基数組集合による木節点の番号付け

佐藤 隆士，李 正勲 ，居 行和，張　鵬
大阪教育大学大学院総合基礎科学専攻数理情報コース
〒582-8582 大阪府柏原市旭ヶ丘 4-698-1
{sato, jhsrlove, k11,zp}@ss.osaka-kyoiku.ac.jp

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の提案である．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．本稿では，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．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，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アルゴリズムを示している．

キーワード　XML-DB，テキスト DB，木節点の番号付け，索引付け，正規経路表現

Numbering by Set of Radix Groups for Tree Structures

Takashi SATO, Jeong Hun LEE, Yukikazu KYO, Peng ZHANG
Course of Mathematical and Information Science, Division of Pure and Applied Science,
Graduate School of Education, Osaka Kyoiku University

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．

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]．本稿では，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する．

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]．更に本稿では，(C)による改良を行う．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，

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

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．

(C) 基数組による木節点の番号付け

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，本稿で 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．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 2 パス構成で設定する．最初のパスでは，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．

３．１　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 $A_i=(a_{i1},a_{i2},\ldots,a_{ihi})$ を作る．h_i は文書 i の木の高さである．数の組を $B_j=(b_{j1},b_{j2},\ldots)$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．

[アルゴリズムA] XML 文書集合の節点数の分布を求める

入力：XML 文書集合(文書数 d)およびコード化ビット長 m

出力：基数組集合，基数組の吸収関係の記録，および各 XML 文書が属する基数組

方法：

(1) 基数組集合を空集合とし，k=0 とする．調査対象の文書番号を i=0 とする．

(2) i=i+1 としする．i>d なら終了．そうでなければ，文書 i の各レベルの節点数を調べ，A_i=(a_{i1},a_{i2},…,a_{ihi})を求める．

(3) 文書 i の節点のコード化ビット長 L_i=log(a_{i1}+1)+log(a_{i2}+1)+…+log(a_{ihi}+1)[1] が m 以上 の場合は，文書 i は m ビット内でコード化できないことを記録する．属する基数組は A_i とする． (2)に進む．

(4) A_i が基数組 B_j(j=1,2,…,k)でカバーされる(即ち，A_i≦B_j の)場合，文書 i が属する基数組を B_j とする．複数の候補ある場合はそのいずれを用いてもよい．(2)に進む．

(5) カバーされない，あるいは k=0 場合は，k=k+1 とし，B_k=A_i を基数組集合の要素として加えると共に，文書 i が属する基数組を B_k とする．

(6) B_j≦B_k(j=1,2,…,k-1)となる基数組 j については，k でカバーされるので，B_j は B_k に吸収されることを記録(j→k)しておく．B_j は基数組集合から削除する．(2)に進む．

B_1,B_2,…,B_kのうち最終的に他に吸収されず残った組から基数組集合が得られる．また，アルゴリズムの途中で記録した各文書 i の属する基数組も出力になる．但し，(3)でオーバフローを生ずるとした文書 i の基数組は暫定的に A_i とする．[2]

[例１] コード化ビット長 m=16，文書数 d=10，A_1=(3,3,6,5,1)，A_2=(5,3,4,6,2,2)，A_3=(10,8,4,6)，A_4=(7,6,2)，A_5=(20,33,42,50,20,1,1)，A_6=(2,2,5,5,3,2,1,1)，A_7=(10,11,6,6)，A_8=(5,2,2,4,1)，A_9=(7,6,5,4,2)，A_{10}=(1,2,4,4,3,2,1)のとき，B_1=A_1，B_2=A_2，B_3=A_3，B_4=A_6，B_5=A_7，B_6=A_9 となる．基数組集合は{B_1,B_2,B_4,B_5,B_6}となる．B_3 は B_5 に吸収される．各文書が属する基数組は，文書番号の順に，B_1,B_2,B_5,B_5,A_5,B_4, B_5,B_2,B_6,B_4 となる．5 番目の文書が A_5 であるのは，L_5=26.9 で m 以上となるからである． □

[1] 本稿での対数の底はすべて 2 とする．

[2] 実際のコード化は，A_iをもとにしたオーバフローコーディングの方法に従う．

３．２ 基数組のマージと設定

3.1 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，3.1 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．以下に，基数組マージのための欲ばり(Greedy)アルゴリズムを示す．

[アルゴリズムB] 基数組のマージと調整

入力：アルゴリズム A で得られた基数組集合およびコード化ビット長 m

出力：マージおよび調整された基数組集合およびマージの記録

方法：

(1) 基数組集合をアルゴリズム A の基数組集合で初期化する．

(2) 集合内のすべての基数組の組合せ B_i,B_j について，各レベルのビット長の差の絶対値和 $\Delta_{i,j}$=|log(b_{i1}+1) -log(b_{j1}+1)|+|log(b_{i1}+1)-log(b_{j1}+1)|+… を求める．

(3) {$\Delta_{i,j}$}のうち最小のものを$\Delta_{k,l}$とするとき，基数組 B_k,B_l を共にカバーする $B_{k:l}$ を作成する．但し，{$\Delta_{i,j}$}が空であれば(6)に進む．

(4) $B_{k:l}$によるコード化によるビット長$L_{k:l}$がm-1 以下であれば，マージしたものに置き換える．即ち，基数組集合から，B_k および B_l を削除し，代わりに$B_{k:l}$を挿入する．マージを記録(k,l→k:l)し，(2)に進む．

(5) {$\Delta_{i,j}$}から$\Delta_{k,l}$を削除し，(3)に進む．

(6) 基数組集合の各要素 B_i=(b_{i1},b_{i2},…)を調整する．即ち，b_{i1}=b_{i1}+1, b_{i2}=b_{i2}+1,…とする．終了．

各レベル内の節点番号は，2.でも述べたように 0 からでなく 1 から始めるので，以上で得られた各基数組内の値は「+1」だけ調整しておく．

[例 2] m=16，基数組集合は例 1 の出力{B_1,B_2,B_4,B_5,B_6}とする．B_1=(3,3,6,5,1)，B_2(5,3,4,6,6,2,2)，B_4=(2,2,5,5,3,2,1,1)，B_5=(10,11,6,6)，B_6=(7,6,

5,4,2)である．$\{\Delta_{i,j}\}$は，$\Delta_{1,6}$が2.87で最小となり，$B_{1:6}$=(7,6,6,5,2) を作成する．$L_{1:6}$=12.8 で 15 以下なので，B_1 と B_6 をマージしたものに置き換えることができる．基数組集合は，$\{B_{1:6},B_2,B_4,B_5\}$となる．次に，$B_{1:6}$ と B_5 がマージされ$\{B_{1:5:6},B_2,B_4\}$となる．$B_{1:5:6}$=(10,11,6,6,2)である．それ以降のマージはコード化ビット長の制限を満たさなくなるのでできない．最終調整して，$B_{1:5:6}$=(10,12,7,7,3), B_2=(6,4,5,7,7,3,3,), B_4=(3,3,6,6,4,3,2,2) の基数組からなる基数組集合が得られる． □

４．基数組集合の妥当性

アルゴリズムAにより，吸収されずに残る基数組は，他の基数組に比べ１箇所以上のレベルで大きい値をとっている．レベルの最大値を h とするとき，1箇所のレベルが他の基数組より大きいものの個数が ${}_hC_1$ 以下，2 箇所のものが ${}_hC_2$ 以下などである．従って，基数組集合の大きさ N の上限は，

$$N = {}_hC_1+{}_hC_2+\ldots+{}_hC_{h-1} = 2^h - 2 \qquad (2)$$

となるが，実際のケースでははるかに少なくて済む．また，コード化ビット長 m に余裕をもたせれば，基数組のマージが加速し，更に減少するため，基数組をメモリ上の置くことは容易である．文書番号からその文書に用いられた基数組を知るには，文書番号から基数組IDへの変換テーブルが必要である．基数組IDは整数で表現されるので，文書数 d に比例するが，現在のコンピュータのメモリ上に置くことは容易であろう．また，文書番号を基数組IDの順になるように番号を振りなおすことが許されれば，各基数組を利用する文書の先頭の文書番号を持てばよいから，変換テーブルのエントリ数は，基数組集合の要素数にまで圧縮される．

コード化ビット長 m に余裕をもたせると，各レベルの基数を大きめに設定することができる．木構造の変化に対して必要となる基数組の再設定および再番号付けの可能性が下がるため，更新の頻繁なXML文書に対応し易くなると考えられる．

５．実験結果

Xmark[15]のXMLジェネレータにより予め作成されホームページ上に公開されているサイズ100Mbyte，節点総数が 1,527,590 の XML 文書(URL:<http://monetbd.cwi.nl/xml/Assets/standard.gz>)を用いて実験を行った．全体が１文書になっているため，木の３レベル目から上部を切り取り，複数個の副木に分割し，152,757 文書を得た．各文書木の最大および平均のレベルはそれぞれ 8 および 2.83 である．上部を切り取ったことにより低い木となった．コード化ビット長 m を変化しながら，アルゴリズムAにより，基数組集合を求めた．表１に，m=16,15,14,13,12 対する，基数組集合の組数とオーバフロー文書数を示す．基数組数は，h=8 から(2)式で求まる上限(N=254)に比べの一桁程度小さかった．m が小さくなるに従い，一旦は増加した組数が減少に転じているのは，オーバフロー文書が多くなる一方，これらは基数組集合からはずして集計したためである．

表１　基数組数とオーバフロー数

Table1 1 Number of radix groups and overflows

m	16	15	14	13	12
基数組数	14	24	31	32	22
オーバフロー	0	11	195	1182	2695

６．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．本稿では，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を示した．更に，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，基数組集合による番号付けの妥当性を考察し，実験結果を示した．

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