Number List-based Interval Labeling Scheme for Structural Index Building of Encrypted XML Data

Baolong Liu
School of Computing Science & Engineering, Xi'an Technological University, Xi'an, China,
email:liu.bao.long@hotmail.com

Hua Chen
School of Computing Science & Engineering, Xi'an Technological University, Xi'an, China,
email:liu.bao.long@hotmail.com

***Abstract*-XML documents changing will lead to global index information updating in existing literatures for encrypted XML data querying schemes. There should have a scheme considering updating efficiency of index information for encrypted XML data. The paper presents the number list-based interval labeling scheme (NLBILS) for encrypted XML data. The basic idea is that if there have not enough space available for nodes inserting, the labeling process assigns a number for the sub-tree to be inserted, and then start with a new labeling process for each node in the sub-tree. The labeling result of each node will be consisting of a number list with its parent's nodes label. The proposed scheme makes index information maintenance more efficient, and it is easy to update XML data with decreasing the number of affected nodes to the lowest. The evaluation results show that the proposed scheme supports index information maintenance in an efficient way. The proposed scheme can be used to build structural index information for encrypted XML data query processing.**

Keywords-XML data encryption; XML data query; structural index; number list, range query.

I. INTRODUCTION

With the widely applications of XML, it is necessary to handle sensitive information in XML data, and XML data confidentiality becomes an important issue [1]. The sensitive parts of the XML data have to be protected in case unauthorized access. There are two approaches to protect the sensitive information in XML data. One is using access control mechanism, and the other is using encryption technology, especially XML encryption technology. Most cases, the access control mechanism can be bypassed and encryption technology is a must [2]. When XML data is transmitting through an untrusted channel, it needs encryption technology to protect sensitive information [3, 4]. W3C has published specifications to illustrate the processing of XML data encryption. However, how to query encrypted XML data has not been addressed in XML encryption specification.

Querying encrypted XML data schemes or survey can be found in [4-8]. The basic idea for encrypted XML data query is to build index information for encrypted XML data. Two types of index information are deployed for encrypted XML data. One is the structural index information, and the other is the value index information [9]. Structural index is used to determine the XPath matching any paths in a submitted query. The value index is used to support the range query. These indexes are deployed in either at the server side or client side [9]. Maintaining index at the server side can be found in [5, 6, 7, 10], and maintaining index at the client side can be found in [8, 1].

Two major issues should be considered in a scheme for encrypted XML data query based on index information mechanism. The first is that avoids unnecessary encrypted blocks being decrypted, and most of existing schemes achieved this objective. Considering frequently changing of XML data, the efficiency of index information updating should be considered. However, the second issue has not been taken into account by researchers. Existing literatures indicate that the XML document update will lead to global index information updating. There should have a scheme with considering updating efficiency for index information.

Motivated by problem above, a number list-based interval labeling scheme (NLBILS) is proposed for encrypted XML data in this paper based on interval-based labeling scheme. The basic idea is that if there have not enough space available for inserting, the labeling process assigns a number for the sub-tree to be inserted, and starts with a new labeling process for each node in the sub-tree. The labeling result of each node will be consisting of a number list with its parent's nodes label. Proposed scheme provides spare space for node insertion, and makes management of index information more efficiency. Therefore, it is easy to update XML data without affecting other nodes.

II. NUMBER LIST-BASED INTERVAL LABELING SCHEME (NLBILS)

*A.Interval-based labeling scheme.*The interval-based labeling scheme is described by Li in 2001 [11]. In this scheme, each node is assigned two values: start position value and the end position value. The values are positive numbers during the depth first traverse of an XML data as shown in Fig. 1 [11, 12]. The step size of increment is set as 3 in Fig. 1.

This technique aims at determining if there exists a relationship ascendance/precedence between two given nodes. A pair $(order(x), size(x))$ is associated to each node x in the document in such a way that, for each child node y of x, $order(x) < order(y)$ and $(order(y) + size(y)) \leq (order(x) + size(x))$

It has the following property: $[order(y), order(y) + size(y)] \subset [order(x), order(x) + size(x)]$ if and only if y is the child of x.

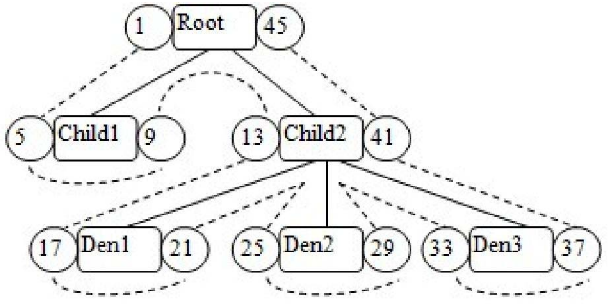

Fig. 1. Example of interval-based labeling

When inserting a child to an existing node, it is always possible to find an interval that satisfies that property above. The computation of a new interval for a sibling between two nodes depends on the available remaining space. However, it is difficult to predict the XML data updating. It means that it is difficult to reserve the space which is used to insert XML data. After data updated several times, the space required to contain inserted data will exceed the reserved space, and the re-labeling of the whole XML data is needed [12].

*B.Number list-based interval labeling scheme (NLBILS).*This section improves the interval-based labeling scheme on labeling the nodes when there have not enough space for inserting. The basic idea is that if there is not enough space for inserting, the labeling process assigns a number for the sub-tree to be inserted, and then start with a new labeling process for each node in the sub-tree. The labeling result of each node will be consisting of a number list with its parent's nodes label.

Definition 1: Number list NL

Let NL be the number list, and $NL = p_1.p_2.\cdots.p_n (n \geq 1)$, where p_i, i is a positive integer.

In definition 1, if $i = 2$, this means that the situation of not having enough space occurs first time. If $i > 2$, the situation of low inserting space has happened several times, and $p_1.p_2.\cdots.p_n (n \geq 1)$ contains the label of parent node. With the number list, it can overcome the space problem of insertion, and avoid to re-label the whole XML data, therefore, it improves the efficiency of XML data updating.

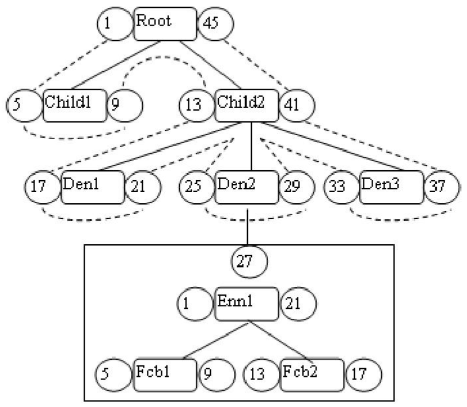

Fig. 2. Example of XML data inserting

Fig. 2 shows an example of XML data inserting. The sub XML data in rectangle will be inserted into original data tree, and there have not enough inserting space. The sub XML data will be assigned a new number 27, which is in the range of interval-based labeling scheme. Each node in sub XML data will be labelled with a new start number. The labeling result of each node in sub-tree is a number list. For example, the label of node "Enn1" is (27.1, 27.21), "Fcb1" (27.5, 27.9), and "Fcb2" (27.13, 27.17).

Definition 2: Node label

The label of each node is denoted as the 3-tuple $(left, right, level)$, where $left$ is the left number list of the node, $right$ is the right number list of the node, and the $level$ is the depth of the node in the XML data.

Lemma 1 (Number list relationship). Given two number lists $s = s_1.s_2.\cdots.s_n$, $r = r_1.r_2.\cdots.r_n$, their relationship can be judged by following rules:

- $s > r$, if $s_1 > r_1$, or $(s_i = r_i) and (s_{i+1} > r_{i+1})$, where $i = 2, \ldots, n$
- $s = r$, if $s_i = r_i$, where $i = 1, \ldots, n$
- $s < r$, if $s_1 < r_1$, or $(s_i = r_i) and (s_{i+1} < r_{i+1})$, where $i = 2, \ldots, n$

Lemma 2 (Nodes relationships). Given two nodes x, y, let $(x_{left}, x_{right}, x_{level})$ and $(y_{left}, y_{right}, y_{level})$ are the node label respectively;

- $x = y$, if $x_{left} = y_{left}$, $x_{right} = y_{right}$, and $x_{level} = y_{level}$
- x is the parent of y, if $x_{left} < y_{left}$, $x_{right} > y_{right}$, and $x_{level} = y_{level} - 1$
- x is the ancestor of y, if $x_{left} < y_{left}$,

and $x_{right} > y_{right}$

- x is the descendant of y, if $x_{left} > y_{left}$, and $x_{right} < y_{right}$
- x is the preceding of y, if $x_{right} < y_{left}$
- x is the following of y, if $x_{left} > y_{right}$

The basic rules for updating are that the lemma 2 is still hold. The updating processes include the insertion process and deletion process.

1. Insert process

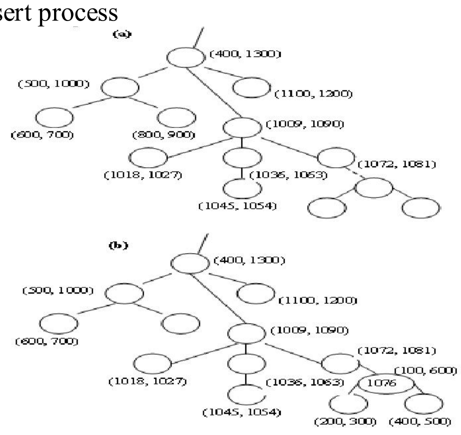

Fig. 4. Insert processing without enough space

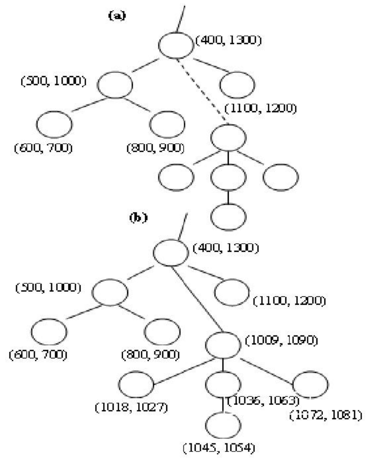

Fig. 3. Insert processing with enough space

This process contains two steps: adding a sub-tree into original XML data, and labeling the inserted sub-tree. Two situations should be considered when labeling inserted sub-tree. If the provided space size is bigger than inserting size, the sub-tree to be inserted with integer numbers should be in the range of the space. If the provided space size is smaller than the insert size, it needs to label the data to be inserted as an interval of parent node.

- Space size>Insert size

Under this situation, the sub-tree to be inserted can be labelled in the range of provided space size. Assuming the space size is 100, and n nodes need to be inserted ($n < 100$), and incremental size $S = [100/2n+1]$. Fig. 3 shows the case. In Fig. 3 (a), there are 5 nodes to be inserted, and space is 100. Because $2n = 10 < 100$, it means that there has enough space to insert these nodes. The incremental size is $S = [100/(2*5+1)] = 9$. The inserted result is shown in Fig. 3 (b), and it still holds the lemma2.

- Space size <=Insert size

The root of the sub-tree to be inserted will be denoted an integer number r, and r is in the range of space. The descendants of r will be labelled with a new data range. Fig. 4 (a) represents three nodes need to be inserted, and space size is smaller than insert size. The root of sub-tree denotes an integer number "1076" in range of space. The descendants of root is labelled with a new start as shown in Fig. 4 (b), and the labelled result of inserted nodes are (1076.100, 1076.600), (1076.200, 1076.300), and (1076.400, 1076.500). The results indicate that the lemma 2 has not been broken.

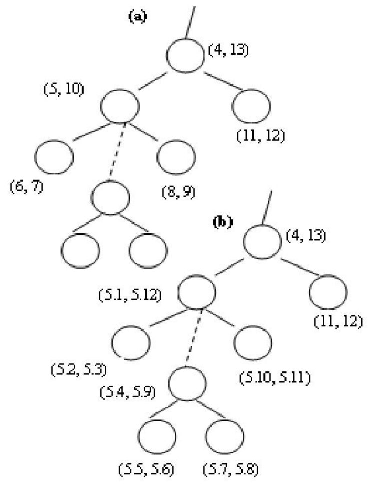

Fig. 5. Insert processing with space=0

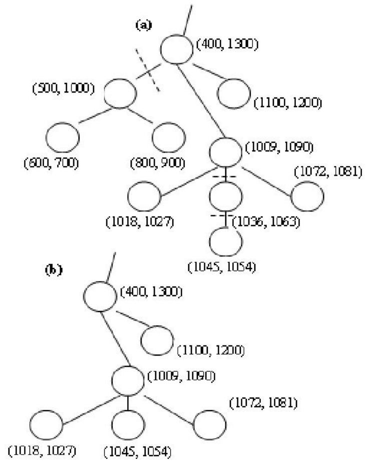

Fig. 6. Delete processing

- Space=0

When space size is equal to 0, the sub-tree to be inserted with its parent will be treated as a new sub-tree [12]. Because the parent of the tree to be inserted has obtained position in the XML data tree, the inserting process is similar to situation of space size smaller than insert size. As shown in Fig. 5 (a), three nodes need to be inserted into data tree, and the insert space is equal to 0. The sub-tree combined with parent node (5,10) as a new sub-tree, and it can obtain insert space as shown in Fig. 5 (b). Although this situation will lead to re-label portions of other nodes, it decreases the affected nodes to the lowest.

2. Delete process

The XML data deletion can be treated as removing a sub-tree from the original XML data. Because the lemma 2 is not broken after deleting a sub-tree, it does not need to do additional performance. As shown in Fig. 6, a sub-tree and an element will be delete in Fig. 6 (a), and Fig. 6 (b) is the deleted result. Fig. 6 shows that lemma 2 will still be hold after deleting process.

III. EVALUATION

With the frequency updating of XML data, it will lead to a changing of index information. The advantage of the proposed scheme considers the efficiency of index information updating. In order to evaluate the efficiency on index information updating, this section gives an XML data, which contains 242 elements based on XMark dataset [13]. The selected scaling factor is 1, and created XML data size is 113MB. Another kind of XML data, which contains 321 elements based on DBLP dataset. Through inserting the same number of elements (from 10 to 60) as shown in Fig. 7, the proposed scheme has been compared to the scheme of Query-Aware, and hash scheme approaches (The details of hash scheme and Query-Aware can be found in [5, 7] respectively). The position of XML data to be inserted is generated randomly.

Based on XMark dataset, the proposed scheme has an average of 199.95ms updating time cost. However, the time cost for Query-Aware and hash scheme are 372ms, and 445.6 respectively. As to DBLP dataset, the average time cost for proposed scheme is 189.65ms. Query-Aware, and hash scheme are 367.4ms and 452.58ms respectively. Although the time cost of index information updating is increasing as the numbers of inserted nodes increasing, the proposed scheme has almost 48% higher efficiency than Query-Aware, and 57% higher efficiency than hash scheme.

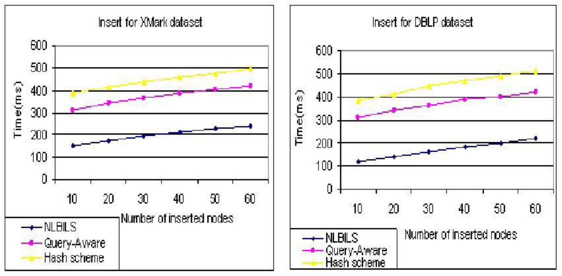

Fig. 7. Efficiency of index information updating

Considering the efficiency of index information updating, especially XML data changing with a high frequency, Query-Aware approach needs to re-label the whole XML data to generate encoding values. Based on number list-based interval labeling scheme, this problem is avoided. This means that the XML data updating cannot lead to re-label the whole index information. Only the elements to be inserted into the original XML data tree will be labeled. Therefore, the proposed scheme has higher efficiency than Query-Aware scheme. Furthermore, hash function based scheme needs to hash each possible XPath when XML data changing, it will cost a huge of time. This means that the hash function based scheme has to compute hash value of each inserted nodes, and re-compute hash value of each affected nodes in the inserting process. Without these additional performances, the presented scheme has higher efficiency than hash function based scheme.

IV. CONCLUSIONS

A new labeling scheme for encrypted XML data is presented to improve the efficiency of index information maintenance which is applied to support encrypted XML data query processing. The proposed scheme makes index information maintenance more efficient, and it is easy to update XML data with decreasing the number of affected nodes to the lowest. The evaluation results indicate that the proposed scheme supports index information maintenance with an efficient way. The proposed scheme can be used to build structural index information for encrypted XML data query processing.

REFERENCES

[1] Y. Yang, W. Ng, H.L. Lau, J. Cheng, An efficient approach to support querying secure outsourced XML information. CaiSE 2006, LNCS 4001, 2006, pp 157-171.
[2] W. Fan, C. Chan, M. Garofalakis,Secure XML querying with security views. In: SIGMOD Conference, 2004, pp 587-598.
[3] R. Agrawal, J. Kiernan, R. Srikant, Y. Xu, Order preserving encryption for numeric data. In: SIGMOD Conference, 2004, pp 563-574.

[4] R. Brinkman, L. Feng, J. Doumen, P. H. Hartel, W. Jonker, Efficient Tree Search in Encrypted Data. Information systems security, 13 (3), 2004, pp 14-21.

[5] L. Feng, W. Jonker, Efficient processing of secured XML metadata, OTM workshop 2003, LNCS 2889, pp 704-717.

[6] H. Wang, L.V.S. Lakshmanan, Efficient secure query evaluation over encrypted XML databases, in proceedings of the 32nd international conference on Very large data bases, Seoul, Korea, 2006, pp 127-138.

[7] J-G. Lee, K-Y. Whang, Secure query processing against encrypted XML data using Query-Aware decryption, Information sciences, 176 (2006), pp 1928-1947.

[8] J. Gao, T. Wang, D. Yang, Xflat: Query-friendly encrypted XML view publishing, information sciences 178 (2008), pp 774-787.

[9] O. Ünay, T.I. Gündem, A survey on querying encrypted XML documents for databases as a service, SIGMOD Record, Vol.37 No. 1, 2008, pp 12-20.

[10] R.C. Jammalamadaka, S. Mehrotra, Querying encrypted XML documents. Proceedings of the IEEE International Database Engineering & Applications Symposium, IDEAS, 2006, pp129-136.

[11] Li, Q., Moon, B., 2001. Indexing and querying XML data for regular path expressions. In proceedings of the VLDB 2001, pp. 361-370.

[12] J. Yun, C. Chung, Dynamoc interval-based labeling scheme for efficient XML query and update processing, The journal of systems and software 81 (2008), pp 56-70.

[13] A. R. Schmidt, F. Waas, M. L. Kersten, D. Florescu, I. Manolescu, M. J. Carey and R. Busse, The XML Benchmark Project. Technical Report INSR0103, CWI, Amsterdam, The Netherlands, 2001. http://monetdb.cwi.nl/xml/ (Accessed on April 2011).