XMLデータベースのラベル付け手法VLEIの評価

長良　香子† 　小林　一仁†† 　小林　大†† 　横田　治夫†††,††

† 東京工業大学 工学部 情報工学科
†† 東京工業大学 大学院 情報理工学研究科 計算工学専攻
††† 東京工業大学 学術国際情報センター　〒 152–8552 東京都目黒区大岡山 2–12–1
E-mail: †{nagara,kkobayashi,daik}@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で RDBMS の様々な機能が容易に利用可能となるため，XML の各ノードへラベル付けして RDB へ格納する手法が注目されている．しかし，単純な数値を用いたラベル付けでは更新に伴って大規模なラベルの付け変えが必要となることから，更新のコストが高くなる．そこで，我々は更新コストを抑えるために，挿入制限のない VLEI コードとそのラベル付け手法への適用を提案し，簡単な検索と挿入の実験を通してその有効性を示してきた．しかし，これまでの提案手法では，包含関係の判定においてパターンマッチを行う必要があることなどから，検索性能に関して改善の余地があった．本稿では，XML ラベル付け手法として VLEI コードを用いた Dewey Order を前提にして，包含関係の判定に Dewey Order の特徴を活かした手法を提案する．さらに，XML データベースの評価で良く用いられるベンチマークを用いた評価実験を行う．その結果，提案手法が特定の検索に関しては従来の手法に比べて極めて高い性能改善を示すことを報告する．

キーワード　XML，データベース，性能評価，問い合わせ処理，インデックス

Evaluation of XML Labeling Methods using VLEI Code

Kyoko NAGARA†, Kazuhito KOBAYASHI††, Dai KOBAYASHI††, and Haruo YOKOTA†††,††

† Department of Computer Science in Tokyo Institute of Technology
†† Department of Computer Science Graduate School of Information Science and Engineering in Tokyo Institute of Technology
††† Global Scientific Information & Computing Center in Tokyo Institute of Technology
Ookayama 2–12–1, Meguro-ku, Tokyo, 152–8552 Japan
E-mail: †{nagara,kkobayashi,daik}@de.cs.titech.ac.jp, ††yokota@cs.titech.ac.jp

Abstract　When XML documents are stored into an RDB, many useful functions of an RDBMS can be used. Therefore labeling methods for XML nodes to store them into an RDB attract attentions. However, since labels using straightforward numerical values incur numerous relabeling, costs for update for them become very high. Therefore, we have proposed the Variable Length Endless Insertabel (VLEI) code for XML labeling to reduce the update costs. We also proposed methods for applying VLEI code to XML labeling, and evaluated its effectiveness by simple experimentation of retrieval and update. However, the previous methods required pattern match operations to find the ancestor-decedent relationship, which could be improved. In this paper, we propose new methods to find the relationship using the combination of VLEI code and Dewey Order by utilizing the characteristics of Dewey Order. The experimental results indicate that the proposed methods quite improve the retrieval performance.

Key words　XML, Database, Performance Evaluation, Query Processing, Index

1. はじめに

最近，様々なデータが XML で記述されることが多くなって来ている．そのような大量の XML 文書を効率良く管理し，高速に検索するために，XML 文書を RDB に格納する方法が注目されている [1]．RDB に格納することにより，検索機能や同時実行制御、障害処理など現在の RDBMS が持つ様々な機能を容易に XML 文書の管理に利用することができる．

XML 文書はタグによって表現されたノード間に包含関係があり，木構造と見なすことができる [2]．RDB に XML 文書を格

納する場合に，この木構造を認識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 文書中の位置を判定することができる．

木構造とその要素の包含関係をラベル付けする手法として，従来から，前順後順法 [3] や Dewey Order 法 [4] が提案されている．これらの手法を XML のノードに適用することで，XML 文書の木構造を表現することが可能となる．しかし，それらのラベル付けに単純な整数を用いたのでは，更新に伴って大規模なラベルの付け変えが必要となる．このため，更新時のラベル付け変えのコストを抑える手法として，範囲ラベリング法 [5]、ラベルに浮動小数点数を用いる方法 [6]，予め挿入に備えて数値間に間隔を持たせたラベルを割り当てる方法 [2] などが提案されている．しかし，これらの手法では無制限に挿入を続けることは不可能で，予め用意された許容範囲を越えた挿入後にはやはり大規模なラベルの付け変えが必要となる．

我々は，他の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 (Variable Length Endless Insertable) コードと，VLEI コードを前述の前順後順法と Dewey Order 法に適用したラベル付け手法を提案した [7]．また，簡単な検索と挿入の実験によって，VLEI コードを用いたラベル付け手法と他手法との比較を行い，VLEI コードを用いたラベル付け手法の有効性を示した．しかし，これまでの手法では，包含関係の判定においてパターンマッチを行う必要があることなどから，検索性能に関して改善の余地があった．

本稿では，XML ラベル付け手法として VLEI コードを用いた Dewey Order 法を前提にして，包含関係の判定に Dewey Order の特徴を活かした包含関係の判定手法を提案する．さらに，XML データベースの評価で良く用いられるベンチマークである Michigan Benchmark [8] と XMark [9] を用いた評価実験を行い，提案手法が特定の検索に関しては従来の手法に比べて極めて高い性能改善を示すことを報告する．

本稿の構成は次の通りである．まず 2. 節では，XML 文書のラベル付けの概要と VLEI コードについて述べる．次に，3. 節において，Dewey Order 法を前提に包含関係の判定の改善手法を提案する．更に，4. 節で Michigan Benchmark と XMark の 2 つのベンチマークを用いた評価実験の結果について報告し，5. 節でまとめと今後の課題について述べる．

2. XML 文書ラベル付けと VLEI コード

前提知識として XML 文書のラベル付けの手法と VLEI コード，および VLEI コードをラベル付けに適用する手法に関して，その概要を述べる．詳細については [7] を参照されたい．

2.1 XML 文書の木構造の表現

XML 文書はタグによってノード毎に表現され，ノード間には包含関係があって，木構造と見なすことができる．RDB に XML 文書を格納する場合，XML の各ノードにラベルを割り当て，ラベルとノードをタプルとして格納することによって，この木構造を認識することができる．木構造とその要素の包含関係をラベル付けする手法として，従来から，前順後順法 [3] や Dewey Order 法 [4] が提案されている．

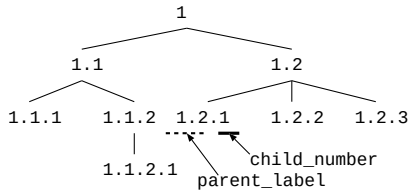


図 1　木に Dewey Order のラベルを付けた様子

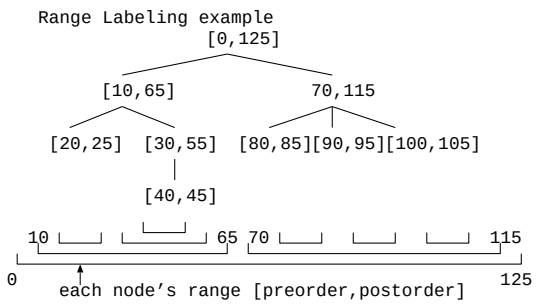


図 2　範囲ラベリング法の一例

前順後順法は，各ノードに前順 (preorder) と後順 (postorder) の 2 種類の値の組を割当てるラベル付け手法である．2 つの番号の大小関係によって，ノード間の包含関係を判定することができる．

例えば，ある XML 文書中でタグが $\langle A\rangle\cdots\langle B\rangle\cdots\langle\backslash B\rangle\cdots\langle\backslash A\rangle$ のようになっていた場合，この XML 文書に対応する木構造ではノード A がノード B の先祖であり、それぞれのノードのラベル間には以下の大小関係が成立する．

A の前順 $<$ B の前順 AND A の後順 $>$ B の後順

一方，Dewey Order は図 1 のように，親ノードのラベルの後ろに何らかのデリミタ（多くは “.”）を挟んで兄弟間の順序を示すコードを追加していくラベル付け手法である．

前順後順法も Dewey Order 法も，XML 文書の木構造を表現することが可能であるが，それらのラベル付けに単純な整数を用いたのでは，更新に伴って大規模なラベルの付け変えが必要となる．このため，更新時のラベル付け変えのコストを抑える手法として，範囲ラベリング法 (図 2 参照) [5]、ラベルに浮動小数点数を用いる方法 [6]，あらかじめ挿入に備えて数値間に間隔を持たせたラベルを割当てる方法 [2] などが提案されている．なお，これらの手法は全て前順後順法を前提としている．

しかし，これらの手法では無制限に挿入を続けることは不可能で，予め用意された許容範囲を越えた挿入後にはやはり大規模なラベルの付け変えが必要となる．そこで我々は，他の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 (Variable Length Endless Insertable) コードと，VLEI コードを前述の前順後順法と Dewey Order 法に適用したラベル付け手法を提案している [7].

2.2 VLEI コードの概要

まず，以下に VLEI コードの定義を示す．

アルゴリズム InsertVLEI(v_l, v_r)
入力: 挿入する要素の左側のコード v_l, 右側のコード v_r (但し $v_l < v_r$)
出力: 挿入された要素のコード v_i

if $length(v_l) \leqq length(v_r)$
$v_i = v_r \cdot 0$
else
$v_i = v_l \cdot 1$
endif
return v_i

図 3　アルゴリズム： 挿入要素のコードの決定

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．

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

すなわち，あるラベル v に対して，右に 0 のつくものは v よりも小さく，1 のつくものは v よりも大きい．

任意の $v_l < v_r$ なる VLEI コードの組みに対し，図 3 に示すアルゴリズムは，v_l と v_r の間を表す VLEI コードを容易に無制限に作成することができる．つまり，任意の大小関係をもつ $v_l < v_r$ に対して，$v_l < v < v_r$ となるコード v が存在する（証明は [7] 参照）．これは，木構造の挿入要素のコードが容易に求まり，挿入されるノード以外のコードを変更する必要がないことを示している．

このため，VLEI コードを前順後順法や Dewey Order 法に適用することによって，他のノートのラベルの付け変えが不要な更新コストの低いラベル付けが可能となる．

2.3　VLEI ラベル

以降，XML 文書の各ノードに割り当て，ノード間の包含関係を表現するラベルに VLEI コードを用いたものを VLEI ラベルと呼ぶことにする．更に，VLEI コードは，前順後順 (Preorder-Postorder) 法と Dewey Order 法の両方に適用できるので，それぞれ PP-VLEI ラベル，DO-VLEI ラベルと呼ぶことにする．

2.3.1　PP-VLEI ラベル

PP-VLEI ラベルは，前順と後順の値を VLEI コードで表現したものである．挿入の場合には，前後のノードの前順の間に入る VLEI コードを前順の値とし，後順の間に入る VLEI コードを後順の値とすればよい．

2.3.2　DO-VLEI ラベル

Dewey Order 法では，デリミタを用いてノードのひとつ下の子ノードをサフィックスとして表現する (図 1)．DO-VLEI ラベルでは，兄弟ノード間の順序の表現に VLEI コードを用いる．[7] では，格納効率を上げるため，VLEI コードを 8 進数に変換し，デリミタに 9 を用いて，全体を 10 進数で表現する Suffixed VLEI コードを提案した．本稿では，これを DO-VLEI ラベルとして用いる．

定義 2. DO-VLEI ラベル

root ノードの DO-VLEI ラベルを $L_{root} = 19$ とする．兄弟間の順序を VLEI コードの大小関係で表し，これを 8 進数に変換したものを C_{child} とする．このとき，親ノードの DO-VLEI ラベルを L_{parent} とすると，

$$\text{DO-VLEI ラベル} = L_{parent} \cdot C_{child} \cdot 9$$

例えば，1.111.101.1111 は 197959179 になる．この DO-VLEI ラベルでは，先祖の DO-VLEI ラベルで始まる DO-VLEI ラベルを探すことで子孫を求めることができる．また，DO-VLEI ラベルの右から (n+1) 個目の 9 から右の桁数を取り除けば，n 番目の先祖ラベルがわかる (n=1 のときは親ノードのラベルとなる)．

2.4　VLEI ラベルの性能

[7] では，VLEI ラベルに関し，要素ノード数が 6×10^3 程度の XML 文書を用いて，3 つの簡単な Path の検索処理時間の測定と，要素への子ノード挿入処理に要する時間を測定した．比較の対象として，以下の 4 種類のラベル付け手法について測定した．

- PP-VLEI ラベル
- DO-VLEI ラベル
- QRS: 浮動小数点数を用いた範囲ラベル
  前順後順の値を浮動小数点で表現
- SPARSE: 間隔をあけた整数を用いた範囲ラベル
  前順後順の値を予め間隔をあけて整数で表現

RDBMS としては PostgreSQL を用い，先祖子孫判定は DO-VLEI ラベルでは，文字列に変換して DO-VLEI ラベル同士のパターンマッチ，DO-VLEI ラベル以外では数値の大小比較により行った．DO-VLEI ラベル同士のパターンマッチでは，

’子孫ラベル’ LIKE ’先祖ラベル’ ‖ ’残りの文字列’

を調べる必要がある．ここで，LIKE は左の文字列が右のパターンにマッチしたときに真を返し，‖ は文字列の結合を表す．実際には，PostgreSQL で任意の文字列（空文字でない）とマッチする%というパターンを用いて，以下のように記述した．

ノード A がノード B の木構造における先祖であるとき，
「‘A ラベル’ LIKE ‘B ラベル’ ‖ ‘%’」が真となる．

実験の結果，検索処理においてはそれぞれの手法の間に大きな差は無かったが，挿入処理においては VLEI コードを用いた Dewey Order ラベル付け手法が最も良い性能を示した．しかし，データのサイズ的には必ずしも十分な実験ではなかった．

そこで，XML 文書の大きさと構造や問い合わせの内容による影響を含めて，VLEI コードの詳しい性能分析を行なうため，XML の評価で良く用いられるベンチマークである XMark [9] と Michigan Benchmark [8] を用いた実験を行った．なお，VLEI ラベルとしては DO-VLEI ラベルのみを対象とした．その結果，検索処理において DO-VLEI ラベルは他のラベル付け手法と比較して十分な性能を示すことができなかった．これは，数値として格納した DO-VLEI ラベルを判定のために文字列に変換していること，および LIKE を用いたパターンマッチングを行っているためであると思われる．また，逆に Dewey Order の特長

を活かしきれていないこともその原因と思われる．

以降，検索処理において Dewey Order 法の特長を活かして、条件判定を効率良く行なう方法を検討する．

3. 検索処理の改善

以下，2 つの手法を用いて DO-VLEI ラベルの問合せ処理の効率化を図る．1 つは，DO-VLEI ラベルを文字列に変換することなく数値として扱う方法であり，もう 1 つは，文字列を用いて格納し，パターンマッチングを使用せずに文字列操作により判定する方法である．

これらの 2 つの手法では，Dewey Order の，ラベルの一部から先祖のラベルを判定可能であるという特長を利用している．つまり，ラベル間の先祖子孫関係を判定するのではなく，与えられた条件から子孫ノードのラベルの候補を求め，求めた子孫ノードの先祖のラベルを取り出し，先祖ノードの候補を求める方法である．これまでの手法と異なる点は，一つ一つのラベルを比較して先祖子孫関係があるかを調べるのではなく，子孫ラベルから求めた先祖ラベルの候補と，先祖条件から求めた先祖ラベルが一致するかを判定するという点である．

3.1 アプローチの概要

以下に本改善手法による先祖子孫選択の流れを示す．

条件 B を満たす子孫を持つ，条件 A を満たす先祖を求める場合．

（1） 先祖条件 A から，先祖候補ノードのラベルの長さ $A_{length}\{l_1, l_2, ...\}$ を求める．

（2） 子孫条件 B を満たすラベルを取り出す $B_{label}\{b_1, b_2, ...\}$

（3） (2) (1) で求めた B_{label} と A_{length} から先祖候補ラベルを作る．先祖候補ラベル＝ B_{label} のそれぞれのラベルから A_{length} の長さを取り出したものとする．

（4） 先祖条件 A を満たすノードのうち，ラベルが (3) で求めた先祖候補ラベルと一致するノードを選ぶ．

(3) において Dewey Order では，子孫ラベルに先祖ラベルが含まれているので，条件 B を満たす子孫のラベルから先祖候補のラベルを取り出すことができる．以下で説明する 2 つの提案手法は，この処理の方法が異なる．DO-VLEI ラベルを数値として mod() を取って下桁を取り出す方法と，DO-VLEI ラベルを文字列として扱い、文字列関数 substring() により取り出す方法である．

3.2 mod() による判定法

前述の流れで説明した (3) において，mod を用いる方法を説明する．この方法では文字列のパターンマッチングは行なわず，数値として格納した VLEI ラベルを数値のままで用いて先祖子孫関係の判定を行うことができる．また，先祖ラベルとの比較においても “=” を用いて数値が等しいか，等しくないか，を判定すればよい．まずそのために，逆 DO-VLEI ラベルを提案する．

定義 3. 逆 DO-VLEI ラベル

Dewey Order において，デリミタを 9 で表し，root ノードの逆 DO-VLEI ラベル $L_{root} = 91$ とする．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 8 進数に変換したものを C_{child} とする．このとき，親ノードの逆 DO-VLEI ラベルを L_{parent} とし，

$$\text{逆 DO-VLEI ラベル} = 9 \cdot C_{child} \cdot L_{parent}$$

とする．図 4 に例を示す．

	DO-VLEIラベル	逆DO-VLEIラベル
定義	親ラベル･兄弟コード･9	9･兄弟コード･親ラベル
親ノードラベル	192939	939291
兄弟順序コード	13	13
ラベル	192939139	913939291

図 4　DO-VLEI ラベルの例

逆 DO-VLEI ラベルでは，子孫候補のラベルを (10 の (先祖候補の長さ) 乗) で割った余りとして先祖候補ラベルを取り出すことができる．

例） 先祖候補ノードの逆 DO-VLEI ラベル：9291 (length=4)
10 ^ (先祖候補ノードのラベル長)：10^4=10000
子孫候補の逆 DO-VLEI ラベル：9139291
mod(9139291，10000)=9291

3.3 substring() による判定法

DO-VLEI ラベルをそのまま文字列として扱い，子孫ラベルから先祖候補を取り出す際に mod() ではなく，文字列関数 substring() を用いる．DO-VLEI ラベルでは，[親ノードのラベル]+[兄弟順 VLEI コード（8 進数）]+9 となっており，左端から部分文字列を取り出すことで先祖ラベルを取り出すことができる．

この方法では，逆 DO-VLEI ラベルでの判定のような 10 の (先祖候補の path 長さ) 乗を計算する必要がない．文字列処理であるが，以前の先祖子孫関係の判定方法と異なりコストが大きいパターンマッチングを行わない．文字列の一致判定でよいので，＝により判定可能である．LIKE よりも＝で判定するほうが速いことから，文字列一致の判定には LIKE ではなく=を使う．

例） 先祖候補ノードの DO-VLEI ラベル：1929 (length=4)
子孫候補 DO-VLEI ラベル：1929139
substring(192913，1，4)= 1929

4. 評価実験

改良した DO-VLEI ラベルの先祖子孫関係の判定方法を評価する実験を行う．評価においては，2.4 で述べた 2 種類のベンチマーク，Michigan Benchmark [8]，及び XMark [9] を用いる．

4.1 Michigan Benchmark

Michigan Benchmark は，XML データベースの size，fanout，depth が問い合わせ処理 (XML query processing) に与える影響を測定し，データベースの処理性能を評価することができる．

Query Set は，その性能評価の内容によって分類されている．XML data は，node の選択度 (selectivity)，depth，fanout，などの特性を指定し，それぞれの特性が query processing に与える影響を測定できるようになっている．先祖子孫関係判定の処理を評価することが目的であることから，Michigan Benchmark の

表 1　Michigan Benchmark に用いた XML 文書

ds0.1x　　file size =　　47,504,697　byte

Nodes	name	
Element	eNest	number of nodes = 66655
	eOccasional	number of nodes = 1041
Attribute	aUnique1	unique integer (id)
	aUnique2	unique integer (random)
	aLevel	store the level of th node
	aFour	aUnique2 mod 4
	aSixteen	(aUnique1+aUnique2) mod 4
	aSixtyFour	aUnique2 mod 64

表 2　XML 文書を格納するテーブルのスキーマ

ID	ノードに割り当てる unique な ID の値（INTEGER）
label	Dewey Order の VLEI ラベル
length	label の長さ（INTEGER）
name	ノード名 (TEXT)
その他	上の他に XML 文書によって value や必要な構造を加える

表 3　selectivity

aLevel	fanout	nodes
1	2	1
2	2	2
3	2	4
4	2	8
5	4	16
6	4	64
7	4	256
8	1/4	1024
9	2	256
10	2	512
11	2	1024
12	2	2048
13	2	4096
14	2	8192
15	2	16384
16	-	32768

column name	selectivity	nodes
aFour	1/4	16663
aSixteen	1/16	4206
aSixtyFour	1/64	1041

↑
aFour=0,1,2,3 の値をとり，各値の selectivity は等しい
selectivity ＝ 1/4 は全体のノードの 1/4 が同じ値を持つということを意味する．

← aLevel の値は深さを表す
fanout は一つのノードの子の数

pscaling factor=0.1(66655nodes)

中の Structural Selection Query(QS) を使用した．また，Update Query(QU) を用いて更新操作の評価も行なった．

4.1.1　実験準備

実験に用いた XML 文書は，Michigan Benchmark の DataSets の一つである ds0.1x を用いた (file size：45MB，要素ノード数：67696nodes)．表 1 に ds0.1x の情報を示す．

ds0.1x を PostgreSQL の RDB に格納した．XML 文書を格納するテーブルのスキーマを表 2 に示す．

Michigan Benchmark を行なうに当たり，ノードの Selectivity (選択度) の指標となる (“aFour”,“aSixteen”,“aSixtyFour”) という column を加え，それぞれに値を格納した．表 3 に Selectivity を示す．例えば，属性 “aFour” の Selectivity は 1/4 であり，「aFour=0」という条件からは，66655 nodes のうち 16663 nodes(=66655×1/4) が選ばれる．

逆 DO-VLEI ラベルの格納データ型には，PostgreSQL の可変長数値型である NUMERIC を用いた．文字列 DO-VLEI ラベルは，可変長文字列型 TEXT で格納した．NUMERIC で格納した場合の検索速度も測定したが，TEXT で格納した場合とほとんど差はなかった．

表 4　実験環境

CPU	Intel(R) Xeon(TM) CPU 2.40GHz
HDD	IBM IC35L018(18.4GB 15KRPM SCSI) × 6 (RAID5)
Memory	3.2GB
OS	Linux 2.4.27
DBMS	PostgreSQL 7.4.1
java	SunSDK1.4.2_03

本実験では，次の 4 つの先祖子孫関係の判定方法による検索処理時間を比較した．

- VLEI-mod:逆 DO-VLEI ラベルで mod() を用いた判定法
- VLEI-sub:DO-VLEI ラベルで substring() を用いた判定法
- VLEI-like:文字列 DO-VLEI ラベルを使用し，従来のパターンマッチングによる判定法
- PP(int):整数を用いた範囲ラベリング法 (2 つの番号 [preoder,postorder] の大小関係比較により判定する方法)

QS21 を例として，先祖子孫関係の判定方法の違いを図 5 に示す．提案した 2 つの判定方法は，先祖ラベルの取り出し方が異なるのみであるため SQL は同様であるといえる．図 5 では substring() による方法のみを示す．先祖ラベルを取り出す部分を mod() にすれば mod() による方法となる．

実験環境は表 4 の通りである．

本実験では，Michigan Benchmark の問い合わせのうち，先祖子孫判定を必要とする問い合わせを用いた．Michigan Benchmark には問い合わせのセットに，Ancestor-Descendant Selection (QS21～27，QS31～35) があり，これを用いた．この Ancestor-Descendant Selection は，先祖の条件と子孫の条件が与えられ，条件を満たす先祖を求めるものである (QS(A) とする)．本実験では，同じ条件で子孫を求めた場合の処理時間も測定した (QS(D)) とする．QS21～27 の各問い合わせの先祖条件，子孫条件を表 5 に示す．例）QS21:Select nodes with aLevel=13 that have a descendant with aSixteen=3.(先祖条件 aLevel=13，子孫条件 aSixteen=3) QS31～33 は，図 6 のように複数の先祖子孫の組がある問い合わせである．

4.1.2　実験結果

問い合わせ処理時間 (QueryTime) を計測した結果を表 6 にまとめた．図 7～9 では，クエリタイプ別に範囲ラベリング法と DO-VLEI ラベリングのパフォーマンス比をグラフで示す．
各 QS ごとに VLEI-mod，VLEI-sub，VLEI-like，PP(int) の順番でグラフ棒が並んでいる．VLEI-mod，VLEI-sub が本稿で改良を試みた判定手法である．

グラフの縦軸は，DO-VLEI ラベリング (VLEI-mod,VLEI-sub,VLEI-like) と範囲ラベリング法のパフォーマンス比である．ここでパフォーマンスとは (1/query 処理時間) とし，DO-VLEI による問い合わせ処理時間を T_{VLEI}，PP(int) による問い

表 5　QS21～27 の選択条件

Query	先祖条件		子孫条件		nodes	Sel(%)
QS21(A)	aLevel=13	M	aSixteen=3	M	2403	3.6
QS22(A)	aLevel=15	H	aSixtyFour=3	L	489	0.7
QS23(A)	aLevel=11	L	aFour=3	H	1024	1.5
QS24(A)	aSixteen=3	M	aSixteen=5	M	717	1.1
QS25(A)	aFour=3	H	aSixtyFour=3	L	1168	1.8
QS26(A)	aSixtyFour=9	L	aFour=3	H	337	0.5
QS27(A,D)	aSixtyFour=9	L	aFour=3	H	3032	4.5
QS21(D)	aLevel=13	M	aSixteen=3	M	3544	5.3
QS22(D)	aLevel=15	H	aSixtyFour=3	L	494	0.7
QS23(D)	aLevel=11	L	aFour=3	H	15864	23.8
QS24(D)	aSixteen=3	M	aSixteen=5	M	2241	3.4
QS25(D)	aFour=3	H	aSixtyFour=3	L	959	1.4
QS26(D)	aSixtyFour=9	L	aFour=3	H	2914	4.4

A: select ancestors　　M: Midium selectivity (6%)

D: select descendants　　H: High Selectivity (25%)

L: Low Selectivity (1.5%)

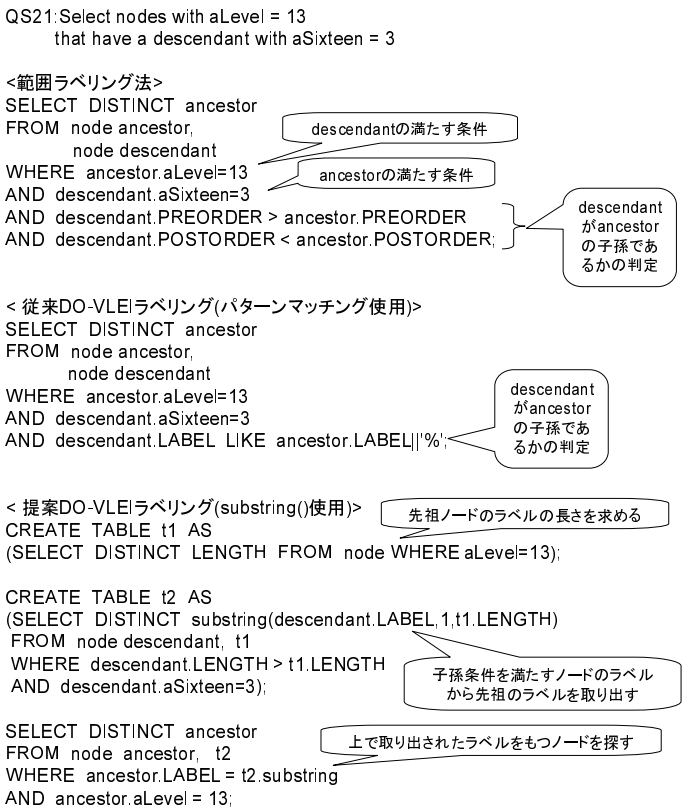


図 5　SQL：先祖子孫関係の判定法の比較 (QS21 の例)

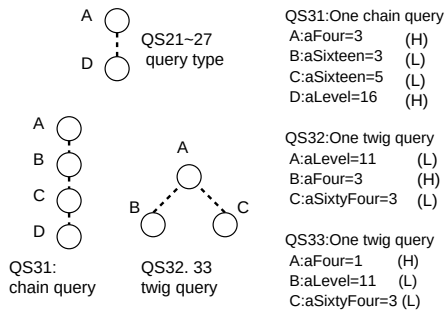


図 6　実験に用いた QS の Query Type

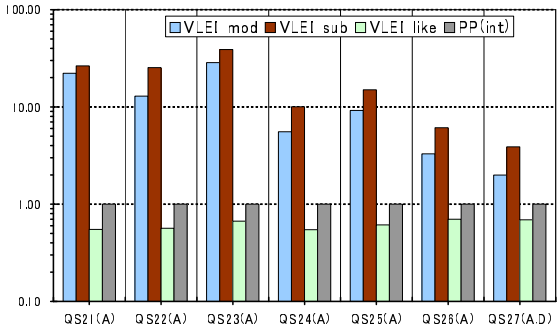


図 7　QS(A) の範囲ラベリング法と DO-VLEI の性能比

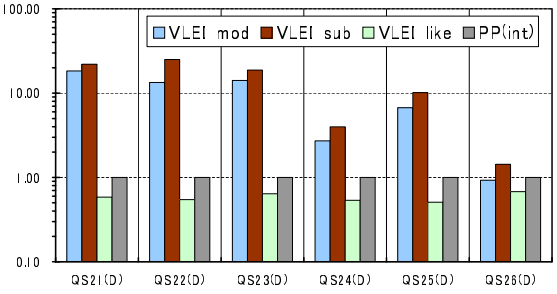


図 8　QS(D) の範囲ラベリング法と DO-VLEI のパフォーマンス比

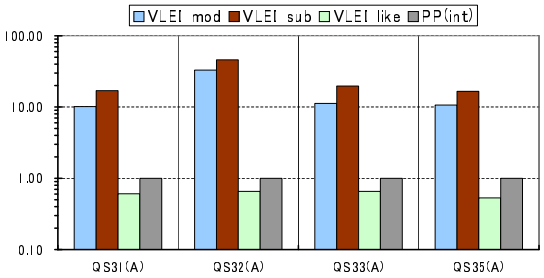


図 9　QS31～33，35 の範囲ラベリング法と DO-VLEI のパフォーマンス比

合わせ処理時間を T_{PP} とすると，縦軸のパフォーマンス比は，

$R=(1/T_{VLEI})/(1/T_{PP})=T_{PP}/T_{VLEI}$

である．図 7～9 では，この値の対数をとって示している．

グラフから，特に先祖選択の問い合わせ QS(A) において，新しく用いた先祖子孫判定法 VLEI-mod，VLEI-sub が格段速いことがわかる．VLEI-mod と VLEI-sub では，文字列関数 substring() を用いる方法 VLEI-sub のほうが速い．これは，VLEI-mod の処理における mod() で大きな桁数の割算が生じているためである．今回は，数値の格納に可変長数値型 NUMERIC を用いており，計算にコストがかかっている．

4.1.3　更新処理

Michigan Benchmark の Update Query(QU1～QU7) を用いて，更新操作の問い合わせのベンチマークを行なった．ここでは，DO-VLEI ラベリング法と範囲ラベリング法で浮動小数点数を用いた手法 (QRS) [6] を比較した．結果を表 7 に示す．QU1 と QU3 はある条件を満たすノードにそれぞれ 1 つの子ノードを

表 6　QS の問い合わせ処理時間の測定結果（second）

Query	VLEI-mod	VLEI-sub	VLEI-like	PP(int)	nodes	Sel(%)
QS21(A)	0.37	0.31	14.92	8.21	2403	3.6
QS22(A)	0.64	0.32	14.53	8.22	489	0.7
QS23(A)	0.58	0.42	24.56	16.45	1024	1.5
QS24(A)	1.47	0.81	14.93	8.16	717	1.1
QS25(A)	0.99	0.61	14.90	9.12	1168	1.8
QS26(A)	5.04	2.71	23.65	16.57	337	0.5
QS27(A,D)	8.46	4.35	24.44	16.89	3032	4.5
QS31(A)	11.92	7.11	199.32	120.85	314	0.5
QS32(A)	0.76	0.54	38.26	25.04	635	1.0
QS33(A)	1.28	0.73	21.99	14.40	317	0.5
QS35(A)	3.31	2.12	66.44	35.27	60946	91.4
QS21(D)	0.45	0.37	14.04	8.19	3544	5.3
QS22(D)	0.61	0.33	15.06	8.22	494	0.7
QS23(D)	1.16	0.87	25.53	16.35	15864	23.8
QS24(D)	3.02	2.06	15.31	8.20	2241	3.4
QS25(D)	1.25	0.83	16.56	8.40	959	1.4
QS26(D)	17.76	11.50	24.29	16.47	2914	4.4

単位：second
QS（A）：select Ancestors
QS（D）：select Descendants
VLEI-mod：mod() による数値の DO-VLEI ラベリング
VLEI-sub：substring() による DO-VLEI ラベリング
VLEI-like：LIKE によるパターンマッチング DO-VLEI ラベリング
PP(int)：整数をラベルに用いた範囲ラベリング法

表 7　QU（Update）のクエリ処理時間の測定結果

Query	Query Dscription	VLEI	QRS	VLEI/QRS
QU1	Point Insert	2.24	2.63	0.85
QU2	Point Delete	11.41	2.52	4.53
QU3	Bulk Insert	13.45	54.64	0.25
QU4	Bulk Delete	5.85	5.05	1.16
QU5	Bulk Load	170.70	164.92	1.04
QU6	Bulk Construction	1.93	2.49	0.78
QU7	Restructuring	0.17	0.34	0.49

挿入する Query である．単純な挿入であるため DO-VLEI ラベルの挿入処理が速いことがわかる．また，QU1 を連続して行ったところ，QRS では 18 回目で精度限界となった．DO-VLEI ラベリング法では精度限界にはならかったが，3 回の挿入に対して 1 桁ずつラベルが長くなった．QU2 は 1 つのノードを削除する更新である．Dewey Order 法では，あるノードが削除されるとその子孫ノードのラベルを変更する必要が生じる．そのため，DO-VLEI ラベルにおける処理時間が大きくなっている．

4.1.4　考　　察

実験結果から，提案した改良手法が検索処理速度においては，かなりの効率化を実現できたといえる．しかし，選択対象のノード数が大きくなると，この効果は小さくなった．子孫ノードを選択する場合と先祖ノードを選択する場合では，ノード数が大きい場合に差が大きかった．これは，比較するラベルの数が大きくなるからである．

ベンチマークにあたり，固定長整数型 (INTEGER や BIGINT) では桁数が足りなかったため可変長型を用いたが，格納効率，処理速度，メモリ消費量を考慮すると，改善する必要がある．

更新処理の実験では，単純な挿入処理では DO-VLEI ラベリングが QRS と比べて速かった．これは，これまでの評価実験 [7] で示されているとおり挿入の際のラベル付けアルゴリズムのコストが小さいからである．また，連続した挿入に対しても QRS のように精度限界は生じない．しかし，先祖ノードが削除される場合や，結合される場合などには，Dewey Order では大きなラベル書き換えが生じ，更新コストが大きくなった．

表 8　XMark に用いた XML Data Set

scaling factor	document size[KB]	table rows		
		element	attribute	text
0.000	27	382	78	236
0.001	116	1729	357	1203
0.005	571	8517	1972	5945
0.010	1161	17131	3917	12004
0.100	11670	167864	38266	118141

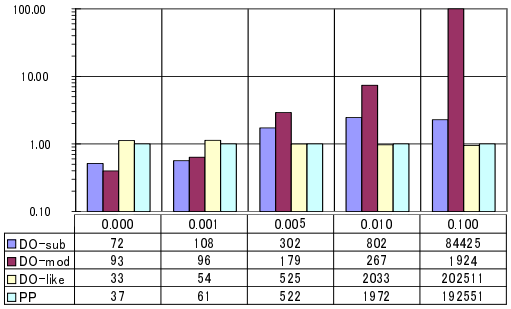


図 10　XMark Query8 における PP(int) と DO-VLEI の性能比

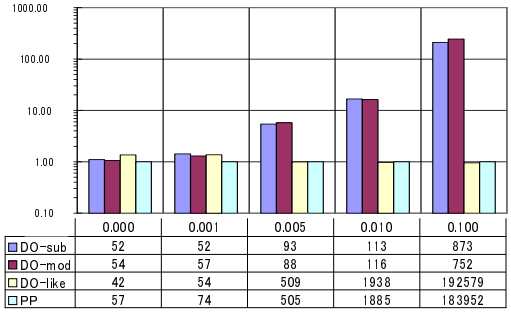


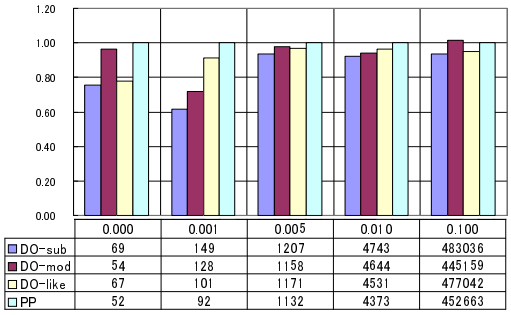
図 11　XMark Query5 における PP(int) と DO-VLEI の性能比

4.2　XMark

4.2.1　実 験 結 果

XMark では，5 つの scaling factor の XML 文書を用いた．表 8 に各 XML 文書の情報を示す．XML 文書の格納では，要素ノードを element table，属性ノードを attribute talbe，テキストノードを text table に格納する．Query Set は XMark の Query1〜Query20 を用い，各 Query に対して，処理時間を測定した．実験では Michigan Benchmark と同じ 4 つの検索法を比較した．

XMark の実験結果の一部を図 10〜12 に示す．横軸は使用した XML ファイルの scaling factor である．縦軸は Michigan Benchmark と同様に，範囲ラベリング法との性能比を示す．グ



	0.000	0.001	0.005	0.010	0.100
DO-sub	69	149	1207	4743	483036
DO-mod	54	128	1158	4644	445159
DO-like	67	101	1171	4531	477042
PP	52	92	1132	4373	452663

図 12　XMark Query19 における PP(int) と DO-VLEI の性能比

ラフの下には，クエリ処理時間 [ms] のデータテーブルを置いた．

実験には Query1〜20 を用いたが，実験の結果は大きく 3 パターンに分けられる．図 10 と図 11 のパターンが最も多かった．ノード数の増大に伴い，提案手法の性能比が上がっているパターンである．この結果から，VLEI-sub と VLEI-mod の提案手法が対象となるノード数が大きい検索において，有効であることがわかる．また，図 12 のように，十分な効果が見られない Query もあった．

4.2.2　考　　察

XMark の結果から，提案手法によるラベル比較回数の削減の効果が大きいクエリでは，ノード数が大きいほどラベル他手法との性能差が顕著であり，極めて高い性能改善を示すことがわかった．しかし，ノード数が小さい場合は，ラベルの比較にかかるコストよりも，ノードの絞込みにおけるテーブル作成のコストが大きくなる場合があり，提案手法のコストが大きくなった．

5. ま と め

Dewey Order に適用した DO-VLEI ラベルの先祖子孫判定方法の改良を試みた．このとき，子孫ノードのラベルから先祖ノードのラベルがわかるという Dewey Order の特徴を利用した．本稿では 2 つの方法を提案し，評価した．1 つは逆 DO-VLEI を数値として扱い，mod() により先祖ラベルを取り出す方法である．もう 1 つは，文字列の DO-VLEI ラベルを文字列関数 substring() により先祖ラベルを取り出す方法である．これらの方法により，従来文字列のパターンマッチングで比較処理していた判定方法を改め，検索処理を効率化することができた．実際のベンチマークを用いた評価実験により，先祖子孫関係の判定を含む問い合わせに対しては，改良手法が有効であることがわかった．さらに，インデックスを作成するなどの工夫を加えることで，さらに効率的な処理が実現される可能性が大いにある．今後，DO-VLEI ラベルの格納方法やメモリ消費量を考慮すればより実用性が高くなるといえる．

DO-VLEI ラベルの有用性

＊先祖条件，子孫条件が与えられ，先祖ノードあるいは子孫ノードを求めていく問い合わせの効率がよい．

DO-VLEI ラベルが考慮するべき課題

＊複雑な問い合わせの場合．

＊ラベルの格納方法．

＊中間テーブル作成のコスト，メモリ消費量．

＊先祖ノードが削除される場合．

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 (16016232)，独立行政法人科学技術振興機構 CREST，及び 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の構築」の助成により行なわれた．

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,No.1 in Vol.1,2002*, 2002.

[3] Paul F. Dietz. Maintaining order in a linked list. In *Proceedings of the fourteenth annual ACM symposium on Theory of computing* pp. 122–127, 1982.

[4] Online Computer Libraly Center. Introduction to the dewey decimal classification. http://www.oclc.org/oclc/fp/about/about_the_ddc.htm.

[5] Edith Cohen, Haim Kaplan, and Tova Milo. Labeling dynamic xml trees. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART*, pp. 271–281, 2002.

[6] Toshiyuki Amagasa, Masatoshi Yoshikawa, and Shunsuke Uemura. Qrs: A robust numbering scheme for xml documents. In *19th International Conference on Data Engineering (ICDE 2003)* pp. 705–707, 2002.

[7] Kazuhito Kobayashi, Wenxin Liang, Dai Kobayashi, Akitsugu Watanabe, and Haruo Yokota. Update conscious xml labeling methods using dedicated codes. Technical report, CS Technical Reports,Tokyo Institute of Technology, 2004.

[8] Kanda Runapongsa, Jignesh M.Patel, H.V.Jagadish, Yun Chen, and Shurug Al-Khalifa. Michigan benchmark: Towards xml query performance diagnostics. In *Proceedings of the 29th VLDB Conference*, 2003.

[9] 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.

[10] 江田毅晴 , 天笠俊之 , 吉川正俊 , 植村俊亮. XML のための動的範囲ラベル付け手法:その評価および XRel への適用について. 情処研報, 情報処理学会, DBS-129-16, 2002.

[11] 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.

[12] 小林一仁 , 横田治夫. 挿入制限のないコード vlei を用いた xml ラベリング手法の検討とその評価. 第 15 回電子情報通信学会データ工学ワークショップ (DEWS2004) 論文集.