XML データベースにおける構造要約索引を用いた Tree Pattern 問い合わせ処理方式に関する検討

朱　佳俊† 　金子 邦彦‡

†九州大学大学院システム情報科学府
‡九州大学大学院システム情報科学研究院
E-mail: †shukashun@db.is.kyushu-u.ac.jp, ‡kaneko@ait.kyushu-u.ac.jp

あらまし 近年，XML データベースのニーズが高まっている．XML のノードを選択するために，Tree Pattern 問い合わせは多く使われている．問い合わせを処理するために，索引手法及び XML のノード間の関係を判断できるラベルを利用する構造結合と呼ばれる手法は広く研究されている．本稿では，構造結合における結合回数が多い場合の効率低下を改良する方法を提案する．提案手法では，模倣関係により，索引を構成し，それを利用することで，構造結合の回数を削減する．また，B+-Tree を利用することで，必要なノードのみをメモリに読み出し，メモリにおける XML のノード数を削減できる．

キーワード XML データベース，構造要約，構造結合，TPQ

An Investigation on Tree Pattern query processing with XML Summaries in XML database

Jiajun ZHU† and Kunihiko KANEKO‡

†Graduate Faculty of Information Science and Electrical Engineering, Kyushu University
‡Graduate School of Information Science and Electrical Engineering, Kyushu University
E-mail: †shukashun@db.is.kyushu-u.ac.jp, ‡kaneko@ait.kyushu-u.ac.jp

1. はじめに

XML はデータ交換フォーマットとして使われている．多くの企業では，XML を用いてデータを管理，交換することになっている．また，XML データベースのニーズが高まり，多くの研究者に研究されている．

XML データへアクセスするために，XPath や XQuery といった問い合わせ言語が開発されている．また，高速化のために，構造要約索引と構造結合等の手法は多数提案されている．DATAGUIDE[6]，1-index[7]，A(k)-index[8]，D(k)-index[9]等の構造要約索引は XML 文書の要約を構築し，XML 文書を使わず，索引を用いてデータを検索することができる．しかし，こういう手法では，ノード間の関係を判断できない．また，文献[4][10]では，ラベル付け手法に基づいた構造結合といった手法の研究を行っている．こういう手法では，ノードのラベルを比較することでノード間の関係を判断できるが，構造結合操作の回数が重要な課題になり，特に，問い合わせのパス式が長くなる場合は検索の効率が問題になる．そこで，文献[1]では，こういう問題を解決するため，XPath 問い合わせをシングルパスと小枝パターンに別けて処理する手法を提案している．特に，シングルパスには DATAGUIDE という構造要約索引を使い，高速化を実現している．しかし，DATAGUIDE はあらゆる XPath に対応できず，特に，述語を使う問合せに対しては，評価できない場合が多い．また，小枝パターンには構造要約索引の応用が考えられていない．そこで，本稿では，模倣索引といった構造要約索引と XB-Tree[4]を統合し，小枝パターンにおける構造結合操作を減らすことで，小枝パターン処理の高速化を実現する手法を提案している．また，使われる頻度が高い先祖/子孫軸，親子軸などの高速化に注目するため，XPath 問い合わせの代わりに，XPath 問い合わせのサブセットである Tree Pattern 問い合わせを使う．

2. 準備

本稿で使う基本的な概念を説明する．

2.1. XML モデルと問い合わせ木

XML 文書は木モデルで表されることができる.XML

```
<a>
  <b>
    <c/>
    <c>
      <d/>
    </c>
  </b>
  <b>
    <c>
      <d/>
    </c>
  </b>
</a>
```

図 1. XML 文書 D

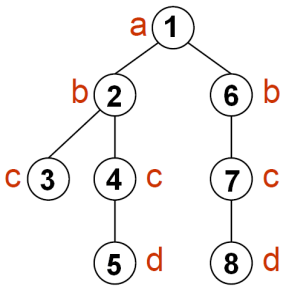

図 2.XML 木

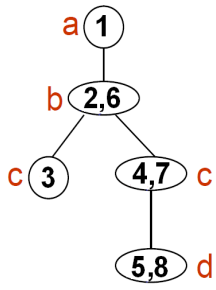

図 3.模倣索引

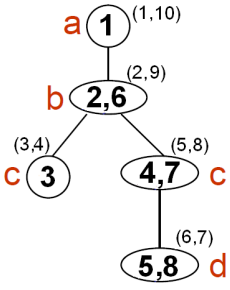

図 4. ラベル付き索引

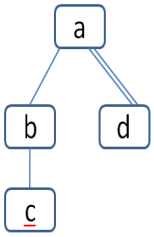

図 5. 問い合わせ木

木 T を，*T=(V,E)*で与える．V はノードの集合，E は枝の集合である．図 2 は図 1 の XML 文書の木構造を示している．

問い合わせ *Q=//a[d]/b/c* は図 5 のような問い合わせ木で表すことができる．問い合わせ木のノードは問い合わせにおけるノードテストであり，1 重線はノード間の親子関係を示しており，2 重線はノード間の先祖/子孫関係を示している．c は問い合わせ Q の出力なので，アンダーラインが付いている部分は問い合わせの出力である．

2.2. 模倣索引と Tree Pattern 問い合わせ

XML における模倣関係を以下のように定義する．便利のため,ここではフォワード模倣関係のみを考える．

1．ノードの型 T(タグ名)：

node1 = root(D) ならば *node2 = root(D)*

それ以外は *T(node2) = T(node1)*

2．外向エッジ：

外向エッジ(node1,node1') ごとに対して，

外向エッジ(node2,node2') が存在するならば，node1'は node2'に模倣されたと呼ぶ

図 3 は図 2 の XML 木の模倣索引を示している．

文献[3]における有限グラフの模倣計算のアルゴリズムを使うことで，O(||E||V||)時間で模倣索引を構築できる．

Tree Pattern 問い合わせとは XML 問合せ言語 XPath のサブセットであり，Self，Child，Descendant，Descendant-or-self という四つの軸を提供している．これから Tree Pattern 問い合わせを TPQ と呼ぶ．また，文献[2]の結論により，XML 文書の模倣索引は全ての TPQ 問い合わせを評価できる．

2.3. 構造結合

構造結合とは，ラベリング手法に基づいた問い合わせ手法である．本稿では，Region Numbering Schema といったラベリング手法を用いている．Region Numbering Schema は，先行順の深さ優先探索により，XML のノードに(StartPos,EndPos)を与える．StartPos は,探索がノードに入るときに与えられた番号であり，EndPos は，探索がノードを出るときに与えられた番号である．図 4 は Region Numbering Schema により，ラベルを付けた索引である．ノード間の関係はラベルを用いて判断可能になる．すなわち，問い合わせの軸処理はラベルの比較操作になる．

2.4. XB-Tree

文献[4]は構造結合における小枝結合を効率的に処理するために XB-Tree といったデータ構造を提案している．小枝結合とは，図 5 のような分岐がある問い合わせのことである．XB-Tree は XML 文書の StartPos, EndPos による検索に適した索引であり，B-Tree と同じように構築することができる．XB-Tree の葉ページは XML のノードを保存し，ノードの StartPos によってソートされている．中間ノードはラベル(StartPos, EndPos)とその子供ページを指すポインターを持っている．中間ノードの StartPos は子供ページの要素の StartPos の最小値を持ち，EndPos は子供ページの要素の EndPos の最大値を持っている．しかし，XB-Tree はタグ名を保存しないので，タグごとに，XB-Tree を構築する必要がある．図 7 は図 6 の XML の d 要素に対して,c 要素の XB-Tree である.XB-Tree を用いて XML のノード(20,30)d の先祖ノード c を探すには，XB-Tree のルートから XB-Tree の要素のラベルは (20,30)を含んでいるかどうかをチェックし，含んでいる場合はその要素の子供ページへ移動し，また判断する．図 7 の例は(3,58) (3,50) (3,50)と(3,58) (3,50) (6,48)と (3,58) (10,58) (10,45)三つの経路が条件に満たしているので，(3,50)c と(6,48)c と(10,45)c は(20,30)d の先祖ノードである．

また，文献[5]によると，索引を用いた構造結合においては，XB-Tree が B+-Tree と XR-Tree よりも優れた性能を示している．

3. 提案手法

提案手法では，文献[1]のシングルパスと小枝パターンのアイデアーを導入している．図 8 の左側は従来手法のフローで,右側は提案手法のフローを示している．

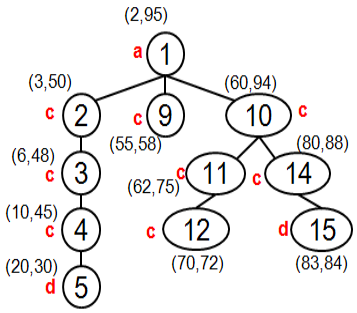

図 6. XML 木

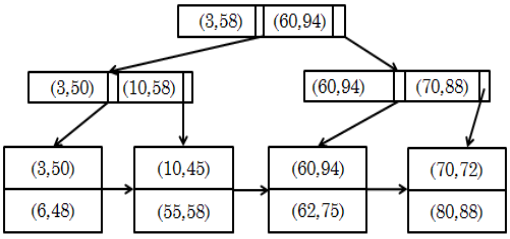

図 7. XML 木に対応する XB-Tree

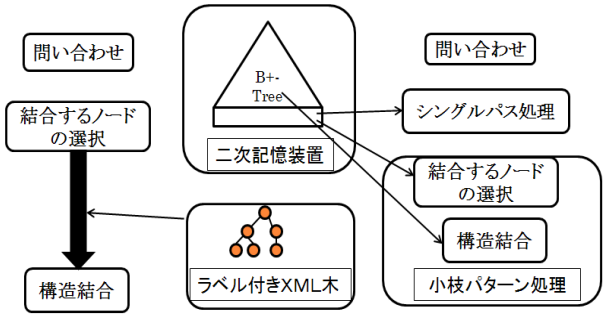

図 8.　従来手法と提案手法の比較

従来手法では，問い合わせ木におけるあらゆる二項パターンに対して，問い合わせのノードテストと同じタグを持つ XML ノードを全部メモリに呼び出し，これらに対して，構造結合を行う．

これに対して，提案手法では，問い合わせに対して，分岐が存在する場合，分岐をシングルパスに分解し，分解されたシングルパスごとに対して，模倣索引を用いて，探索を行う．得られたシングルパスの出力に simIdx-XB+索引を用いて構造結合を行い，結果を出力する．提案手法はシングルパスに対して，構造結合を行わない一方，分岐に対する処理を行うときに発生する構造結合の回数を simIdx-XB+索引を使うことで減らすことができる．

3.1. SimIdx-XB+索引

本稿では XB-Tree の性質を持つ B+-Tree を提案している．

本稿における B+-Tree の性質：

(1) 中間ノードと葉ノードの二種類のノードが存在する

(2) 中間ノードのラベルは XB-Tree と同じように与える

(3) 葉ノードは XML 文書に対応する模倣索引のノードを保存している

(4) 葉ノードはラベルの StartPos によってソートされている

模倣索引は文献[3]の HHK アルゴリズムを用いて，構築することができる．また，構築した模倣索引に Region Numbering Schema を用いて，ラベルも付ける．

また，メモリにおけるノード数を減らすため，B+-Tree を二次記憶装置に格納する．格納された索引を SimIdx-XB+索引と呼ぶことにする．

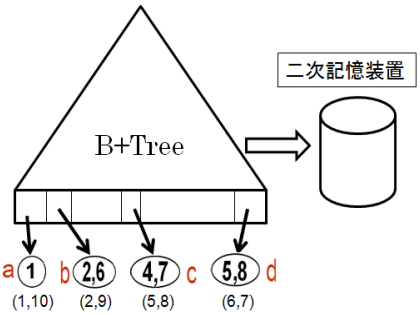

図 9. SimIdx-XB+索引

3.2. SimIdx-XB+を用いた問い合わせ

文献[1]と同じく，本稿では問い合わせをシングルパスと小枝パターンに分別し，処理を行う．シングルパスには，必ず最大の長さを保持する．小枝パターンには，必ずシングルパスに分解する．

SimIdx-XB+索引を用いたシングルパスと小枝パターンの処理を順番に説明する．

(1) シングルパス：

分岐がないパスについては，従来の方法と同じ手順で行う．図 3 の模倣索引を利用し，探索を行う．文献[2]の結論は結果の正確性を保証している．最後に，問合せの結果を二次記憶装置から取り出す．

(2) 小枝パターン：
小枝パターンをシングルパスに分解し，シングルパスごとに，(1)の操作を行う，得られたシングルパスの結果に構造結合を行う．

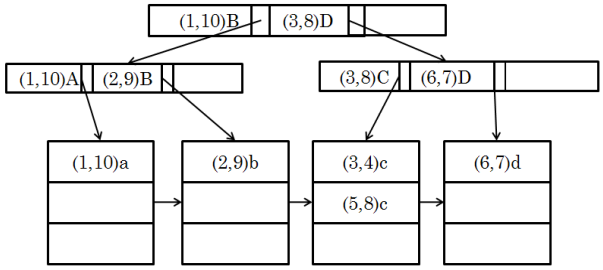

図 10.　simIdx-XB+索引

例えば，図 10 のような simIdx-XB+索引が構築されたとする．問い合わせ *Q=//a[d]/b/c* が与えられ，問い合わせ木をシングルパス問い合わせに分解すると，*q1=a*，*q2=b/c*，*q3=d* になる．q1 にシングルパス処理を行う結果は{(1,10)a}になる．次に，分岐であるノードテスト a と b に simIdx-XB+索引を用いて構造結合を行う．即ち(1,10)a の子ノード b を探す．索引のルートから探索を行い，b ノードなので，中間ノード B の下へ探索しに行く．次に(1,10)a と(2,9)B のラベルを比較し，a のラベルの範囲は B のラベルの範囲を含まれば，下の葉へ探索しに行く．最後に葉ノードである(2,9)b を取り出す．次に，(2,9)b を持って，*q2=b/c* にシングルパス処理を行い，出力は(3,4)c，(5,8)c になる．分岐である a と d にも同じ手順で処理を行う．問い合わせ *Q=//a[d]/b/c* の出力は(3,4)c，(5,8)c になる．それらに対応する XML のノードを二次記憶装置から取り出して返す．

4. まとめ

本稿では，XML の構造要約索引を用いて構造結合の回数を削減する手法を提案している．提案手法では，構造要約索引である模倣索引のノードを B+-Tree の葉に格納することで，構造結合におけるラベルの比較を削減する索引を提案し，さらに，B+-Tree を二次記憶装置に格納することで，メモリにおける索引のノード数を削減することができる．

また，今後の課題としては，XML の構造要約を用いて問い合わせにおける無駄な述語を削除し，より簡潔な問い合わせを求める．これは，問い合わせ性能にかかわる重要な課題と考えられる．

謝辞

本研究は科研費（22500092）の助成を受けたものである．

参　考　文　献

[1] 江田毅晴，鬼塚真，山室雅司，“XML データの要約情報を用いた高速な XPath 処理方法”, DEWS 2005.

[2] Prakash Ramanan. “Covering Indexes for XML Queries: Bismulation – Simulation = Negation.” In Proceeding of the 29th VLDB, pp.165-176, 2003.

[3] Monika R. Henzinger, Thomas A. Henzinger, and Peter W. Kopke, “Computing Simulations on Finite and Infinite Graphs” In Proceedings of the 36th Annual Symposium on Foundations of Computer Science , IEEE Computer Society Press, pp. 453-462, 1995.

[4] Bruno N, Srivastava D, Koudas N. “Holistic twig joins: optimal XML pattern matching”, In Proceeding of ACM SIGMOD International Conference on Management of Data. New York: ACM Press, pp.310-321, 2002.

[5] Hanyu Li, Mong Li Lee, Wynne Hsu, Chao Chen. “An evaluation of XML indexes for structural join”, In ACM SIGMOD Record Volume 33 Issue 3, 2004.

[6] Goldman R, Widom J. “DataGuides: Enabling querying formulation and optimization in semistructured database.” In Proceeding of the 23th VLDB, pp.436-445, 1997.

[7] Milo T, Suciu D, “Index Structures for Path Expressions.” In Proceedings of the 7th International Conference on Database Theory , pp.277-295, 1999.

[8] Kaushik R, et al. “Exploiting local similarity for efficient indexing of paths in graphstructured data.” In Proceedings of the 18th International Conference on Data Engineering, pp.129, 2002.

[9] Chen Q, et al. “D(k)-index: An adaptive structural summary for graph-structured data.” In Proceedings of the 2003 ACM SIGMOD international conference on Management of data, pp.134-144, 2003.

[10] Al-Khalifa S, et al. “Structural joins: A primitive for efficient XML query pattern matching.” In Proceedings of the 18th International Conference on Data Engineering, pp.141, 2002.