均等の計算も，動的なスキーマに格納する手法と比較して困難である．

一方，動的なスキーマに格納する手法は，各要素の値に数値型や固定長文字列等のデータ型を定義することができる．しかし，要素の種類が多くなるとテーブル数も増加するため，問い合わせに対してテーブル結合が頻繁に発生してしまう．その結果，問い合わせの処理コストが高くなってしまう．この解決策として，テーブル数の削減が考えられる．このような研究として，XML の文書モデルを記述した DTD（Ducument Type Definition）を利用してスキーマを生成する手法 [1], [6] もある．また，Shanmugasundaram 等の手法 [6] は，非常に有効であることがわかっている [4]．しかし，この手法は DTD が必須であり，すべてのエンドユーザが DTD を記述するとは考えにくい．類似研究として，DTD，XML の要素の出現回数，頻出クエリから，オブジェクト関係データベース（ORDB:Object-Relational DataBase）のスキーマ生成を補助する手法 [7] も提案されているが，テーブル分割手法は記述されていない．

DTD は，関係データベーススキーマを記述することが目的ではないため，数値型や固定長文字列等のデータ型や親要素に対する子要素の最大出現回数の定義ができず，データベーススキーマを生成するためには効率的ではない．同一の DTD から得られる XML は多様であり，そのような XML に対する効率的なデータベーススキーマが一意に決定するとは限らない．一方，DTD より高い表現力を持つ XML Schema（2. 3 章）は，データベーススキーマ生成に利用できる情報が多い．従って，これらの情報を利用したデータベーススキーマの生成手法が良いと思われるが，XML Schema は，DTD と同様にエンドユーザが利用するとは考えにくい．本研究では，XML Schema 等のモデルが存在しない XML も考慮するため，XML 文書の統計量を算出して，データベーススキーマの生成手法を提案する．

1.2 論文構成

本研究では，テーブルに不必要な Null 値が多くなることを冗長性が高いと考え，出来るだけ少ないテーブル数と低い冗長性である XML のデータベーススキーマを動的に生成することを目的とする．提案手法の特徴は，データベーススキーマを生成する際に DTD や XML Schema ではなく，XML の開始タグの出現回数や親要素に対する子要素の最大出現回数を利用してデータベーススキーマを生成することである．提案手法は，以下の 2 つのアプローチからテーブルスキーマを生成する．

- テーブル数を削減させるために，文字修飾のような複雑なモデルになる構造や親要素に対する子要素の最大出現回数が多い葉要素をテキスト型として格納する．
- テーブル数の削減により生じる冗長性を抑えるために，統計量から Null 値の割合を示す評価関数を定義し，評価関数を小さくするようなデータベーススキーマを生成させる．

```
<history>
  <period title="江戸">
    <year attr="開始">
      <seireki>1603</seireki>
      <wareki>慶長8年</wareki>
      <shogun>
        <name first="松平" last="元康" />
        <name first="徳川" last="家康" />
        <law name="武家諸法度" />
        <law name="キリスト教禁止" />
        <law name="士農工商" />
      </shogun>
    </year>
    <year attr="終了">
      <seireki>1868</seireki>
      <wareki>慶応4年</wareki>
      <commnet>
        <line>この時代は<b>非常に</b>安定<i>！</i>！</line>
        <line>260年も続いた<i>！！</i></line>
        <line><b>参勤交代</b>がよかったのかな<b>？</b></line>
        <line>家康<b><i>万歳</i>！</b></line>
      </commnet>
    </year>
  </period>
  <shogun>
    <name first="徳川" last="綱吉" />
    <law name="生類憐みの令">
      <summary>
        <point>極端な動物愛護令</point>
        <point>金魚や犬の戸籍の作成</point>
        <point>鳥を撃った者は切腹</point>
      </summary>
    </law>
  </shogun>
</history>
```

図 1　サンプル XML

本論文の構成は，以下の通りである．2. 章では，XML の関連技術ついて，本論文に関連している重要な項目を説明する．3. 章では，提案手法の前提条件，以降に用いる定義や利用する統計量についてを説明する．4. 章では，データベーススキーマのテーブル数を削減させることを目的とし，XML の特徴的構造の発見手法とそのような特徴的構造に対する適切な格納手法について説明する．5. 章では，データベーススキーマの冗長性を削減させることを目的とし，Null 値の割合を示す評価関数の定義，評価関数を利用したモデルの分割処理について説明する．6. 章では，問い合わせに対する提案手法の処理の概要を示す．7. 章では，まとめと今後の研究予定について記述する．

2. 関連技術

2.1 XML

XML とは，明快な構造と単純さを目標に，複雑な構造を表現可能な SGML（Standard Generalized Markup Language）と記述が簡単な HTML を基に誕生したものである [8]．XML は，要素（element）・属性（attribute）・文字データ（PCDATA:Parsed Character DATA）から構成される．要素とは，記述する文字データを開始タグ < 要素名 > と終了タグ < /要素名 > で囲うものであり，入れ子にすることで構造化されたデータを表現できる．要素名は記述する文字データのメタデータとするのが一般的である．属性とは要素に属する情報であり，ある要素に重複した属性名を定義することは許されていない．PCDATA とは，マークアップ文字を含まないパースされた文字データのことである．XML は開始要素タグで始まり，終了要素タグで終わる必要がある．このように XML の文法は単純明快であり，XML に準拠した文書は整形式（Well-Formed）と呼ばれる（図 1）．以降，図 1 を用いて説明する．

```
<!ELEMENT shogun ( name+, law+ ) >

<!ELEMENT name EMPTY >
<!ATTLIST name last NMTOKEN #REQUIRED >
<!ATTLIST name first NMTOKEN #REQUIRED >

<!ELEMENT law ( summary? ) >
<!ATTLIST law name NMTOKEN #REQUIRED >
<!ELEMENT summary ( point+ ) >
<!ELEMENT point ( #PCDATA ) >
```

図 2　要素 “shogun” をルートとしたときの DTD

2.2　DTD

DTD とは，SGML から用いられていたものであり，XML の要素名と属性の組み合わせや順序の規則を記述する XML モデルである．ある親要素に対する子要素の出現回数を『?』,『*』,『+』の記号で表現する．それぞれの記号は，0 回または 1 回，1 回以上，0 回以上の出現回数を意味する．また，ID（識別子）・IDREF（識別子参照）・IDREFS（識別子参照リスト）型の規則も記述される．ID とは，XML 文書内の識別子であり一意であることを保証するものである．IDREF や IDREFS は ID 型を参照するデータ型であり，必ず参照先の ID が存在しなければならない．このような，DTD を満たしている XML 文書は，妥当である（Validity）と呼ばれる．

［例 1］　図 1 の要素 “shogun” をルートと仮定したときの DTD の記述は図 2 となる．

2.3　XML Schema

XML Schema [9] とは，DTD の表現力をより強力にした XML モデルであり，XML Schema 自体も XML で記述される．DTD では数値型・文字列型の定義はできないが，XML Schema ではこのような記述が可能であり，複雑な入力規則等も記述が可能である．要素間の関係に関しても，DTD より強力に表現できる．その 1 つとして，ある親要素に対する子要素の出現回数は，最小出現回数（minOccurs）と最大出現回数（maxOccurs）により記述される．

本研究では，この最大出現回数を統計量の 1 つとして利用する．また，DTD や XML Schema などの XML モデルを満たす XML 文書を XML インスタンスと呼ぶが，本論文では XML インスタンスを XML と記述する．

［例 2］　図 1 の要素 “shogun” をルートと仮定したときの XML Schema の記述は，図 3 となる．ここで，最大出現回数の記述が，図 2 の『?』,『*』,『+』から，具体的な数である maxOccurs となっていることに注目して頂きたい．

3.　提案手法の概念

データベーススキーマは低い冗長性と問い合わせ処理に対する少ないテーブルの結合回数であるのが理想である．しかし，冗長性を削減するとテーブル数が増加し，結合回数は増えてしまう．一方，結合回数を抑えるために，テーブル数を削減すると，冗長性が高くなってしまう．通常は，冗長性を少なくするために関係データベースの原則である正規化を行う．最近では，正規化を完全に行わず，配列型やオブジェクト型としてデータを格納するオブジェクト指向関係データベースが標準で用いられている．オブジェクト指向関係データベースという視点から

```
<schema>
 <element name="shogun">
  <complexType>
   <sequence>
    <element name="name" type="typeName"
                                  minOccurs="0" maxOccurs="2" />
    <element name="law" type="typeLaw"
                                  minOccurs="0" maxOccurs="4" />
   </sequence>
  </complexType>
 </element>

 <complexType name="typeName">
  <sequence></sequence>
  <attribute name="first" type="string" />
  <attribute name="last" type="string" />
 </complexType>

 <complexType name="typeLaw">
  <sequence>
   <element name="summary" type="typeSum"
                                  minOccurs="0" maxOccurs="1" />
  </sequence>
  <attribute name="name" type="string" />
 </complexType>

 <complexType name="typeSum">
  <sequence>
   <element name="point" type="string"
                                  minOccurs="0" maxOccurs="3" />
  </sequence>
 </complexType>
</schema>
```

図 3　要素 “shogun” をルートとしたときの XML Schema

考えると，提案手法は XML から配列型やオブジェクト型にふさわしい構造を発見することと似ている．オブジェクト型や配列型の記述法や機能がデータベースソフトに依存してしまうため，本研究ではオブジェクト指向関係データベースを利用せず，関係データベースで定義されているデータ型で格納する．

提案手法は，列挙構造をした葉要素や最大出現回数の少ない幹要素ならば，同一テーブルに格納する方が効率的であると考え，DTD では表現できなかった親要素と子要素の最大出現回数を利用して，テーブル数を削減を試みる．さらに，文字修飾等の複雑な構造は，テキストとして格納することで，テーブル数を削減する．そして，異なる幹要素を同一テーブルに格納すると生じる冗長性に対しては，最大出現回数と開始タグの出現回数から冗長性を示す評価関数を定義し，これを利用して，冗長性を削減する．

［例 3］　最大出現回数の少ない幹要素を同一テーブルに格納する場合を考える．図 1 の “shogun”，“name”，“first”，“last” を例にする．従来手法では，テーブル shogun（shogunID）とテーブル name（shogunID, first, last）とすることが多い．しかし，提案手法では，shogun（shogunID, name.first1, name.last1, name.first2, name.last2）と同一テーブルにする．

3.1　前 提 条 件

ここでは，提案手法の前提条件について説明する．提案手法では，XML の統計量を利用し，効率的なデータベーススキーマを生成するので，統計量の正確さが重要である．そのため，解析する XML は，ある程度のデータが存在するものとする．XML の統計量は，以下に従って算出した．

i.　XML の属性名と属性値は，要素名とその値とみなす．

ii.　直接要素で包含されていない文字データは子要素 <#PCDATA> で包含する．

iii.　子要素とした属性と文字データは親要素名も一致しなければ，同一データとはみなさない．

DTD に関して，DTD の ID や IDREF が定義されている場合

は，冗長を効率的に削減することができる（5.1 章），提案手法では仮定しない．テーブルスキーマに関して，親要素に対する子要素の順序は，同要素名の順序は保持するが，全体の要素名の順序は保持しない．それは，関係データベースには列の順序は重要視されていないため，そのような機能を実現できないからである．しかし，<order>要素等で擬似的に囲うことで，全体の要素名の順序は保持できる．

［例 4］ 図 1 を見ていただきたい．<law name="武家諸法度" /> の属性名 “name” と属性値 “武家諸法度” は，前述の（i），（iii）より，<law><(law.)name>武家諸法度</(law.)name></law>となる．また，<line>260 年も続いた！！</line>の，直接要素で包含されていない文字データ “260 年も続いた” は，（ii），（iii）より，<line><(line.)#PCDATA>260 年も続いた</(line.)#PCDATA> !! </line> となる．

3.2 語句解説

DTD のように XML の要素と親子関係を表現したものをモデル $\mathcal{M}$ という．本論文のモデルは，要素を表すノード集合と，親要素と子要素の最大出現回数をラベルにもつエッジ集合からなる（図 4）．簡単のため，XML には要素，モデルにはノードというが，ノードは XML の要素である．また，子孫ノードを持つノードを幹ノード，持たないノードを葉ノードとする．モデル $\mathcal{M}$ に対する，ノード A の関係を以下のように定義をする．

［定義 1］
i. $|\cdot|$ ： モデルやノード集合のノード数
ii. $r(\mathcal{M})$ ： $\mathcal{M}$ のルートノード
iii. $\mathcal{P}(A)$ ： A の親ノード集合
iv. $p(A)$ ： A の親ノード集合かつ ルートからの深さが最小であるノード
v. $\mathcal{C}(A)$ ： A の子ノード集合
vi. $\mathcal{D}(A)$ ： A の子孫ノード集合
vii. $\mathcal{L}(A)$ ： A の子孫ノード集合かつ 葉ノードである集合

［例 5］ 図 1 のモデルをグラフ表現する（図 4）．このモデル $\mathcal{M}$ のノード A を “shogun” とする．定義 1 の各値は以下のようになる．

$$
\begin{aligned}
&|\mathcal{M}| = 21\\
&r(\mathcal{M}) = \text{history}\\
&\mathcal{P}(\text{shogun}) = \text{history, year}\\
&p(\text{shogun}) = \text{history}\\
&\mathcal{C}(\text{shogun}) = \text{name, law}\\
&\mathcal{D}(\text{shogun}) = \text{(shogun.)name, law, first, last,}\\
&\qquad\qquad\qquad\qquad \text{(law.)name, summary, point}\\
&\mathcal{L}(\text{shogun}) = \text{first, last, (law.)name, point}
\end{aligned}
$$

3.3 XML の統計量

ここでは，提案手法で用いる XML から取得する統計量の説明をする．XML から取得する統計量は，要素の開始タグの出現回数（cnt:count）と最大出現回数（mo:MaxOccurs），Null 率（np:Null Probability）とする．以下にその定義を示す．

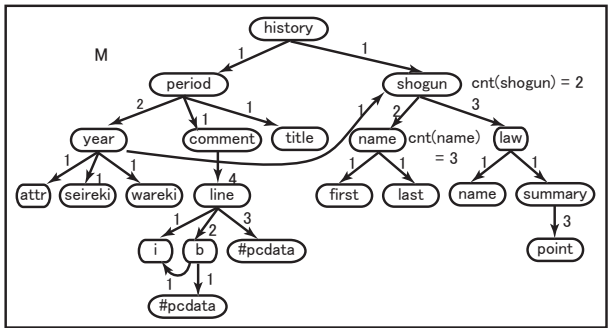


図 4 図 1 の統計量付モデル（最大出現回数と一部の *cnt* のみ表示）

```
CREATE TABLE shogun
(
   name1.first   text,
   name2.first   text,
   name1.last    text,
   name2.last    text,
   law1.name     text,
   law2.name     text,
   law3.name     text,
   law1.summary.point1   text,
   law2.summary.point1   text,
   law3.summary.point1   text,
   law1.summary.point2   text,
   law2.summary.point2   text,
   law3.summary.point2   text,
   law1.summary.point3   text,
   law2.summary.point3   text,
   law3.summary.point3   text
)
```

図 5 図 4 の “shogun” をルートとした場合のテーブルスキーマ

［定義 2］
i. $cnt(e)$: 要素 e の開始タグの出現回数
ii. $mo(p(e), e)$: 親要素 $p(e)$ に対する子要素 e の の最大出現回数
iii. $np(p(e), e) = 1 - cnt(e) / \big(cnt(p(e)) \cdot mo(p(e), e)\big)$

図 1 の統計量付モデルは図 4 となる．但し，表示しているのは，最大出現回数と一部分の開始タグの出現回数のみである．

［例 6］ 図 4 のように，$cnt(\text{shogun}) = 2$, $cnt(\text{name}) = 3$, $mo(\text{shogun, name}) = 2$ とする．このとき，ノード “shogun” に対するノード “name” の Null 率は，以下となる．

$$
\begin{aligned}
np(\text{shogun, name}) &= 1 - cnt(\text{name}) / (cnt(\text{shogun}) \cdot mo(\text{shogun, name}))\\
&= 1 - 3/(2 \cdot 2)\\
&= 0.25 \qquad (1)
\end{aligned}
$$

3.4 モデルからデータベーススキーマへの変換

データベーススキーマは，複数のテーブルスキーマからなる．各テーブルスキーマは，要素名に順序を付加した名前のフィールド名からなる．また，親要素と子要素の区切りは “.” としている．これは，5. 章で特に重要になってくる．

［例 7］ 図 4 の “shogun” をルートと仮定したときに，生成されるテーブルスキーマは，図 5 となる．

4. テーブル数の削減

一般に，動的なスキーマに格納する手法は，テーブル数が多くなってしまう．DTD を利用したスキーマ生成手法では，親要素に対する子要素の最大出現回数が 2 回でも『*』や『+』と表記されてしまうため，最大出現回数がわからない．そのため，列挙構造をした葉要素や最大出現回数の少ない幹要素は，テーブルを分割することにより冗長性が大幅に削減できるため，分割対象となる場合が多い．また，文字修飾の再帰が複雑に入り

組んだ特長的な構造を分散してテーブルに格納することは，明らかに効率的ではなく，テーブル数も増加してしまう．そこで，提案手法は，葉要素の列挙構造や文字修飾などの特徴的な構造は，そのままテキストとして格納する．最大出現回数の少ない幹要素に関しては，5.2 章で説明する．

以下，最大出現回数を利用して，テーブル数の削減を期待できる特徴的構造の定義，格納手法，発見手法について説明する．

4.1 テーブル数の削減を期待できる特徴的構造

ここでは，特徴的な構造をテキスト型として格納することで，テーブル数を削減し，冗長性に関係している最大出現回数を変更する．この最大出現回数は，5. 章で定義する評価関数に用いられる．提案手法では，以下の特徴的構造に対し，そのような処理を行う．

i. 文字修飾の複雑な構造
ii. 葉要素を列挙している構造

4.1.1 文字修飾の複雑な構造

文字修飾の複雑な構造とは，ある要素内に#PCDATA,<i>, <b>, <sup>, <sub> 等が複雑に入り組んだ構造のことである．このような構造をテーブルに分散させて，格納することは明らかに効率的でないため，このような構造はそのままテキスト型として格納する．このようなデータを RAW データと呼ぶ．

このような文字修飾は#PCDATA の存在が特徴的である．従って，#PCDATA に注目した以下のようなアルゴリズムを使用して，文字修飾の発見を行う．

STEP 1． #PCDATA の最大出現回数が 2 回以上の要素を発見する．
STEP 2． STEP 1 以外の要素で，#PCDATA の最大出現回数が 1 回以上，または要素数（属性は含まない）が 2 個以下の要素を発見する．
STEP 3． STEP 2 に含まれる要素を持つ step 1 の要素の子孫要素全てを RAW データとする．
STEP 4． STEP 1 の要素に，最大出現回数 1 回の子要素 <RAW> を追加し，RAW データの子孫要素を削除する．

［ルール 1］ ある要素 e がテキストや修飾タグ等の複雑な構造を持つ親要素ならば，その子孫要素はそのままテキスト型に格納する．そして，$mo(p(e),e)$ に 0 を，$mo(p(e),\mathrm{RAW})$ に 1 を代入する．

$$e \text{ is RAW} \Rightarrow mo(p(e),e) := 0\ ,\ mo(p(e),\mathrm{RAW}) := 1\ \ \big(e \in \mathcal{M}\big) \tag{2}$$

［例 8］ 図 4 の要素 “comment” と子孫要素を例にして説明する（図 6）．STEP 1 に該当する要素は “line” であり，STEP 2 に該当する要素は “i”，“b”，“(line.)#PCDATA”，“(b.)#PCDATA” となる．“line” は，STEP 2 に該当する要素を持つため，それらは RAW データとして 1 つの要素（図 6 右）となる．

4.1.2 葉要素を列挙している構造

葉要素の列挙構造に対して，その葉要素のみ（実際のデータが格納されるフィールド数が 1 つ）のテーブルを生成することは効率ではない．なぜならば，検索時には，その葉要素の親・祖先情報も取得する場合が一般的であるため，結局テーブルの結合が頻繁に発生すると考えられるからである．そのため，同一テーブルに格納しても許容と考えられる葉要素の最大出現回数の閾値を MMO（Maximum MaxOccurs）とし，ある要素が葉要素でかつ最大出現回数が MMO より多ければ，テキスト型に列挙して格納する．このようなデータを ENUM データと呼ぶ．

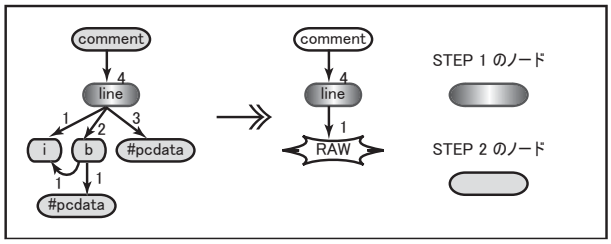


図 6 文字修飾構造の処理過程

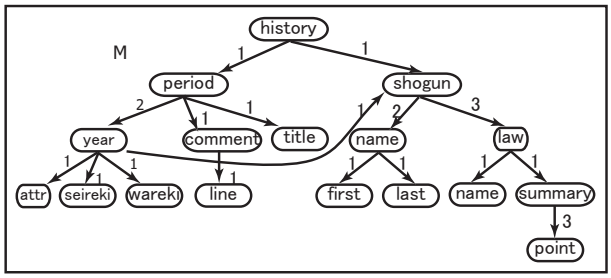


図 7 モデル図 4 のテーブル削減後

［ルール 2］ ある要素 e が葉要素でかつ最大出現回数が MMO より多ければ，テキスト型に格納する．そして，$mo(p(e),e)$ に 1 を代入する．

$$mo(p(e),e) > \mathrm{MMO} \Rightarrow mo(p(e),e) := 1\ \ \big(e \in \mathcal{L}(M)\big) \tag{3}$$

［例 9］ 図 4 の要素 “line” は，ルール 1 により，図 6 右となる．mo(line, RAW)= 1 で，$|\mathcal{D}(\text{line})| = 1$ となっているため，要素 “line” を要素 “line(.RAW)”，mo(line, RAW)= 0 とする．そして，MMO = 3 とした場合，ルール 2 により $mo(\text{commnet}, \text{line(.RAW)}) = 1$ となる（図 7）．

5. 冗長性の削減

テーブル数の削減処理（4. 章）を行っただけのモデル $\mathcal{M}$ から生成されるデータベーススキーマは，冗長性が高い．そこで，幹要素の列挙構造等の構造は分割し，モデルから冗長性の低いデータベーススキーマの生成手法を提案する．このように，分割された個々のモデルをテーブルモデルと呼ぶ．

テーブルモデルはモデルの部分グラフとなるため，定義 1 を満たす．また，テーブルモデル $\mathcal{T}$ は，モデル $\mathcal{M}$ から複数のテーブルモデル $\mathcal{T}^1, \mathcal{T}^2, \ldots$ が生成される．テーブルモデル数は生成されるテーブルスキーマのテーブル数，テーブルモデルのルート名はテーブル名と一致し，テーブルモデルの大きさはテーブルのフィールド数にも依存する（図 5）．以下に，テーブルモデルの数を抑えながら，冗長性を効率的に削減する手法について説明する．

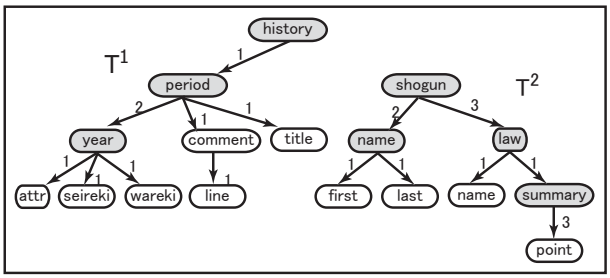


図 8　テーブルモデルの生成過程（網掛け:$e \in \overline{\mathcal{L}(M)}$）

5.1　従来手法で分割を行う構造

ここでは，明らかに別テーブルにする必要のある構造を説明する．そのような構造として，幹要素の列挙，再帰や要素間の参照関係が存在する閉路を含む構造等がある．これらの構造は冗長性が高いため，様々な格納手法でも分割対象となっている．しかし，提案手法は従来手法と異なり，幹要素の列挙構造の分割は最大出現回数により決定する．これは，提案手法の概念である最大出現回数の少ない幹要素ならば同一テーブルに格納した方が効率的という考えからである．従って，幹要素の最大出現回数が MMO より多い構造と閉路を含む構造は別テーブルモデルとする．

［例 10］　図 7 では，$|\mathcal{P}(\text{shogun})| \neq 1$ であり，再帰・参照構造を成しているルートノードである．そのため，別テーブルモデルとして扱う．その結果，エッジは削除され，$mo(\text{history}, \text{shogun}) = mo(\text{year}, \text{shogun}) = 0$ となる（図 8）．

5.2　冗長性を考慮したテーブルモデルの分割手法

最大出現回数の少ない幹要素を同一テーブルに格納すると，テーブル数は削減するが，必然的に Null 値が増加し，テーブルスキーマの冗長性が高くなってしまう．このことは，図 5 から容易に想像できる．そのため，5.1 章から得られるテーブルモデルから，テーブルモデルの数を抑えながら，冗長性を効率的に削減する手法について説明する．

従来のいくつかの格納手法は，DTD を利用してテーブル（本論文でいうテーブルモデル）の分割を行っている．提案手法は，DTD では表現できない情報を XML インスタンスから取得し，これを利用してテーブルモデルの分割を行う．あるテーブルモデルの冗長性を定式化するため，XML の統計量から，Null 値の列数を示す評価関数を Null 値列として定義する．この Null 値列を利用して，少ないテーブルモデル数と低い冗長性である分割候補を発見する．

テーブルモデル $\mathcal{T}$ の分割候補は $2^{\text{エッジの数}-1} = 2^{|\overline{\mathcal{L}(T)}|-1}$ 通り存在する．以下，$I = |\overline{\mathcal{L}(T)}| - 1, J = 2^I$ とし，分割候補をテーブルモデル集合 $\mathcal{TS}_1 = \{\mathcal{T}_1^1\}, \mathcal{TS}_2 = \{\mathcal{T}_1^2, \mathcal{T}_2^2\}, \mathcal{TS}_3 = \{\mathcal{T}_1^3, \mathcal{T}_2^3\}, \ldots, \mathcal{TS}_J = \{\mathcal{T}_1^J, \mathcal{T}_2^J, \ldots \mathcal{T}_I^J\}$ とする．この中から，

i. Null 値の割合が多い
ii. 同名葉ノード数が多い
iii. ある子ノードに関するフィールド数が多い

に該当するテーブルモデル $\mathcal{T}_i^j$ を含んだテーブルモデル集合 $\mathcal{TS}_j$ は除外する．以下に，その理由を述べる．

5.2.1　Null 値列による制限

Null 値の割合が多い場合は，明らかに効率的なスキーマではない．そのため，テーブルモデルに対して，Null 値列 NF（Null Fields）を定義し，その閾値 MNF（Maximum Null Fields）を設ける．Null 値列とは，テーブルモデルの Null の値の割合の指標であり，少ない方が望ましいが，テーブルモデルの大きさ $|\mathcal{T}|$ が大きいほど増加する指標である．Null 値列 $\text{NF}(\mathcal{T})$ が MNF より大きいテーブルモデルを持つテーブルモデル集合は除外する．

以下，$\prod mo\big(p(n), n\big) = mo\big(p(n), n\big) \cdot mo\big(p(p(n)), p(n)\big) \cdots$，$\prod np\big(p(n), n\big) = np\big(p(n), n\big) \cdot np\big(p(p(n)), p(n)\big) \cdots$ とする．

［ルール 3］　$\text{NF}(\mathcal{T}) > \text{MNF}$ をとなるテーブルモデル $\mathcal{T}$ を持つテーブルモデル集合 $\mathcal{TS}$ は除外する．但し，テーブルモデル $\mathcal{T}$ の Null 値列 $\text{NF}(\mathcal{T})$ は以下の式で求める．

$$\text{NF}(\mathcal{T}) = \sum_{l \in \mathcal{L}(T)} \Big(\prod_{n=l}^{p(n)=r(\mathcal{T})} mo\big(p(n), n\big) \Big) - \sum_{l \in \mathcal{L}(T)} mo(p(l), l) \prod_{n=p(l)}^{p(n)=r(\mathcal{T})} mo(p(n), n)(1 - np(p(n), n)) \quad (4)$$

ここで，第一項はテーブルモデル $\mathcal{T}$ のフィールド数と一致する．

$$\text{F}(\mathcal{T}) = \sum_{l \in \mathcal{L}(T)} \Big(\prod_{n=l}^{p(n)=r(\mathcal{T})} mo\big(p(n), n\big) \Big) \quad (5)$$

［例 11］　図 10 で，テーブルモデル集合 $\mathcal{TS}_4$ のテーブルモデル $\mathcal{T}_2$ の Null 値列 $\text{NF}(\mathcal{T}_2)$ は，ルール 3 から，以下のように導出される．

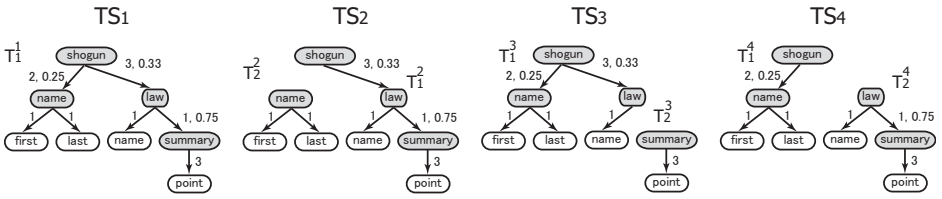
$$\begin{aligned} \text{NF}(\mathcal{T}_2) &= mo(\text{summary, point}) \cdot mo(\text{law, summary}) \\ &\quad - mo(\text{summary, point}) \cdot mo(\text{law, summary}) \cdot \big(1 - np(\text{law, summary})\big) \\ &= 3 \cdot 1 - 3 \cdot (1 \cdot 0.25) \\ &= 2.25 \end{aligned}$$

5.2.2　同名葉ノード数による制限

フィールドに同名葉ノード数が多い場合は，SQL の Where 句に OR 句の数が非常に多くなるので，葉ノード数が多い場合はテーブルを分割した方が適切と考えられる．図 5 では，フィールド “point” を検索する場合は 9 つのフィールドを検索する必要があり，多数の OR 句により問い合わせが長くなるため可読性が失われる．そのため，フィールド内に最も出現する葉ノード名を最大同名葉ノード数 LLF（Large Leaf Fields）と定義し，その閾値 MLLF（Maximum Large Leaf Fields）を設ける．最大同名葉ノード数が MLLF より大きいテーブルモデルを持つテーブルモデル集合は除外する．

［ルール 4］　$\text{LLF}(\mathcal{T}) > \text{MLLF}$ をとなるテーブルモデル $\mathcal{T}$ を持つテーブルモデル集合 $\mathcal{TS}$ は除外する．但し，テーブルモデル $\mathcal{T}$ の最大同名葉ノード数 $\text{LLF}(\mathcal{T})$ は以下の式で求める．

$$\text{LLF}(\mathcal{T}) = \max_{l \in \mathcal{L}(T)} \Big(\prod_{n=l}^{p(n)=r(\mathcal{T})} mo\big(p(n), n\big) \Big) \quad (6)$$



TableSet	|TS|	Table	F(T)	NF(T)	LLF(T)	LCF(T)	ΣF(T)	ΣNF(T)	ΣLLF(T)	ΣLCF(T)
TS1	1	T1	16	9.5	9	12	16	9.5	9	12
TS2	2	T1	12	8.5	9	12	12	8.5	9	12
		T2	-	-	-	-				
TS3	2	T1	7	2	3	4	7	2	3	4
		T2	-	-	-	-				
TS4	2	T1	4	1	2	4	8	3.25	5	7
		T2	4	2.25	3	3				

図 10　テーブルモデルの分割候補

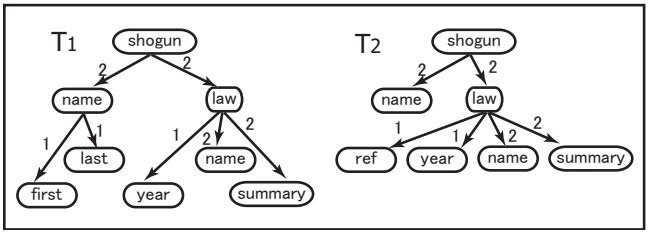


図 9　最大同子ノード名数の例

5.2.3　ある子ノードに関するフィールド数による制限

図 9 の $\mathcal{T}_1, \mathcal{T}_2$ のようなモデルを仮定する．それぞれのモデルは，$np(p(e), e) = 0, (e \in \mathcal{T}_1, \mathcal{T}_2)$ とする．従って，$\mathrm{NF}(\mathcal{T}_1) = \mathrm{NF}(\mathcal{T}_2)$，$\mathrm{LLF}(\mathcal{T}_1) = \mathrm{LLF}(\mathcal{T}_2)$ となる．そして，テーブルモデル $\mathcal{T}_1$ と $\mathcal{T}_2$ をそれぞれ 2 つに分割することを考える．この場合，$\mathcal{T}_1$ も $\mathcal{T}_2$ も “shogun” と “law” のエッジを削除し，モデルを分割することが効率的と思われる．ここで，$\mathcal{T}_1$ と $\mathcal{T}_2$ を比較すると，$\mathcal{T}_2$ のほうがモデル構造が不均衡なので，より分割しやすいと考えられる．不均衡であれば，NF も大きくなる可能性があるが，NF は Null 値の割合も依存するものである．

このような場合の指標として，フィールド内に最も出現するある子ノードに関するフィールド数を最大同子ノード名数 LCF(Large Child Fields) と定義し，その閾値 MLCF（Maximum Large Child Fields）を設ける．すなわち，最大同名葉ノード数が MLCF より大きいテーブルモデルを持つテーブルモデル集合は除外する．

［ルール 5］　$\mathrm{LCF}(\mathcal{T}) > \mathrm{MLCF}$ をとなるテーブルモデル $\mathcal{T}$ を持つテーブルモデル集合 $\mathcal{TS}$ は除外する．但し，テーブルモデル $\mathcal{T}$ の最大同子ノード名数 $\mathrm{LCF}(\mathcal{T})$ は以下の式で求める．

$$\mathrm{LCF}(\mathcal{T}) = \max_{c \in \mathcal{C}(T)} \Big\{ \sum_{l \in \mathcal{L}(c)} \Big(\prod_{n=l}^{p(n)=T} mo\big(p(n), n\big) \Big) \Big\} = \max_{c \in \mathcal{C}(T)} \big(F(\mathcal{T}_c)\big) \quad \big(\mathcal{T}_c = c \cup \mathcal{D}(c), r(\mathcal{T}_c) = c\big) \quad (7)$$

［例 12］　図 9 の $\mathcal{T}_1, \mathcal{T}_2$ を考える．$\mathrm{MLCF} = 11$ とすると，$\mathrm{LCF}(\mathcal{T}_1) = 10$，$\mathrm{LCF}(\mathcal{T}_2) = 12$ であるため，$\mathcal{T}_2$ を含むテーブル集合は除外される．

5.3　最終的なテーブルモデル集合の決定手法

ここでは，ルール 3-5 を満たすテーブルモデル集合の中から最終的な分割の選択手法について説明する．まず，提案手法の目的はテーブルの削減であるため，最もテーブル数の少ないテーブルモデル集合を選択する．そのような，テーブルモデル集合が複数存在する場合は，Null 値列の総和 $\big(= \sum_{i=1}^{|\mathcal{TS}_j|} \mathrm{NF}(\mathcal{T}_i)\big)$ が最小である候補を選択する．このように，冗長性とテーブル数を削減するような分割を行い，効率的なテーブルスキーマを生成する．

［例 13］　図 7 のノード “shogun” のテーブルモデルを考える（図 10）．分割候補は，$2^{|\overline{\mathcal{L}(T)}|-1} = 8$ 通り存在するが，ここではテーブル数の少ない 4 通りのみを説明する．

各パラメータは，$\mathrm{MNF} = 3, \mathrm{MLLF} = 4, \mathrm{MLCF} = 8$ とする．まず，全く分割しないテーブルモデル集合 $\mathcal{TS}_1$ は，閾値の条件を満たさない．次に，1 つのエッジを削除するので，分割候補は 3 つとなる．その中で，$\mathcal{TS}_2$ はルール 3-5 を満たさないが，他の $\mathcal{TS}_3, \mathcal{TS}_4$ は満たす．そして，Null 値列の総和が最小であるテーブルモデル集合 $\mathcal{TS}_3$ に基づき分割を行い，テーブルモデル $\mathcal{T}_1, \mathcal{T}_2 \in \mathcal{TS}_3$ から効率的なテーブルスキーマが生成される．

6.　問い合わせ

ここでは，提案手法の XML データベースへの問い合わせについて，概要を説明する．提案手法のテーブルスキーマは，図 11 のように，ノード情報テーブル（NodeInfo），コンポーネント情報テーブル（ComponentInfo），提案手法により生成されるテーブルからなる．コンポーネント情報テーブルは，祖先や子孫関係の複雑な問い合わせに対して，効率よく検索するためのものであり, [2] 等で利用されているスキーマと似ている．異なる点は，テーブルモデルのルートノードの情報しか保持しないことである．つまり，位置情報は，コンポーネント情報テーブルとノード情報テーブルで分割して管理する．コンポーネント情報テーブルは，テーブルモデルのルートノードの XML 全体に対する位置情報を格納する．ノード情報テーブルは，テーブルモデルのルートノードでないノードとテーブルモデルのルートノードの相対的な位置情報を格納する．

ノード情報テーブルは，識別子，名前空間，属性か要素かを判別できるようなノード名，親ノード，ルートまでの最短パス，格納されているテーブル名，ENUM か否か，RAW か否か，テーブルのフィールド名，フィールド名の最大反復回数，テーブル

TABLE : NodeInfo

列名	説明
NID	ノード識別子
NSID	名前空間識別子
Name	ノード名
ParentNID	親のノード識別子
AncestorPath	ルートタグまでの最短パス
TableName	格納されているテーブル名
Enum	列挙型か否か
Raw	RAW型か否か
FieldName	テーブルのフィールド名
MaxOccurs	フィールド名の最大反復回数
Distance	テーブル名タグからの相対距離

TABLE : ComponentInfo

列名	説明
CID	コンポーネント識別子
NID	ノード識別子
DID	XMLインスタンスの識別子
eCID	コンポーネントの終了識別子
Depth	ルートノードからの距離
rCID	参照コンポーネント

TABLE : table A-C

列名	説明
CID	コンポーネント識別子
SameAP	TagInfoのAncestorPathと一致するか否か
----	以下はXMLインスタンスによる

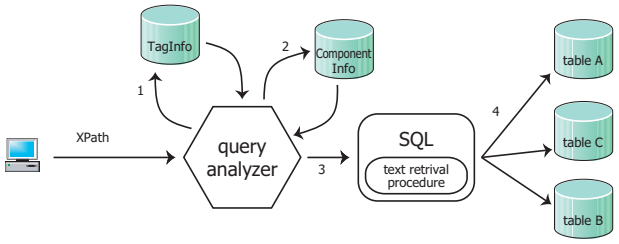


図 11　提案手法のテーブルスキーマとその問い合わせに対する処理

名ノードからの相対距離からなる．コンポーネント情報テーブルは，識別子，ノード識別子，XML インスタンス識別子，コンポーネントの終了タグの位置，ルートノードからの距離，参照コンポーネントからなる．生成されたスキーマには，コンポーネント識別子やノード識別子の最短ルートパスと一致するか否かのフィールドを追加する．

提案手法の問い合わせは XPath [10] をクエリ解析し，SQL に変換することで結果を得る．テーブルモデルから生成された各テーブルは，コンポーネント識別子と祖先パスと一致するかの情報を保持するため，祖先パスと一致する問い合わせでは，位置情報テーブルを参照しなくても処理ができる．また，祖先や子孫関係の複雑な問い合わせは，位置情報テーブルの範囲関係から処理できる．

提案手法は，ノードの位置情報を分割することで，データの挿入が効率的になる．従来手法では，データを挿入する場合，挿入したい箇所にその上下の識別子の平均の識別子をつけて，挿入するのが一般的である．しかし，提案手法では，ルートノードに一致するデータを挿入する場合しか，そのような処理は生じない．また，ルートノードでないデータを挿入する場合も効率的に処理できる．最もデータ操作が多いと思われる，RAW データや ENUM データのデータ挿入は，値を単に更新するだけの処理となる．それ以外のデータの更新は，フィールド値の単なる更新かフィールドの追加処理となる．フィールドの追加は望ましくないが，ある程度データの記述された XML を前提としているため，このようにフィールドの追加が必要となるクエリは，あまり発生しないと考えられる．

7. おわりに

本研究では，XML を関係データベースに格納する際，問題となっていたテーブル数の増加に対して，異なる幹要素を可能な限り同じテーブルに格納する手法を提案した．テーブル数の削減には，テーブル数増加の主因と考えられる，文字修飾などの複雑な構造・葉要素の列挙構造・親要素に対する子要素の最大出現回数が少ない構造に対し，それぞれの特徴に合わせた適切なデータ型を提案して対処した．また，可能な限り同じテーブルに格納することで生じてしまうテーブル内の冗長性は，評価関数として XML の統計量である Null 値列を定義し，この評価関数を用いて分割・結合の決定を行った．

問い合わせに対する処理時間やテーブルの結合回数・冗長性による性能評価，生成されたデータベーススキーマに対する問い合わせ処理の実装，類似構造の統合に関しては，今後研究する予定である．

謝辞

本研究の一部は，文部科学省科学研究費特定領域研究 (2)[情報学:A02]（課題番号:14019075），東京都立大学総長特別研究費 (特別重点研究)，日本データベース学会・マイクロソフト株式会社共催 2002 年度データベース研究支援プログラムによる．

文　献

[1] Alin Deutsch, Mary Fernandez, Dan Suciu, “Storing Semistructured Data with STORED,” *In Proceeding of SIGMOD Conference*, pp.431–442, June, 1999.

[2] Chun Zhang, Jeffrey F. Naughton, David J. DeWitt, Qiong Luo, Guy M. Lohman, “On Supporting Containment Queries in Relational Database Management Systems,” *In Proceedings of the VLDB Conference, Santa Barbara*, 2001.

[3] Daniela Florescu, Donald Kossmann, “Storing and Querying XML Data using an RDBMS,” *IEEE Data Engineering Bulletin*, vol.22, no.3, 1999.

[4] Feng Tian, David J. De Witt, Jianjun Chen, Chun Zhang, “The Design and Performance Evaluation of Alternative XML Storage Strategies,” *ACM Sigmod Record*, vol.31, no.1, pp.5–10, March, 2002.

[5] Haifeng Jiang, Hongjun Lu, Wei Wang, Jeffrey Xu Yu, “Path Materialization Revisited: An Efficient Storage Model for XML Data,” *Thirteenth Australasian Database Conference*, 2002.

[6] Jayavel Shanmugasundaram, Kristin Tufte, Gang He, Chun Zhang, David DeWitt, Jeffrey Naughton, “Relational Databases for Querying XML Documents: Limitations and Opportunities,” *In Proceedings of the VLDB Conference, Edinburgh, Scotland*, 1999.

[7] Meike Klettke, Holger Meyer, “XML and Object-Relational Database Systems — Enhancing Structural Mappings Based On Statistics,” *Lecture Notes in Computer Science*, vol.1997, 2001.

[8] XML : `http://www.w3.org/XML/`

[9] XML Schema : `http://www.w3.org/TR/xmlschema/`

[10] XPath : `http://www.w3.org/TR/xpath/`