社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2007－DBS－143（17）
2007／7／2

挿入によって他ラベル変更を起こさない可変長XMLラベルの容量評価

高橋 昭裕† 梁 文新††,††† 横田 治夫†††,†

† 東京工業大学大学院情報理工学研究科計算工学専攻 〒152-8552 東京都目黒区大岡山 2-12-1
†† (独) 科学技術振興機構 〒332-0012 埼玉県川口市本町 4-1-8
††† 東京工業大学学術国際情報センター 〒152-8550 東京都目黒区大岡山 2-12-1
E-mail: {†akihiro, ††,†††wxliang}@de.cs.titech.ac.jp, †††,†yokota@cs.titech.ac.jp

あらまし RDBにXML文書を格納することにより，RDBMSの機能が利用可能になるため，XML木の各ノードにラベル付けを行い，RDBへ格納するための研究が注目されている．XML文書の更新が頻繁に起こる場合，単純なラベリング手法では大規模なラベルの付け替えが発生し，更新コストが高くなる．そこで，我々は更新時にラベルの付け替えを必要としないVLEIコードと，Dewey Orderとを組み合わせたラベル付け手法であるDO-VLEIを提案し，更新処理および問い合わせ処理の性能向上を図ってきた．ラベルの格納容量低減を企図してDO-VLEIコードをbit列に変換する圧縮bit列DO-VLEIを提案している．ここで，圧縮bit列DO-VLEIコードは可変長であるため，任意のbit列からコードを取り出す方法が必要になると考えられる．本稿では，圧縮bit列DO-VLEIを可変長コードが取り出せるように改良を行う．また，DO-VLEIと同じく更新時にラベルの付け替えを発生させないORDPATHとラベル格納に必要な容量において比較評価を行った結果を報告する．

キーワード XML，性能評価，ラベリング

Storage Consumption of Variable Length XML Labels Uninfluenced by Insertions

Akihiro TAKAHASHI†, Wenxin LIANG††,†††, and Haruo YOKOTA†††,†

† Department of Computer Science, Graduate School of Information Science and Engineering, Tokyo Institute of Technology 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552, Japan
†† Japan Science and Technology Agency 4-1-8 Honcho, Kawaguchi-shi, Saitama 332-0012, Japan
††† Global Scientific Information and Computing Center, Tokyo Institute of Technology 2-12-1 Ookayama, Meguro-ku, Tokyo 152-8550, Japan
E-mail: {†akihiro, ††,†††wxliang}@de.cs.titech.ac.jp, †††,†yokota@cs.titech.ac.jp

Abstract In recent years, the method of assigning labels to the node of XML tree is getting more attraction. Various fanctions in an RDBMS can be easily utilized by storing the labeled XML documents into the RDB. However, in simple labeling methods, a number of nodes need to be re-labeled, when the XML documents are updated frequently. To avoid this, we proposed DO-VLEI method combining VLEI code with the Dewey Order method. Compressed-bit-string DO-VLEI created by applying DO-VLEI on bit string has a variable length. In order to derive the length, we have to provide the size of code at the head of the string or a terminal symbol at the end of it. In this paper, we propose improved compressed-bit-string DO-VLEI labeles to handle the length efficiently, and compare the storage consumption of the labels with the wellknown ORDPATH.

Key words XML, performance evaluation, labeling

1. はじめに

大量のXML文書を効率よく管理するために，RDBに格納する方法が注目されている[1]．XML文書は，一組のタグをノードとして扱うことで木構造と見なすことができる[2]．図1にXML文書を木構造とみなしたXML木を示す．RDBに格納する際，XML文書の木構造情報を保持するための方法として，XML文書の各ノードにラベルを割り当て，ラベルとノード情報をタプルとして格納する方法がある．ラベルの規則性により，各ノード間の包含関係や，ノードのXML木における位

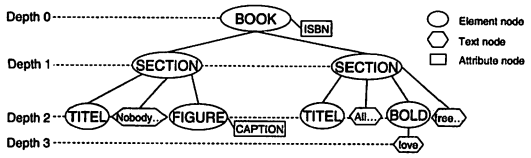

図 1 XML 木

置判定を可能にする.

ラベル付け手法として，前順後順法 [3] や Dewey Order [4] などが提案されている．しかし，それらのラベル付けに単純な連続する整数を用いると，更新時に大規模なラベルの付け替えが必要となる．そこで，更新時のラベル付け替えのコストを抑える手法として，浮動小数点数法 [5] や，範囲ラベリング法 [6]，範囲ラベリング法を改良した更新に強い節点ラベル付け手法 [2]，などが提案されている．

このようなラベル付け替えが発生する問題に対して，我々は他ノードのラベルを変更することなく無制限に挿入することができる VLEI コード (Variable Length Endless Insertable code) と，VLEI コードを用いたラベル付け手法を提案し，更新・検索処理に関する改良を行ってきた [7]〜[9]. [7] では VLEI コードを前順後順法に適用した PP-VLEI と，Dewey Order に適用した DO-VLEI を提案した. [8] では，包含関係の判定において，子孫ノードのラベルは必ず先祖ノードのラベルよりも長いという特徴を利用することで，検索処理の性能改善を行った. [9] では，ラベル格納に必要な記憶容量低減を企図し，DO-VLEI コードを bit 列で表現する，圧縮 bit 列 DO-VLEI を提案した.

ここで，圧縮 bit 列 DO-VLEI コードは可変長であるため，コードの最初に bit 列の長さ，もしくは，コードの終端を示すことで bit 列の中からコードの部分だけを取得可能にする必要がある．そこで本稿では可変長の bit 列コードを取得可能にするため，圧縮 bit 列 DO-VLEI の改良を行う．さらに，DO-VLEI と同じく XML 文書更新時にラベルの付け替えが発生しないラベル付け手法である ORDPATH [10] とラベル格納に必要な記憶容量について比較を行う.

本稿の構成は次の通りである．まず，2 節で DO-VLEI および，圧縮 bit 列 DO-VLEI について説明し，3 節では関連研究として比較実験の対象となる ORDPATH について説明する．4 節で圧縮 bit 列 DO-VLEI の性質について述べる．次に 5 節では圧縮 bit 列 DO-VLEI の改良について述べ，6 節で改良された圧縮 bit 列 DO-VLEI と ORDPATH のラベル格納に必要な記憶容量について比較実験を行う．最後に 7 節でまとめと今後の課題を述べる.

2. DO-VLEI

我々がこれまで提案してきた VLEI コードと，VLEI コードに Dewey Order を組み合わせた DO-VLEI，DO-VLEI コードを bit 列に変換する圧縮 bit 列 DO-VLEI について説明する.

2.1 VLEI コードの定義

VLEI コードは 1 から始まる 0,1 の可変長 bit 列であり，特殊な大小関係を持つ．以下に VLEI コードの定義を示す.

定義 1. VLEI コード

v を 1 から始まる {0,1} からなる bit 列とし，以下の大小関係を満足する場合に VLEI コードと呼ぶ.

$$v \cdot 0 \cdot \{0|1\}^* \prec_v v \prec_v v \cdot 1 \cdot \{0|1\}^*$$

すなわち，あるラベル v に対して，右に 0 のつくものは v より

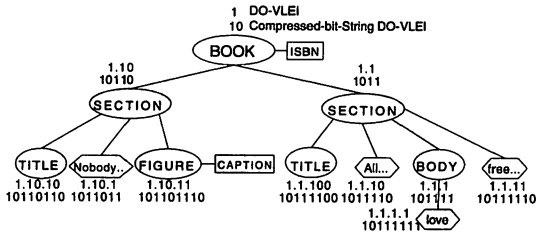

図 2 DO-VLEI コードによるラベル付け

も小さく，1 のつくものは v よりも大きい.

VLEI コードでは，任意の 2 つの VLEI コードの間に存在する新しい VLEI コードを無制限に作り出すことができるため，他ノードのラベルを付け替える必要がない．よって，更新コストの低いラベル付けが可能となる.

2.2 DO-VLEI

Dewey Order では，親ノードのラベルの末尾に，デリミタ (区切り文字) と兄弟ノード間の順序を表現するコード (兄弟コード) とを加えて，子ノードのラベルを表現する．DO-VLEI では兄弟コードに VLEI コードを用いる.

定義 2. DO-VLEI

（1） root ノードの DO-VLEI ラベルを $D_{root} = 1$ とする.

（2） 兄弟ノード間の順序を VLEI コードの大小関係で表し，これを v_{child} とする．このとき，親ノードの DO-VLEI ラベルを D_{parent} とすると，次のようになる.

$$D_{child} = D_{parent} \cdot v_{child}$$

図 2 に DO-VLEI によってラベル付けされた XML 木を示す.

2.3 圧縮 bit 列 DO-VLEI

圧縮 bit 列 DO-VLEI は DO-VLEI コードをビット列表現に変換することにより，実際に扱うラベルのサイズを小さくし，また，bit 列による操作を可能にする.

2.3.1 DO-VLEI コードの構成要素

DO-VLEI コードは「.」と「1」と「0」の三つの記号から構成される文字列である．そして，三つの記号から構成される文字列が DO-VLEI コードであるためにはいくつかの条件が存在する.

（1）「.」は連続しない.

（2）「.」がラベルの末尾にあることはない.

（3） VLEI コードは「1」で始まる.

条件 (1)-(3) から「.」の後には必ず VLEI コードの先頭の文字「1」が存在する．ゆえに，DO-VLEI コードは「.1」と「1」と「0」の三つの記号から構成されると言い換えることができる.

これら三つの記号にビット列を割り当てることにより，圧縮 bit 列 DO-VLEI コードを生成する.

2.3.2 圧縮 bit 列 DO-VLEI の定義

以下に圧縮 bit 列 DO-VLEI の定義を示す.

定義 3. 圧縮 bit 列 DO-VLEI

DO-VLEI コードにおいて，「.1」を bit 列 (11) に，「1」を bit 列 (10) に，「0」を bit 列 (0) に変換したものを，圧縮 bit 列 DO-VLEI コードと呼ぶ.

三つの記号「.1」「1」「0」それぞれに，語頭符号 [11] となる bit 列 (0)，(10)，(11) を割り当て，符号化する.

ここで，三つの記号にどの符号語を割り当てるかが問題となる．特に符号語 (0) は，1bit で表現されるため，ラベルサイズを小さくするためにはもっとも出現頻度の高い記号に割り

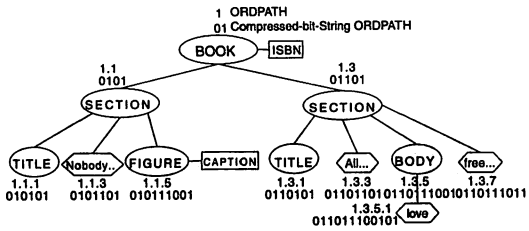

図 3 ORDPATH によるラベル付け

表 1 圧縮 bit 列 ORDPATH で用いる接頭辞スキーマ

L_i Bitstring	O_i length	O_i value range
000000001	20	[-1118485, -69910]
00000001	16	[-69909, -4374]
0000001	12	[-4373, -278]
000001	8	[-277, -22]
00001	4	[-21, -6]
0001	2	[-5, -2]
001	1	[-1, 0]
01	0	[1, 1]
10	1	[2, 3]
110	2	[4, 7]
1110	4	[8, 23]
11110	8	[24, 279]
111110	12	[280, 4375]
1111110	16	[4376, 69911]
11111110	20	[699912, 1118487]

当てる必要がある．まず，挿入が行われると兄弟コードである VLEI コードの長さが増大し，デリミタの個数が増大することは少ないため，デリミタが含まれる記号「.1」の出現率は低いと考える．ゆえに (0) に割り当てる候補として「.1」は除外する．insertVLEI アルゴリズム [9] では，比較するラベルサイズが等しいときに「0」付け加えるため，挿入時には「0」の出現率が高くなると予想する．したがって「0」に (0) を割り当てる．

(11) と (10) にラベルサイズの差異はない．しかし，連続する (1) の数を数えることで DO-VLEI コードへ変換することなくデリミタを抽出することができるよう「.1」に (11) を，「1」に (10) を割り当てる．図 2 に圧縮 bit 列 DO-VLEI でラベル付けした XML 木を示す．

3. 関連研究

DO-VLEI 同様に，他ノードのラベルを付け替えることなく無制限にノードの挿入を行うことのできるラベル付け手法として，素数を用いた手法 [12] や Dewey Order を用いた手法 [10] [13] [14] などが提案されている．本節では以下，ORDPATH について簡単に説明する．詳しくは [10] を参照されたい．

3.1 ORDPATH の概要

ORDPATH は，DO-VLEI と同様に，他のラベルを付け替えることなく挿入を無制限に行うことのできるラベル付け手法である．図 3 に ORDPATH によってラベル付けされた XML 木を示す．

3.2 ORDPATH におけるノードの挿入

ORDPATH は初期のラベル付けにおいて，正の奇数のみを用いている．これは，負の整数や偶数を，後で XML 木に挿入が行われたときに用いるためである．また兄弟順を決定する兄弟コードは必ず奇数で終了する．例えば，1.3.1 と 1.3.3 の間に新しく挿入されるノードの ORDPATH は 1.3.2 のあとに，1 を加えた，1.3.2.1 となる．

表 2 アルゴリズム： DO-VLEI コードの大小比較

アルゴリズム compearDO-VLEI(v, w)
入力: 比較される圧縮 bit 列 DO-VLEI v, 比較対象の圧縮 bit 列 DO-VLEIw (但し length(v)≦length(w))
出力: v が w より大きいかどうか

$l' = length(w) - length(v)$
$v' = v \cdot 1 \cdot \{0\}^{l'-1}$
if $v' \geqq w$
　return true
else
　return false
endif

3.3 圧縮 bit 列 ORDPATH

ORDPATH は内部では 0,1 の bit 列で表現される．これを圧縮 bit 列 ORDPATH と呼ぶこととする．表 1 のような接頭辞スキーマを用いて，整数を bit 列に変換する．一つの整数を表すために，一組の bit 列コンポーネント L_i-O_i を用いる．i はデリミタで区切られた整数のラベル中における順番である．L_i は頭から逐次解析することで決定する bit 列であり，続く O_i の bit 数と整数の範囲を接頭辞スキーマにより決定する．そして O_i は L_i によって指定される範囲の数を 2 進数によって表す．

圧縮 bit 列 ORDPATH は bit 列を接頭辞スキーマに従って区切ることでデリミタを表現し，ラベルサイズの圧縮を行っている．図 3 に，圧縮 bit 列 ORDPATH によるラベル付けの例を示す．

4. 圧縮 bit 列 DO-VLEI の性質

圧縮 bit 列 DO-VLEI における DO-VLEI コードから bit 列への変換では，DO-VLEI の構成要素を最小単位ごとに bit 列に変換を行っている．そのため，子ノードラベルが頭に親ノードラベルを含むという，Dewey Order の性質を保持している．

また，bit 列への割り当て方により，XML 文書格納時のラベルサイズやラベルを用いた問い合わせ性能に違いが出る．「0」$= v_0$，「.1」$= v_d$，「1」$= v_1$，と変換したときに，

$$1 \cdot v_0 \prec_v 1 \prec_v 1 \cdot v_d \prec_v 1 \cdot v_1 \qquad (1)$$

ならば，表 2 のアルゴリズムで大小比較を行うことにより，昇順が XML の文書順と等しくなる．証明は付録の定理 6 を参照されたい．本稿では評価を行わないが，この性質により，ラベルを用いた問い合わせの性能が向上すると考えられる．

5. 可変長に対応した圧縮 bit 列 DO-VLEI

[9] における圧縮 bit 列 DO-VLEI では可変長である bit 列コードを任意の bit 列から抽出することが出来ない．そこで，(1) DO-VLEI コードの前に bit 長を示す，もしくは，(2) DO-VLEI コードの最後に終端記号「EOL」を加えることにより，可変長の bit 列を抽出可能にする．

5.1 bit 長を示す

圧縮 bit 列 DO-VLEI コードを生成した後に，その bit 長を調べ，ラベルの先頭に bit 長を示す部分を付加する．以下，bit 長を付加した圧縮 bit 列 DO-VLEI を vlei-ABL，圧縮 bit 列 ORDPATH を ordpath-ABL とする．

bit 長を示すために圧縮 bit 列 ORDPATH の bit 列生成手法同様，一組のビット列コンポーネント L_i-O_i を用いる．今回実験で利用する接頭辞スキーマを表 3 に示す．

表 3　bit 長を示す接頭辞スキーマ

L_i Bitstring	O_i length	O_i value range
0	3	[1, 7]
100	4	[8, 23]
101	6	[24, 87]
1100	8	[88, 343]
1101	12	[344, 4439]
11100	16	[4440, 69975]
11101	20	[69976, 1118551]
11110	24	[1118552, 17895767]
11111	31	[17895768, 2165379414]

表 4　bit 列の対応

	vlei-ABL	vlei1	vlei2	vlei3	vlei4	vlei5	vlei6
0	0	00	0	0	0	00	00
1	10	11	10	110	11	1	1
.1	11	10	110	10	10	01	01
EOL		01	111	111	1111	0101	0110

5.2 終端記号

DO-VLEI コードに終端記号を付加すると，DO-VLEI コードの構成要素は，「.1」「1」「0」「EOL」の四つとなる．そこで，四つの構成要素に bit 列を割り当てることによって DO-VLEI コードを bit 列に変換する．本稿では，特に XML 文書格納時のラベルサイズに焦点を当て，bit 列の割り当て方を考える．

5.2.1 語頭符号化

XML 文書をラベル付けするときの各要素の出現頻度の関係は次のようになると考えられる．

$$「0」 > 「1」 <> 「.1」 > 「EOL」 \qquad (2)$$

「EOL」は各ラベルに対し一回のみ出現するため，他の記号に比べ，出現頻度は低い．「.1」は各ラベルに関して，(深さ) − 1 回出現する．一方，「0」「1」は各深さの兄弟コード中に複数回出現すると考えられる．また，VLEI コードはコードの大小関係のみを定義しているため，「0」「1」の VLEI コード中における出現頻度は自由に変えることが出来る．たとえば，兄弟要素が三つだった場合，VLEI コードの付け方は，$10 \prec_v 1 \prec_v 11$ や，$100 \prec_v 10 \prec_v 1$ のように複数存在する．したがって初期状態における DO-VLEI コード中の「0」「1」の出現頻度の比は調節が可能である．

式 (2) は，「0」の出現頻度を高くした場合である．式 (2) より，ハフマン符号を生成すると次の三つのパターンのいずれかとなる．

vlei1　「0」:(00),「1」:(11),「.1」:(10),「EOL」:(01)
vlei2　「0」:(0),「1」:(10),「.1」:(110),「EOL」:(111)
vlei3　「0」:(0),「1」:(110),「.1」:(10),「EOL」:(111)

5.2.2 特定出現パターンを終端とする符号化

四つの構成要素の中で「EOL」は明らかに出現頻度が低い．そこで，他の三つの構成要素を bit 列に対応させ，「EOL」は他の三つの構成要素の特定出現パターンを割り当てることを考える．具体的には，「.1.1」が出現しないようにラベル付けを行い，このパターンを「EOL」とする．こうすることによって，「EOL」にくらべ頻出する他の三つの構成要素を，短い bit 列で表現することが可能となる．

vlei4　「0」:(0),「1」:(11),「.1」:(10),「EOL」:「.1.1」:(1010)
vlei5　「0」:(00),「1」:(1),「.1」:(01),「EOL」:「.1.1」:(0101)
vlei6　「0」:(00),「1」:(1),「.1」:(01),「EOL」:「.11(0)」:(0110)

表 4 は，構成要素と bit 列の対応をまとめた表である．vlei1,3,4 は式 (1) を満たしているため表 2 のアルゴリズムによる圧縮 bit 列 DO-VLEI の大小比較が可能である．

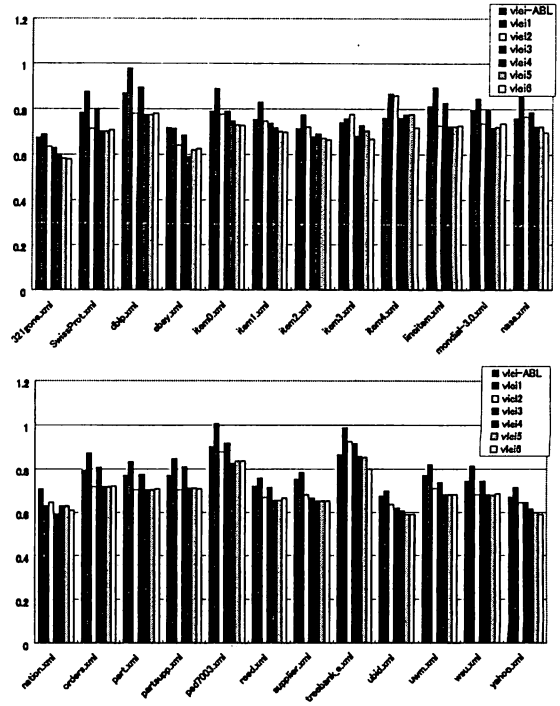

図 4　ordpath-ABL を 1 としたときの各手法のラベルサイズ比

表 6　各手法の平均値

vlei-ABL	vlei1	viel2	vlei3	vlei4	vlei5	vlei6
0.7638	0.8221	0.7289	0.7505	0.7015	0.6979	0.6911

6. 実　　験

表 4 で示した圧縮 bit 列 DO-VLEI と圧縮 bit 列 ORDPATH を用いて，実際の XML 文書にラベル付けを行い，ラベルサイズの総和を求め，比較を行った．実験には，XML データベースのベンチマークとして知られる XMark [15] 用の XML 文書作成ツールである xmlgen [16] によって作成した XML 文書と，XMLData Repository [17] の XML 文書を利用した．実際に実験に使用した XML 文書の情報を表 5 に示す．

また，圧縮 bit 列 DO-VLEI 同様，圧縮 bit 列 ORDPATH も可変長の bit 列であるため，実験では，圧縮 bit 列 ORDPATH を生成した後，bit 長を付加した．

DO-VLEI コードの兄弟数分の VLEI コードを生成する際，VLEI コードの bit 数の総和が最小になるようにした．例えば vlei1 で兄弟数 5 個分の VLEI コードを生成することを考える．vlei1 は「0」「1」に対応する bit 列が同じ長さなので，VLEI コードの長さが bit 列の長さに対応するため，$100 \prec_v 10 \prec_v 101 \prec_v 1 \prec_v 11$ となる．最長 bit 数 (この場合は 100 と 101) となる VLEI コードは他に 110 と 111 が存在するが，小さい VLEI コードを優先的に利用した．

6.1 実験結果

実験結果を図 4 に示す．縦軸は圧縮 bit 列 ORDPATH のラベルサイズを 1 としたときの各圧縮 bit 列 DO-VLEI のラベルサイズ比率である．表 6 は各手法の値の平均値である．

表 6 より，ラベルサイズが特定出現パターンを終端とした手法 (vlei4-6) で，ordpath-ABL と比べ，約 30% 小さい．ordpath-ABL と同様，ラベル長を付加した vlei-ABL でも約 24% 小さいことが分かる．また，ラベル長を付加するよりも終端記号を

表 5 XML 文書一覧

	size	elements	max-depth	avr-depth	max-fanout	avr-fanout
321gone.xml	24479	311	5	3.765273	98	2.988142
SwissProt.xml	114820211	2977031	5	3.556711	50000	2.034511
dblp.xml	133862772	3332130	6	2.902279	328858	1.943555
ebay.xml	35525	156	5	3.75641	380	5.391305
item0.xml(xmlgen SF=1)	118552713	1666315	12	5.547796	25500	2.10692
item1.xml(xmlgen SF=0.1)	11875066	167865	12	5.548244	2550	2.103751
item2.xml(xmlgen SF=0.01)	1182547	17132	12	5.506012	255	2.102092
item3.xml(xmlgen SF=0.001)	118274	1729	12	5.717756	25	2.04965
item4.xml(xmlgen SF=0.0001)	34455	396	12	6.078283	15	2.015152
lineitem.xml	32295475	1022976	3	2.941175	60175	1.941175
mondial-3.0.xml	1784862	22423	5	3.59274	955	2.386347
nasa.xml	25050288	476646	8	5.583141	2435	2.000728
nation.xml	4584	126	3	2.785714	25	1.785714
orders.xml	5378845	150001	3	2.899987	15000	1.899987
part.xml	618181	20001	3	2.899905	2000	1.899905
partsupp.xml	2241868	48001	3	2.833295	8000	1.833295
psd7003.xml	716853012	21305818	7	5.15147	262526	1.818513
reed.xml	283619	10546	4	3.199791	703	1.809579
supplier.xml	29250	801	3	2.872659	100	1.872659
treebank_e.xml	86082517	2437666	36	7.872788	56384	1.571223
ubid.xml	20283	342	5	3.766082	62	4.176471
uwm.xml	2337522	66729	5	3.952435	2112	1.952276
wsu.xml	1647864	74557	4	3.157866	3924	2.077534
yahoo.xml	25384	342	5	3.766082	41	2.682432

付加したほうがラベルサイズを小さく出来る.

vlei2 と vlei3 を比較したときに vlei2 の方が良い結果が出ている. これは,「1」と「.1」でどちらが出現頻度が高いかを示しており, vlei2 が良いことで,「1」の方が出現頻度が高いことが分かる.

7. おわりに

本稿では, 圧縮 bit 列 DO-VLEI において可変長 bit 列を取得できるよう改良を行い, バルクロード時のラベル格納に必要な記憶容量について圧縮 bit 列 ORDPATH と比較を行った. その結果, ラベル長を付加した圧縮 bit 列 ORDPATH に比べ必要な記憶容量の平均は, ラベル長を付加した圧縮 bit 列 DO-VLEI で約 76%, 終端記号を加える方法では約 70% になることを示した.

今後の課題として, また, 4. 節で述べた圧縮 bit 列 DO-VLEI の性質を基にした親子判定や深さの判定など, ラベルを用いた操作にかかる時間の評価を行う必要がある.

謝辞

本研究の一部は, 文部科学省科学研究費補助金特定領域研究 (19024028), 独立行政法人科学技術振興機構 CREST, および東京工業大学 21 世紀 COE プログラム「大規模知識資源の体系化と活用基盤構築」の助成により行われた.

文献

[1] Igor Tatarinov, Stratis Viglasand Kevin S. Beyer, Jayavel Shanmugasundaram, Eugene J. Shekita, and Chun Zhang. Storing and Querying Ordered XML using a Relational Database System. In *Proc. of SIGMOD Conf.*, pp. 204–215, 2002.

[2] 江田毅晴 , 天笠俊之 , 吉川正俊 , 植村俊亮. XML 木のための更新に強い節点ラベル付け手法. In *DBSJ Letters Vol.1*, 2002.

[3] Paul F. Dietz. Maintaining Order in a Linked List. In *Proc. of the 14th annual ACM symposium on Theory of computing*, pp. 122–127, 1982.

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

[5] Toshiyuki Amagasa, Masatoshi Yoshikawa, and Shunsuke Uemura. QRS: A Robust Numbering Scheme for XML Documents. In *ICDE*, pp. 705–707, 2002.

[6] Edith Cohen, Haim Kaplan, and Tova Milo. Labeling Dynamic XML Trees. In *Proc. of the 21st ACM SIGMOD-SIGACT-SIGART*, pp. 271–281, 2002.

[7] Kazuhito Kobayashi, Wenxin Liang, Dai Kobayashi, Akitsugu Watanabe, and Haruo Yokota. VLEI code: An Efficient Labeling Method for Handling XML Documents in an RDB. In *ICDE*, 2005.

[8] 長良香子 , 小林一仁 , 小林大 , 横田治夫. XML データベースのラベル付け手法 VLEI の評価. 第 16 回電子情報通信学会データ工学ワークショップ (DEWS2005) 論文集, 2005.

[9] 村上翔一, 小林大, 横田治夫. DO-VLEI を用いた XML 格納におけるラベルサイズと問い合わせ性能. 第 17 回電子情報通信学会データ工学ワークショップ (DEWS2006) 論文集, 2006.

[10] P. O'Neil, E. O'Neil, S. Pal, I. Cseri, G. Schaller, and N. Westbury. ORDPATHs: Insert-friendly XML Node Labels. In *Proc. of the ACM SIGMOD International Conference on Management of Data 2004*, pp. 903–908. ACM Press New York, NY, USA, 2004.

[11] 韓 太舜 , 小林欣吾 , 金谷文夫 , 山本博資 , 横尾英俊 , 鈴木譲 , 森田啓義 , 佐藤創 , 伊藤秀一. 情報源符号化 無歪みデータ圧縮. 情報理論とその応用シリーズ 1-1. 培風館, 1998.

[12] Wynne Hsu Xiaodong Wu and, Mong Li Lee and. A Prime Number Labeling Schema for Dynamic Ordered XML Trees. In *ICDE*, pp. 66–78, 2004.

[13] Michael Peter Haustein, Theo Härder, Christian Mathis, and Markus Wagner. Deweyids - The Key to Fine-Grained Management of XML Documents. In *SBBD*, pp. 85–99, 2005.

[14] C. Li and T.W. Ling. QED: a Novel Quaternary Encoding to Completely Avoid Re-labeling in XML Updates. In *Proc. of the 14th ACM international conference on Information and knowledge management*, pp. 501–508. ACM Press New York, NY, USA, 2005.

[15] Albrecht Schmidt, Florian Waas, Martin Kersten, and Daniela Florescu. The XML Benchmark Project. Technical report, Technical Report INS-R0103, `http://monetdb.cwi.nl/xml/index.html`, April 2001.

[16] xmlgen. http://monetdb.cwi.nl/xml/downloads.html.

[17] XML Data Repository. `http://www.cs.washington.edu/research/xmldatasets/`.

付録：DO-VLEIと数値のマッピング

圧縮bit列DO-VLEIコードのbit列を整数として比較する方法に関する定理を示す．定理1-4で，VLEIコードを二進数としてみたときの大小比較方法についての定理を示し，定理5，6で，圧縮bit列DO-VLEIコードを二進数としてみたときの大小比較に関する定理を示す．本節では，$l(v)$(v:VLEIコード)をVLEIコードの長さとする．

[定理1] $v \prec_v w \cap l(v) = l(w) \Rightarrow v \cdot \{0|1\}^* \prec_v w \cdot \{0|1\}^*$

証明 $v \prec_v w \cap l(v) = l(w)$ より，以下を満たす $h = 1 \cdot \{0|1\}^*$

$$v = h \cdot 0 \cdot \{0|1\}^{(l(v)-l(h)-1)} \quad w = h \cdot 1 \cdot \{0|1\}^{(l(v)-l(h)-1)}$$

このとき，$v \cdot \{0|1\}^*$，$w \cdot \{0|1\}^*$ を考えると，定理より

$$v \cdot \{0|1\}^* = h \cdot 0 \cdot \{0|1\}^* \prec_v h \cdot 1 \cdot \{0|1\}^* = w \cdot \{0|1\}^*$$

である．よって，定理1は証明された．

[定義1] VLEIコードの集合を V，自然数の集合を N とし，$v = b_{l-1} \cdot b_{l-2} \cdots b_0$ としたとき，写像 $f : V \rightarrow N$ を次のように定義する．

$$f(v) = \sum_{i=0}^{l-1} b_i \times 2^i$$

この式はVLEIコード，すなわち1から始まるbit列を2進表記と読み替える操作であり，写像 f が全単射である事は自明である．以下，整数は二進表記を用いて示す．

[定理2] $v \prec_v w \Leftrightarrow f(v) < f(w) \quad (l(v) = l(w))$

証明 $V^l = \{v | v = 1 \cdot \{0|1\}^{l-1}\}$ とする．

（1） $l = 2$ の時

$V^2 = \{10, 11\}$ で，VLEIコードの定義より，$10 \prec_v 11$．また，$f(10) = 10 < 11 = f(11)$ であるため，定理2は成り立つ．

（2） $l = l + 1$ のとき

$V^l = \{v_0^l, v_1^l, \cdots, v_n^l\}$ のとき，定理2が成り立つと仮定すると，

$$v_0^l \prec_v v_1^l \prec_v v_2^l \prec_v \cdots \prec_v v_n^l \tag{3}$$
$$f(v_0^l) < f(v_1^l) < f(v_2^l) < \cdots < f(v_n^l) \tag{4}$$

となる．$v_i^l \cdot 0 = v_{2i}^{l+1}$，$v_i^l \cdot 1 = v_{2i+1}^{l+1}$ とすると，$V^{l+1} = \{v_0^{l+1}, v_1^{l+1}, \cdots, v_{2n+1}^{l+1}\}$ となる．式(3)，定理1とVLEIの定義より，

$$v_{2\cdot 0}^{l+1} \prec_v v_0^l \prec_v v_{2\cdot 0+1}^{l+1} \prec_v v_{2\cdot 1}^{l+1} \prec_v v_1^l \prec_v v_{2\cdot 1+1}^{l+1} \prec_v \cdots \prec_v v_{2n}^{l+1} \prec_v v_n^l \prec_v v_{2n+1}^{l+1} \tag{5}$$

$$v_0^{l+1} \prec_v v_1^{l+1} \prec_v \cdots \prec_v v_{2n}^{l+1} \prec_v v_{2n+1}^{l+1} \tag{6}$$

一方，$f(v_i^l \cdot 0) = 2f(v_i^l)$, $f(v_i^l \cdot 1) = 2f(v_i^l) + 1$ より，

$$f(v_i^l \cdot 0) < f(v_i^l \cdot 1) \tag{7}$$

である．また，$f(v_i^l)$ は整数なので，$f(v_i^l) + 1 \leqq f(v_{i+1}^l)$ であるため，

$$f(v_i^l \cdot 1) = 2f(v_i^l) + 1 < 2(f(v_i^l) + 1) \leqq 2f(v_{i+1}^l) = f(v_{i+1}^l \cdot 0)$$
$$f(v_i^l \cdot 1) < f(v_{i+1}^l \cdot 0) \tag{8}$$

したがって，式(7)，(8)より，

$$f(v_0^{l+1}) < f(v_1^{l+1}) < f(v_2^{l+1}) < \cdots < f(v_{2n}^{l+1}) < f(v_{2n+1}^{l+1}) \tag{9}$$

である．式(6)，(9)より，$l = l + 1$ の時も定理2は成り立つ．

よって定理2は成り立つ．

[定義2] $\mathrm{next}(v, l) = \min(\{v' | v \prec_v v', l(v') = l\})(l > l(v))$

[定理3] $\mathrm{next}(v, l) = v \cdot 1 \cdot \{0\}^{l-l(v)-1}$

証明 $v \prec_v w \prec_v v \cdot 1 \cdot \{0\}^{l-l(v)-1}$ となる $w(l(w) = l)$ が存在すると仮定する．

定理2より，$f(w) < f(v \cdot 1 \cdot \{0\}^{l-l(v)-1})$ である．$f(w)$ は自然数であるため，$f(w) \leqq (f(v \cdot 1 \cdot \{0\}^{l-l(v)-1}) - 1)$ であり，

$$f(v \cdot 1 \cdot \{0\}^{l-l(v)-1}) - 1 = f(v) \times 2^{(l-l(v))} + 2^{(l-l(v)-1)} - 1$$
$$= f(v) \times 2^{(l-l(v))} + \sum_{i=0}^{l-l(v)-2} 2^i = f(v \cdot 0 \cdot \{1\}^{l-l(v)-1})$$

よって，$f(w) \leqq f(v \cdot 0 \cdot \{1\}^{l-l(v)-1})$ となる．ここで，VLEIコードの定義より $v \cdot 0 \cdot \{1\}^{l-l(v)-1} \prec_v v$ である．したがって，

$$w \preceq_v v \cdot 0 \cdot \{1\}^{l-l(v)-1} \prec_v v$$

のどちらかとなり，仮定と矛盾する．

したがって，仮定が誤りであり，定理3は証明された．

[定理4] $v, w \in V(l(v) < l(w))$ があるとき，$f(\mathrm{next}(v, l(w))) \leqq f(w) \Rightarrow v \prec_v w$，$f(\mathrm{next}(v, l(w))) > f(w) \Rightarrow v \succ_v w$ である．

証明 定理3より，$\mathrm{next}(v, l(w))$ は w と同じ長さのVLEIコードの v より大きいコードの中で最も小さいVLEIコードである．また，定理2より，$f(\mathrm{next}(v, l(w)))$ と $f(w)$ を数値比較した結果はVLEIコードによる比較結果と等しい．よって定理が証明された．

[定理5] DO-VLEIコードに以下の大小関係を導入すると，DO-VLEIラベルの昇順は文書順に等しくなる．

$$v \cdot 0 \cdot \{0|.|1\}^* \prec_{dv} v \prec_{dv} v \cdot . \cdot \{0|.|1\}^* \prec_{dv} v \cdot 1 \cdot \{0|.|1\}^* \tag{10}$$

証明 文書順はXML木を順序木としてみたときのpreorderによる順序に相当する．文書順であるための必要十分条件は次の式で表される．

$$\mathrm{prevSibling}(v) \prec_{dv} v \prec_{dv} \mathrm{descendant}(v) \prec_{dv} \mathrm{nextSibling}(v) \tag{11}$$

（1） $\mathrm{prevSibling}(v) \prec_{dv} v$

右辺，左辺両方同じ親を持つので，親のラベルを D_{parent} とすると，DO-VLEIの定義より，

$$\mathrm{prevSibling}(v) = D_{parent} \cdots C_{prev}$$
$$v = D_{parent} \cdots C_v$$

であり，$C_{prev} \prec_v C_v$ である．式(10)より，$v \prec_v w \Rightarrow v \prec_{dv} w$ が成り立つので，$\mathrm{prevSibling}(v) \prec_{dv} v$ が成り立つ．

（2） $v \prec_{dv} \mathrm{descendant}(v)$

DO-VLEIの定義より，$\mathrm{descendant}(v) = v \cdot . \cdot 1 \cdot \{0|.|1\}^*$ である．したがって式(10)より，$v \prec_{dv} \mathrm{descendant}(v)$ が証明された．

（3） $\mathrm{descendant}(v) \prec_{dv} \mathrm{nextSibling}(v)$

上の証明より，$v \prec_{dv} \mathrm{nextSibling}(v)$ である．これは，次の三つのパターンに分けることが出来る．以下 $\mathrm{nextSibling}(v) = w$ とする．

（a） $v = h \cdot 0 \cdot v' \prec_{dv} h \cdot 1 \cdot w' = w$ のとき

$\mathrm{descendant}(v) = h \cdot 0 \cdot v' \cdot . \cdot 1 \cdot \{0|.|1\}$ より $\mathrm{descendant}(v) \prec_{dv} w$ である．

（b） $v \prec_{dv} v \cdot 1 \cdot w' = w$ のとき

$\mathrm{descendant}(v) = v \cdot . \cdot 1 \cdot \{0|.|1\}$ より $\mathrm{descendant}(v) \prec_{dv} w$ である．

（c） $v = w \cdot 0 \cdot v' \prec_{dv} w$ のとき

$\mathrm{descendant}(v) = w \cdot 0 \cdot v' \cdot . \cdot 1 \cdot \{0|.|1\}$ より $\mathrm{descendant}(v) \prec_{dv} w$ である．

よって，$\mathrm{descendan}(v) \prec_{dv} \mathrm{nextSibling}(v)$ が証明された．

ゆえに，式(11)が成り立つため，定理5は証明された．

[定理6] 「0」$= v_0$，「.1」$= v_d$，「1」$= v_1$ $(1 \cdot v_0 \prec_v 1 \prec_v 1 \cdot v_d \prec_v 1 \cdot v_1)$ とする圧縮bit列DO-VLEIコードをVLEIコードとしてみたときの大小関係はそれに対応するDO-VLEIコードの大小関係と等しくなる．

証明 DO-VLEIの性質より，「.」の後は必ず「1」なので，式(10)は次のように書き換えることが出来る．

$$v \cdot 0 \cdot \{0|.1|1\}^* \prec_{dv} v \prec_{dv} v \cdot .1 \cdot \{0|.1|1\}^* \prec_{dv} v \cdot 1 \cdot \{0|.1|1\}^*$$

これを圧縮bit列DO-VLEIの表記に変換すると，

$v \cdot v_0 \cdot \{v_0|v_d|v_1\}^* \prec_{dv} v \prec_{dv} v \cdot v_d \cdot \{v_0|v_d|v_1\}^* \prec_{dv} v \cdot v_1 \cdot \{v_0|v_d|v_1\}^*$

となる．圧縮bit列DO-VLEIコードは「1」から始まるbit列であるため，VLEIコードとして比較することができ，条件より

$v \cdot v_0 \cdot \{v_0|v_d|v_1\}^* \prec_v v \prec_v v \cdot v_d \cdot \{v_0|v_d|v_1\}^* \prec_v v \cdot v_1 \cdot \{v_0|v_d|v_1\}^*$

と書き換えることが出来る．したがって，DO-VLEIの大小関係は圧縮bit列DO-VLEIコードをVLEIコードとしてみたときの大小関係と同じである．ゆえに，定理6は証明された．