A Novel Encoding Scheme for XML Document Update-supporting

Yuanzi He

School of Information, Nanchang Institute of science & Technology, Nanchang, 330108, China

Keywords: XML, XML Document Update, Fraction Encoding, Dewey Encoding, XML data updating.

Abstract. In order to improve the efficiency of XML document query and support XML document dynamic update, etc, this paper proposes a dynamic prefix encoding scheme based on fraction (DPESF), which uses the unlimited extensibility of fraction to implement XML document dynamic updating without the second encoding in according with retaining the excellent characteristics of Dewey encoding. Finally, this paper implements the relevant experiment, the experiment results show that the DPESF encoding has better time and space performance compared to the existing the dynamic prefix encoding schemes.

Introduction

As more and more network data are expressed by XML form, XML has become the de facto standard of network data storage and exchange, and is also an essential of the web and the basis of technical development in the future. How to quickly implement data query of XML documents is hotspot of the current XML researches. However, the data query of XML document depends on encoding of the XML document tree. Therefore, the study of XML document encoding is of great significance. The encoding mechanism of XML document refers to each nodes of document tree is given only encoding so as to determine quickly the relationship between any two nodes(eg. parent-children relationship, ancestor-descendant relationship, sibling relationship, etc)[1]. At present, there is XML encoding sheme, which cannot support the dynamic updating of XML nodes. The existing encoding schemes need significantly adjust and recode their encodings when XML data implements the operation of inserting, deleting and updating, which leads to high cost of data updates.

The paper is arranged as follows. In the next section, we propose a dynamic prefix encoding scheme based on fraction (DPESF) model that we research in this paper, and some definitions and assumptions are given. In Section 2, the related work is given. In section 3, the some related definitions are given. Section 4 presents the related algorithm. In Section 5, some experiments and performance are given. Finally, we conclude our paper in section 6.

The Related Work

At present, there is more common encoding schemes, such as interval encoding [2-3] and prefix encoding [4]. The interval encoding is that each nodes of XML document tree is given an interval encoding [start, end], and encoding of anyone must contains interval encoding of its descendant nodes. The literature [2] and literature [3] proposes an interval encoding method, which can effectively support calculating of contained relationship and document location relationship. In addition, its encoding length is smaller, and query efficiency is higher. However, they cannot implement completely dynamic update of nodes. The interval encoding needs the second encoding when insufficient reserved space, which will lead to massive overhead of space and time. The prefix encoding is that encoding of parent nodes is taken as prefix of its children encoding. Dewey encoding is the most common prefix encoding, but prefix encoding does not directly support update operation. The literature [5] proposes an ORDPATH, which is analogous to Dewey, which is expressed in form of binary. The original encoding of ORDPATH is expressed in positive odd number. The reserved minus of placeholder implements dynamic update of node when inserting new nodes, however, judgement of the location relationship is more complex and encoding scale is

more massive in ORDPATH . The literature [6] proposes a TDE algorithm, which maps the real number to two dimensional arrays. The TDE algorithm the characteristic of any two real number contain infinite real number, so as to avoid the second encoding when node updating, however, the node length of TDE encoding is longer, and it wastes storage space.

In order to improve the query efficiency, this is paper proposes a dynamic prefix encoding scheme based on fraction (DPESF). The DPESF coding has not only better time and space performance, but also supports dynamic updating operation. According to thinking of Dewey encoding, this paper proposes a dynamic prefix encoding scheme based on fraction (DPESF), which can reserves the excellent characteristic of Dewey encoding, and implements the dynamic updating of XML data.

The DPESP Encoding

Definition1. Numeric-Character Corresponding Table (NCCT). Let number set $N=\{0,1,2,3,4,5,6,7,8,9\}$, character set $C=\{'A','B','C','D','E','F','G','H','I','J'\}$, any $n\in N$ exists correspond to only one $c\in C$. The corresponding rule function $f=\{<0,A>,<1,B>,<2,C>,<3,D>,<4,E>,<5,F>,<6,G>,<7,H>,<8,I>,<9,J>\}$.

Definition2. Fraction Encoding. Any a fraction $\frac{x}{y}$ is expressed in form of hy, where $h=\{a_0,a_1,a_2,....a_n \mid a_i\in C,i,n\in n\}$, C is character set of definition 1 that numerator x can be expressed as character form in accordance with definition 1 while expressing fraction, and denominator y stays the same. For instance, $\frac{123}{11}$ is expressed as BFH11.

Definition3. Static DPESF Encoding. Static DPESF encoding refers each nodes of XML document tree is given only a coding when initializing XML document. The encoding rules are determined by following rules.

i) The encoding of root node is 1;

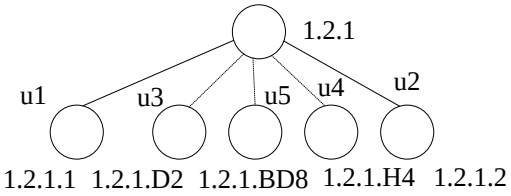
ii) In the process of depth-first walk in a tree, if node v is the ird children node of node u, then the DPESF of node v is $c(u).i$, where $c(u)$ represents encoding of node u.

Definition4. Dynamic DPESF Encoding. The dynamic DPESF encoding refers that the DPESF supports operation of insertion, deletion and updating in accordance with the static DPESF encoding. Considering the operation of insertion and deletion does not influence other node encoding, the dynamics mainly refers to insertion operation. The encoding of inserted node is expressed by fraction when inserting new nodes in DPESF encoding. Considering the fraction is not easy to be express in encoding, the fraction is encoded by FC encoding. There is three ways in inserting node.

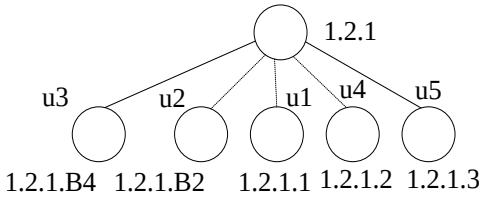
i) Inserting a node between $u1$ and $u2$. Let encoding of node $u1$ to be $p(u).a$, Let encoding of node $u2$ to be $p(u).b$($p(u)$ is encoding of parent node of $u1$ and $u2$), then the encoding of new insertion node to be $p(u).((a+b)/2)$. In figure 1(a), in turn inserting the $u3,u4$ and $u5$ between $u1$ and $u2$, then encoding of node $u3$ is $1.2.1.((1+2)/2)$, according to the expressed definition of definition 2, the final encoding can be expressed as $1.2.1.D2$.Using the same method, encoding of node $u4$ is $1.2.1.((3/2+2)/2)$, the final encoding can be expressed as $1.2.1.((3/2+7/4)/2)$, and the final encoding can be expressed as $1.2.1.BD8$.

ii) Inserting the nodes left-most node of $u1$. Let the encoding of node $u1$ to be $p(u).a$, then encoding of new inserted node is $p(u).a/2$. In figure 1 (b), in turn inserting $u2$ and $u3$, then the encoding of node $u2$ is $1.2.1.1/2$, according to expressing method of definition 2, the final encoding can be expressed as $1.2.1.B2$. The encoding of node $u3$ is $1.2.1.((12)/2)$, the left-most encoding is $1.2.1.B4$.

iii) Inserting the nodes right-most node of $u1$. Let the encoding of node $u1$ to be $p(u).a$, then encoding of new inserted node is $p(u).(a+1)$.In figure 1 (b), in turn inserting $u4$ and $u5$, then the encoding of node $u4$ is $1.2.1.2$, the encoding of node $u5$ is $1.2.1.3$.



(a) Inserting nodes left-most



(b) Inserting nodes right-most

Fig.1: Insertion Operation of DPESP Encoding

The Relevant Algorithm

It can be seen that the form of DPESF encoding is similar to Dewey encoding. Therefore, the DPESF encoding has still excellent characteristic of Dewey encoding, such as simplicity of algorithm implementation and encoding contains location relationship, etc. The DPESF uses the fraction form to express node encoding when updating, so as to implement dynamic update operation. Insertion operation process is shown in algorithm 1and determining of nodes location is shown in algorithm 2.

Algorithm 1. Inserting node algorithm.
Input: Encoding of left node1 and right node2 of inserting node.
Output: Encoding newnode of inserting new node.

```
InsertNode(Node node1, Node node2)
{
code1= getDPESF(node1); // get DPESF encoding of node1;
pcode1=parentCode (node1); // get parent node encoding of node1;
lcode1=lastCode(node1); // get value of the last "." of node1 encoding;
code2=getDPESF(node2); // get DPESF encoding of node2;
pcode2=parentCode(node2);
 lcode2=lastCode(node2);
If (code1! =null&& code2==null)
Newcode =pcode1&FractionCoding (node1/2);
else if (code1==null&& code2! =null)
 // Inserting node after right-most node;
newcode=pcode2&FractionCoding (node2+1);
else if (code1!=null&& code2!=null)
 // Inserting node between node1 and node2;
  newcode=pcode1&FractionCoding ((lcode+node2)/2);
else exit();
return newcode;
 }
```

Algorithm 2. Determining location relationship algorithm.
Input : DPESF encoding of node1 and right node2.
Output: Location relationship between node1 and node2;

```
DataType Nodejudge(Node node1, Node node2)
{
code1=getDPESF (node1);  // get DPESF encoding of node1;
```

```
code2=getDPESF (node2);  // get DPESF encoding of node2;
level1=getLevel (node1);  // get level number of node1;
level2=getLevel (node2);  // get level number of node2;
if (prefix(code1,code2)>0)&&abs(level1,level2)==1)
 return parent;
 else if (prefix(code1,code2)<0)&&abs(level1,level2)==1)
 return child;
else if (prefix(code1,code2)>0)
 return descendant; // node1 is the descendant node of node2;
else if (prefix(code1,code2)==0)
 return descendant;    // node1 is brother  node of node2;
else return other ;  // other relationship;
 }
```

In the algorithm 1, because the time complexity of DPESF coding of getting the left-right node is O(n) , the time complexity of the parents coding and inserting a node are O(1), and the time complexity of the algorithm1 is O(n). In the algorithm 2, the time complexity of getting DPESF coding and calculating the lever of node is O (n). For judging from relationship between nodes and nodes, it is required that implementing operation of string pattern matching. Therefore, the time complexity of the algorithm 2 is also O (n).

Analysis of Experiment and Performance

Experiment Environment

Hardware environment: AMD Athlon 7750 dual core 2.7 GHZ Processor; 2G Memory; 160G HDD;

Software environment: Windows 7, 32 bit; Development Platform :Eclipse 3.6.2; The XML document is parsed with Java DOM4J, and the XML document data are stored in MySQL5.5. The dataset of experiment is shown in table1. Five datasets of experiment are given in table 1. The dataset D1 is provided by the literature [8], and the dataset D2 and D3 are provided by literature [9]. The D4 and D5 are generated by XMARK[10], the D4 takes 0.04 on as *f* of the generated factor, and D5 takes 0.08 on as *f* of the generated factor.

Table.1.Testing Dataset

Dataset	File name	File size	Maximum depth	Average depth	Total nodes
D1	Hamlet.xml	261K	6	4.86	6163
D2	Sigmod.xml	456K	7	5.66	15259
D3	NASA.xml	2015K	8	5.0	40786
D4	XMARK1.xml	4726K	12	5.16	83479
D5	XMARK2.xml	9370K	12	5.17	165539

Experiment Result Analysis

Figure2 shows three different encoding in space occupation. TDE encoding is encoded by Two-dimensional coding, each node is made up of two-dimensional tuple. Each nodes of the DPESF coding and the ORDPATH is made up of some unit value. Space occupation of TDE coding is larger, while DPESF coding is not much different from ORDPATH coding. But the even bits of the ORPATH coding are vacant, the average value of the final bit is larger than DPESF coding when many nodes need encoding. It can be seen that the DPESF occupies less space.

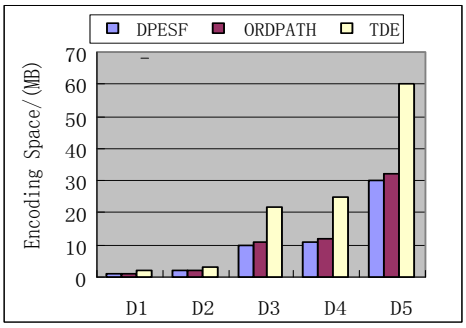


Fig2.Space occupation

Figure 3 shows time comparison of three static coding. The DPESF coding and ORDPATH coding are similar to Dewey coding. The calculation method of TDE coding is much complex, and so it will consume much time.

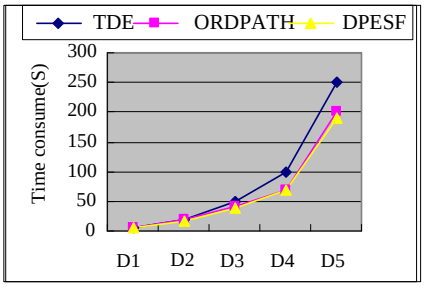


Fig3.Time consume of static coding

Figure 4 shows comparison of average performance. When inserting operation, the DPESF coding and TDE coding need calculate once to get new encoding, while ORDPATH coding need to introduce placeholders before inserting new node, so updating efficiency is lower.

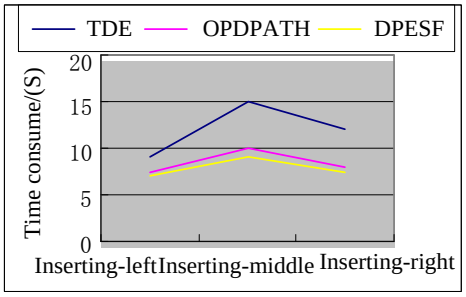


Fig4.Time consume of dynamic coding

Conclusions

According to analysis of existing the interval coding and prefix coding, this paper proposes a dynamic prefix encoding scheme based on fraction, the DPESF coding reserves the excellent characteristic of Dewey, and implements the dynamic updating of XML data. The experiment results show that the DPESF coding has a good performance in space occupation, encoding time and efficiency. The next step in our research is to design and implement indexing structure in accordance with DPESF coding, so as to support the dynamic update in Native XML database.

Acknowledgements

The research work is supported by Supported by Program for New Century Excellent Talents in University of China under Grant no.NCET10-0787.

References

[1] HU.J.M, LI.J,H, DU.Z.H,etc. A Dynamic XML Document Tree Encoding Scheme. Computer Engineering, vol,19,36 (2010).pp:75-77.

[2] LI.Q.Z, MOON B. Indexing and Querying XML Data for Requlars Path Expressions [C]. Proc. of the 27th International Conference on very Large Data base. Roma, Italy. [s.n.], 2001, pp:361-370.

[3] ZHANG C, Naughton J,David D, et al. On Supporting Contain-ment Queries in Relational Database Management Systems[C]. Proc. of the SIGMOD ’01. Santa Barbara, California,USA. [s.n.], 2001,pp:425-436.

[4] Cohen E, Kaplan H. Milo Labeling Dynamic XML Trees [C]. Pro of PODS’02. Madison Wisconsin. SA:[S.n.],2002,pp:271-281.

[5] Wu. X, Lee,M.L, Hsu, W. A prime number labeling scheme for dynamic ordered XML trees. In: proceedings of ICDT 2004,PP.66-78.

[6] LIU Z Y, CHEN Y. Identifying Meaningful Return Information for XML keyword Search. Processings of the 2005 ACM SIGMOD Internetional Conference on Management of Data.NY(2007),pp:19-30.

[7] SOLTAN S, RAHGOZAR M. A clustering based scheme for labeling XML trees. International Journal of Computer Science and Network Security, vol,6 (9) 2006,pp: 84-89.

[8] Dong Chan An, Seog Park. Efficient labeling scheme of XML data considering update operations [C].The 8th IEEE International Conference on Computer and Information Technology, CIT 2008.Sydney, Australia: NSW, 2008:438-443.

[9] NIAGARA Experimenttal Data [EB/OL]. Available at http://www.cs.wisc. edu /niagara/data.html.

[10] Xmark-An XML Benchmark Project[EB/OL]. Available at http:// monetdb. ewi.nl/xml/ downloads.html.