IP-labeling：接尾辞経路に対する極小なXMLラベル付け手法

根本 潤[†] 遠山元道[††]

本論文では，XML文書の接尾辞経路に対する新しいラベル付け手法IP-labelingを提案する．XML問合せ処理を効率的に行うためには範囲ラベル付け手法だけでは不十分であり，接尾辞経路に対するラベル付けが重要視されている．しかしながら，接尾辞経路に対する既存のラベル付け手法であるP-labelingはラベルとして非常に大きな数値を用いるため，ラベルサイズが大きくなるという問題点がある．そこで，提案するIP-labelingでは接尾辞経路に対して極小な区間をラベルとして付与することによりラベルサイズの低減をはかる．さらに，IP-labelingの拡張をし，ラベル付けの際に値ノードも接尾辞経路の一部として扱うことで，等価条件を含む問合せを効率良く処理する手法VIP-labelingを提案する．実験結果により，IP-labelingによるラベルサイズの低減が問合せ処理を高速化すること，および，VIP-labelingが等価条件を含む問合せ処理を高速化することを示す．

IP-labeling: Minimal Range Labeling for Suffix Path of XML Documents

Jun Nemoto[†] and Motomichi Toyama[††]

We propose IP-labeling, a novel labeling scheme for suffix path of XML documents. Since range labeling schemes are not enough to efficiently process XML queries, suffix path labeling schemes have been considered as one of the most important research topics. However, P-labeling which is a previous work for suffix path labeling schemes has a problem that the size of suffix path labels are too large since it uses huge numbers as suffix path labels. In order to avoid this problem and reduce the size of the label, IP-labeling annotates suffix path with minimal-ranged labels. Moreover, we propose VIP-labeling which is an extension of IP-labeling and also a labeling scheme for suffix path. It treats value nodes as a part of suffix path and enables to efficiently process queries, which have equality predicates. Experimental results demonstrate that reducing the size of labels using IP-labeling have a large increase in speed of query processing and VIP-labeling have a large increase in speed of query processing with equality predicates.

1. はじめに

XML[7]はビジネスや科学の分野における半構造データの表現やデータ交換のためのディファクトスタンダードの言語である．XMLの重要性の拡大にともない，XML文書に対する様々な索引手法や問合せ手法が研究されてきた．本論文では，既存のアプローチよりも優れたパフォーマンスを得ることができる新しいラベル付け手法を提案する．

XMLは木構造で表現されるため，XML文書の特定の部分を指し示す構文を規定したXPath[5]やXQuery[6]といったXML問合せ言語では子供ノードを指定するChild軸（/）や子孫ノードを指定するDescendant軸（//）が問合せの中核を占める．加えて，複数のノードに対して選択条件を付与した，枝分かれを含む小さな木の形で表現される問合せ（枝分かれ問合せ）も重要な問合せクラスである．たとえば，XPathで与えられる `//inproceedings[/author = 'John Doe']/title` という問合せは，authorノードの値がJohn Doeであるようなinproceedingsノードの子供ノードtitleを求める枝分かれ問合せである．つまり，この問合せにより，著者名がJohn Doeである論文のタイトルを得ることができる．XMLデータベースにおいて，こうした経路や枝分かれパターンにマッチするものをすべて検索することは，XML問合せ処理におけるコアオペレーションである．すなわち，先祖子孫関係や親子関係の判定，および述語の処理をいかに効率良く行うかがきわめて重要である．

これまで，先祖子孫関係や親子関係を効率良く処理するために様々なラベル付け手法が提案されてきた[1),3),10),15),21)]．最も一般的なのは，各XMLノード

† 慶應義塾大学大学院理工学研究科
Graduate School of Science and Technology, Keio University
†† 慶應義塾大学理工学部
Faculty of Science and Technology, Keio University

の開始タグの位置と終了タグの位置でラベル付けを行い，位置の包含関係から先祖子孫関係を判定するものである．先祖子孫関係と親子関係はノードの深さ情報によって区別される．問合せ中の各先祖子孫関係や親子関係は結合操作として処理されるが，結合操作は非常に高価な処理であるため，Chen らは，連続する Child 軸を効率良く処理することができる P-label を提案している[9]．P-label は接尾辞経路と呼ばれる，経路に対するラベル付けである．

しかしながら，開始タグと終了タグのみによるラベル付け手法では大規模な XML 文書に対する効率的な XPath 問合せ処理を実現できない．なぜなら，そのようなラベル付け手法では，先祖子孫関係，親子関係を多く含む問合せにおいて，それらを判定するためには，結合操作を複数回行わなければならず，大きなノード集合どうしの結合処理は非常に高価だからである．

また，その問題点を解決するために提案された接尾辞経路に対するラベルである P-label も非効率なラベル生成を行っているため，ラベル長が増大し，結果として問合せの際にディスク IO が増加するので，索引としての効果が十分得られないという問題がある．さらに，これまでのラベル付け手法は値ノードについて深く議論がなされていないため，述語の処理において改善すべき余地がある．

そこで，本研究では接尾辞経路に対する新しいラベル付け手法 IP-labeling を提案する．提案手法では，まず，XML 文書から IP-Tree と呼ばれるラベル付けのための木構造を構築する．そして，それに基づいて各 XML ノードに対してラベル付けを行う．本研究では，ラベル付けされた XML ノードの保存先を便宜的に RDB とするが，ラベル付けそのものはバックエンドのストレージを選ばない．そして，XPath 問合せが与えられた場合，IP-Tree を用いて問合せ中の接尾辞経路に該当する区間を取得し，それを範囲条件として XML ノードを検索する．

本論文の主な貢献は次のとおりである．

- XML 文書から IP-Tree を構築し，接尾辞経路に対して極小なラベル IP-label を生成する手法を提案する（IP-labeling）．
- IP-Tree の構築において，値ノードも統一的に扱うことで VIP-Tree を構築し，等価条件を含む XPath 問合せを効率的に処理可能にする（VIP-labeling）．
- 接尾辞経路に対する極小なラベル付けが，ディスク IO および問合せ時間においてパフォーマンスの改善をもたらすことを示す．

IP-labeling は，連続する Child 軸を含む XPath 問合せ処理の高速化を目的として，接尾辞経路に対してラベル付けを行う．この結果として得られるラベルは，XML 文書に対する索引ととらえることができる．

VIP-labeling は等価条件を含む問合せの高速化を目的としているが，値ノードに関する情報を索引として追加的に保持するかたちをとるため，前方一致検索や後方一致検索，全文検索なども通常どおり行うことが可能である．ただ，評価の際にも示すように，追加的に必要とする索引容量が大きいため，等価条件検索がきわめて重要であるようなアプリケーションに対して用いることを想定している．

本論文の構成は次のとおりである．2 章では，前提事項として，既存のラベル付け手法について述べる．また，関連研究についても触れる．3 章では，提案するラベル付け手法，IP-labeling について述べ，4 章でその拡張について述べる．5 章では，関係データベースをベースに実装したシステムについて述べる．6 章では，実装したシステムを用いて実験を行ったうえで，その評価を行う．最後に 7 章で結論を述べる．

2. 前 提 事 項

本章では提案する次章で IP-labeling について述べるため，まず，XML のデータモデルと問合せに関する定義について述べる．次に，既存のアプローチとして，範囲ラベル付け手法，接尾辞経路に対する区間ラベル付け手法について説明する．また，ラベル付け手法以外の関連研究についても触れる．

2.1 データモデルと問合せ

XML 文書はラベル付きの根付き順序木として表現され，各ノードは要素あるいは値に対応する．また，エッジは要素と子要素あるいは要素と値の関係に対応する．そして，ノードのラベルは，要素名および値の集合からなる．兄弟ノード間の順序関係は木構造を前置順にたどるときの順序により定める．なお，属性は通常ある要素の子要素として表現されるため，本論文では属性と要素を特に区別しない．XML 文書を木構造で表現した例を**図 1** に示す．

本論文では XML 文書に対する問合せとして，主に XPath のサブセットを想定する．すなわち，Child 軸，Descendant 軸および述語（[]）による分岐を含む XPath を想定する．このような XPath 問合せは木構造のパターンで表現されるため，枝分かれ問合せと呼ぶ．また，分岐を含まない経路のみによって表現される問合せを経路問合せと呼ぶ．たとえば，前述の XPath 問合せ `//inproceedings[/author = 'John`

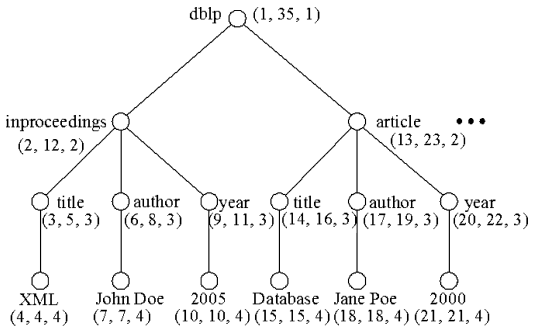


図 1 XML 文書の木構造表現
Fig. 1 Tree representation of XML documnet.

`Doe']/title` は，author ノードの値が John Doe であるような inproceedings ノードの子供ノード title を求める枝分かれ問合せである．

ここで，経路問合せについて定義する．

定義 1（経路問合せ） XML 木 T に対して経路問合せ P を評価すると，XML 木における根ノードから経路 P によって到達可能なノードの集合が得られる．この集合を $[[P]]$ と表す． □

次に，経路問合せの包含関係について定義する．

定義 2（経路問合せの包含関係） 経路問合せ P が経路問合せ Q に含まれるとき，$P \subseteq Q$ と表し，そのときに限り，すべての XML 木について $[[P]] \subseteq [[Q]]$ である．経路問合せ P と経路問合せ Q が非被覆であるとき，$P \cap Q = \emptyset$ と表し，そのときに限り，すべての XML 木について，$[[P]] \cap [[Q]] = \emptyset$ である． □

本論文では，複雑な XPath 問合せを処理するために接尾辞経路（suffix path）[9] と呼ばれる経路に分割して処理を行う．接尾辞経路は次のように定義される．

定義 3（接尾辞経路） 接尾辞経路は 0 個または 1 個の Descendant 軸で始まり，0 個以上の Child 軸が続く経路である．単純経路は Child 軸のみを含む接尾辞経路である． □

たとえば，`//inproceedings/title` は接尾辞経路である．また，`/dblp/inproceedings/title` は接尾辞経路であり，単純経路でもある．さらに，起点経路を次のように定義する．

定義 4（起点経路） XML 木 T におけるノード n の起点経路は $SP(n)$ で表し，根ノードからノード n までの一意な単純経路とする． □

以上の定義より，接尾辞経路問合せ Q を評価することは，$SP(n) \subseteq Q$ なるすべてのノード n を求めることに帰着する．

2.2 範囲ラベル付け手法

XPath の 13 種類の軸のうち，Descendant 軸の処理は問合せ処理の中核を占める．Descendant 軸を効率良く処理するためのラベルとしては，文献 11) や 25) で用いられているような，XML 文書における各ノードの開始タグと終了タグの位置（範囲）および深さの三つ組 (d_1, d_2, d_3) で表現する範囲ラベル付け手法が一般的である．本論文でも，この範囲ラベル付け手法を部分的に利用する．先に示した図 1 は XML 木の各ノードが三つ組でラベル付けされた例である．なお，簡略化のために本論文では単一の XML 文書を想定するが，三つ組の範囲ラベルに文書 ID を導入することで複数文書に対しても簡単に対応することが可能である．

範囲ラベル付け手法を用いた場合，異なる 2 つのノード m と n において n が m の子孫ノードであるとは，次の条件で表される．

$m.d_1 < n.d_1$ かつ $n.d_2 < m.d_2$

また，n が m の子供ノードであるとは，以下と同値である．

n が m の子孫ノードでかつ $m.d_3 + 1 = n.d_3$

一方，n と m が先祖子孫関係を持たないことは，以下と同値である．

$n.d_2 < m.d_1$ または $m.d_2 < n.d_1$

たとえば，図 1 において，$(3, 5, 3)$ のラベルが付与された title ノードと $(1, 35, 1)$ のラベルが付与された dblp ノードについて考える．ここで，$1 < 3$（dblp の開始位置 < title の開始位置）かつ $5 < 35$（title の終了位置 < dblp の終了位置）であるので，title ノードは dblp ノードの子孫である．また，この title ノードと $(2, 12, 2)$ のラベルが付与された inproceedings ノードは，$2 < 3$（inproceedings の開始位置 < title の開始位置）かつ $5 < 12$（title の終了位置 < inproceedings の終了位置）で，さらに $2 + 1 = 3$（inproceedings の深さ $+ 1 =$ title の深さ）であるから，親子関係にある．一方，同じ title ノードでも $(14, 16, 3)$ のラベルが付与された title ノードと inproceedings ノードは 12<14（inproceedings の終了位置 < title の開始位置）であるので，先祖子孫関係は存在しない．

範囲ラベル付けは Descendant 軸を効率良く処理するためのものであるが，XPath 問合せは一般に複数の Child 軸を含む．その場合，それぞれの Child 軸を処理するごとに結合操作を行わなければならない☆．

2.3 接尾辞経路に対する区間ラベル付け手法

連続する Child 軸を効率良く処理するため，接尾辞

☆ ただし，XRel [24]，XParent [12]，Microsoft SQL Server 2005 [16] で用いられているような，経路情報も保持するアプローチにおいてはその限りではない．

経路に対するラベル付け手法が提案されている[9]．各ノードの起点経路に付与された区間ラベルと接尾辞経路問合せに付与される区間ラベルを比較することで経路の包含関係を判定し，結合操作なしに問合せ結果を得ることができる．

定義 5（接尾辞経路に対する区間ラベル） 任意の接尾辞経路 P，Q に対する区間ラベル I_P は次の性質を満たすような区間 $[p_1, p_2)$ で表される．

- $P.p_1 < P.p_2$
- $P \subseteq Q \Leftrightarrow Q.p_1 \leq P.p_1$ かつ $P.p_2 \leq Q.p_2$
- $P \cap Q = \emptyset \Leftrightarrow$
  $P.p_1 \geq Q.p_2$ または $P.p_2 \leq Q.p_1$ □

P-labeling[9] は，接尾辞経路に対するラベル付けの実装の 1 つである．しかしながら，P-labeling には 2 つの問題点がある．

まず，1 つは区間の割当て方法である．P-labeling では XML 文書におけるタグの種類数を n，XML 木の高さを h とすると，用意すべき区間は 0 から $(n+1)^h$ となる．そして，想定されうるすべての接尾辞経路に対してその部分区間を割り当てていく．これは，実際には存在しえない接尾辞経路に対しても区間が予約されてしまうことを意味し，結果として不要に長いラベルを付与していることになる．この問題は XML 木の高さが高いほど顕著になり，ラベルを格納するのに通常の整数型の 2 倍から 3 倍程度の記憶容量を必要とする．予備実験では，こうした属性にアクセスする問合せのパフォーマンスが著しく低下した．

もう 1 つの問題は，値ノードについて考慮していないことである．たとえば，`//author[.='John Doe']` という問合せを考える．author ノードは多数あるけれどもその値が John Doe であるようなノードは少数であるという状況では，解と関係のない多数の author ノードにアクセスすることとなり，非効率である．この例では，値についての 2 次索引を B+木などで構築しておくことにより対応することは可能であるが，該当する値ノードの数やその経路によっては，経路と値を統一的に扱うことでアクセスすべきノードを絞り込むほうが効率が良い場合がある．

こうした従来技術の問題点をふまえ，本論文では接尾辞経路に対する極小な区間ラベル付け手法を提案する．また，経路と値ノードを統一的に扱うことを可能にし，その有用性について検証する．

2.4 その他の関連研究

XML 文書に対する問合せに関する研究は数多くなされている．以下では本研究にも適用しうる結合アルゴリズムに関する研究と，本研究とはまったく異なるアプローチであるシーケンスベースの問合せ処理に関する研究について述べる．

2.4.1 効率的な結合アルゴリズムに関する研究

前置順と後置順もしくは開始位置と終了位置でラベル付けされた XML 要素の親子関係や先祖子孫関係を判定するには 2.2 節で述べたように，構造条件に基づく結合操作が必要である．しかしながら，関係データベースに一般的に実装されている通常の結合アルゴリズムでは効率が悪い．

Zhang らは従来のマージ結合アルゴリズムの改良である “Multi-predicate Merge Join（MPMGJN）” アルゴリズムを提案している[25]．Al-Khalifa らは，2 項間の構造結合を行うための，ディスク IO および CPU に関して最適なアルゴリズムを提案している[2]．

しかしながら，これらの 2 項間の構造結合アルゴリズムは，一般的な問合せの形式である枝分かれ問合せには不向きである．枝分かれ問合せの場合には，結合コストに基づいて結合の順序を考慮したり，選択度，中間結果のサイズなどを推定したりしなければならない．そこで，Bruno らは不要な中間結果を生成せずに枝分かれ問合せを処理可能な構造結合アルゴリズムを提案している[8]．このアルゴリズムは TwigStack と呼ばれ，スタックを用いて解に関係のないノードを結合せずに効率良くスキップする．

本研究では，プロトタイプシステムのバックエンドのストレージとして関係データベースを利用し，問合せエンジンも関係データベースが担っている．よって，結合のアルゴリズムは XML ノードの構造結合に適したものではない．したがって，ここで述べた TwigStack アルゴリズムを用いるなどして，ファイルシステム上にネイティブに問合せエンジンを実装すれば，結合処理のさらなるパフォーマンスの向上が期待できる．

2.4.2 シーケンスベースの問合せ処理に関する研究

XML 文書に対する効率的な問合せ処理手法としては，シーケンスベースのアプローチもあげられる．

Wang らの提案する ViST[23] や Rao らの提案する PRIX[17] では，XML 文書と問合せの双方を，構造情報を保持したシーケンスに変換する．そして，シーケンスの部分一致を求めることで，枝分かれ問合せを分割することなく，かつ，結合操作もせずに処理することができる．ただ，シーケンスにおける一致が必ず問合せにおける一致に結び付かない場合もあり，そうした後処理が煩雑であるという欠点を持つ．

最近では Wang らが問合せとシーケンスの等価性について定式化することで，この等価性のもとではシーケンスマッチング後の後処理などの必要もないことを

示している[22]．さらに，索引付けや問合せにおける時間，空間において最適なシーケンス生成方法についても検証している．

3. IP-labeling

本章では，接尾辞経路に対する極小な区間ラベル付け手法 IP-labeling について述べる．IP-labeling は 2 つの段階からなる．最初の段階ではラベル付けに使用するための木構造 IP-Tree を構築する．次の段階では IP-Tree にラベル付けを行い，そのラベルを用いて接尾辞経路にラベル付けを行う．

ここで，まず，接尾辞経路に対するラベルの極小性について定義する．

定義 6（接尾辞経路に対するラベルの極小性） P, Q を XML 木 T において出現しうる接尾辞経路とし，P, Q のための整数区間をそれぞれ $I_p = [P.p_1, P.p_2)$, $I_q = [Q.p_1, Q.p_2)$ とする．I_p より小さな整数区間が $P \supseteq Q$ なるすべての I_q を含むことができないとき，I_p は極小であるという． □

3.1 IP-Tree の構築

IP-labeling の第 1 段階として，IP-Tree を構築する．IP-Tree（Inverted Path Tree，逆経路木）は IP-labeling を行うための一種の索引木である．まず逆経路を次のように定める．

定義 7（逆経路） XML 木 T におけるノード n の逆経路とは，そのノードから根ノードまでのノード名の列であり，$IP(n) = root.t_l.t_{l-1}.\cdots.t_1$ と表現する．ここで，$root$ は仮想的な根ノードを表す．また，$t_1, \cdots, t_l$ は $SP(n)$ 上の各ノード名を表し，l はノード n の深さを表す． □

IP-Tree の定義は次のとおりである．

定義 8（IP-Tree） XML 木 T における $IP(n)$ の集合を保持するトライ木を IP-Tree とする． □

IP-Tree の構築アルゴリズムを**図 2** に示す．IP-Tree 構築アルゴリズムでは XML 木を入力として，木を深さ優先探索でたどりながら，各ノード n を訪れるたびに，逆経路 $IP(n)$ がトライ木に存在するかを確認する（7 行目）．もし，存在しなければ addInvertedPath 関数を用いて $IP(n)$ をトライ木に追加する（8 行目）．なお，addInvertedPath 関数は追加しようとする経路の一部がすでに存在する場合，その差分の経路のみを追加する．

IP-Tree のサイズは，経路の種類に対してほぼリニアであり，メモリ上に十分保持可能である．

図 3 は，図 1 の XML 文書を用いて構築した IP-Tree である．たとえば，範囲ラベルが $(3, 5, 3)$ である title ノードの逆経路は `root.title.inproceedings.dblp` であり，図 3 の $[2, 3)$ というラベルが付いた dblp ノードに対応する．

Algorithm IP-Tree Construction

input: an XML tree T_{XML}
output: an IP-Tree T_{IP}
01: Stack s
02: Trie T_{IP}
03: $node = T_{XML}.root$
04: Depth-First Search($node$){
05: **if** $node$ is an element **then**
06: push(s, $node.tag$)
07: **if** T_{IP} does not have $IP(n)$ **then**
08: addInvertedPath(s, T_{IP})
09: **end if**
10: Depth-First Search($node$)
11: pop(s)
12: **end if**
13: }

図 2 IP-Tree 構築アルゴリズム
Fig. 2 Construction algorithm of IP-Tree.

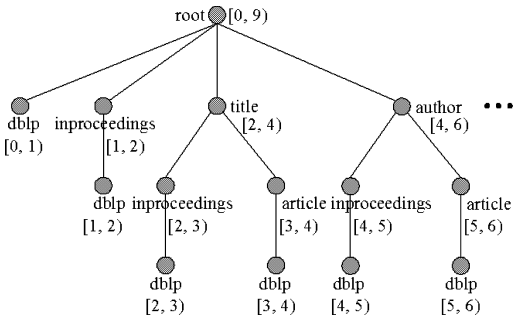


図 3 IP-Tree の例
Fig. 3 Example of IP-Tree.

また，アルゴリズムが範囲ラベルが $(14, 16, 3)$ の title ノードに到達するときには，すでに

```
root.title.inproceedings.dblp
```

という経路が登録されている．よって，この title ノードの逆経路 `root.title.article.dblp` のうち，`article.dblp` の部分のみが新たに追加される．

なお，図 3 の各ノードに付与されたラベルのラベル付け方法とその利用方法に関しては，それぞれ 3.3 節と 3.4 節で述べる．

3.2 IP-Tree の性質

IP-Tree の各ノードから根ノードへの経路は，XML 木 T 中の各ノードの起点経路の最初のノードもしく

は途中のノードから終端までの経路と考えることができるので，次の補題が成立する．

補題 1 IP-Tree の根ノードから葉ノードへの経路は，XML 木に含まれる根ノードから葉ノードへのすべての経路の逆経路を表し，かつそれを転置したものは単純経路と等価である． □

証明 IP-Tree 構築アルゴリズムにより，IP-Tree は XML 木の根ノードから葉ノードへのすべての経路を転置・合成して構築しており，自明． □

たとえば，図 3 における逆経路 `root.title` は接尾辞経路 `//title` と等価である．同様に，逆経路 `root.title.inproceedings` は接尾辞経路 `//inproceedings/title` と等価である．

さらに，IP-Tree の任意のノードとその子ノードの経路に関して以下が成立する．

補題 2 IP-Tree における任意のノードから根ノードへの経路は，その子ノードから根ノードへの経路の接尾辞である． □

証明 IP-Tree における任意のノード m を考える．ノード m から根ノードまでの経路は $t_1, t_2, \cdots, t_l, t_{root}$ と表される．ノード m の子ノード n のタグが t_0 であるとすると，ノード n から根ノードまでの経路は $t_0, t_1, t_2, \cdots, t_l, t_{root}$ となる．よって，IP-Tree における任意のノードから根ノードへの経路は，その子ノードから根ノードへの経路の接尾辞である． □

先の例と同様に，図 3 における title ノードとその子ノードの inproceedings ノードについて考える．title ノードから根ノードへの経路は title,root であり，inproceedings ノードから根ノードへの経路は inproceedings,title,root であるので，title,root は inproceedings,title,root の接尾辞となっていることが分かる．

補題 1 と補題 2 は IP-Tree のノードと接尾辞経路に関する次の定理を導く．

定理 1（IP-Tree のノードと接尾辞経路） 根ノードを除く IP-Tree の異なる 2 つのノード m と n について，これらのノードによって表される接尾辞経路をそれぞれ P_m，P_n としたとき，n が m の子ノードであるならば，$P_n \subseteq P_m$ である． □

証明 補題 1 より，もとの XML 木には IP-Tree の葉ノード i に対応する単純経路 P_i が存在する．ノード i の親ノード j から根ノードへの経路はノード i から根ノードへの経路の接尾辞．よって，ノード j に対応する接尾辞経路 P_j が存在し，$P_i \subseteq P_j$．IP-Tree のすべての葉ノードについて，同様のことが成立し，かつ，補題 2 より，根ノードを除く IP-Tree の異なる 2 つのノード m と n について，これらのノードによって表される接尾辞経路をそれぞれ P_m，P_n としたとき，n が m の子ノードであるならば，$P_n \subseteq P_m$ である． □

この定理は IP-Tree の各ノードどうしの親子関係および先祖子孫関係が接尾辞経路の包含関係に対応することを示している．たとえば，逆経路が `root.title` である title ノードとその子ノード inproceedings（逆経路は `root.inproceedings.title`）との関係は接尾辞経路において，`//inproceedings/title` ⊆ `//title` に対応する．

3.3 IP-Tree へのラベル付け

IP-Tree を構築した後，次の段階として，**図 4** のアルゴリズムに従い，IP-Tree に対して $(start, end)$ の組でラベル付けを行う．

IP-Tree に対するラベル付けアルゴリズムでは，IP-Tree を深さ優先探索しながら，現在位置を行きがけに開始位置として（4 行目），帰りがけに終了位置として付与していく（10 行目）．葉ノードを訪れたときにだけ，現在位置を 1 インクリメントする（6 行目）．図 3 は，このアルゴリズムに従ってラベル付けされた IP-Tree である．また，このアルゴリズムから，次の補題が成り立つ．

補題 3 IP-Tree のあるノード m に付与されたラベルに基づく整数区間を $I_m = [p_1, p_2)$ とすると，IP-Tree の任意の異なる 2 つのノード m と n について以下の性質を満たす．

- $m.p_1 < m.p_2$
  特に m が葉ノードであるならば，

Algorithm IP-Tree Labeling

```
input: an IP-Tree T_IP
output: labeled IP-Tree T_IP
01: node = T_IP.root
02: position = 0
03: Depth-First Search(node){
04:     node.start = position
05:     if node is a leaf then
06:         position++
07:     else
08:         Depth-First Search(node)
09:     end if
10:     node.end = position
11: }
```

図 4 IP-Tree ラベル付けアルゴリズム
Fig. 4 Labeling algorithm for IP-Tree.

$m.p_1 + 1 = m.p_2$

- n が m の子ノードであるならば，
  $m.p_1 \leq n.p_1 < n.p_2 \leq m.p_2$
  特に n が m の最も左の子ノードであるならば，
  $m.p_1 = n.p_1 < m.p_2$
  そうではなく，もし n が m の最も右の子ノードであるならば，$m.p_1 \leq n.p_1 < n.p_2 = m.p_2$
- n が m の直近の following-sibling ノードであるならば，$m.p_2 = n.p_1$ □

証明 IP-Tree 構築アルゴリズムより自明. □

この補題から，IP-Tree に付与される整数区間に関して次の定理が成り立つ．

定理 2（IP-Tree に付与された整数区間の極小性） IP-Tree の任意のノード m に付与された整数区間を $I = [p_1, p_2)$ とすると，I より小さな整数区間は，m のすべての子ノードに付与された整数区間を含むことはできない. □

証明 IP-Tree の任意のノード m に付与された整数区間 I より小さな区間を $I' = [p_1', p_2')$ とすると，

$m.p_1 < p_1'$ または $p_2' < m.p_2$ $\cdots$ (1)

また，m の子ノードを $n_1, n_2, \cdots, n_x$ とする．ここで I' が m のすべての子ノードに付与された区間を含むことができたと仮定すると，

$p_1' \leq n_1.p_1 < n_1.p_2 = n_2.p_1 < n_2.p_2$
$\cdots < n_{x-1}.p_2 = n_x.p_1 < n_x.p_2 \leq p_2'$ $\cdots$ (2)

補題 3 より，

$m.p_1 = n_1.p_1 < n_1.p_2 = n_2.p_1 < n_2.p_2$
$\cdots < n_{x-1}.p_2 = n_x.p_1 < n_x.p_2 = m.p_2$ $\cdots$ (3)

(2)，(3) より，

$p_1' \leq n_1.p_1 = m.p_1$ かつ $m.p_2 = n_x.p_2 \leq p_2'$

すなわち，$m.p_1 \geq p_1'$ かつ $p_2' \geq m.p_2$

これは (1) に矛盾する．よって，仮定は誤りであり，IP-Tree の任意のノード m に付与された整数区間を $I = [p_1, p_2)$ とすると，I より小さな整数区間は，m のすべての子ノードに付与された整数区間を含むことはできない. □

3.4 IP-labeling

定理 1 および定理 2 より，IP-Tree に付与されたラベルは，定義 6 の性質を満たすので，XML 木の各ノードの起点経路，および接尾辞経路問合せに対して極小な区間ラベルとすることができる．この極小な区間ラベルを IP-label と呼び，IP-Tree を用いた極小な接尾辞経路ラベル付け手法を IP-labeling と呼ぶ．XML 木に対して IP-label を付与するアルゴリズムを**図 5** に，接尾辞経路に対して IP-label を付与するアルゴリズムを**図 6** に示す.

Algorithm IP-label

input: an XML tree T_{XML}, an IP-Tree T_{IP}
output: XML tree T_{XML} with IP-label

```
01: Stack s
02: node = T_XML.root
03: Depth-First Search(node){
04:     if node is an element then
05:         push(s, node.tag)
06:         < p1, p2 > = getIPlabel(s, T_IP)
07:         Depth-First Search(node)
08:         pop(s)
09:     end if
10: }
```

図 5 XML 木に対する IP-labeling アルゴリズム
Fig. 5 Algorithm IP-labeling for XML tree.

Algorithm IP-label

input: a suffix path query Q : $\{/|//\}t_1/\cdots/t_n$, an IP-Tree T_{IP}
output: an IP-label

```
01: Stack s
02: for i = 1; i ≤ n; i++
03:     push(s, t_i)
04: end for
05: < p1, p2 > = getIPlabel(s, T_IP)
06: return < p1, p2 >
```

図 6 接尾辞経路問合せに対する IP-labeling アルゴリズム
Fig. 6 Algorithm IP-labeling for suffix path query.

図 5 と図 6 のそれぞれ 6 行目と 5 行目にある getIPlabel 関数は，起点経路あるいは接尾辞経路が格納されたスタックを入力として，その逆経路に該当するノードを IP-Tree 中から検索する．見つかった場合には，そのノードに付与されている区間 $[p_1, p_2)$ を戻り値として返す.

XML 木に対して IP-labeling を行う場合には，XML 木を深さ優先探索でたどる．行きがけにはスタックにノードのタグ名を積み，その時点でのスタックの中身を用いて getIPlabel 関数を実行し，IP-label を得る．また，帰りがけにはスタックからタグ名を取り除く.

XML 木に対する IP-labeling では，IP-Tree の構築という前処理と実際のラベル付けで，2 度 XML 文書を読むことになるが，前処理，本処理ともに入力サイズに対してほぼリニアな時間で処理可能である.

接尾辞経路に対して IP-labeling を行う場合には，

先頭から順にタグ名をスタックに積んでいき，終端に達したら getIPlabel 関数を実行し，IP-label を得る．

そして，接尾辞経路問合せを処理するには，XML 木におけるあるノードの IP-label が問い合わせた接尾辞経路に対する IP-label に含まれるかをチェックすればよい．形式的には次のように表される．Q を接尾辞経路問合せとすると，

$$||Q|| = \{n|Q.p_1 \leq n.iplabel < Q.p_2\}$$

を満たす $||Q||$ が問合せの解である．

ここで，iplabel は IP-label の開始位置を表す．IP-label の開始位置および終了位置の大小関係と，区間どうしが交わることがないという性質から開始区間だけが Q の区間に含まれればよい．

IP-labeling の例 図 3 の IP-Tree を用いて，図 1 の XML 文書に IP-label を付与する場合を例として説明する．まず，XML 文書における dblp ノードは，逆経路が `root.dblp` であるので，IP-Tree を参照して，IP-label は $[0, 1)$ となる．また，inproceedings ノードは，逆経路が `root.dblp.inproceedings` であるので，IP-label は $[1, 2)$ となる．同様に，XML 文書中の他のすべてのノードに対しても IP-label を付与することができる．図 1 の XML 木のすべてのノードに IP-label を付与した結果を**図 7** に示す．

IP-label を用いた接尾辞経路問合せ例 `//title` という接尾辞経路問合せについて考える．この接尾辞経路は逆経路 `root.title` と等価である．この逆経路を IP-Tree で検索することで，$[2, 4)$ という IP-label を得ることができる．したがって，XML 文書の各ノードが関係 *nodes* として保存されており，属性 *iplabel* が IP-label の開始区間を保持しているとすると，$2 \leq nodes.iplabel < 4$ を満たすようなすべてのノード n が問合せの解となる．すなわち，この問合せは次のような SQL を評価すればよい．

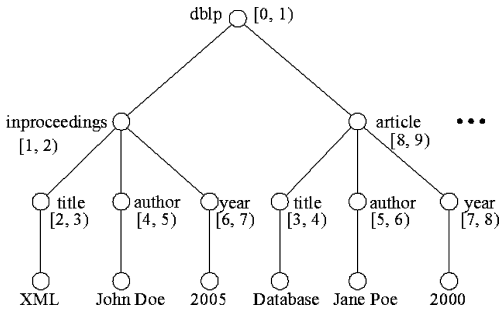


図 7 IP-label を付与した XML 文書
Fig. 7 XML document tree with IP-label.

```
SELECT *
FROM nodes
WHERE nodes.iplabel >= 2
AND nodes.iplabel < 4
```

この SQL を図 7 の XML 文書に対して問い合わせれば，$[2, 3)$ という IP-label が付与された title ノードと $[3, 4)$ という IP-label が付与された title ノードがマッチすることになる．

4. IP-Tree の拡張

3 章では，接尾辞経路に対して極小な区間ラベル付けを行うために IP-Tree を構築した．これを Regular IP-Tree（RIP-Tree）または単に IP-Tree と呼ぶ．これに対し，IP-Tree には用途に応じて他に 2 つの拡張が存在する．1 つは，等価条件を高速に処理することを目的として値ノードを考慮して構築する Value-included IP-Tree（VIP-Tree）である．もう 1 つは，XML 文書の更新について考慮して構築する Updatable IP-Tree（UIP-Tree）である．UIP-Tree は，現在のプロトタイプシステムではサポートしないが，その導入手法を述べる．

4.1 VIP-Tree：値ノードを考慮した IP-Tree

VIP-Tree による接尾辞経路に対するラベル付け VIP-labeling では値ノードを他の要素ノードと統一的に扱うことができる．しかしながら，値ノードに対してもラベル付けを行い，その情報を 1 タプルとして格納するため，追加的に必要とする索引容量が大きくなってしまう．また，前方一致検索や後方一致検索，全文検索などの各種検索も通常どおり行うことができるが，VIP-labeling によって構築される索引は接尾辞経路を含む問合せと等価条件を含む問合せのみを高速化する．以上より，XML 文書の構造と全文検索に関係する問合せを高速化する索引手法[19] も提案されているが，VIP-labeling に関しては，等価条件検索がきわめて重要であるようなアプリケーションに対して用いることを想定する．

XML 文書に対して問合せを行う場合，要素内の値の取扱いは非常に重要である．なぜなら，XPath では述語を評価することでノードの絞り込みを行うことができるため，該当する値を持つようなノードを検索しようとすることは少なくない．

接尾辞経路問合せにおいて，値ノードを取り扱うには値ノードも要素ノードと同様に経路の一部であると考えればよい．つまり，P-labeling や IP-labeling では要素ノードのみを経路の対象としてきたが，それを値ノードまで含めて考えるのである．たとえば，図 1 に

おける “John Doe” という値ノードは，P-labeling や IP-labeling では /dblp/inproceedings/author という経路の要素ノードが持つ値として処理されるが，それを /dblp/inproceedings/author/John Doe という経路の要素ノードとして処理するのである．そうすることで，//inproceedings/author[.='John Doe'] といった等価条件を含む XPath 問合せを //inproceedings/author/John Doe という単純な接尾辞経路問合せとして処理することが可能となる．

VIP-Tree を構築するに際して，注意しなければならないことがある．それは，単純にすべての値ノードを VIP-Tree に加えるべきではないということである．なぜなら，値ノードの種類数が多かった場合，VIP-Tree のサイズが大きくなり，接尾辞経路に対するラベルも長くなってしまうからである．

そこで，図 2 のアルゴリズムに変更を加えて，VIP-Tree を構築する．図 2 のアルゴリズムでは要素ノードのみしか考慮していないが，訪れたノードが値であった場合には，ハッシュ関数 $h(x)$ を用いて値を整数にエンコードする．そして，その整数から根ノードまでのパスを逆経路とし，これをトライ木に格納していく．たとえば，経路が /dblp/inproceedings/author である要素ノードの値 “John Doe” について，$h(\text{'John Doe'}) = 206$ であったとすると，206.author.inproceedings.dblp がこの値ノードの逆経路となる．このようなハッシュ関数の利用により，値の種類数の増加によって VIP-Tree のサイズが肥大化するのを防ぐことができる．

VIP-Tree を用いて VIP-labeling を行う場合，値ノードに対しても接尾辞経路を求め，ラベル付けをする．したがって，値ノードも 1 タプルとして $\{name, start, end, level, iplabel, data\}$ というスキーマを持つテーブルへ格納する．このとき，*name* 属性にはエンコードされたハッシュ値を，*data* 属性には実際の値を格納しておく．また，検索の効率を高めるため，要素ノードのタプルにおける *data* 属性にも値を含む場合には冗長に格納しておく．

等価条件を含む接尾辞経路問合せを処理する場合，まず，VIP-labeling を行った際に用いたものと同じハッシュ関数を用いて値のハッシュ値を求める．そして，そのハッシュ値を含めた接尾辞経路（の逆経路）を VIP-Tree から検索し，VIP-label の区間を得る．たとえば，//inproceedings/author[.='John Doe'] という接尾辞経路問合せを考える．“John Doe” のハッシュ値が 206 であったとすると，206.author.inproceedings がこの問合せの逆経路となる．これを VIP-Tree で検索し，$[20, 21)$ という区間が得られたとすると，次のような SQL を評価すれば求める結果を得られる．

```
SELECT *
FROM nodes
WHERE nodes.viplabel >= 20
AND nodes.viplabel < 21
AND data = 'John Doe'
```

4.2 UIP-Tree：更新を考慮した IP-Tree

XML 文書は半構造データであるために，スキーマが変更されることは一般的にありうることである．IP-Tree は XML 文書に更新が起こり，かつ，新しい逆経路が出現したときに，更新する必要が出てくる．しかしながら，極小な区間で接尾辞経路に対しラベル付けを行っていた場合には，もはや新しい逆経路を追加できる数値区間がなく，再度 IP-Tree に対してラベル付けを行わなければならない．IP-Tree に対するラベルを変更した場合には，XML 文書の各ノードに付与された IP-label もすべて更新しなければならないため，効率が悪い．

そこで，IP-Tree の各ノードに区間をラベル付けする際に，あらかじめ区間を大きく確保しておくことで，新しい逆経路の追加に対応することが可能である．これは，接尾辞経路に対する割当て区間の極小化とは相反することであるが，更新が想定される環境においては，トレードオフを考慮したうえでのラベルサイズの増大はやむをえないものと思われる．

更新に対応した UIP-Tree を構築するには，通常の IP-Tree を構築した後，図 4 のアルゴリズムによって区間をラベル付けする際に，必要な区間を考慮したうえで，6 行目の position の増分を 1 以上にすればよい．

ただ，この手法では，更新が一部の箇所に集中したりすると，用意していた区間を消費しつくしてしまう．このような問題をさけるため，ビット列による接頭辞ラベル[10]，浮動小数点を用いたラベル[4]，ノードを無制限に挿入可能な VLEI コード[13] といった更新に強いラベル付け手法を IP-Tree のラベル付けに応用することも考えられる．

5. 実　　装

3 章および 4 章で説明した手法をもとに，XPath を処理するプロトタイプシステムを実装した．本章ではその実装について述べる．

まず，実装したプロトタイプシステムを**図 8** に示す．

プロトタイプシステムは大きくわけて 2 つのモジュールで構成される．1 つは索引生成部であり，XML 文書を入力として IP-Tree の構築および IP-label の生

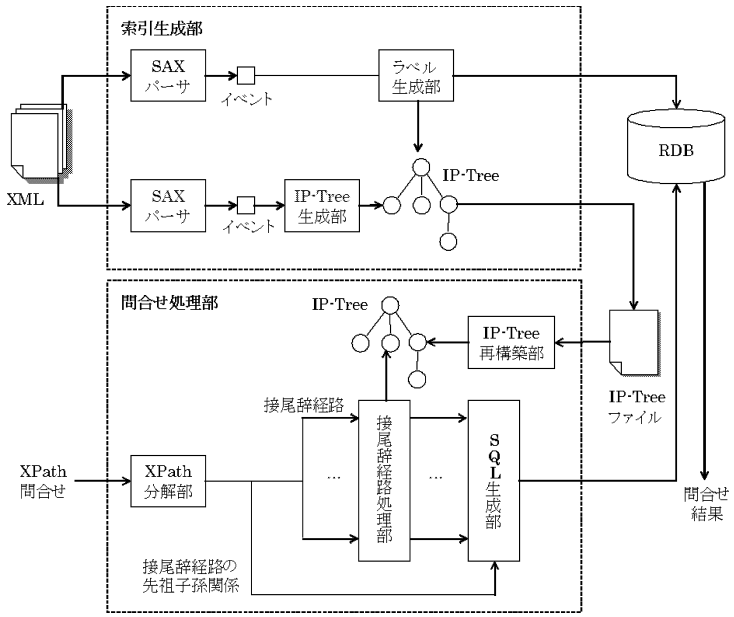


図 8 プロトタイプ XPath 処理システム
Fig. 8 Prototype XPath processing system.

成を行う．もう 1 つは問合せ処理部であり，XPath 問合せを入力として IP-label を含んだ SQL を生成し，関係データベースに問合せを行う．

索引生成部では，まず，XML 文書が SAX パーサを通じて読み込まれる．そして，発生した各イベントが IP-Tree 生成部にわたり，IP-Tree が構築され，同時にラベル付けも行われる．なお，ここで構築する IP-Tree の種類には，4 章で述べたように 3 つの選択肢がある．構築された IP-Tree はその時点から問合せ処理に利用できるようになるが，XML 形式で外部ファイルにも書き出される．効率性のため，問合せの際，IP-Tree はメモリ上に保持しておく．一度システムを終了し，再び起動したときに利用するためには，IP-Tree ファイルを読み込む必要がある．図 3 の IP-Tree を外部ファイルとして書き出した例を**図 9** に示す．ここで，ipt 要素は IP-Tree の根ノードを示し，各要素の start および end はそれぞれ IP-label の開始区間，終了区間を示す．

続いて，再度 SAX パーサを用いて XML 文書が読み込まれ，発生した各イベントはラベル生成部へ渡される．ここで，XML タグの開始位置および終了位置による範囲ラベルと IP-label の両方が付与され，バックエンドのストレージに格納される．実装したプロトタイプシステムでは関係データベースを用いているが，IP-labeling はバックエンドのストレージを選ばない．ファイルシステム上に実装することも可能である．それにもかかわらず，関係データベースを選択したのは，問合せ処理，索引，データの信頼性など様々な点においてすでに技術が確立されているからである．関係のスキーマは $\{name, start, end, level, iplabel, data\}$ である．ここで，$name$ は要素もしくは属性の名前である．$start$，end はその要素の開始タグの位置および終了タグの位置である．$iplabel$ は IP-Tree を用いて得られた IP-label の開始位置である．開始位置しか保持しないのは，3.4 節で述べたように開始位置だけであっても，IP-label の区間どうしの比較を行うことができ，不要な記憶領域を割かないようにするためである．また，$level$ はその要素のレベルを示し，$data$ はその要素の値を示している．値がなければ null 値が格納される．また，検索の効率を高めるために，属性の組 $\{iplabel, start\}$ および $start$ と $data$ に対しては B+木を構築する．さらに，$\{iplabel, start\}$ の B+木でクラスタ化する．

問合せ処理部では，最初に IP-Tree ファイルを読み込み，IP-Tree 再構築部でメモリ上に IP-Tree を展開する．その後，XPath 問合せが与えられると，XPath 分解部に渡され，複数の接尾辞経路問合せに分解される．そして，各接尾辞経路問合せは IP-Tree を用いて IP-label を取得し，それに基づいて 1 つの SQL を生成する．最後に，SQL 生成部が接尾辞経路問合せ間の先祖子孫関係に基づいて複数の SQL を統合する．統合の際には Chen らの提案する PushUp アルゴリ

```
<?xml version="1.0" standalone="yes"?>
<ipt start="0" end="9">
  <dblp start="0" end="1"/>
  <inproceedings start="1" end="2">
      <dblp start="1" end="2"/>
  </inproceedings>
  <title start="2" end="4">
      <inproceedings start="2" end="3">
          <dblp start="2" end="3"/>
      </inproceedings>
      <article start="3" end="4">
          <dblp start="3" end="4"/>
      </article>
  </title>
  <author start="4" end="6">
      <inproceedings start="4" end="5">
          <dblp start="4" end="5"/>
      </inproceedings>
      <article start="5" end="6">
          <dblp start="5" end="6"/>
      </article>
  </author>
  ...
</ipt>
```

図 9 IP-Tree ファイルの例
Fig. 9 Example of IP-Tree file.

ズム[9] を利用する．そして，この SQL が関係データベースによって処理され，問合せ結果が返る．

6. 実験と評価

本章では，提案する IP-labeling の有用性を示すため，従来の P-labeling との比較実験を行う．実験には，関係データベース上に実装したプロトタイプシステムを用いる．また，対象は，スキーマが異なり，それぞれ違った特徴を持つ 3 種類のデータセットである．パフォーマンスを測る際には，XPath 問合せとして，単純な接尾辞経路問合せ，等価条件を含む接尾辞経路問合せ 2 種類，枝分かれ問合せの計 4 種類を用いる．実験では，IP-labeling および VIP-labeling の違いが与える影響について検証も行う．

6.1 実験環境

実験は CPU：Pentium III 1.4 GHz Dual，メモリ：2 GB，OS：Linux（kernel 2.4.18）のマシンで行い，関係データベースには PostgreSQL 8.1.1 を利用した．

表 1 XML 文書の概要
Table 1 Summary of XML dataset.

	Auction	DBLP	SwissProt
サイズ	111 MB	268 MB	109 MB
ノード数	2,048,180	7,926,463	5,166,890
タグ種類数	77	41	99
値種類数	405,559	2,715,303	677,412
最大深度	12	6	5

6.1.1 データセット

実験に用いるデータセットとして，Auction[18]，DBLP[14]，SwissProt[20] の 3 種類を用いた．それぞれのデータの概要は次のとおりである．

Auction オークションの情報が記述されているデータセットである．XMark ベンチマークプロジェクトが提供するツールを用いて人工的に生成される．データセットの生成は同じく提供される DTD に基づいて行われる．DTD は再帰的であり，インスタンスは比較的深さのあるデータセットとなる．

DBLP DBLP はコンピュータサイエンスの分野における論文誌や会報の書誌情報を XML 形式で表現したデータセットである．タグの種類数に対して値の種類数が多く，木の高さは比較的低い．

SwissProt SwissProt はタンパク質の情報を XML で表現したデータセットである．分岐が多い木構造で，高さは比較的低い．

これらのデータセットの特徴を**表 1** に示す．表 1 におけるサイズとはオリジナルの XML 文書ファイルのサイズである．ノード数は属性ノードの数も含めている．最大深度は XML 文書における最も長い経路の長さである．

6.1.2 関係データベースの構成

P-labeling，IP-labeling，VIP-labeling の 3 種類のラベル付け手法に基づいて，各データセットに対しそれぞれ 3 つのテーブルを用意した．これらをそれぞれ P，IP，VIP とする．スキーマはいずれも $\{name, start, end, level, plabel, data\}$ とした．ただし，属性 $plabel$ には，テーブル P，IP，VIP の場合，それぞれ，P-label，IP-label，VIP-label が格納される．

また，検索の効率を高めるために，各テーブルにおける属性の組 $\{plabel, start\}$ および属性 $start$ に対して B+木を構築した．さらに，テーブル P，IP については，属性 $data$ に対しても B+木を構築した．そして，いずれのテーブルも最後に $\{iplabel, start\}$ の B+木でクラスタ化した．

なお，VIP-labeling を行う際のハッシュテーブルの

表 2 属性 *plabel* の最大値とデータ型
Table 2 The data type and maximum value of the attribute *plabel*.

		P	IP	VIP
Auction	最大値	50,697,762,500,009,282,441,473	547	86,925
	データ型	NUMERIC(23, 0)	INTEGER	INTEGER
DBLP	最大値	5,454,877,177	158	21,854
	データ型	BIGINT	INTEGER	INTEGER
SwissProt	最大値	9,930,020,101	263	20,613
	データ型	BIGINT	INTEGER	INTEGER

サイズは各データセットとも 467 とした．ハッシュ値がなるべく均等に散らばるよう素数を用いているが，そのサイズが与える影響に関しては検証していない．

6.2 データサイズの比較

6.2.1 ラベルサイズの比較

P, IP, VIP の各テーブルの属性 *plabel* のデータ型とそこに格納されている値の最大値を**表 2** に示す．データ型のうち，NUMERIC は任意の精度で数値を格納可能なデータ型のことであり，(23, 0) という指定により，23 桁の整数として定義している．INTEGER は通常使用される 32 ビットの整数であり，BIGINT は広範囲にわたる数値データを格納するために使用される 64 ビットの整数である．

表 2 から分かるように，テーブル P ではいずれのデータセットにおいても，接尾辞経路のための区間が最も大きく，テーブル IP, VIP の場合より大きなデータ型を使用している．P-labeling においてこのようにデータ型の違いが出る理由は，P-labeling における区間の割当て方法には無駄があり，ラベルサイズが大きくなってしまうからである．XML 木が 3 種類のデータセットのうち最も深い Auction のデータセットにおいてはそれが顕著に現れ，あらかじめ用意されている固定長のデータ型では対応できなかった．また，VIP-label の最大値が，IP-label よりも大きいのは，値ノードを含めた接尾辞経路に対してラベル付けを行った結果である．

6.2.2 テーブル・索引サイズの比較

各データセットにおける P, IP, VIP の 3 つのテーブルおよびそれに付随する B+木索引のサイズを測定した．その結果を**表 3** に示す．なお，A，D，S はデータセットの頭文字を示し，単位は MB である．

まず，VIP テーブルのサイズが他の 2 つのテーブルに比べて非常に大きくなっているのが分かる．これは，VIP-labeling では値ノードに対しても接尾辞経路にラベル付けを行い，1 つのタプルとしてテーブルに格納するためである．また，VIP テーブルの *data* 属性に対する B+木索引のサイズが N/A となっているのは，VIP テーブルでは等価条件を含む問合せであっても接尾辞経路として処理可能であるため，*data* 属性には B+木を付与していないためである．

表 3 テーブル・索引サイズの比較
Table 3 Size of table and index.

		P	IP	VIP
A	テーブル	240	177	357
	B+木 $\{plabel, start\}$	80	44	77
	B+木 $\{start\}$	35	35	62
	B+木 $\{data\}$	89	89	N/A
	テーブル・索引合計	444	345	496
D	テーブル	725	663	1,304
	B+木 $\{plabel, start\}$	204	170	338
	B+木 $\{start\}$	136	136	270
	B+木 $\{data\}$	306	306	N/A
	テーブル・索引合計	1,371	1,275	1,912
S	テーブル	414	374	680
	B+木 $\{plabel, start\}$	133	111	201
	B+木 $\{start\}$	88	88	160
	B+木 $\{data\}$	141	141	N/A
	テーブル・索引合計	777	715	1,041

次に，P テーブルと IP テーブルの索引を含めた合計サイズを比較すると，Auction で約 20%強，DBLP，SwissProt で約 10%弱，IP テーブルの方がサイズが小さくなっているのが分かる．P テーブルと IP テーブルで異なるのは *plabel* 属性のみであるから，これは接尾辞経路に対する極小な区間ラベル付けの効果であるととらえることができる．また，テーブルサイズの違いは *plabel* 属性に使用するデータ型が影響していると思われ，Auction データのように XML 木が深い場合において特に差が出ることが分かる．

6.3 問合せにおけるパフォーマンスの比較

6.3.1 問合せの準備

問合せにおけるパフォーマンスを比較するために，各データセットに対して，それぞれ 4 つのタイプの XPath 問合せを用意した．1 つ目は接尾辞経路問合せだけで構成されて等価条件を含まないもの，2 つ目は接尾辞経路問合せだけで構成されて等価条件を含むもの，3 つ目は接尾辞経路問合せだけで構成されて等価条件を含む問合せであるが，等価条件の値としてタイプ 2 よりも選択度の低い値を指定したもの，そして，4 つ目は枝分かれ問合せである．

表 4 XPath 問合せ
Table 4 XPath queries.

No.	XPath
QA1	/site/regions/asia/item/name
QA2	/site/regions/asia/item/location[.='Thailand']
QA3	/site/regions/asia/item/payment[.='Cash']
QA4	/site/regions/asia/item[/location ='Thailand']/name
QD1	//inproceedings/tilte
QD2	//author[.='Yoko Maeda']
QD3	/dblp/article/author[.='Serge Abiteboul']
QD4	//inproceedings[/author='Yoko Maeda']/title
QS1	/root/Entry/Keyword
QS2	//Entry/Ref/Author[.='Mueller P']
QS3	/root/Entry/Features/DOMAIN/to[.='13']
QS4	//Entry/Ref[./Author ='Mueller P'][./Author='Keller M']

これらのタイプの問合せを用いた理由は以下のとおりである．まず，1 つ目の問合せは，接尾辞経路問合せにおいて P-labeling と IP-labeling のパフォーマンスを比較するためのものである．2 つ目および 3 つ目の問合せは，等価条件を含む問合せにおいても提案手法が有効であるか検証し，なおかつ，VIP-labeling で値ノードを統一的に扱うことの意義を検証するためのものである．4 つ目は実用上一般的であると考えられる枝分かれ問合せについて，パフォーマンスを比較するためのものである．

用いる問合せのリストを**表 4** に示す．問合せに付与されている “QXY” という名前は，“X” が “A”（Auction），“D”（DBLP），“S”（SwissProt）のうちのいずれかを表す．また，“Y” は問合せタイプの番号であり，“1”（接尾辞経路問合せ），“2”（選択度の高い等価条件付き接尾辞経路問合せ），“3”（選択度の低い等価条件付き接尾辞経路問合せ），“4”（枝分かれ問合せ）のうちのいずれかである．

タイプ 2 およびタイプ 3 の問合せの選択度の違いについては**表 5** に示す．表 5 では，その問合せの等価条件のみに該当するタプルの数と接尾辞経路まで含めた実際の解の数を記している．たとえば，QA2 の場合，“Thailand” に該当するタプル数は 30 であるが，そのうち `/site/regions/asia/item/location[ .= 'Thailand']` という問合せにマッチするタプル数は 2 である．

なお，問合せ処理時間に関しては，キャッシュの影響が出ないよう，つねにデータベースを起動した直後の状態で測定を行った．また，問合せはいずれも 10 回繰り返して行い，最大値，最小値をはずした残りの値から処理時間の平均値を算出した．結果出力にかかる時間，すなわち，関係形式の問合せ結果から XML 文書を生成するため時間は，P-labeling，IP-labeling，VIP-labeling ともに共通であるので，ここでは考慮しないこととする．

表 5 タイプ 2 とタイプ 3 の問合せの選択度
Table 5 Selectivity of type 2 query and type 3 query.

	等価条件の該当タプル	問合せの該当タプル
QA2	30	2
QA3	1,455	40
QD2	1	1
QD3	176	52
QS2	11	11
QS3	3,706	65

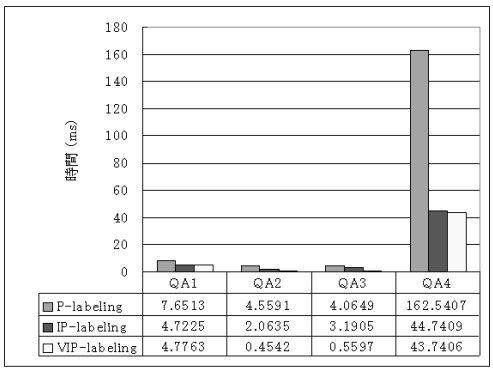


図 10 Auction データに対する問合せ処理時間
Fig. 10 Execution time for Auction dataset.

6.3.2 パフォーマンスの比較

6.3.1 項の条件下で問合せ処理の時間を測定し，P-labeling，IP-labeling，VIP-labeling の各手法別に比較した．結果のうち，Auction データに関するものを**図 10** に，DBLP データに関するものを**図 11** に，SwissProt データに関するものを**図 12** に示す．

まず，タイプ 1 の問合せについて見てみる．Auction のデータセットで，P-labeling を用いた場合よりも IP-labeling と VIP-labeling を用いた場合のパフォーマンスが良くなっているが，DBLP および SwissProt

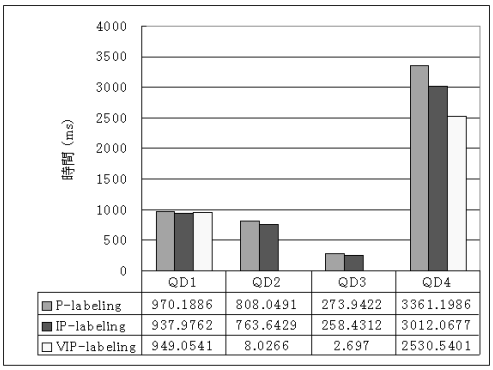


図 11 DBLP データに対する問合せ処理時間
Fig. 11 Execution time for DBLP dataset.

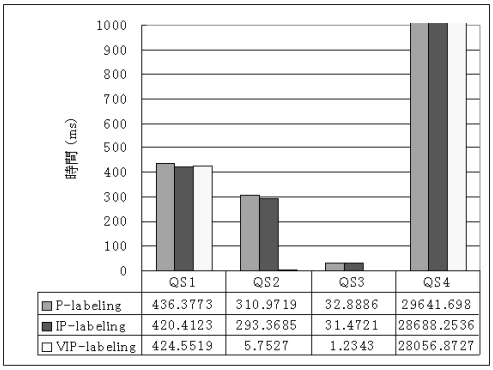


図 12 SwissProt データに対する問合せ処理時間
Fig. 12 Execution time for SwissProt dataset.

表 6 タイプ 2 の問合せのディスク IO の比較
Table 6 Disk IO of type 2 query.

		P	IP	VIP
A	テーブル	2	2	1
	B+木 $\{plabel, start\}$	14	10	4
	B+木 $\{data\}$	6	6	N/A
	テーブル・索引合計	22	18	5
D	テーブル	1	1	38
	B+木 $\{plabel, start\}$	4,870	4,059	15
	B+木 $\{data\}$	5	5	N/A
	テーブル・索引合計	4,876	4,065	53
S	テーブル	8	10	26
	B+木 $\{plabel, start\}$	1,861	1,552	11
	B+木 $\{data\}$	4	4	N/A
	テーブル・索引合計	1,873	1,566	37

表 7 タイプ 3 の問合せのディスク IO の比較
Table 7 Disk IO of type 3 query.

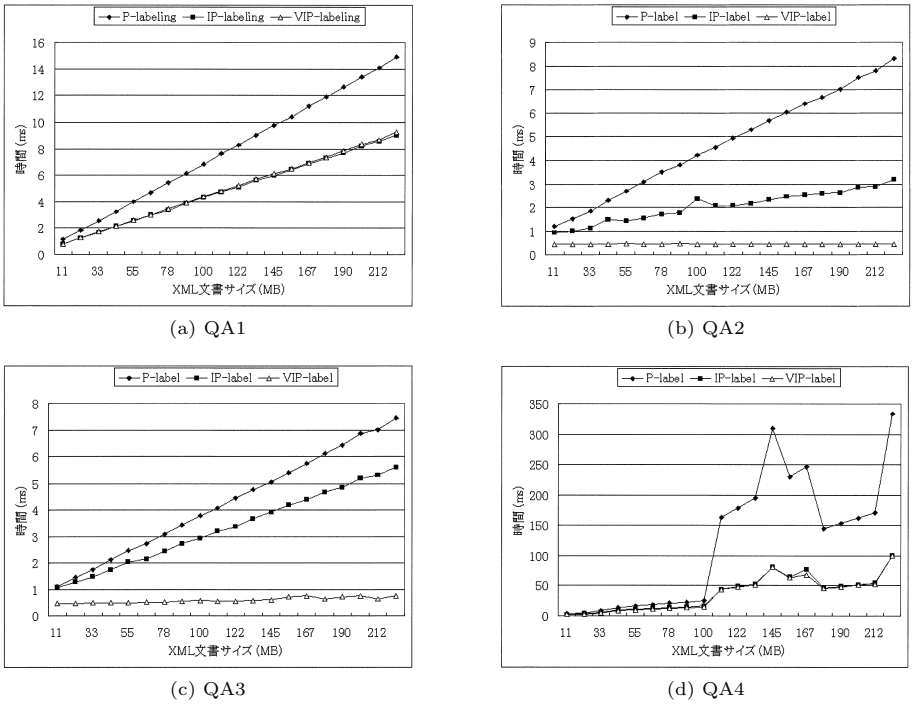
		P	IP	VIP
A	テーブル	9	7	1
	B+木 $\{plabel, start\}$	6	6	4
	B+木 $\{data\}$	10	10	N/A
	テーブル・索引合計	25	23	5
D	テーブル	34	34	11
	B+木 $\{plabel, start\}$	1,636	1,634	7
	B+木 $\{data\}$	6	6	N/A
	テーブル・索引合計	1,676	1,674	18
S	テーブル	37	34	3
	B+木 $\{plabel, start\}$	38	38	5
	B+木 $\{data\}$	14	14	N/A
	テーブル・索引合計	89	86	8

のデータセットではほんのわずかしか改善が見られないことが分かる．これは，Auction データの P テーブルのディスクページあたりのタプル数が影響していると考えられる．Auction データの P テーブルにおける *plabel* 属性のデータ型は NUMERIC(23,0) であり，IP，VIP テーブルの INTEGER 型とはページあたりの記憶効率に大きな違いがある．一方，DBLP，SwissProt のデータの場合の *plabel* 属性は，P テーブルが BIGINT 型，IP，VIP テーブルが INTEGER 型である．これら 2 つのデータ型では，タイプ 1 の問合せの処理の場合，パフォーマンスに大きな影響を与えない．

次に，タイプ 2 およびタイプ 3 の問合せについて見てみる．ここでは，明らかに VIP-labeling を用いた場合のパフォーマンスが非常に優れていることが分かる．P-labeling と IP-labeling の違いに関しては，タイプ 1 の問合せの場合と同様の傾向が見られる．P，IP テーブルの *data* 属性には B+木が構築されているにもかかわらず，VIP-labeling とのパフォーマンスの差が大きいので，問合せを実行したときの実行プランとディスク IO を調べた．ディスク IO についてはタイプ 2 およびタイプ 3 の問合せについて，それぞれ**表 6** と**表 7** に示す．なお，A，D，S はデータセットの頭文字を表し，単位はページ数である．

ディスク IO の測定の結果，タイプ 2 とタイプ 3 のいずれにおいても P-labeling，IP-labeling では属性の組 $\{plabel, start\}$ に構築した B+木へのアクセスが多いことが分かった．そこで，問合せの実行プランを参照すると，PostgreSQL のオプティマイザは，BitmapAnd と呼ばれる問合せ演算子を使用していた．これは，*start* と *data* に複数列索引が存在しないが，それぞれにおいては索引が存在する場合，両方の索引を検索し，結果をメモリ内で組み合わせ，*start* と *data* に関する両方の条件に一致する行のみをヒープから取り出すという操作を行う．しかしながら，*data* と $\{plabel, start\}$ で極端に *data* の方が選択度が高い場合，このようなプランは不適切である．

そこで，表 6 と表 7 のディスク IO のうち，P-labeling，IP-labeling における $\{plabel, start\}$ へのディスク IO を無視したとする．この場合，QD2 と



(a) QA1 (b) QA2

(c) QA3 (d) QA4

図 13 XML 文書サイズを変化させたときの問合せ処理時間
Fig. 13 Execution time for various size XML dataset.

表 8 問合せパフォーマンスの比較
Table 8 Comparison of query performance.

	P	*IP*	*VIP*
データサイズ	○	◎	△
標準的な深さの XML 文書への問合せ	○	○	○
比較的深い XML 文書への問合せ	○	◎	◎
等価条件付き問合せ（選択度高）	○	○	○
等価条件付き問合せ（選択度低）	○	○	◎
枝分かれ問合せ	○	◎	◎

QS2 において，接尾辞経路で値ノードを統一的に扱わない P-labeling と IP-labeling のパフォーマンスが若干良く，それ以外においては VIP-labeling のパフォーマンスが良いということになる．この結果を表 5 と照らし合わせると，等価条件で一致しても解と関係のない複数の経路が存在する場合には，接尾辞経路と値ノードを統一的に扱うことができる VIP-labeling が有効であることが分かる．

最後に，タイプ 4 の問合せについて見てみる．タイプ 1 の問合せの場合と同様に Auction データセットにおいて，P-labeling に比べて提案手法のパフォーマンスが良いことが示された．DBLP データセット，SwissProt データセットにおいても若干パフォーマンスの向上が見られる．

以上の比較をまとめたものを**表 8** に示す．◎，○，△はそれぞれ，P-labeling を○とした場合の相対的な評価である．

6.3.3 スケーラビリティ

P-labeling，IP-labeling，VIP-labeling のスケーラビリティを調べるため，Auction データセットを生成するツールのスケールファクタを 0.1 から 2.0 まで 0.1 ごとに変化させて計 20 個のデータセットを用意した．なお，表 1 の Auction データセットはこのスケールファクタを 1.0 として生成したものである．

6.3.1 項で述べた条件のもとで，生成したデータセットそれぞれに対して，QA1，QA2，QA3，QA4 の 4 つの問合せを行い，問合せ処理時間を測定した．それぞれの問合せの，XML 文書サイズごとの処理時間を**図 13** に示す．

QA1 の問合せでは，IP-labeling と VIP-labeling がほぼ同じパフォーマンスを示していることが分かる．これは，問合せに等価条件を含んでいないため同じ問合せ実行プランで処理が行われたことに加えて，*P* テーブルと *IP* テーブルの *plabel* 属性のデータ型が同じであり，ディスク IO が等しくなったためであると考えられる．一方，P-labeling では，*plabel* 属性のデータ型として 23 桁の NUMERIC 型を使用しており，ディスク IO が増大してしまうため，つねに

IP-labeling，VIP-labeling 以上の問合せ処理時間がかかっている．また，XML 文書のサイズが大きくなるにつれて，P-labeling の問合せ処理時間と IP-labeling，VIP-labeling の問合せ処理時間の差が拡大している．これは，提案手法が P-labeling に比べて高いスケーラビリティを持つことを示している．

QA2 の問合せでも IP-labeling，VIP-labeling が P-labeling に比べて高いスケーラビリティを持つことを示している．ただ，6.3.2 項で述べたとおり，P-labeling と IP-labeling におけるこの問合せの実行プランには問題がある．したがって，適切に実行プランが決定された場合は，P-labeling と IP-labeling の問合せ処理時間と VIP-labeling の問合せ処理時間の差はもっと小さくなると思われる．しかしながら，少なくとも VIP-labeling の処理時間はほぼ一定に保たれており，絶対的に見て十分なスケーラビリティがあるといえる．

QA3 の問合せにおいても QA2 の問合せの場合と同じような傾向が見られるが，P-labeling と IP-labeling の問合せ処理時間の差が小さいにもかかわらず，VIP-labeling の問合せ処理時間は XML 文書サイズが大きくなってもほぼ一定である．これは，値ノードに関しても接尾辞経路と統一的に扱うことで，解と関係のない経路のノードにアクセスすることなく効率良く目的の解を得ることができるようになったためであると考えられる．

QA4 の問合せでは，XML 文書サイズが 100 MB に達するまでは微小ではあるが，P-labeling と提案手法との問合せ処理時間の差が拡大していることが確認される．XML 文書サイズが 100 MB を超えた後は，変動が激しいためスケーラビリティに関しては一概にはいえないが，少なくとも IP-labeling と VIP-labeling の方が良いパフォーマンスを維持していることが分かる．

7. ま と め

本論文では XML 文書における接尾辞経路に対する新しいラベル付け手法 IP-labeling を提案した．提案手法では，XML 文書から IP-Tree と呼ばれるラベル付けのための木構造を構築する．IP-Tree は対象とする XML 文書に出現しうるすべての接尾辞経路（逆経路と等価）を保持する．これらに対して極小な区間を割り当てることでサイズの小さな接尾辞経路ラベルを生成することを可能とした．

また，XML 文書における値ノードの取扱いに着目し，接尾辞経路と値ノードを統一的に扱うことができる VIP-labeling を提案した．VIP-labeling では，値をハッシュ関数を用いてエンコードすることで，値の種類数が膨大な場合でも，接尾辞経路のための区間および IP-Tree のサイズを小さく抑えることを可能とした．

そして，提案手法に基づいて，XPath のサブセットを処理することができるプロトタイプシステムを実装し，それを用いて実験を行った．実験は，データサイズと問合せ処理時間の 2 つの観点から行った．

実験の結果，P-labeling，IP-labeling，VIP-labeling のうち，データサイズに関しては IP-labeling が最も優れていることが示された．特に，深さのある XML 文書に対してその差が顕著に現れる．一方，VIP-labeling は値ノードをタプルとして冗長に保持しているため，データサイズでは P-labeling に劣る．

問合せ処理時間に関しては，等価条件を含まない接尾辞経路問合せの場合，IP-labeling と VIP-labeling のパフォーマンスが良いことを示した．ただし，データサイズの場合と同様，深さのある XML 文書に対して特に良い結果が得られた．等価条件を含む接尾辞経路問合せの場合，該当する値や経路の選択度が高い場合には，通常の IP-labeling と値に対する B+木が有効であるが，解と関係のない経路が多数ある場合には，値まで含めた接尾辞経路問合せの処理，すなわち VIP-labeling が有効である．枝分かれ問合せに関しても，深さのある XML 文書を中心に IP-labeling，VIP-labeling が有効であることを示した．

Auction データセットについては，異なるサイズの XML 文書に対しても同様の問合せを行い，IP-labeling と VIP-labeling が P-labeling に比べて高いスケーラビリティを有していることを示した．

以上を総括して，提案する IP-labeling は従来の P-labeling と比べ，データサイズ，問合せ処理時間いずれにおいても優れている．特に，深さのある XML 文書においてより有用である．データサイズが大きくなるというトレードオフを受け入れるならば，等価条件を含む問合せを効率良く処理することができる VIP-labeling の利用も有用である．今後は，値ノードを接尾辞経路と統一的に扱うことができる VIP-labeling を用いて，XQuery における値ベースの結合を効率良く行う手法について検討していきたい．

参 考 文 献

1) Abiteboul, S., Kaplan, H. and Milo, T.: Compact labeling schemes for ancestor queries, *Proc. SODA*, pp.547–556 (2001).
2) Al-Khalifa, S., Jagadish, H.V., Patel, J.M.,

Wu, Y., Koudas, N. and Srivastava, D.: Structural Joins: A Primitive for Efficient XML Query Pattern Matching, *Proc. IEEE ICDE*, pp.141–152 (2002).
3) Alstrup, S. and Rauhe, T.: Improved labeling scheme for ancestor queries, *Proc. SODA*, pp.947–953 (2002).
4) Amagasa, T., Yoshikawa, M. and Uemura, S.: QRS: A Robust Numbering Scheme for XML Documents, *Proc. IEEE ICDE*, pp.705–707 (2003).
5) Berglund, A., Boag, S., Chamberlin, D., Fernández, M.F., Kay, M., Robie, J. and Siméon, J.: XML Path Language (XPath) 2.0, Technical report, World Wide Web Consortium (2005).
6) Boag, S., Chamberlin, D., Fernández, M.F., Florescu, D., Robie, J. and Siméon, J.: XQuery 1.0: An XML Query Language, Technical report, World Wide Web Consortium (2005).
7) Bray, T., Paoli, J., Sperberg-McQueen, C.M., Maler, E. and Yergeau, F.: Extensible Markup Language (XML) 1.0 (3rd Edition), Technical report, World Wide Web Consortium (2004).
8) Bruno, N., Koudas, N. and Srivastava, D.: Holistic twig joins: optimal XML pattern matching, *Proc. ACM SIGMOD*, pp.310–321 (2002).
9) Chen, Y., Davidson, S.B. and Zheng, Y.: BLAS: An Efficient XPath Processing System, *Proc. ACM SIGMOD*, pp.47–58 (2004).
10) Cohen, E., Kaplan, H. and Milo, T.: Labeling Dynamic XML Trees, *Proc. PODS*, pp.271–281 (2002).
11) DeHaan, D., Toman, D., Consens, M.P. and Özsu, M.T.: A Comprehensive XQuery to SQL Translation using Dynamic Interval Encoding, *Proc. ACM SIGMOD*, pp.623–634 (2003).
12) Jiang, H., Lu, H., Wang, W. and Yu, J.X.: XParent: An Efficient RDBMS-Based XML Database System, *Proc. IEEE ICDE*, pp.335–336 (2002).
13) Kobayashi, K., Liang, W., Kobayashi, D., Watanabe, A. and Yokota, H.: VLEI code: An Efficient Labeling Method for Handling XML Documents in an RDB, *Proc. IEEE ICDE*, pp.386–387 (2005).
14) Ley, M.: DBLP Computer Science Bibliography. http://dblp.uni-trier.de/
15) O'Neil, P.E., O'Neil, E.J., Pal, S., Cseri, I., Schaller, G. and Westbury, N.: ORDPATHs: Insert-Friendly XML Node Labels, *Proc. ACM SIGMOD*, pp.903–908 (2004).
16) Pal, S., Cseri, I., Schaller, G., Seeliger, O., Giakoumakis, L. and Zolotov, V.V.: Indexing XML Data Stored in a Relational Database, *Proc. VLDB*, pp.1134–1145 (2004).
17) Rao, P. and Moon, B.: PRIX: Indexing And Querying XML Using Prüfer Sequences, *Proc. IEEE ICDE*, pp.288–300 (2004).
18) Schmidt, A., Waas, F., Kersten, M., Florescu, D., Manolescu, I., Carey, M.J. and Busse, R.: The XML Benchmark Project, Technical report, Centrum voor Wiskunde en Informatica (2001).
19) Shimizu, T. and Yoshikawa, M.: Full-Text and Structural XML Indexing on B^+-Tree, *Proc. DEXA*, pp.451–460 (2005).
20) Swiss Institute of Bioinformatics: Swiss-Prot Protein knowledgebase. http://us.expasy.org/sprot/
21) Tatarinov, I., Viglas, S., Beyer, K.S., Shanmugasundaram, J., Shekita, E.J. and Zhang, C.: Storing and querying ordered XML using a relational database system, *Proc. ACM SIGMOD*, pp.204–215 (2002).
22) Wang, H. and Meng, X.: On the Sequencing of Tree Structures for XML Indexing, *Proc. IEEE ICDE*, pp.372–383 (2005).
23) Wang, H., Park, S., Fan, W. and Yu, P.S.: ViST: A Dynamic Index Method for Querying XML Data by Tree Structures, *Proc. ACM SIGMOD*, pp.110–121 (2003).
24) Yoshikawa, M., Amagasa, T., Shimura, T. and Uemura, S.: XRel: A path-based approach to storage and retrieval of XML documents using relational databases, *ACM TOIT*, Vol.1, No.1, pp.110–141 (2001).
25) Zhang, C., Naughton, J.F., DeWitt, D.J., Luo, Q. and Lohman, G.M.: On Supporting Containment Queries in Relational Database Management Systems, *Proc. ACM SIGMOD*, pp.425–436 (2001).

(平成 18 年 3 月 20 日受付)
(平成 18 年 7 月 9 日採録)

(担当編集委員　浦本 直彦)

根本　潤（正会員）
平成 16 年慶應義塾大学経済学部経済学科卒業．平成 18 年同大学大学院開放環境科学専攻修士課程修了．同年より，株式会社日立製作所システム開発研究所に勤務．

遠山　元道（正会員）

昭和 54 年慶應義塾大学工学部管理工学科卒業．昭和 56 年同大学大学院修士課程修了．昭和 59 年慶應義塾大学工学部管理工学科助手．平成 4 年専任講師．平成 8 年情報工学科に移籍．現在に至る．博士（工学）．平成 8 年オレゴン大学院大学客員研究員．平成 10〜13 年科学技術振興事業団さきがけ研究 21「情報と知」領域研究員．主にデータベースの研究に従事．電子情報通信学会，日本ソフトウェア科学会，IEEE Computer Society，ACM 各会員．