2004－DBS－134　(I)　(28)
2004／7／13

XML-DB 用の基数組による節点の番号付け索引構造の実装

張　鵬　　佐藤 隆士

あらまし

XML-DB における経路式質問を効率的に処理するための索引構造の実装である．従来，XML データの木構造の節点に振った番号を索引に格納し，索引のみで節点間の先祖子孫関係を知る方法が提案されている．著者らは，木構造中での絶対的な位置を識別可能な番号付けを提案している．具体的には，レベルごとに異なる基数を用い，経路を整数にコーディングしている．また，文書グループごとに異なる基数組を選択できる方法を提案している．本稿では，提案中の「基数組集合による階層構造を識別可能な木節点の番号付け」に基づき XML 文書の検索部を実装した．そして，索引のデータ構造や経路式質問に対して検索するアルゴリズムを示している．

Numbering Index Structure by Set of Radix Groups for XML-DB

Peng ZHANG　　Takashi SATO

Abstract

We have made the index structure that processes requirements of Regular Path Expression efficiently. It has been proposed that the assigned numbers of the nodes of tree structure for XML-DB is stored in the Index. We can know the relation between the nodes by the Index. In this paper, we propose a numbering that discernment of an absolute position in a tree structure is possible. We show the numbering by the different radix group for every level, and the method that we select the radix group by the document group. We propose the data structure of the Index and the algorithm to processes requirements of Regular Path Expression.

1．　はじめに

ＸＭＬ(拡張可能なタグ付き言語)は，インターネットの標準的なデータ交換手段として用いられるようになってきた．これに伴い，ＸＭＬデータを効率よく格納および検索する必要性が増している．ＸＭＬ文書は木で表現できる階層構造を成している．このため，階層をたどる経路式質問と呼ばれる特徴的な質問がなされる．

大阪教育大学大学院総合基礎科学専攻
Course of Mathematical and Information Science, Osaka Kyoiku University

経路式質問を効率よく処理するため，経路に基付く索引が提案されている[1-5]．経路の途中は任意で，階層の上位と下位を指定する正規経路式質問で効果的な索引も提案されている．木構造の節点に対応する要素，属性を表す節点に番号を振り，節点ごとに分解して番号と共に格納する．検索時には，付けられた節点番号から節点間の先祖子孫関係などの情報を得ることができる索引である[6-11]．文献[7-9]の方法は番号のコンパクトさを追求したものであり，更に Cohen ら[10]の方法は，更新を考慮したものである．しかし，

文献[6-10]の方法では，節点対が，先祖子孫関係であるかどうかは分かるが，節点番号だけではその関係が親子であるか，あるいは何世代離れた関係であるかに答えることはできない．また，k-ary 完全木に基付く番号付けによる Lee ら[11]の方法は，節点に付けられた番号から，木における絶対的な位置がわかるので，先祖子孫関係でなく，世代の間隔や兄弟節点間の位置関係も答えることができるはずである．しかし，番号を幅方向に順に振っているため親子以外の関係の計算は容易ではない．また，多くの利用されない仮想節点に番号を割り当てるため，arity と木の高さが大きくなると，現実的でないことが指摘されている．Kaplan と Miro[12]は，親子関係を知ることができる拡張機能を提案している．また，Peleg[13]は，２節点間の最も近い共通先祖(LCA: lowest common ancestor)を知ることができる番号付けを提案している．

著者らは，節点の木における絶対的な位置を記憶しており，先祖子孫関係をはじめ上にあげた節点間の関係を含むすべての位置関係を容易に計算できる方法を提案した[14]．更に進め，レベル(文書を表す木構造における深さ)だけでなく，文書グループごとに異なる基数組を使用することにより，多様な木構造に番号付けできる方法を提案した[15]．本稿では，提案した方法で索引構造を実装した．そして，正規経路表現の質問を検索する方法や索引構造の処理を示している．

2. 階層構造を識別可能な木節点の番号付け

木の節点に対して，絶対位置を表す番号付けを以下の(A)の考えをもとに(B)の方法で行う[14]．なお，具体的に番号付けした例は，文献[14]を参照されたい．

(A) 数字の組による節点の位置表現

高さ h までの木について，h 組の数字からなる番号を付ける．このうち，レベル n にある節点の下から h-n 個の数字は 0 とする．従って，根(レベル 0)は h 個の数すべてが 0 である．レベル n にある節点 s の親を p とするとき，s の位置を表す数字の組の先頭から n-1 組は p を受け継ぐ．n 番目には，p の子を左から数えたときの s の順を入れる．本の章，節などに付ける番号の組と同じであると考えれば分かり易い．

(B)節点のコード番号

数字の組と一対一に対応する単一整数へのコード化を行う．レベル $i(0 \leqq i \leqq h\text{-}1)$の基数を k_i，あるレベル $n(\leqq h\text{-}1)$にある節点の数字の組を$(a_1, \ldots, a_n, 0, \ldots, 0)$とするとき，

$$((\ldots(a_1k_1+a_2)k_2+\ldots+a_n\text{-}1)k_{n-1}+a_n)k_n\ldots k_{h-1} \qquad (1)$$

とコード化する．但，$1 \leqq a_i \leqq k_i\text{-}1$ である．レベル h 以上あるいは a_i が k_i 以上の節点はオーバフローとして扱う[14]．(1)式によるコードとオーバフローによるコードを区別するために 1 ビット使用するので，コード化ビット長を m とするとき，コード化は m-1 ビット以内で行う．

ここで，各レベル i にある節点の子の数をあらかじめ調べておくこととし，レベル i の基数 k_i は，ほとんどすべての子の数が，k_i-1 以下になるように決める．１が引かれているのは，先頭から i 番目の数が 0 をレベル i-1 より浅いレベルの節点に使うことにし，レベル i における子の番号を 1 から始めたためである．

3. 基数組による木節点の番号付け

更に，(2)の方法に対して以下の基数組集合による改良を行う．

XML 文書を表す木構造は，文書により様々な形をとる．同じ節点数で表される木についても，幅広で浅い木，逆に幅は狭いが深い木がある．細かく見れば，各レベルの節点数が異なっており変化に富んでいる．このため，１セットの基数組のみでは，番号空間を効率的に使用するコード化を行うことは不可能である．そこで，XML 文書集合を構成する各文書の構造に適した基数組を選択できるようにする方法を提案する．番号付けの際問題となるイレギュラーな木構造にも対応しやすい特徴を有す．基数組は１セットではなく，複数セットもつことが可能となるため，番号付けに先立ち，様々な木構造を，より値が小さく，且つより少ない数の組でカバーする基数組集合を求める必要性が生じる．

3.1 基数組集合の設定

木節点の番号付けのもととなる基数組集合を以下の 2 パス構成で設定する．最初のパス(A)では，XML 文書を表す木構造の各レベルの節点数分布を調査し，コード化に用いる基数組の候補を上げる．続くパス（B）では，先のパスの情報をもとに，定められたコード化ビット長の範囲内において，マージして候補を絞り基数組集合を決める．

なお，具体的に基数組集合の設定した例は，文献[15]を参照されたい．

(A) 節点数分布調査と文書のグループ化

XML 文書ごとに，各レベルの節点数を記録する．i 番目の文書(i=1,2,…,d)のレベル l(l=1,2,…)で兄弟関係にある節点数の最大値（言い換えるとレベル l-1 の子の最大値）を a_{il} とし，組 $A_i=(a_{i1},a_{i2},..,a_{ihi})$ を作る．h_i は文書 i の木の高さである．数の組を $B_j=(b_{j1},b_{j2},...)$とし，文書 i のすべてのレベルにおいて，$a_{il}\leqq b_{jl}$ であるとき，文書 i は，この数の組（これを j 番目の「基数組」という）でカバーされるといい，$A_i\leqq B_j$ で表す．ただし，≦の比較において，どちらかの項数が足りない部分は 0 として比較するものとする．

(B) 基数組のマージと設定

(A)で得られたオーバフローを生じない文書に対する基数組の集合は，各組のレベルの基数値が必要最小の条件下での組数(要素数)最小の集合である．一方，オーバフローを生じない基数組によるコード化は，m-1 ビット以下なので，m-1 に満たない部分については，下位ビットから 0 を埋めてもよいが，より効果的に利用することを考える．即ち，**(A)** で得られた，基数組の対を $B_i=(b_{i1},b_{i2},...)$，$B_j=(b_{j1},b_{j2},...)$とするとき，B_i,B_j を共にカバーする $B_{i:j}=(b_{i:j1},b_{i:j2},...)$によるコード化が m-1 ビット以下になるならマージし，基数組集合をコンパクトにする．ここで，$b_{i:j1}=\max(b_{i1},b_{j1})$, $b_{i:j2}=\max(b_{i2},b_{j2})$, … である．

4． 索引構造の実装

Berkeley DB[17]で Btree を利用して索引を作成する．理由は，Berkeley DB が高速であることに加え，可変長のレコードが使えるからである．可変長レコードが使用できれば，XML 文書の中で，頻出する単語が何回現れても，格納可能である．また，Berkeley DB は Perl で直接に操作できるという特長を持っている．

4.1 データ構造

索引は(**単語**(name)=> **番号組集合**(value))という形式のレコードで構成されている．番号組集合はその単語を含む全ての節点の番号組（文書番号と節点番号を組み）で構成されている．次のようになる．

```
%dbm=(
  'office' => '1,307200;2,1719;6,18102;  ...',
                    ・
                    ・
                    ・
);
```

4.2 位置関係の計算

節点に付けられた番号の 2 進数表現から，節点間の位置関係を計算する方法を説明した[14]．本稿では，実装の中に利用した 10 進数表現からの計算方法を示す．索引のデータ構造が利用した節点番号の 10 進数表現は式(1)で計算された．そして，式(1)を逆に利用して番号から絶対位置に複元することができる．以下に絶対位置の複元のアルゴリズムを示す．

[アルゴリズム A] 節点の絶対位置の複元

入力：節点に付けられた番号の 10 進数表現およびその文書の基数組

出力：節点のレベルと位置に関して数字組

方法：

(1) レベル 1 の基数 k_1 からすべての基数の掛け算を行う，積 P を求め，レベル数 K=0 とする

(2) 節点番号 N を積 P で割り，商を a とする，(a)a<1 の時，K は節点のレベルｎ(≦h-1)の位置数，a>=1 のとき，a がレベルｎの位置数，レベル数 K は +1(K=K+1)する

(3) 節点番号 N は(2)の余りになり，積 P はレベルｎの基数を割り(P=P/ k_n)，(2)へ

最後に，節点のレベル数 K と各レベルの位置数を得ることができる，絶対位置を復元する．従って，節点間の関係や先祖子孫関係にある節点間の距離および兄弟と従兄弟関係などを簡単に計算することができる．

[例１] 基数組(7,2,16,3,6,2,4,2,5)の文書に対して76920と76930と番号付けられた二つの節点の関係を求め．ここに積Pは46080で，二つの節点のレベルはそれぞれ7および8であることが分かる．レベル差が１，即ち親子の関係にあることが分かる．

4.3 検索方法

文書集合の全ての単語と付けられた番号を Berkeley DB に入れた，「基数組集合による階層構造を識別可能な木節点の番号付け」に基づき XML 文書の索引を作成した．そして，この索引による正規経路表現式質問を検索することができる．以下に検索のアルゴリズムを示す．

[アルゴリズム B] 正規経路表現式質問(テキスト

ノード)を効率よく処理する XML 文書の検索方法
入力：正規経路表現式質問(テキストノード)の文字列
出力：条件に合う文書番号集合
方法：
(1) 質問の文字列を'/'による分解し，配列 K に入れる.
(2) 最初の一つ文字列の番号組を索引から取り出し，配列 J に入れる.
(3) 次の文字列の番号組を索引から取り出し，配列 J と比べる, 同じ文書番号の番号組を残す.
(4) (３)の中に残った番号組の節点番号を利用し，二つ文字列の関係を判断しながら，質問に合う文書番号を配列 J に残す.
(5) もし次の文字列があったら，(３)へ
(6) 配列 J 中に, 質問に合う文書番号集合を見つけることができる.

4.4 索引の更新

XML-DB が変わる時，伴って索引を更新しなければならない．基本的に XML-DB の変化に削除, 更新，追加の三つの場合がある.

削除と文書構造に変化のない更新の場合には, 索引構造に大きな影響はない. 索引構造の番号に余裕が生じる可能性があるが, 索引の構造は変わらない. 変化する文書内容に対してのみ索引を更新することができる.

文書構造に変化がある更新と追加の場合には, 以下二つの可能性がある. 一つは新たな文書の基数組を必要としない場合である. つまり XML 文書で各レベルの兄弟関係にある節点数の最大値（言い換えるとレベル *l*-1 の子の最大値）を基数として基数組を作ってあるので既に存在する基数組みが利用できる場合である．また，各レベルに番号の余裕を持っているので, 一定の範囲に追加ノードの番号付けを行うこともできる. もう一つは新たな文書の基数組を必要とする場合である. つまり追加ノードが一定的な範囲を超えて, 既存の基数組は利用できなかった場合である. 文書集合全体ではなく変わった文書だけに対して最初から索引構造を再構築する. 一つ文書に対して再構築のコストは大きくないである．著者らは，表１のコンピュータでレベル 10 サイズ 20kb 文書(XMLgen で作った)を用いて実験を行った．索引再作成の時間は 2 秒間以内であった.

CPU	メモリ	OS	ハードディスク	ハードディスクの SPEED
AMD Duron 856Mb	256Mb	FreeBSD 4.5	Maxtor 4D080H4	5400 RPM

表１　実験環境

5．おわりに

XML-DB における経路式質問を効率的に処理するため, 索引に文書の階層構造を反映した番号を格納する方法を採用した. 木構造のレベルごとに基数を設定できるだけでなく, 文書グループごとに異なる基数組を選択できる方法を提案した. そして，各レベルの基数値最小のもとでの，基数組数最小の集合を求める具体的方法や定められたコード化ビット長内に基数組をマージする欲ばり(Greedy)アルゴリズムを示した．本稿では，以上の方法を利用して索引構造を実装した. そして, 索引のデータ構造や正規経路表現の質問を検索するアルゴリズムを示した．更に，データを更新する時に索引構造の処理方法を示した.

但し, 現階段に実験対象のサイズが大きくないという欠点が有る．従って，大規模な XML-DB また分散型 XML-DB に対して索引構造の性能や改良を行うことが今後の課題となる.

文　献

[1]Brian F. Cooper, Neal Sample, Michael J. Franklin, Gisli R. Hjaltason and Moshe Shadmon: A Fast Index for Semistructured Data, in *Proceedings of the 27th VLDB Conference*, Roma, 2001.

[2]Chun Zhang, Jeffrey Naughton, David DeWitt, Qiong Luo: On Supporting Containment Queries in Relational Database Management Systems, in *Proceedings of the 2001 ACM SIGMOD Conference*, Santa Barbara, CA, May 2001.

[3]Chee-Yong Chan, Minos Garofalakis, Rajeev Rastogi: RE-Tree: An Efficient Index Structure for Regular Expressions, *in Proceedings of the 28th VLDB Conference*, Hong Kong, 2002.

[4]Chin-Wan Chung, Jun-Ki Min, Kyuseok Shim: APEX: An Adaptive Path Index for XML Data, in *Proceedings of the 2002 ACM SIGMOD Conference*, Madison, Wisconsin, June 2002.

[5]Raghav Kaushik, Phillip Bohannon, Jeffrey F Naughton,

Henry F Korth: Covering Indices for Branching Path Queries, in *Proceedings of the 2002 ACM SIGMOD Conference*, Madison, Wisconsin, June 2002.

[6]Quanzhong Li and Bongki Moon: "Indexing and Querying XML Data for Regular Path Expressions", in *Proceedings of the 27th VLDB Conference*, Roma, 2001.

[7]Serge Abiteboul Haim Kaplan and Tova Milo: "Compact Labeling Schemes for Ancestor Queries", in *Proceedings of 12th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp.547-556, 2001.

[8]Stephan Alstrup and Theis Rauhe: "Improved Labeling Scheme for ncestor Queries", in *Proceedings of 13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp.947-953, 2002.

[9]Hain Kaplan, Tova Milo and Ronen Shabo: "A Comparison of Labeling Schemes for Ancestor Queries", in *Proceedings of 13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp.954-963, 2002.

[10]Edith Cohen, Haim Kaplan and Tova Milo: "Labeling Dynamic XML Trees", in *Proceedings of 21st ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems (PODS)*, pp.271-281, 2002.

[11]Yong Kyu Lee, Seong-Joon Yoo and Kyougro Yoo: "Index Structures for Structured Documents", in *Proceedings of the ACM 1st Conference on Digital Libraries*, Bethesda, MD, March 1996, pp.91-99.

[12]Hain Kaplan, Tova Milo: "Short and Simple Labels for Small Distances and Other Functions", in *7th International Workshop on Algorithms and Data Structures (WADS)*, Vol. 2125 of LNCS, Springer,.

[13]D. Peleg : "Informative Labeling Schemes for Graphs", in *Mathematical Foundations Computer Science (MFCS)*, pp.579-588, 2000.

[14]佐藤,里本,小畑, 潘:階層構造を認識可能な木節点の番号付け, DEWS2002, 109, Mar. 2002.

[15]佐藤,李,居,張: 基数組集合による木節点の番号付け, DEWS2003, 3-C-1, Mar. 2003.

[16]Xmark - An XML Benchmark Project: URL: <http://monetdb.cwi.nl/xml/generator.html>.

[17]Berkeley DB - Sleepycat Software Inc.: URL: <http://www.sleepycat.com>