Send Orders for Reprints to reprints@benthamscience.ae

Fraction and Prefix Encoding Scheme of Supporting Updating Data Efficiently

Houliang Xie[1,*] and Liang Lei[2]

[1]*Information Engineering Department, Zhangjiajie Institute of Aeronautical Engineering, Zhangjiajie 427000, China*

[2]*School of Computer and Information Technology, Xinyang Normal University, Xinyang 464000, China*

Abstract: At present, more and more data are expressed in the form of XML format, and how to manage these data efficiently becomes an important issue. In order to update and query XML data efficiently, we proposed a new encoding scheme called MPES (modify prefix encoding scheme). MPES makes full good use of the advantages of fraction encoding and prefix encoding and it supports updating data efficiently. Furthermore, MPES also supports the representation of sibling relationship, parent-children relationship and ancestor-descendant relationship between any two nodes. The experimental results show that, compared with fraction encoding scheme, MPES improves the updating efficiency of XML data. As compared with prefix encoding scheme, MPES improved the querying efficiency of XML data.

Keywords: Fraction encoding scheme, prefix encoding scheme, update data, XML, XML format.

1. INTRODUCTION

With the rapid development of computer network technology, more and more online resources are represented in XML format. Compared with the hypertext markup language HTML, XML language has good scalability, interexchangeable and easy system to follow strict syntax requirements of information and so on. However, when the mass-like information is stored in XML format, fast update and fast query become extremely important. At present, domestic and foreign scholars have done a lot of research in this area, and made a number of coding algorithms at the same time. But there are still some deficiencies for efficient data updates and queries.

2. RELATED RESEARCH

In managing XML [1, 2] data, coding techniques [3] is extremely important. At present, domestic and foreign scholars have done a lot of research in this regard and made a variety of coding schemes. Overall, these coding schemes can be divided into two categories, the first category is the interval coding [4-7], and the second is a prefix code [8-11]. These codes have good performance, but on the other hand there are some deficiencies. Based on the in-depth study of other encoding schemes, we propose a new coding scheme - MPES (modify prefix encoding scheme). MPES is an improved prefix code; it scores coding and prefix code together. To sum up, the work for this paper is as follows:

(1) Proposing a new coding scheme i.e. MPES. MPES fractional prefix code LDSX coding to the combine advantages.

(2) MPES supports unlimited updates between nodes. MPES takes advantage of the theory that the scores between the two points can be inserted into an infinite number of points, in order to support unlimited updates between nodes. Thus, it effectively avoids secondary coding.

(3) The experimental results show that, compared with fractional coding, MPES improves updating efficiency of taking points. Compared with prefix code LDSX, MPES improves query efficiency of taking points.

3. MPES ENCODING

3.1. MPES Encoding Method

MPES codes are consisted of numbers and letters with XML document with each node with a triple (Numerator / Denominator, NodeCode, Level) to represent. Numerator parameter represents preorder of the XML node values obtained after each visit of the next XML node and its value is plus 1. The root Numerator is 1. Denominator initial value is set to a parameter. The parameter NodeCode represents coding node. Level represents the level node. The level of the root node is set to 1. Fig. (**1**) is named library.xml document, and MPES encoding is shown in Fig. (**2**).

3.2. For Junction Node Called Lib

The first mode encodes for the <1/1, a1, 1>, since it is the first order of encoding. Numerator is 1; Denominator initial value is set to 1. Because the node is the root node and its node coded as a1, the hierarchy is also 1. For 2 title FIG leftmost node, it is encoded as <3/1, a1a1.a1, 3>. Because it is a third node at the first coding sequence, Numerator is 3 and Denominator initial value is set to 1, and the node is encoded as a1a1.a1. Meanwhile the node is in the third layer, the Level 3.

*Address correspondence to this author at the Information Engineering Department, Zhangjiajie Institute of Aeronautical Engineering, Zhangjiajie 427000; E-mail: 306025482@qq.com

1874-4443/15

```
< lib>
    <book id="TP311">
        <title> Data Structure</title>
        <author> Robert </author >
    </book>
    < magazine>
        <title> Science </title>
        <author> Richard Stone</author >
    </magazine>
    <book id="TP312">
        <title> Operation System </title>
        <author> Andy Rathbone </author>
    </book>
</lib>
```

Fig. (1). Library.xml.

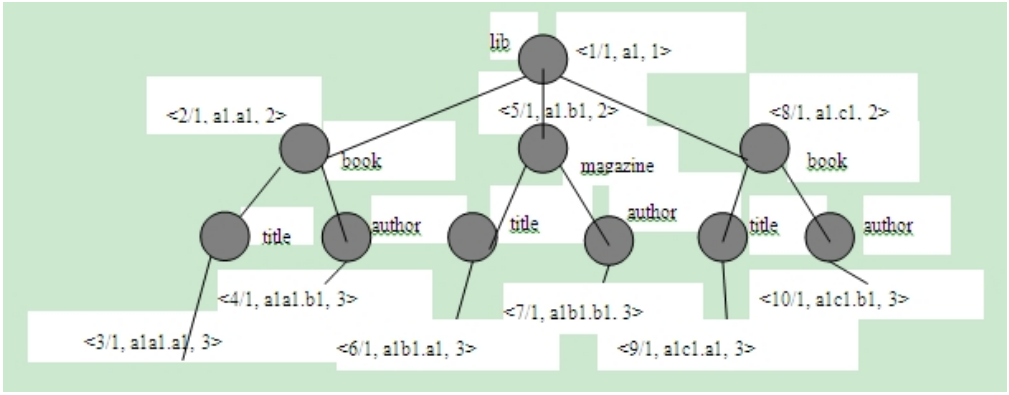

Fig. (2). XML document tree and MPES encoding.

3.2.1. The Judgment of Nodes Relationship of MPES

In XML document, the relationship between nodes includes brothers, father and son relationship, ancestor - descendant relationship and hierarchy nodes located. For any kind of XML, junctions between different coding have great significance. Four relations of MPES encoding were discussed above.

3.2.2. The Judgment of Ancestor-Descendant Relationship

Suppose U〈f1, x1.y1, p1〉 and V〈f2, x2.y2, p2〉 are two nodes of XML document, parameter f1 and f2 are real number, parameter p1 and p2 are integer. If x1+y1 is x2 substring, and p2 > p1+1, then node U is the ancestor of node V, node V is the descendant of node U. In Fig. (**2**), the encoding of node lib is <1/1, a1, 1>, the author node below the magazine code is <6/1, a1b1.a1,3>. Therefore, node lib is the ancestor of the author node.

3.2.3. The Judgment of Parent-Children Relationship

Suppose U〈f1, x1.y1, p1〉 and V〈f2, x2.y2, p2〉 are two nodes of XML document; parameter f1 and f2 are real number, parameter p1 and p2 are integer. If x1+y1＝x2, then node U is the parent of node V, node V is the children of node U. In Fig. (**2**), the encoding of lib node is <1/1, a1, 1>, the encoding of magazine is <5/1, a1.b1, 2>. Therefore, node lib is the parent of node magazine, and node magazine is the children of node lib.

3.2.4. Sibling Relationship Judgment

Suppose U〈f1, x1.y1, p1〉 and V〈f2, x2.y2, p2〉 are two nodes of XML document, parameter f1 and f2 are real number, parameter p1 and p2 are integer. If x1y1= x2y2, p1=p2, f1<f2, then node U is the brother of node V. In fig. 2, the left node title and the node author, its encoding is <3/1, a1a1.a1, 3> and <4/1, a1a1.b1, 3> respectively. According to the rule, we can see that x1y1= x2y2＝a1a1, p1=p2=3, 3/1<4/1. Therefore, the left title node is the left brother of the left author node.

3.2.5. Level Judgment of Node

Suppose U 〈f1, x1.y1, p1〉 is a node of XML document, parameter f1 is a real number, parameter p1 is an integer.

We can judge the level of a node according to its XML encoding. In Fig. (**2**), as for title node, its encoding is

<3/1, a1a1.a1, 3>, then the level of the title node is three.

3.3. XML Data Updating

Before we discuss the XML data updating, let us introduce two important Lemmas.

Lemma 1: If $\frac{c1}{c2} < \frac{d1}{d2}$ (*c1*, *c2*, *d1* and *d2* are greater than 0), then $\frac{c1}{c2} < \frac{c1+d1}{c2+d2} < \frac{d1}{d2}$. For example: $\frac{2}{1} < \frac{2+3}{1+1} < \frac{3}{1}$.

Lemma 2: If $\frac{c1}{c2} < \frac{d1}{d2}$ (*c1*, *c2*, *d1* and *d2* are greater than 0),

Then $\frac{c1}{c2} < \frac{c1+n*d1}{c2+n*d2} < \frac{c1+(n+1)*d1}{c2+(n+1)*d2} < \frac{d1}{d2}$. (*c1*, *c2*, *d1* and *d2* are greater than 0)

For example:

$$\frac{2}{1} < \frac{2+4*3}{1+4*1} < \frac{2+(4+1)*3}{1+(4+1)*1} < \frac{3}{1} \quad (\text{n=4})$$

According to the two lemmas, we can draw a conclusion that $\frac{b}{a} < \frac{b+n*d}{a+n*c} < \frac{b+(n+1)*d}{a+(n+1)*c} < \frac{d}{c}$, so we can insert unlimited nodes between $\frac{b}{a}$ and $\frac{d}{c}$ without changing existing encoding nodes. For example, we can insert unlimited nodes between $\frac{b}{a}$ and $\frac{d}{c}$ without re-encoding.

Example: $\frac{2}{1} < \frac{2+4*3}{1+4*1} < \frac{2+(4+1)*3}{1+(4+1)*1} < \ldots\ldots < \frac{3}{1}$

The following four different cases will be discussed for inserting new nodes:

(1) The new inserting node has right brother without left brother.

The new inserting node has right brother without left brother. If we want to insert a new node B in Fig. (**3**), supposing the encoding of node A is <2/1,a1.a1,2>, the encoding of node C is <3/1,a1a1.a1,3>, then the encoding of node B is $<\frac{2+3}{1+1}$, a1a1.a0, 3>, that is to say, the encoding of node B is <5/2, a1a1.a0, 3>.

2) The new inserting nodes has left brother without right brother.

Inserting a new node D below node B is shown in Fig. (**4**). Node E is the subsequence node of node C according to the preorder traversal. If the encoding of node C is <3/1,a1a1.a1, 3>, and the encoding of node E is <4/1,a1.b1,2>, then the encoding of node D (suppose its node name is "price") is $<\frac{3+4}{1+1}$, a1a1.b1,3>, that is to say, the encoding of node D is <7/2, a1a1.b1,3>.

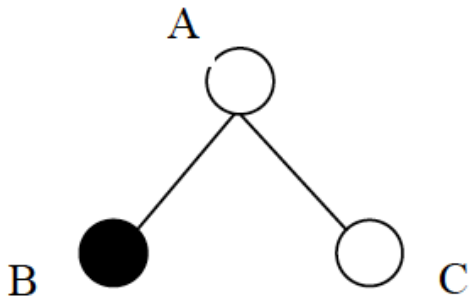

Fig. (3). Has right brother without left brother.

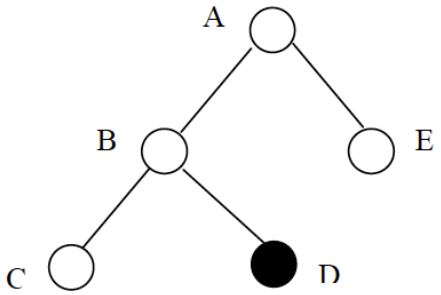

Fig. (4). Has left brother without right brother.

*3) The new inserting nodes both have left brother and right brother as shown below in Fig. (**5**).*

If we want to insert a new node C between node B and node D, suppose the encoding of node B is <3/1, a1a1.a1, 3>, and the encoding of node D is <4/1, a1a1.b1, 3>, then the encoding of node C is $<\frac{3+4}{1+1}$, a1a1.b0, 3>, that is to say, the encoding of node C is <7/2, a1a1.b0, 3>.

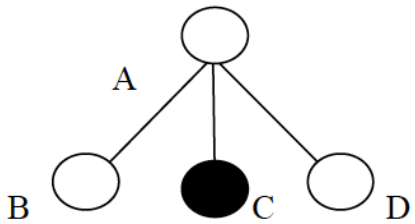

Fig. (5). Both have left brother and right brother.

4) The new inserting nodes have no brother (neither left brother nor right brother).

If we want to insert a new node B below node A as shown in Fig. (**6**), suppose the encoding of node A is 〈3/1, a1a1. a1, 3〉, then the encoding of node B is 〈4/1, a1a1a1. a1, 4〉.

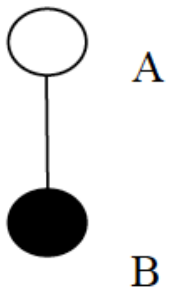

Fig. (6). Have no brother.

Summary: Concerning any of four cases above, if we know the prior node encoding and next node encoding by preorder traversal when a new node is inserted, we can get the node encoding.

4. EXPERIMENT AND ANALYSIS

4.1. Experiment Parameter and Data Set

Experiments are conducted in a single processor PC, Pentium (R) 4 3.0GHZ, 512M of memory, using the Windows XP operating system. The database uses Microsoft SQL SERVER 2000 database, and the programming language Java. Experimental data uses XMark [12] generated XML test data.

4.2. Time Efficiency of MPES

In the first experiment, we use XMark to generate the related data. By setting coefficient to 0.000009, 0.018 and 0.045, corresponding XML documents are 1KB, 2MB and 5MB in size respectively. MPES, fraction encoding [7], and prefix code LSDX [9] were used for experimental comparison. The time efficiency of the three algorithms is shown in Table **1**.

The experimental results show that fraction coding takes the longest time, with MPES, following prefix code. The reason is that MPES and prefix code are known only for one's access to XML node, and fraction coding required scores twice traverse the nodes. That is once before all the descendants of node access, the other in all descendants knot after access point. Due to the combination of advantages of prefix code and fraction coding, MPES coding needs to spend more time than prefix code.

4.3. Data Updating Efficiency of MPES

The second experiment used to detect updated efficiency of MPES coding, using XMark dataset to generate test data. By setting coefficient to 0.000009 and 0.018, corresponding XML documents 1KB and 2MB in size respectively, were generated. MPES, fractional coding [7], and prefix code LSDX [9] were used for experimental comparison. The relevant update time is shown in Table **2**.

Table **2** shows that fraction coding spends the maximum time. MPES is followed by prefix encoding. The reason is that when updating the scores, encoded insertion position of the node should be found, which will consume most of the time of update. While the prefix code encoding implicit the location of position-taking point leading to least time consumption. Due to the advantages of retaining the prefix code, MPES has better coding efficiency than updated scores.

Table 1. The comparison of time and efficiency.

Document Size	MPES/ms	fraction Encoding /ms	LSDX/ms
1kb	326	470	321
2MB	25711	29750	24903
5MB	84665	95546	82723

Table 2. Data updating efficiency of MPES.

Document Size	The Number of Updating Nodes	Time/ms		
		MPES	Fraction Encoding	LSDX
1kb	1	15	32	15
1kb	10	18	62	16
1kb	100	130	156	110
1kb	1000	908	956	859
2M	1	19	41	19
2M	10	20	78	20
2M	100	137	170	125
2M	1000	980	1012	945

4.4. Query Efficiency of FPES

In this experiment, the size of 30MB XML document was generated. The document comprises 1,022,976 nodes, setting six queries cases for different query conditions, as shown in Fig. (**7**). The results are shown in Figure 8.

As can be seen from Fig. (**8**), fractional coding spends the least time, and MPES is inferior to prefix code. The reason is the relatively fast speed of comparison between scores than between prefix. It can also be seen from the table that MPES has a better encoding efficiency than query prefix code. The reason is that MPES fractions is part of the phase-out of the magnitude relationship between nodes, such as query follows the path expression "// SigmodRecord / / articles // authors". According to the coding rules, the value of authors node must be greater than the value of the articles of nodes. Meanwhile, the nodes of the same value articles must be greater than the value of the node SigmodRecord. By doing this, parts of the node which do not meet the requirements will be eliminated, leading to improved query efficiency.

Case	Path name
Q1	/SigmodRecord/issue/volume
Q2	/SigmodRecord/issue/articles/article/title
Q3	//SigmodRecord//articles//authors
Q4	/SigmodRecord/issue/articles/article/authors/author [@position=00@]
Q5	/SigmodRecord/issue/articles/article/initPage[15]
Q6	//article[@id=B001@]/title[Annotated]

Fig. (7). Six query case.

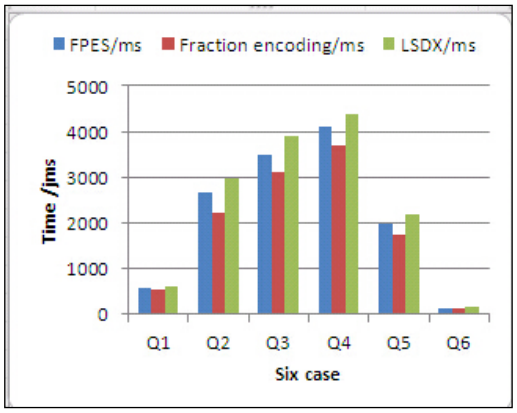

Fig. (8). Query efficiency of the three algorithms.

CONCLUSION

This paper analyzed performance MPES time, updating efficiency and query efficiency from the experimental point of view. The results show that compared with scores, MPES has improved time efficiency and update efficiency of the nodes. Compared with the prefix code, MPES also has improved query efficiency.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

Declared none.

REFERENCES

[1] C. Su, J. He, H. Xi, and W. Mao, "General data interface testing tool based on xml," binggong zidonghua/ ordnance industry automation, vol. 31, no.8, pp. 72-77, 2012.

[2] E. Cohen, H. Kaplan, and T. Milo, "Labeling dynamic XML trees," *SIAM Journal on Computing*, vol. 39, no. 5, pp. 2048-2074, 2010.

[3] J. Zhao, "XML and Database," *Acta Scientiarum Naturalium Universitatis Neimongol*, vol.34, no.3, pp. 345-349, 2003.

[4] C. Zhang, J. Naughton, D. Dewitt, Q. Luo, and L. Guy, "On supporting containment queries in relational database management systems," In: *ACM SIGMOD Conference on Management of Data*, 2001, pp. 425-436.

[5] D. Dao, M. Kha, Y. Wa, and U. Shunsuke, "An XML indexing structure with relative region Coordinate," In: *Proceedings 17th Information Conference on Data Engineering*, 2001, pp. 313-320.

[6] J. D. Ren, X. P. Yin, and X. D. Guo, "A Dynamic Labeling Scheme for XML Document," *Journal of Communication and Computer*, pp. 61-65, 2006.

[7] Y. Sun, J. Gao, T. Wang, and D. Yang, "Update Friendly Fraction Number Encoding Scheme for XML Document," *Computer Science*, vol. 35, no.10A, pp. 165-169, 2008.

[8] X. Yang, D. Li, and W. Zhou, "Extended dewey encoding scheme for reducing update costs for XML data," *Journal of Shenyang Normal University (Natural Science)*, vol. 28, no. 2, pp. 214-217, 2010.

[9] M. Duong, and Y. C. Zhang, "LSDX: a new labelling scheme for dynamically updating XML data," *Australian Computer Society*, pp. 185-193, 2005.

[10] D. C. An, J. Y. Kim, and S. Park, "Access control and labeling scheme for dynamic XML data," *Computer Society*, pp. 329-334, 2008.

[11] G. Xie, "Study on coding schemes of the XML document," *Science Technology and Engineering,* vol. 9, no. 5, pp. 1294-1297, 2009.

[12] A. Schmidt, F. Waas, M. Kersten, M. J. Carey, I. Manolescu, and R. Busse, "XMark: a benchmark for XML data management," In: *Proceedings of the 28th VLDB Conference*, Hong Kong, China, pp. 974-985, 2002.

Received: September 16, 2014 Revised: December 23, 2014 Accepted: December 31, 2014