DEWS2004 7-A-04

文書スキーマに基づくノード識別子を用いた関係データベースへのXML文書の効率的な格納

藤本　圭[†]　清水　敏之[††]　吉川　正俊[†††]

† 名古屋大学 工学部 電気電子情報工学科
†† 名古屋大学 情報科学研究科
††† 名古屋大学 情報連携基盤センター
E-mail: {†fujimoto, ††shimizu}@dl.itc.nagoya-u.ac.jp, †††yosikawa@itc.nagoya-u.ac.jp

あらまし　本論文では DTD 情報を取り入れた XML 木のノードを一意に識別できるラベル付け手法である SPIDERS-Dewey Order を提案する．これまでに提案されてきたラベル付け手法は DTD 情報を利用していなかったが，DTD が存在する場合は DTD 情報を利用してデータベースの検索性能を下げることなく更新性能を上げることができる．我々が開発している XML データベースである XRel を拡張することにより SPIDERS-Dewey Order によるラベル付け手法を利用して XML 文書を格納した．XML 文書を格納するための関係スキーマを定義し，様々な XPath 式をその関係スキーマ上の SQL 問合せに変換した．それらの SQL 問合せを実行し，SPIDERS-Dewey Order を用いた場合のデータベースの性能評価を行った．

キーワード　XML, XML データベース, 文書スキーマ, ノードラベル付け

Efficient Storage of XML Documents in Relational Databases using a Node Identifier based on Document Schemas

Kei FUJIMOTO[†], Toshiyuki SHIMIZU[††], and Masatoshi YOSHIKAWA[†††]

† Dept. of Information Engineering, School of Engineering, Nagoya University
†† Graduate School of Infomation Science, Nagoya University
††† Infomation Technology Center, Nagoya University
E-mail: {†fujimoto, ††shimizu}@dl.itc.nagoya-u.ac.jp, †††yosikawa@itc.nagoya-u.ac.jp

Abstract　In this paper, we propose SPIDERS-Dewey Order, a node labeling scheme for XML documents using DTD. Node labeling schemes that have been proposed until now do not use DTDs. However, by utilizing DTDs information, we can improve update performance without loss of retrieval performance. We extend XRel, an XML database we are developing, on top of relational databases, by using SPIDERS-Dewey Order. Then, we define a relational database schema for storing XML documents utilizing SPIDERS-Dewey Order, and translate various XPath expressions into SQL using SPIDERS-Dewey Order. We evaluate the performance of the database using SPIDERS-Dewey Order, by executing translated SQLs.

Key words　XML, XML Database, Document Schema, Node Labeling Scheme

1. はじめに

W3C(World Wide Web Consortium) から仕様が勧告された XML(Extensible Markup Language) はインターネット上で文書やデータを表現するためのメタ言語である．現在，XML を利用したアプリケーションが多数考えられており，今後 XML で記述された文書，データが増加していくと考えられる．XML で記述された文書が大量になった場合，データベースで効率的に管理することが求められる．

XML データベースの実現には関係データベース，オブジェクト指向 (object-oriented) データベース，XML ネイティブなデータベースを用いる方法などが提案されている．我々が開発している XML データベースシステムである XRel [1] [2] は関係データベースに XML 文書を格納するアプローチを採用している．1) 関係データベースが既に多くの応用システムで用いられていること．2) 大量のデータが関係データベース上に蓄積さ

れていること．3) 関係データベースの問合せ最適化やトランザクション管理などの過去 20 年以上に渡って研究された技術蓄積があること．以上のことから XML 文書と既存のデータ資源とを連携して利用できることが関係データベースを用いる利点である．

XML 文書を関係データベースに格納する方法として，XML 文書をそのまま文字列として格納する方法と XML 文書を分解して関係スキーマに写像する方法がある．XRel では後者の方法を採用している．後者の方法はさらに構造写像アプローチとモデル写像アプローチに分けられる．構造写像アプローチは XML 文書の論理構造を表現する関係スキーマを定義する方法で，XML 文書スキーマを関係表の構造に反映させる．モデル写像アプローチは XML データモデルの構成要素を表現する関係スキーマを定義する方法で，固定された関係スキーマを持つ一つ以上の関係表で XML 文書の各要素を表現する．XRel ではモデル写像アプローチを採用しており，4 個の関係表で XML 文書を表現する．

XML 文書は木構造であるが関係は表構造である．従って，XML 文書を関係として関係データベースに格納する際に木構造のトポロジーを明示的に保存する必要がある．トポロジーを保存するために XML の木構造をもとに各要素にラベルを割り当てる方法がある．ラベル付け手法により XML 文書の検索，更新の性能が変化する．

これまでに提案されてきたラベル付け手法は DTD 情報を利用していなかったが，DTD が存在する場合は DTD 情報を利用してデータベースの検索性能を劣化させることなく更新性能を上げることができる．本論文では DTD 情報を取り入れたラベル付け手法である SPIDERS-Dewey Order (Schema-based Path IDentifiER with Sibling enumeration-Dewey Order) を提案し，SPIDERS-Dewey Order を用いて XML 文書を関係へ格納する方法について述べる．SPIDERS-Dewey Order によるラベルは検索性能と更新性能のバランスがとれていること，DTD 情報を利用して更新性能を上げていることが特徴である．

以下，第 2 節では関連研究として XML データベースと XML 文書に対するラベル付けの研究について記述する．第 3 節では SPIDERS-Dewey Order の基礎となるラベル付け手法について説明し，SPIDERS-Dewey Order について述べる．第 4 節では SPIDERS-Dewey Order によりラベル付けされた XML 文書を格納する関係スキーマを定義し，様々な XPath 式をその関係スキーマ上の SQL 問合せに変換する例を挙げる．第 5 節では様々な XPath 問合せを SQL 問合せに変換したものを XRel 上で実行し，本手法の有効性を示す．第 6 節では本研究のまとめと今後の課題について述べる．

2. 関 連 研 究

2.1 XML データベースに関する研究

2.1.1 XParent

XParent は Jiang らが開発した XML データベースシステムである [6] [7]．XML 文書を分解して関係に格納するアプローチを取っており，経路情報を格納する関係，データを格納する関係，要素ノードを格納する関係，データと経路の関連を表す関係の 4 種の関係で XML 文書を表現している．DTD の有無に関わらずどのような XML 文書も同様に取り扱う．

2.1.2 PostgreSQL を用いた多機能な XML データベース

油井らは PostgreSQL を用いた XML データベースを開発した [5]．PostgreSQL と XMLPG SQL に XPath 処理系を組み合わせて多機能な XML データベースを実現している．XML 文書を分解して関係に格納するアプローチを取っている．DTD の有無に関わらずどのような XML 文書も同様に取り扱う．ラベル表現に Dewey Order を採用している．

2.2 ラベル付けに関する研究

2.2.1 Global Order, Local Order, Dewey Order

Tatarinov らが Global Order, Local Order, Dewey Order の 3 種のラベル付け手法を提案した [4]．Global Order は XML 木の各要素を深さ優先の順序でラベル付けする手法である．Local Order は各要素の兄弟間の順序のみを保存する手法である．Dewey Order は各要素にラベルとして親のラベルに兄弟間の順序を連結したものを付与する手法である．

これら 3 種の手法に対して XML 文書更新時のラベル更新範囲を示し，問合せ実行時の検索性能比較を行っている．その結果，1)Global Order は検索性能が高い反面，更新性能が低い，2)Local Order は更新性能が高い反面，検索性能が低い，3)Dewey Order は Global Order と Local Order の中間の性能を持つ，という性質が述べられている．

2.2.2 DTD を利用したラベル付け手法

Dihn Kha らは DTD を利用してラベル付けをする手法 SPIDER (Scheme-based Path IDentifiER) を提案した [3]．この手法では DTD の情報を利用して根ノードから各要素ノードに至る経路を一意に識別する整数をラベルとして各ノードに付与する．根ノードからの経路が同じであるノード全てに同一のラベルが割り当てられる．ラベルからそれらのノードはのうち一つを識別することはできない．SPIDER については 3.2 節で詳しく説明する．

2.3 関連研究との比較

データベースに格納する XML 文書に対して DTD が存在する場合は DTD の情報を利用して検索性能を劣化させることなく更新性能を上げることができる．ノードを一意に識別できるラベル付け手法はこれまで DTD を利用しないものしか存在しなかった．本研究では DTD の情報を利用したより効率的なノードを一意に識別可能なラベル付け手法について提案する．さらにそのようなラベル付け手法を利用して XML 文書を関係へ格納し，問合せを実行する方法について述べる．

3. XML 木に対するラベル付け手法

関係データベースへ XML 文書を格納する際，XML 文書の木構造を保存するために各ノードにラベルを割り当てる．ラベル付け手法により XML 文書の検索，更新の性能が変化する．

本論文で提案するラベル付け手法である SPIDERS-Dewey Order は Dewey Order [4] と SPIDERS を基礎としている．本節では Dewey Order と SPIDER，SPIDERS について説明し，

SPIDERS-Dewey Order について述べる．

3.1 Dewey Order

Dewey Order [4] は Dewey Decimal Classification [8] をもとにしたラベル表現である．Dewey Order では XML 順序木の各ノードのラベルは親のラベルに兄弟間の順序を連結したものとなる．Dewey Order によるラベル付けの例を図 1 に示す．

Dewey Order によるラベル付けの利点は検索と更新の性能のバランスが優れていることである．ノード挿入などの更新処理が行われた場合，XML 木の一部のラベルを付け替える必要がある．Dewey Order によるラベルの更新範囲は挿入ノードの兄弟で挿入ノードより後にあるものとその子孫である．ノード挿入が起きた際の Dewey Order のラベル更新範囲を図 2 に示す．

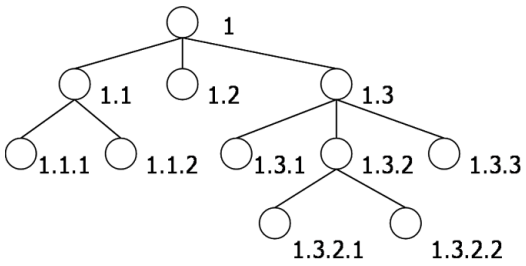

図 1　Dewey Order によるラベル付け

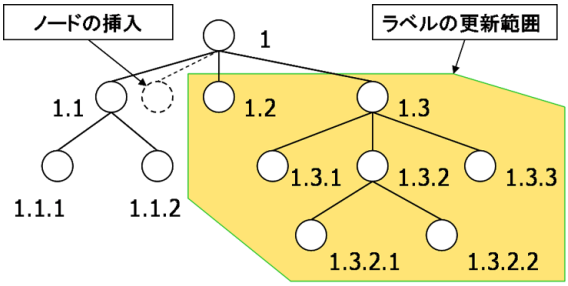

図 2　Dewey Order を用いたラベルの更新範囲

3.2 SPIDER

SPIDER は Din Kha らが提案したラベル付け手法である [3]．DTD の情報を利用して根ノードから各要素ノードに至る経路を一意に識別する整数をラベルとして各要素ノードに付与する．根ノードからの経路が同じであるノード全てに同一のラベルが割り当てられる．

この手法では次の手順で各要素のラベルを算出する．

（1） DTD を元に StruDTD という表を作成する．StruDTD は親要素の名前と子要素の名前，cOrder と呼ばれる数字を列とする表である．cOrder は親要素と子要素の名前の組から cOrder が一意に特定でき、cOrder と子要素の名前の組から親要素の名前が一意に特定でき、親要素の名前と cOrder から子要素の名前が一意に特定できるようにつけられた数字である。

（2） StruDTD をもとに SPIDER を計算する．$p.sid$ を親要素のラベル，$c.sid$ を子要素のラベル，f を cOrder の最大値，childOrd$(p.tag, c.tag)$ を親要素が $p.tag$ であり，子要素が $c.tag$ であるときの cOrder とすると，SPIDER は次の式で計算できる．SPIDER はルートノードのラベルを 1 として再帰的に計算される．

$$c.sid = (p.sid - 1) \times f + 1 + \text{childOrd}(p.tag, c.tag)$$

以上のように計算を行い各ノードにラベルを与えると，子ノードのラベルから親ノードのラベルを一意に計算することができる．

SPIDER の例を挙げる．

```
<!ELEMENT personnel (company, business, person)>
<!ELEMENT person (name+, email?, person*)>
<!ELEMENT name (family, given)>
```

図 3　図 4 の XML 文書の DTD

図 3 の DTD をもとに作成した StruDTD を表 1 に示す．表 1 をもとにラベルを計算すると図 4 のようになる．

表 1　StruDTD の例

PAR	CHI	cOrder
personnel	company	1
personnel	business	2
personnel	person	3
person	name	2
person	email	3
person	person	4
name	family	1
name	given	2

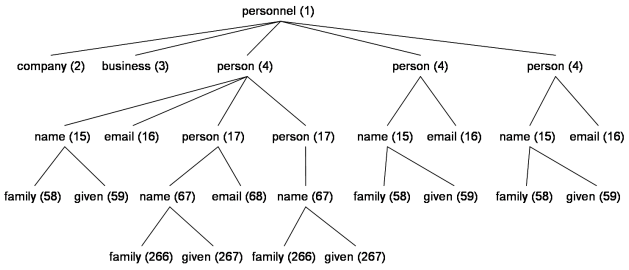

図 4　SPIDER によるラベル付け

3.3 SPIDERS

SPIDER では同一経路に現れる各ノードには同じラベルを付与するため，そのようなノードをラベルにより一意に識別することができない。この問題を解決するため，同一経路に現れる各ノードには SPIDER ラベルの後に兄弟間の順序を表す番号を付与する．このラベル表現を SPIDERS と呼ぶ．

図 4 において，/personnel/person に出現する 3 個の person 要素ノードの SPIDERS ラベルは左から順にそれぞれ 4-1, 4-2, 4-3 となる．

SPIDERS ラベルのみではノードを一意に識別することができないことに注意する．たとえば，person(4-1) の子である name(15) と person(4-2) の子である name(15) の二つのノードはラベルのみでは識別できない．この点を解決するために次に述べる SPIDERS をもとにした SPIDERS-Dewey Order を導入し，ノードを一意に識別できるようにする．

3.4 SPIDERS-Dewey Order

SPIDERS-Dewey Order は Dewey Order と SPIDERS を基礎としたラベル表現である．SPIDERS-Dewey Order では XML 順序木の各ノードのラベルは，そのノードに対して計算した SPIDERS ラベルを親のラベルに連結したものとなる．SPIDERS-Dewey Order によるラベル付けの例を図 5 に示す．

SPIDERS-Dewey Order によるラベル付けの利点はノード挿入の際のラベル更新範囲が狭くなることである．たとえばあるノードが挿入された場合，SPIDERS-Dewey Order によるラベルの更新範囲は挿入ノードと同じ名前の兄弟要素で挿入ノードより後にあるものとその子孫である．ノード挿入が起きた際の SPIDERS-Dewey Order のラベル更新範囲を図 6 に示す．

SPIDERS-Dewey Order では異なる名前の要素ノードが兄弟に混在して出現する場合にラベルによってノードの出現順序を正確に表現できない．このような例を図 7 に示す．図 7 の 2 つの文書はラベルから区別することができない．上記のような要素ノードがコンテクストノードとなるような XPath 式の問合せに対してこの点が問題となるが，そのような問合せが与えられることは稀であると考える．また，DTD と XPath 式から SPIDERS-Dewey Order によるラベル表現で XPath 式の問合せに対応できるかどうか判定できる．したがって，SPIDERS-Dewey Order によるラベル表現で対応できない問合せに対してはラベル表現を用いない処理を行えばよい．具体的な処理に関しては今後の課題である．

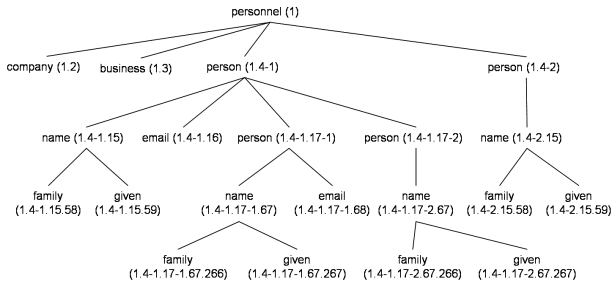

図 5　SPIDERS-Dewey Order によるラベル付け

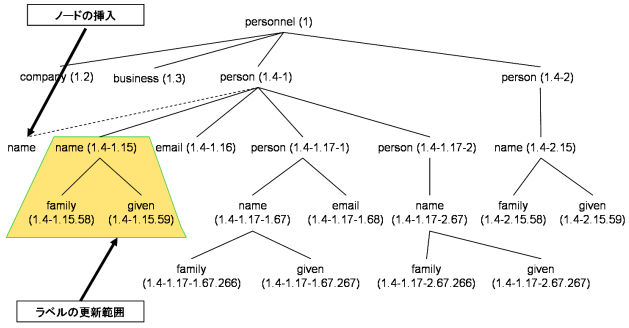

図 6　SPIDERS-Dewey Order を用いたラベルの更新範囲

4. XML 文書の関係への格納

4.1 XRel の概要

XRel [1] [2] は我々が開発している XML データベースシステムである．XRel では XML 文書を分解して固定したスキーマを持つ 4 個の関係に格納する．関係スキーマは DTD や文書の論理構造に現れる要素型に依存しないため，文書の論理構造の変化が関係スキーマに影響を及ぼさない．XML 文書の各要素はラベル表現と XML 木における根からの経路によって管理される．

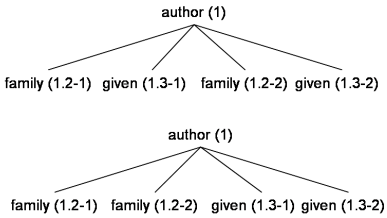

図 7　SPIDERS-Dewey Order が順序を正確に表現できない XML 木の例

XRel では XML 問合せ言語 XPath の問合せを SQL 問合せに変換し，SQL 問合せとして実行している．XRel のアーキテクチャのうち XPath インタフェースの部分の概要を図 8 に示す．図 8 において，システム側は利用者やアプリケーションに対して XML 文書が関係の形で管理されていることを隠す．利用者やアプリケーション側から XML 問合せ言語による問合せが行われた場合，システム側が自動的に SQL に変換して SQL 問合せを実行する．

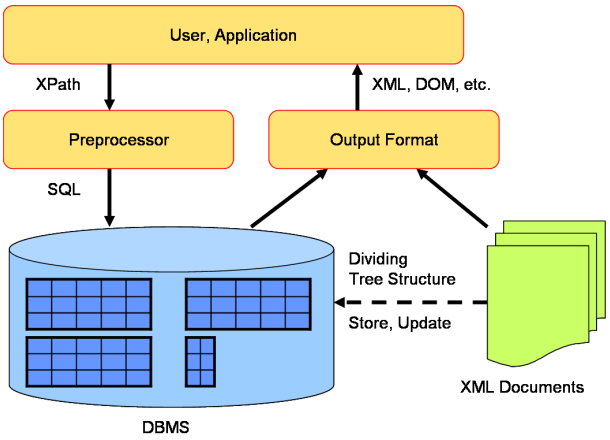

図 8　XRel アーキテクチャの概要

従来の XRel では XML 文書の各ノードの開始タグと終了タグの物理的なバイト位置をデータとして格納することにより XML の木構造を保存していた．この方法では XML 文書にノード挿入が起きた際，挿入位置よりも後に出現する全てのノードの開始タグと終了タグのバイト位置を更新しなければならないため，文書更新のコストが高くなってしまう．また，バイト位置から親子関係を判定する場合には二つのノード間に他のノードが含まれていないことを判定する必要があるが，この計算は時間がかかる．従って，親子関係を判定する必要がある問合せについては検索処理時間がかかる．これらの問題を解決するために，バイト位置ではなく SPIDERS-Dewey Order を利用して XML 文書の木構造を保存することによって検索性能を劣化さ

せることなく文書更新の性能を上げることを目標とする．

4.2 XML 文書を格納する関係スキーマ

XML 文書を格納するための関係スキーマとして次の 4 個を定義する．

```
Element (docID, nodeID, label, omit, parentID, pathID)
  key は (docID, nodeID)
Attribute (docID, nodeID, label, omit,
                              parentID, pathID, value)
  key は (docID, nodeID)
Text (docID, nodeID, label, omit,
                              parentID, pathID, value)
  key は (docID, nodeID)
Path (pathID, pathexp)
  key は (pathID)
```

Element, Attribute, Text, Path はそれぞれ要素ノード，属性ノード，テキストノード，XML 文書に現れる全ての経路を格納する関係表である．XPath 式の経路式の処理は経路情報を利用することにより効率的に行える [1]．

次に各属性の説明を述べる．

docID	関係 Element, Attribute, Text で定義する． XML 文書を識別するための ID である．
nodeID	関係 Element, Attribute, Text で定義する． ノードを一意に識別するために付与した 通し番号である． 順序に意味はなく，更新を行わない．
label	関係 Element, Attribute, Text で定義する． ノードに付与した SPIDERS-Dewey Order の ラベルを格納する．
parentID	関係 Element, Attribute, Text で定義する． 問合せ処理の効率化のために親の nodeID を指定する．
value	関係 Attribute, Text で定義する． Attribute では属性値を，Text では要素内の 文字列をそれぞれ格納する．
pathID	関係 Element, Attribute, Text, Path で定義する． 経路を識別するための ID である．
pathexp	関係 Path で定義する． 実際の経路表現を格納する．

次に XML 文書を関係に格納した例を挙げる．

```
<!ELEMENT books (book+)>
<!ELEMENT book (title, editor, author, summary)>
<!ELEMENT title (#PCDATA)>
<!ELEMENT editor (family, given)>
<!ELEMENT author (family, given)+>
<!ELEMENT summary (#PCDATA|keyword)*>
<!ELEMENT family (#PCDATA)>
<!ELEMENT given (#PCDATA)>
<!ELEMENT keyword (#PCDATA)>
<!ATTLIST book style CDATA>
```

図 9　図 10 の XML 文書の DTD

図 9 の DTD に従った文書の例を図 10 に示す．図 10 の XML 文書を木構造で表現したものが図 11 である．図 10, 11 で表される XML 文書を関係に格納したものが表 2 である．

4.3 問合せ変換

利用者やアプリケーションが関係データベースに格納された XML 文書に対して XPath 式で問合せを行うと，システム側でその問合せを SQL に変換し，関係データベースへの問合せを実行する．これにより，利用者やアプリケーションは関係の形で格納された XML 文書をあたかも実際の XML 文書が存在するかのように取り扱うことができる．

```
<books>
    <book style="textbook">
        <title>Designing XML applications</title>
        <editor>
            <family>Bob</family>
            <given>Kraft</given>
        </editor>
        <author>
            <family>Nick</family>
            <given>Marcus</given>
            <family>Bob</family>
            <given>Pant</given>
        </author>
        <summary>Guide to design<keyword>XML</keyword>applications.</summary>
    </book>
</books>
```

図 10　XML 文書の例

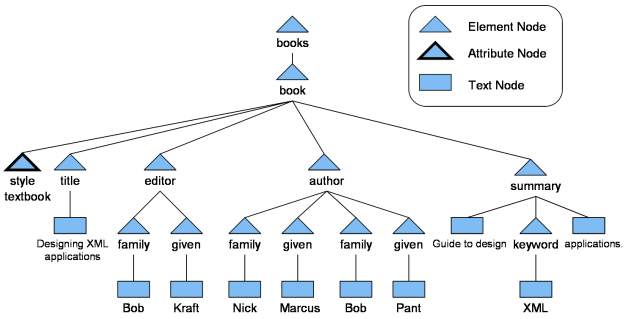

図 11　XML 文書を表現する木構造の例

本節では様々な形の XPath 式を本論文で提案する関係スキーマに対する SQL 問合せに変換した例を示す．基本的な問合せとして単純な経路式，順序の指定がある場合，文書内容に関する条件が指定された場合，軸が指定された場合について取り上げる．例 1, 例 2, 例 3 の XPath 式は XMark [9] ベンチマークで利用される XQuery 問合せから抜き出した．例 4 の XPath 式は XMark で使用される XML 文書スキーマに対する問合せを作成した．

単純な経路式の変換について述べる．XPath 式中に現れる単純経路に対しては関係 Path に対する文字列照合によって経路 ID を求める．SQL 問合せの WHERE 節で LIKE 述語を用いることにより文字列照合を実現できる．この際，単純経路に現れる//は LIKE 述語中で%/に置き換える．このとき LIKE 述語により接頭辞が同じ複数のタグ名が選択されることを防ぐため，経路表現では#をタグ名の終端記号として用いる．関係 Path から得られた経路 ID に対応する要素を関係 Element から求め，文書 ID とノード ID を得る．

例 1　/site/regions//description

```
SELECT   e1.docID, e1.nodeID
FROM     Element e1, Path p1
WHERE    p1.pathexp LIKE '#/site#/regions#%/description'
AND      e1.pathID = p1.pathID
ORDER BY e1.docID, e1.label
```

順序の指定がある場合は問合せの条件を満たすノードに対してラベルの昇順に番号を割り当て，割り当てた番号を指定する．これは SQL-99 で定義された RANK 関数を用いて実現できる．

また，二つのノードの先祖子孫関係の判定に SUBSTRING 関数を利用している．

表 2　XML 文書の格納例

Element

docID	nodeID	label	parentID	pathID
1	1	1.	null	1
1	2	1.2.	1	2
1	3	1.2.8.	2	8
1	4	1.2.9.	2	9
1	5	1.2.10.	2	10
1	6	1.2.11.	2	11
1	7	1.2.9.42.	4	42
1	8	1.2.9.43.	4	43
1	9	1.2.10.48-1.	5	48
1	10	1.2.10.49-1.	5	49
1	11	1.2.10.48-2.	5	48
1	12	1.2.10.49-2.	5	49
1	13	1.2.11.54.	6	54

Attribute

docID	nodeID	omit	parentID	pathID	value
1	1	1.2.7.	2	2	textbook

Text

docID	nodeID	label	parentID	pathID	value
1	1	1.2.8.	3	8	Designing ...
1	2	1.2.9.42.	7	42	Bob
1	3	1.2.9.43.	8	43	Kraft
1	4	1.2.10.48-1.	9	48	Nick
1	5	1.2.10.49-1.	10	49	Marcus
1	6	1.2.10.48-2.	11	48	Bob
1	7	1.2.10.49-2.	12	49	Pant
1	8	1.2.11.	6	11	Guide ...
1	10	1.2.11.54.	13	11	XML
1	9	1.2.11.	6	11	applications.

Path

pathID	pathexp
1	#/books
2	#/books#/book
7	#/books#/book#/@style
8	#/books#/book#/title
9	#/books#/book#/editor
42	#/books#/book#/editor#/family
43	#/books#/book#/editor#/given
10	#/books#/book#/author
48	#/books#/book#/author#/family
49	#/books#/book#/author#/given
11	#/books#/book#/summary
54	#/books#/book#/summary#/keyword

例 2　/site/open_auctions/open_auction/bidder[1]/increase

```
SELECT   e3.docID, e3.nodeID
FROM     Element e3, Path p3,
        (SELECT  e2.docID, e2.nodeID, e2.label
            RANK() OVER (PARTITION BY e2.docID,
                                      e2.parentID
                         ORDER BY e2.label) position
         FROM    Element e1, Element e2, Path p1
         WHERE   p1.pathexp LIKE
         '#/site#/open_auctions#/open_auction#/bidder'
         AND     e1.pathID = p1.pathID
         AND     e2.parentID = e1.nodeID ) AS e
WHERE    e.position = 1
AND      p3.pathexp LIKE
'#/site#/open_auctions#/open_auction#/bidder#/increase'
AND      e3.pathID = p3.pathID
AND      e.label = SUBSTRING(e3.label, 1, LENGTH(e.label))
ORDER BY e3.docID, e3.label
```

XPath 式で文書内容に関する条件が指定された場合，SQL 問合せでは WHERE 節に文書内容に関する条件を追加すればよい．関係 Text や関係 Attribute の属性 value を用いる．

例 3　/site/open_auctions/open_auction
[bidder/personref/@person=”person32”]/reserve

```
SELECT   e3.docID, e3.nodeID
FROM     Element   e1, Path p1,
         Attribute a2, Path p2,
         Element   e3, Path p3
WHERE    p1.pathexp LIKE
'#/site#/open_auctions#/open_auction'
AND      p2.pathexp LIKE
'#/site#/open_auctions#/open_auction#/bidder#/personref
#/@person'
AND      p3.pathexp LIKE
'#/site#/open_auctions#/open_auction#/reserve'
AND      e1.pathID = p1.pathID
AND      a2.pathID = p2.pathID
AND      e3.pathID = p3.pathID
AND      a2.value  = 'person32'
AND      e1.label = SUBSTRING(a2.label,1,LENGTH(e1.label))
AND      e1.label = SUBSTRING(e3.label,1,LENGTH(e1.label))
AND      e1.docID = a2.docID
AND      e1.docID = e3.docID
ORDER BY e3.docID, e3.label
```

軸が指定された場合は，各軸に対する条件をラベル表現を利用して指定する．例えば following-sibling の場合はコンテクストノードと同じ親を持ち，コンテクストノードよりも大きなラベルを持つノードとなる．

例 4　/site/closed_auctions
/closed_auction/following-sibling::*

```
SELECT   e2.docID, e2.nodeID
FROM     Element e1, Path p1,
         Element e2
WHERE    p1.pathexp LIKE
'#/site#/closed_auctions#/closed_auction'
AND      e1.pathID = p1.pathID
AND      e1.parentID = e2.parentID
AND      e1.label < e2.label
ORDER BY e2.docID, e2.label
```

親が同じである異なる名前を持つ要素ノードやテキストノードが混在する場合，SPIDERS-Dewey Order によるラベルの特性により順序の指定や軸の指定がある問合せを処理できない場合がある (3.4 節を参照)．たとえば，/article/bdy/sec を親とするノードで異なる名前のノードが混在する場合，/article/bdy/sec/*[2] や /article/bdy/sec/ss1[1]/following-sibling::node() という問合せをラベルを利用して処理することができない．従って，このような場合はラベルを使用せずに問合せを処理する必要がある．具体的な処理方法については今後の課題である．

5.　性能評価と考察

本節では SPIDERS-Dewey Order によるラベルを付与した XML 文書の検索性能と更新性能に関して評価を行う．比較対象は木構造をタグの出現位置で保存する従来の XRel とする．

XMark [9] で使用される XML 文書を作成し，SQL 文を実行して問合せ処理時間を計測した．実験に使用した XML 文書は一つで容量は 1.12MB，要素ノード数は 17132 個，属性ノード数は 3919 個，テキストノード数は 31089 個，単純経路の種類は 454 種である．

検索性能に関しては様々な特徴を持つ XPath 式を SQL に変換したものを，拡張した XRel と従来の XRel に対して実行した．XPath 式は XMark ベンチマークで利用される XQuery 問合せから抜き出した．実験に使用する問合せを表 3 に示す．

更新性能に関しては文書の一部にノードが挿入された場合に更新すべきラベルあるいは出現位置を修正する SQL 文を実行した．文書の先頭部分，末尾部分の一箇所にノードを挿入した場合を想定し，ラベルあるいは出現位置を書き換えるための SQL Update 文を実行した．

表 3　検索性能評価のための XPath 式

	XPath 式
Q1	/site/open_auctions/open_auction
Q2	/site/regions//description
Q3	//open_auctions//description
Q4	/site/open_auctions/open_auction
	[bidder/personref/@person=”person32”]/reserve

実験環境は CPU: Intel Pentium III Processor 1.26GHz-S (デュアル), メモリ: 2GB, OS: Red Hat Linux 7.1 である．使用した DBMS は Oracle 9i Release 1 である．問合せ実行時間は SQL*Plus の SET TIMING で計測した．問合せ結果を表示しない場合の実行時間である．

従来の XRel と今回の XRel での検索性能評価のための問合せ Q1 から Q4 の実行時間を表 4 に示す．実行時間は問合せを 3 回実行した平均である．

表 4　検索性能評価のための XPath 処理の実験結果

	従来の XRel (秒)	拡張した XRel (秒)
Q1	0.23	0.23
Q2	0.23	0.23
Q3	0.24	0.24
Q4	2.67	7.60

表 4 から次のことがわかる．Q1 から Q3 の問合せに対しては検索性能の違いは見られない．これは従来の XRel でも拡張した XRel でも同じ問合せ処理を行うためである．Q1 から Q3 の問合せは単純経路による問合せであり，関係 Path に対する文字列照合によって検索を行う．

Q4 の問合せに対しては従来の XRel の方が検索性能が高い．これは子孫先祖判定を行う際に，従来の XRel では開始タグと終了タグのバイト位置の比較演算で行うのに対し，拡張した XRel では SUBSTRING 関数を使用するためだと考えられる．

従来の XRel と拡張した XRel での更新性能評価のための問合せ実行時間と更新影響範囲のノード数を表 5 に示す．実行時間は問合せを 3 回実行した平均である．

表 5 からどの場合も従来の XRel よりも拡張した XRel の方

表 5　更新性能評価のための問合せ処理の実験結果

	文書先頭部分		文書末尾部分	
	時間 (秒)	ノード (個)	時間 (秒)	ノード (個)
従来の XRel	16.86	35249	4.14	7480
拡張した XRel	0.12	192	0.51	5320

が更新処理の実行時間が短く，更新性能が高いことがわかる．ラベルあるいはタグ位置の更新範囲の違いが原因である．従来の XRel では挿入されたノード以降に出現する全ての開始タグと終了タグのバイト位置を更新する必要がある．文書の前の方にノードを挿入するほどタグ位置の更新範囲が広くなっていることが表 5 で確認できる．拡張した XRel で利用している SPIDERS-Dewey Order のラベル更新範囲は挿入ノードの後の兄弟とその子孫である．今回の実験では特に文書先頭部分へのノード挿入に対して更新範囲に差が現れている．文書スキーマによっては SPIDERS-Dewey Order によるラベル付け手法が非常に有効であると言える．

従来の XRel では関係 Element, Text, Attribute には属性として開始タグ属性と終了タグ属性が用意されており，文書更新時には二つの属性を更新する必要がある．拡張した XRel では文書更新時はラベル属性のみを更新すればよい．この点でも更新性能の違いが現れたと考えられる．

6. おわりに

本論文では文書スキーマを利用したノード識別子 SPIDERS-Dewey Order を提案し，ラベル情報の関係への格納と，基本的な XPath 問合せから SQL 問合せへの変換を示した．

SPIDERS-Dewey Order は XML 文書に対して DTD が存在する場合に利用可能なラベル表現である．文書にノード挿入が起きた際のラベル更新範囲は Dewey Order よりも狭いため，Dewey Order よりも更新性能が高い．親のラベルを持つという Dewey Order と同じ性質を持つため，Dewey Order とほぼ等しい検索性能を持つ．

開始タグと終了タグによって木構造を保存したデータベースと SPIDERS-Dewey Order によって木構造を保存したデータベースを比較した．前者は先祖子孫判定が必要な問合せを効率的に処理できるため，そのような問合せに対して後者よりも検索性能が高い．後者は前者よりもラベルあるいはタグ出現位置の更新範囲が狭いため，更新性能が高い．

今後の課題としては以下の点を挙げることができる．

- **SPIDERS-Dewey Order で処理できない問合せへの対応**

SPIDERS-Dewey Order には，その特性からラベル情報を利用して処理できない問合せが存在する．このような問合せに答える場合に行う処理を考えることが課題である．

- **XPath 式から SQL 問合せへの変換モジュールの作成**

今回の研究では性能評価実験において XPath による問合せから SQL 問合せへの変換を手動で行った．実際は XPath によって問合せが行われた場合，対応する SQL 問合せにシステム側が自動的に変換して関係データベースに対して問合せを実行する．任意の XPath 式を本論文で提案した関係スキーマ対

応する SQL 問合せに変換するモジュールを作成することが今後の課題である．

- **Dewey Order の省略**

Dewey Order の特性上，文書の先頭部分に要素が挿入された場合に更新範囲が広くなり更新性能が低下するという欠点がある．また，Dewey Order や SPIDERS-Dewey Order によるラベルは可変長であり，データへのアクセスが遅くなるという欠点もある．これらの欠点を補うため一部の部分木のラベルの親や先祖を表す部分を省略することで更新範囲を狭くし，かつラベルの長さを固定長にすることを考える．ラベルを省略する部分木を指定するアルゴリズムを考えることが今後の課題である．

文　　献

[1] 吉川 正俊, 志村 壮是, 植村 俊亮 : オブジェクト関係データベースを用いた XML 文書の格納と検索, 情報処理学会論文誌, Vol.40, No.SIG 6(TOD 3), pp.115-131 (1999).

[2] Masatoshi Yoshikawa, Toshiyuki Amagasa, Takeyuki Shimura, Shunsuke Uemura : XRel: A Path-Base Approach to Strage and Retrieval of XML Documents Using Relational Database, ACM Transactions on Internet Technology, Vol.1, No.1, pp.110-141 (2001).

[3] Dao Dinh Kha, Masatoshi Yoshikawa, Shunsuke Uemura : Virtual Joins for XML Data, Information Science Technical Report NAIST-IS-TR2003012,Graduate School of Information Science, Nara Institute of Science and Technology, ISSN 0919-9527 (2003).

[4] Igor Tatarinov, Stratis D. Viglas, Kevin Beyer, Jayavel Shanmugasundaram, Eugene Shekita, Chun Zhang : Storing and Querying Ordered XML Using a Relational Database System, ACM SIGMOD 2002, pp.204-215 (2002).

[5] 油井 誠, 森嶋 厚行 : PostgreSQL を用いた多機能な XML データベース環境の構築, 情報処理学会論文誌：データベース, Vol.44, No.SIG 12(TOD 19), pp.11-22 (2003).

[6] Haifeng Jiang, Hongjun Lu, Wei Wang, Jeffrey Xu Yu : Path Materialization Revisited: An Efficient Storage Model foe XML Data, AICE2000 (2001).

[7] Haifeng Jiang, Hongjun Lu, Wei Wang, Jeffrey Xu Yu : XParent: An Efficient RDBMS-Based XML Database System, ACDE'02 (2002).

[8] Online Computer Library Center. Introduction to the Dewey Decimal Classification.
http://www.oclc.org/oclc/fp/about/about_the_ddc.htm

[9] XMark - An XML Benchmark Project.
http://www.xml-benchmark.org/