A XML Document Coding Schema Based on Complete Binary Tree Traversal*

Ying Chen[1], Liyong Wan[1, 2], Cheng Luo[1]

[1]School of Information, Nanchang Institute of science & Technology, Nanchang, Jiangxi Province, China
[2]Advanced Computing and System Laboratory, Zhejiang University, Hangzhou, Zhejiang Province, China
{Dragon_ymd, wanliyong, lunwenzju}@163.com

Abstract - In order to resolve the inefficiency for XML data query and support dynamic updates, etc. This paper has proposed a XML document coding schema based of binary tree traversal (BBCTT). Firstly, the XML document tree has been converted into the binary tree of XML document. Secondly, all nodes of the binary of XML document have been encoded with binary. According to characteristic of XML document tree and binary tree, query of XML document can be implemented. Experiment and analysis show that the coding not only has characteristic of a small storage space and relation determination of logic structure, but also can support dynamic update.

Index Terms - XML, Update, query, code, Binary, Binary tree traversal.

1. Introduction

At present, the XML has involved in various fields, and has been widely applied in all kinds of industries. It can not only be applied in traditional finances, securities, scientific research institutions, medical and government departments for process and exchange of data, but also involves some new industries, such as e-government and e-commerce. With the wide application of XML files, the content of research becomes more and more abundant, which includes XML coding, XML query, XML storage, etc, in these researches in which the XML coding is very important, because it is basis of determining the structural relationship of nodes, and effectively supports structure join of XML query.

Today, the coding scheme of XML has two main kinds: the region coding and the prefix coding. Region coding [1, 2] method is applied according to the physical location of nodes, which structure is made up of the [start, end], the start and end represents respectively the start position and end position of node. The region coding is XML coding method of being wide application, which can not support effectively document update. The region can relieves partially the problem of document update with reservation code space, but it is not flexible. The prefix coding [3, 4] uses nodes path, preserve the path information of codes, can support document update, but it length of code is more long, and the space of code is also more large. The literature [5] proposed the PBiTree coding, which proposes a structural join algorithm based on vertical and lateral resolution. Because the algorithm based on the binary coding is more complex and the intermediate conversion is more numerous, it is still not prefect. The literature [6] proposed efficient coding method, which adopt method of recording node path, can support fast query operation, but it needs more auxiliary information

This paper has proposed a XML document coding schema based on complete binary tree traversal (BBCTT coding). The BBCTT coding method encodes all nodes of the XML document tree with binary, can preserve nodes path information. The BBCTT can save storage space, and support document update.

2. The Coding Method

A. The binary operation of XML document tree

The BBTT coding is based on binary tree, the encoding process needs using the complete binary tree structure, therefore, we can need converted the XML document tree into the complete binary tree.

The XML document is a kind of semi-structure data, which can be denoted by tree model, is called as XML document tree. A XML document tree is shown in Fig.1. In figure 1, node r is the root node of the XML document tree; the lowest six nodes are all leaf nodes.

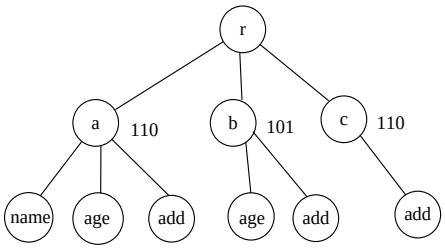

Fig.1. XML document tree

The BBCTT coding method: Firstly, the XML document tree has been converted into the binary tree of XML document. Secondly, all nodes of the binary of XML document have been encoded with binary. The binary tree of XML document can be denoted by T'. There are some inverted steps, as follows:

1) step1: The root node of tree T' is root node of tree T.

2) step2: For all other nodes, if it and its brother nodes totals is no than 2, then denoted the root node's son nodes of tree T' by its son node, otherwise, turns to the step3;

3) step3: Denoting ancestor node by the current node' father node, and bringing the current node and its brother

* This work is partially supported by Education commission of Jiangxi Province Grant # JXJG12-24-2 and Grant #JXJG12-24-3 .

nodes down the next $[\log_2 n]-1$ level, and the empty space are filled with virtual nodes.

4) step4: Repeated step 2 and step3,till all the nodes are converted.

The XML document of the figure 2 is the converted outcome of XML document tree in Fig.1.

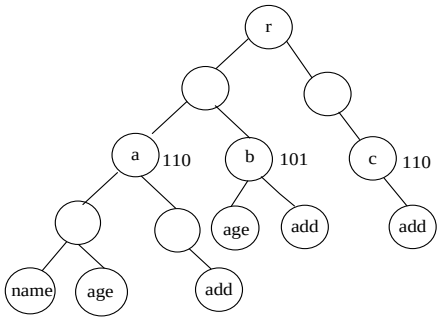

Fig.2.The complete binary tree of XML document tree

B. The encoding method of the BBCTT

All nodes of the XML document are encoded by binary, and the binary digit of each node is equal to its layer. Let encoding of the root node to be 1, for any a node, if the coding of its parents is x, then child node coding of node x are $x\times 2+0/1$, where 0 and 1 represent respectively left child and right child. After the XML document tree is transformed into the binary tree, all nodes of the binary tree need to be encoded, encoding rule should follow these two points:

1) Let current node to be root node, and its coding is 1;

2) Let the current node not to be root, and the code of the current root node can be calculated by $L(u)=2\times fc+x$, where fc represents the parents' coding of node u. The value of node u is 0 or 1, if c is left sub-tree, and the value of node u is 0, otherwise is 1. The encoding model is named as BBCTT coding.

For example, in Figure 2, let the node r to be the root node, and its code is 1, node a code is 100, and the code of the first child is the code of node b and 0, namely, the first code of node b is 1010.

The each code of the BBCTT is a binary string, and each binary bit preserves path branch information. The data can be stored based on the integer of the binary bunch. BBCTT coding is different length code, the level of node is smaller, the length of coding is shorter, conversely, the level of node is bottomer, and the length of coding is longer. In fact, some nodes do not exist in the document tree, which is called as the virtual node. These nodes can be skipped when in encoding.

There algorithm of encoding is shown in algorithm 1.

Algorithm: Encode_tree(T, code)

Input: Let T to be the root node of XML document, and its code is 1.

Output: Coding of every code

```
BEGIN
Printf (code); /* Output the code of current node*/
 ChildNumber=ChildNumber of T;   /*Calculating
          the number of children nodes*/
 temp=log2ChildNumber;
 if (temp! =|tempt|)
    high=temp+1; //*High represents the level of
                  depression*/
else high=temp;
 code=code*(2^high); //The code move the node left high
                  bit when the node is depressed high
                  level, i.e., the end of encoding should
                  be increased 0 of high */
 for (i=1; i<=ChildNumber) // Encoding every node
   c {Temp2=child[i] of T
      Coding_method (Temp2, code); /*Encoding
                        recursively sub-nodes */
      code++; //the value of code +1
   }
END
```

The BBCTT coding can effectively support the document update. The grand-son nodes only need be re-encoded when inserting new node, and others coding are unchanged. For example, if one child node is inserted below node *b*, only need modify the two children node coding of node *b*, and others are unchanged.

3. The Analysis of Coding Properties and Dynamic Updates

A. The coding properties

Property 1: The binary length of each BBCTT coding is equal to level where the node lies in document binary tree.

Proof: It is root node when its length=1, and its coding is 1, the binary length of 1 is 1; Let length=1, and the binary length of L(k) is k; if length $=k+1$, then $L(k+1)=L(k)\times 2+0$ or $L(k+1)=L(k)\times 2+1$,because length of L(k) is k, the binary length of $L(k+1)$ is k+1.

Property 2: If given a coding of node u, its coding of any one ancestor node is front bunch of represented binary of node u.

Proof: According to the principle of the BBCTT encoding, the encoding of any node is obtained from the code of its parents nodes except for the root node, i.e., the coding of the node is the coding of its parents node add one bit: 0 or 1. Therefore, the coding of any node coding is the front bunch of its ancestor node.

Property3: If given a coding *cu* of node u, the coding of ancestor node of its the hth level can be calculated by formula $L(parent)=cu/2k-h$, where $k=\lfloor\log_2 cu\rfloor+1$ (k represents the binary bit number of *cu*).

Proof: According to property 1, binary encoding length of the hth level is h. according to property 2, ancestor coding of the *h*th level of node u is front bunch of represented binary of *cu*. Therefore, ancestor coding of the *h*th level of node u is before h sub-bunch of represented binary of *cu*.

Given node *u* and node *v*, let n1 to be level where node *u* lies in document binary tree, let n2 to be level where node *v* lies in document binary tree. The node *u* is ancestor of node *v*. if and only if the coding of node *u* is equal to the front n1 bit coding of node *v*. The node *u* is parent node of node *v*, if and only if the coding of node *u* is the same as the front n1 bit of coding of node *v*, and n2=n1+1.

B. *Supporting for document dynamic updates*

Let the coding of node u to be *cu*, the children number to be n. if inserting sub-node *v* into node *u*, and the encoding of the new node has three conditions, as follows:

1) Let the coding of node u to be *cu*, the children number to be n. if inserting sub-node *v* into node u, and the encoding of the new node has three conditions, as follows:

If when n=0, i.e., node *u* is leaf node, the new node *v* is taken as child node of node *u* is inserted into the document binary tree, the encoding of the new node *v* is cu× 2, the encoding of others are unchanged.

2) If when n=1, i.e., node *u* is only a child node, the new node v is taken as right-child node of node *u* is inserted into the document binary tree, the encoding of the new node *v* is *cu*×2+1, the encoding of others are unchanged.

3) If when n=1, i.e, node *u* is only a child node, the new node v is taken as left-child node of node *u* is inserted into the document binary tree, the original child encoding of the node *u* is *cu*×2+1, the encoding of the node *v* is *cu*×2, the encoding of others are unchanged.

4) If when n=2, there is no space for insertion of new node, and call algorithm Encode_tree(T, code) to re-encode the sub-tree of node *u*, the encoding of others are unchanged.

5) For deleted nodes, its coding will be deleted, the encoding of others are unchanged.

4. Experiments and Analysis

The current coding schema have some drawbacks, for instance, the region can not effectively support document update, and the Prefix coding can support document update, but it need occupy a great deal of storage space. The BBCTT coding only need modify little node data when document is updated, can save a large number of storage space because of using binary storage.

1) Experiment environment. Hardware system: CPU-Intel Pentium Dual 2.16GHz, RAM 2G; operation system: Windows XP Professional; the development tools: JDK1.6, Eclispe3.4.1 and Java Language.

2) Data set. This experiment takes Xmark as data set of test. The Xmark information is shown in table 1.

TABLE 1 Xmark Information

Name	Document Size (MB)	DTD SIZE (MB)	Node number(ten thousand)	average depth
Xmark	108	5	133.81	7.38

The Xmark has some features, as follows:

(1) It is data set of deep nested structure.

(2) It is well format and meaningful data set.

(3) It requirement of memory is lower, and has nothing to do with size of generated document.

2) The other coding. Region coding XISS[7]and prefix coding LSDX[3]are two classic coding schema.

In experiment, we will compare the BBCTT coding with the region coding XISS and prefix coding LSDX. The length of three different coding is shown in Fig.3.

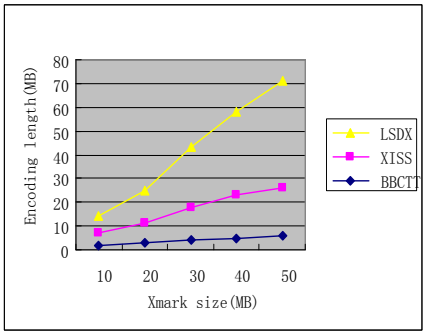

Fig.3. Comparison of coding length

The experiment shows that the BBCTT coding has little space because of using binary encoding.

Next, the experiment of document update is tested. We generate Xmark dataset into six different documents with auto-generation tool of XMLGen. These different documents are respectively named as D1, D2, D3, D4, D5, and D6. The node numbers of these documents are shown in table 2.

TABLE 2 The Number of Nodes of Different Document

D1	D2	D3	D4	D5	D6
620	10204	22403	43256	132423	263590

The secondary encoding rate of node in document tree is shown in Fig.4.

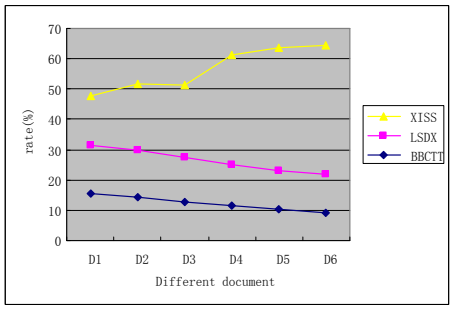

Fig.4. The secondary encoding rate

The figure 4 shows that the XISS is deficient in update ability; the BBCTT and LSDX have effective updated ability.

For XISS, there are many nodes need to be re-encoded when in document update, and the number of the secondary encoding grows larger with the number of nodes; For BBCTT and LSDX, the node number of document is larger, the

secondary encoding rate of node is lower, but the updated ability of BBCTT coding is better than the LSDX coding.

3) Performance analysis.

a) The performance of storage space. The BBCTT coding has much better properties by comparison with the region coding of XISS and the prefix coding of LSDX. The BBCTT coding has little space because of using binary encoding. The XISS encode nodes using a pair of integers, and the LSDX represents the positional relationship with letters, so these need occupy more storage space.

b) The performance of query time. Compared with the region coding of XISS and the prefix coding of LSDX, the BBCTT coding has have many superiority on cost of query time. Because the BBCTT only need scans once XML document tree when it deals with XML document, while the XISS and LSDX coding scan at least twice XML document tree. Therefore from this angle, the CSBTT coding spends less time on query XML document tree.

c) Determinations of relationship. The BBCTT coding can effectively support determination of the relationship among nodes, such as ancestor-grandson, father-son, brotherhood, etc.

5. Conclusion

The BBCTT coding has encoded nodes of the document tree with binary, and each node has a unique coding, which occupy relatively little storage space. It not only supports effectively dynamic update of document, but also has characteristic of a small storage space and relation determination of logic structure.

This is the work in the future:(1) How to decrease costs of converting the XML document tree into the binary document tree;(2) According to the practical application environment, we implement query optimization combined with the relevant indexing mechanism.

Acknowledgment

The author wishes to express thanks to corresponding author Liyong Wan in conducting this study. The study work is supported by Supported by Education commission of Jiangxi Province under Grant no.JXJG12-24-2 and under Grant no.JXJG12-24-3.

References

[1] C Zhang, et al. On Supporting Containment Queries in Relational Database Management Systems. Proc. of SIGMOD, 2001, pp.425-436.

[2] Michael Erdmann, Rudi Studer. How to structure and access XML documents with ontologies. Data & Knowledge Engineering, 2001(36), pp.317-335.

[3] Wang W, Jiang HF, Lu HJ, Jeffrey XY. PBiTree coding and efficient processing of containment joins. Dayal amamritham K, Vijayaraman TM, eds. Proc. of the 19th Int'l Conf.on Data Engineering. Los Alamitos: IEEE Press, 2003, pp.391-402.

[4] L Y Wan, Y Chen. A XML Document Coding Schema Based on Binary Tree Traversal, Computer Application System, vol. 2013,22(2),pp.151-154.

[5] W Wang, H F Jiang, H J Lu. PBiTree coding and efficient processing of containment joins. Dayal U, Ramamritham K, Vijayaraman TM, eds. Proc. of the 19th Int'l Conf.on Data Engineering. Los Alamitos: IEEE Press, 2003, pp.391-402.

[6] H N Wen, X F Liu, W F Li. XML coding scheme for efficient query processing. Computer Application, vol, 30(3), 2010, pp.931-934.

[7] Cohen E, Kaplan H, Milo T. Labeling Dynamic XML Trees. Proc. of PODS, 2002: 271-281.Press, 2003: 391-402.