ISSN:0975-9646

A New Way of Generating Reusable Index Labels for Dynamic XML

P. Jayanthi, Dr. A. Tamilarasi
Department of CSE, Kongu Engineering College, Perundurai 638 052, Erode, Tamilnadu, India.

***Abstract*— XML now becomes a standard for the various businesses in the world. Manipulation of the data and evaluating the queries over the data in the XML documents is very important. The indexing schemes use various labeling schemes for the static and dynamic XML document. The performance of the query system depends on the way of getting the data from the document. The persistent labels assigned to the node will help to get the structural information like ancestor-descendant relationship among the nodes of the XML data while querying. In the case of dynamic XML, the document allows the operations like inserting, deleting and updating nodes in the XML document. Particularly for the frequently changed documents, the labels assigned to the nodes will be affected. Most of the existing labeling schemes require the re-computation while changes take place. The proposed system NSAX(New Scheme for Accessing XML) will avoids this problem and also provides the way of using the deleted labels. Moreover, the system provides a way of authentication to the documents. The access control label is assigned to the nodes such that it describes the roles that can access the nodes as well as the level of the nodes in which the nodes present in the document. The experimental results show that the improvement in the performance of the system.**

***Keywords*— access control, dynamic XML, persistent labeling, querying, re-computation.**

I. INTRODUCTION

XML (eXtensible Markup Language) plays an important role in powering the revolution of the data. Its semi structured nature has a power to express complex data as well as simple enough to understand. The format of the XML in the World Wide Web is useful for representing and exchanging the information. By using the appropriate software systems, the data can be manipulated and queried.

The indexing involves with various numbering and labeling schemes. For each of the node, the unique persistent label value as an index will be assigned in some of the labeling systems. In the case of dynamic XML, the nodes can be inserted, deleted and updated. For these changes the re-computation of the labels of existing nodes is needed for every change which takes unnecessary time of the processor. In these cases, instead of evaluating the query the re-computation of the labels will occupy most of the time in the query processing. The proposed method does not require the re-computation of the labels of the existing nodes in the XML document. The parent-child relationship can be inferred from the label values of the nodes. It is useful for structural query processing [5]. It saves the time and space.

In this paper, section 2 describes the existing schemes of labeling and its limitations. Section 3 describes the new proposed system and its details. Section 4 shows the results and performance analysis of the experimentation involved in the proposed system. Finally, the conclusion and future work of the proposed system is described in Section 5.

II. RELATED WORK

A lot of methods have been used for querying over the XML document. A few examples of labeling and numbering of the nodes falls in the categories like Path indexing, Node indexing and structural indexing, etc. The strengths and weaknesses of several node labeling schemes have been discussed by Haw Su - Cheng and Lee Chien – Sing [2]. The comparison of various labeling schemes for ancestor queries have been discussed by Kaplan, H., Milo, T. and Shabo, R [8]. The re-computation of the labeling problem is dealt in the works of Tatarinov, Viglas, Beyer, Shanmugasundaram, Shekita and Zhang [13] and Li and Moon [9]. The proposed methods by Meuss and Strohmaier[10], Grust[4] , Amato, Debole, Rabitti and Zezula[1] have been using path indexing. These are does not providing the way for the update operation in the XML document.

Dong Chan An and Seog Park [17] proposed a method in which a Role-Based Prime Number is used for accessing the nodes. But it need not be a prime number in the proposed scheme NSAX. The numbering methods proposed by Cohen, Kaplan and Milo [3], used a specific code for each node in the document. In the double-bit growth approach, the binary code will be doubled by adding sequence of zeros. Similarly, Tatarinov, Viglas, Beyer, Shanmugasundaram, Shekita and Zhang [13] used Dewey prefix-based numbering scheme. The parent-child relationship can be found in the Li and Moon [9] system uses range information. In the system PBi Tree proposed by Yu, Luo, Meng and Lu [14], the preserved codes are used. If the reserved code is not enough then renumbering is needed. When the changes are made in the document, it will affect the label values of the existing nodes. The addition and deletion of the nodes in the XML document makes a problem in creating persistent labels of the nodes. The performance of the query processing is increased by indexing methods [7]. The XML document for a course is shown in Fig. 1.

```
<?xml version="1.0"?>
<course>
      <subject>
            <title>Global Issues</title>
            <author>Katterien</author>
            <year>2007</year>
       </subject>
.............
.............
.............
</course>
```

Fig. 1: XML document for a course

The problem due to updating is shown in the Fig. 2. The semi structured nature of the XML document allows the user to add or delete the nodes based on the schema. The order in which the siblings to be maintained is not described in the schema. This will leads to updating problem in the labeled document.

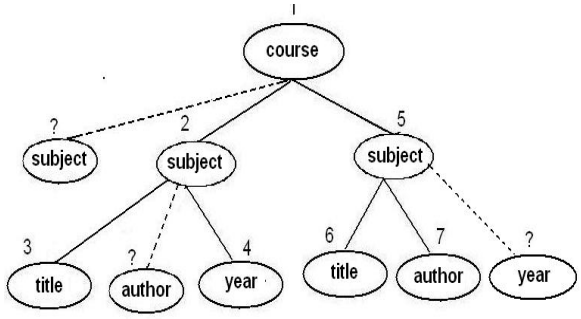

Fig. 2: The effects of updating nodes

But the order of the siblings or child nodes are very important one in the querying process i.e. especially in the structural queries. For example, to get the result of the query such as "Getting the author of the 5th subject in the course document" needs the information about the order of the siblings. The different methods provided in the existing systems are used to overcome this updating problem. But the index size will be increased when the size of the document is increased. The frequently changing document and large size document needs more time for re-computing the label values rather than processing the query. For generating the persistent labels the various combinations of the letters, digits and binary numbers are used. The proposed system NSAX will provide a solution of this re-computation problem as well as an access method of different nodes. The proposed scheme is described in detail in the following sections.

III. NSAX – PROPOSED SYSTEM

The numbers are used in the existing systems like Amato, Debole, Rabitti and Zezula[1], Cohen, Kaplan and Milo [3], Tatarinov, Viglas, Beyer, Shanmugasundaram, Shekita and Zhang [13], Li and Moon[9], Yu, Luo, Meng and Lu[16], Grust[4] for generating the labels of the XML nodes. Some others were used letters in their system like Wang, Jiang, Lu and Yu [14] for labeling scheme. The combination of letters and digits were used in the system LSDX (Labeling Scheme for Dynamically updating XML Data) proposed by Maggie Duong and Yanchun Zhang [18]. In general, by providing the persistent unique labels to the XML nodes will allow the user to insert, delete or update the nodes easily. The persistent labeling scheme should avoid the re-computation. In the proposed scheme NSAX, the digits (0-9), uppercase letters (A-Z) and the lowercase letters (a-z) are used to generate the labels.

A. Algorithm used in NSAX

Definition of a Label of a node (LV): The persistent label of a node includes its self label value and the access control label value which describes the accessibility mode of that node in the document. The access control label also describes the role of the user to which the nodes can be accessed.

LV(*i*) = 0 if "*i*" is the root node of the document

= LV(parent) +self-label$_i$ + access control label$_i$

where, LV(parent) = label value of the parent node of the *i*th node

self-label$_i$ = self-label of *i*th node includes the sibling value of *i*th node retrieved from the sequence in (1) and

access control label$_i$=Access control label of *i*th node is retrieved from the accessibility matrix of the document.

This NSAX is a stack based recursive algorithm which traverses the document in the depth first traversal order and assign the labels. The following part of the algorithm shows how to generating persistent label for a node.

Procedure Generating_Label (node i, parent p, sibling n)
Input:
node i to be labeled in the XML document
parent p denotes the parent node of the ith node
sibling value of ith node according the parent node p
role matrix of the document
access control label matrix of the document
Output:
XML document with labeled nodes
Begin_Generating _Label:
Self-Label(i) =sibling count(i);
access control label(i)=select from accessibility control label matrix;
LV(i)=LV(parent)+Self-Label(i)+
access control label(i);
For each child node (j) of node (i)
Begin-For:
Generating_Label(j, p, n);
End_For:
Return XML document with labeled node(i)
End_Generating_Label:

The following (1) gives the sequence of values used for counting the siblings of a node.

……,000z,001,……,00y,00z,01,…...,09,0A,…,0Z,
0a…….. ,0y,0z,1,2, ……9,A,B,C,...Z.
a,b,c,d,..,z,z0, z1,…z9,zA,……,zZ,
za …….., zz, zz0, zz1,…..….zz9, zzA,…… (1)

Suppose the string "001" is used for a node, then the previous sibling will have the sequence string "000z". Sorting these strings, the string "000z" becomes first. Thus the increased order of the siblings will be maintained. Similarly, the string "zzz" is used as a sequence count of the node, then it's next sibling will have the sequence string "zzz0" and so on. The problem is raised only when the digits are used for numbering the labels. But here the combination of the letters and digits will avoid that re-computation problem.

B. Maintaining Sibling Order

The sequence shown in (1) will produce enough number of labels and can be expanded at any point in any large size document. Using both lowercase, uppercase letters and digits will produce enough number of label values with minimum storage. Thus it will improve the performance of the query processing system. Using the sequence for assigning the label of the siblings, the increased order of the siblings will be maintained. i.e. the sorted of the labels of the siblings will be maintained always.Algorithm used in NSAX

C. Updating Problem – Solution

The existing systems like Amato, Debole, Rabitti and Zezula[1], Cohen, Kaplan and Milo [3], Tatarinov, Viglas, Beyer, Shanmugasundaram, Shekita and Zhang [13], Li and Moon[9], Yu, Luo, Meng and Lu[16], Grust[4] have the problem of re-computation of labels. i.e. these methods do not have the provision of updating the XML nodes with new label values. In the proposed scheme NSAX, this problem is avoided.

The support of updating the nodes will be explained with the Fig. 3. To understand the formulation of self label of a node, the letters and digits except the last digit will be considered. The last digit of the label of a node is used for describing the accessibility of that node. It will be described in the next section. Suppose for the XML document shown in the Fig. 1 is updated. i.e. new 'subject' node is added as the first child node of course. In that case, the string "0z" obtained from (1) will be used as the position of the new 'subject' node. Thus, the self label of the new 'subject' node will become "0z".

Next, a new 'author' node is added in between the siblings of the existing document. The position of the new 'author' node is to be found first. According to the sequence given in (1), there is no string between the strings "1" and "2". In those cases the combination of the label of first sibling node and the string "0" will be used. So the modifications due to the operations like insertion, deletion and updating nodes will not affect the sorted order of the siblings. Thus, the persistent labels need not be changed for the changes in the document when performing these operations.

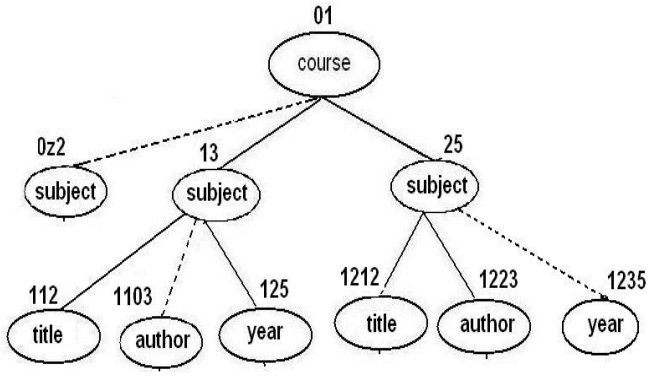

Fig. 3: Updating XML nodes using NSAX

D. Access Control of the nodes

To accessing the nodes the role based prime number is used in [17]. But in NSAX, for accessing the XML document the relationship between roles involved in the business and the levels of the document are determined. i.e. the accessibility control label matrix of an XML document uses the digits which determines the roles who can access that nodes as well as the level of the nodes. Suppose for a document D, the roles R1, R2, R3,.. Rm are assumed. Along with levels the list of roles that can access the nodes in that level is combined. The access control label of a node can be calculated using the matrices given in the Table I and II.

TABLE I
THE SAMPLE ROLE MATRIX

Role Name	List of nodes can be accessed	List of levels in which the node presents
R_1 – CEO	Course\subject\author Course\subject\title ……. ……. Course\syllabus\book Course\syllabus\title ……. …….....	0,1,2
R_2- Director	Course\subject	0,1
……	……	……
R_m-Student	Course\title	0,1

TABLE III
THE SAMPLE ACCESSIBILITY CONTROL LABEL MATRIX

Access Control Label Used	List of roles can access the node	Levels of the node
0	R1, R2, R3, R5	0,1,2
1	R1, R2, R3, R5	0,1,2
2	R1, R2, R3, R5	0,1,2
3	R1, R2, R5	0,1,2
5	R5	0
6	R3, R5	0
7	R1, R5	0

IV. EXPERIMENTAL RESULTS

The NSAX system has been implemented in Java 2 JDK 5.0. Xerces SAX parser is used to manipulate the XML document. The data sets which are similar to the real world applications are considered for the experimentation. So XMark datasets [12] are chosen. They are standard and balanced one. The scaling factors 0.01 to 0.05 are used to generate the different XML documents of different sizes. Genuine Intel CPU, 2140 @1.60GHz, 2.49 GB of RAM on Windows XP system with 75GB hard disk is used for conducting the experiments.

A. Index Size – Analysis

The properties of the XMark data set generated are shown in the Table III.

TABLE IIIII
PROPERTIES OF DOCUMENTS USED

Sl. No.	Scaling Factor	Document Size (MB)	Number of Nodes
1	0.01	1.2	17132
2	0.02	2.3	33140
3	0.03	3.4	50266
4	0.04	4.7	67902
5	0.05	5.6	83533

The index is generated by using the NSAX scheme. To show the effectiveness of the system, a comparison is made with various schemes like LSDX [18], GRP Scheme [19], SP Scheme [3] and shown in the Table IV. It clearly shows that the documents will have the index with reduced size using NSAX.

TABLE IVv
COMPARISON OF THE SIZE OF THE INDEXES

Document Size (MB)	Index Size used (MB)				
	NSAX	***Existing***	***LSDX***	***GRP***	***SP***
1.2	0.16	0.24	0.17	0.64	1.36
2.3	0.34	0.44	0.41	1.20	4.64
3.4	0.56	0.65	0.76	2.00	10.24
4.7	0.82	0.86	1.17	2.72	17.92
5.6	1.09	1.16	1.63	3.44	27.04

To know the performance of the system, a graph is drawn using the results of the comparison of the size of the indexes using GRP[19], Existing Method[17], and NSAX schemes. SP Scheme is not included in the graph because it needs a very large amount of space. The final graph obtained is shown in the Fig. 4. It is clear that the proposed scheme will reduce the space, when compared with the GRP, Existing Method and LSDX schemes respectively.

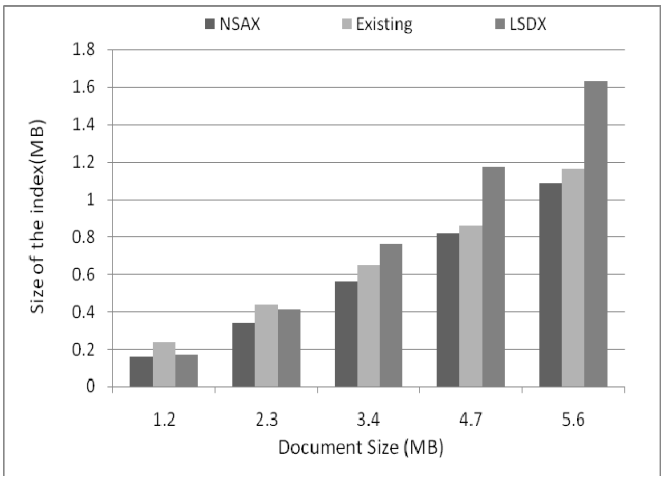

Fig. 4: Comparison of the index sizes

B. Time Analysis for generating labels

The time taken for the various schemes will differ according to the algorithm used. To know the performance of the system, a comparison is made among the various systems like LSDX and NLSX schemes. The results of the comparison are shown in the Table V.

TABLE V
COMPARISON OF TIME TAKEN FOR GENERATING THE LABELS

Sl.No.	Document Size (MB)	LSDX (sec)	NSAX (sec)
1	1.2	1.09	0.250
2	2.3	1.56	0.375
3	3.4	2.25	0.500
4	4.7	2.79	0.609
5	5.6	3.28	0.687
6	6.9	3.90	0.812
7	8.0	4.54	0.938
8	9.2	5.09	1.047
9	10.3	5.59	1.203
10	11.4	6.18	1.297

The graph obtained for the same is shown in the Fig. 5. From the analysis, it is known that 77% of the time is reduced by using NSAX for generating the labels.

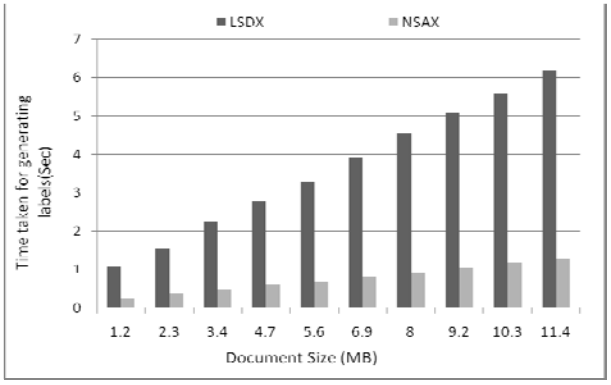

Fig. 5: Comparison of the time taken by various schemes

C. Reusability of the labels

The reusability of the labels of the deleted leaf nodes and the sub trees can be tested. The position of the node to be deleted is chosen randomly. Then at the deleted positions the new leaf nodes or sub trees are inserted. The result is shown in Fig. 6.a and 6.b

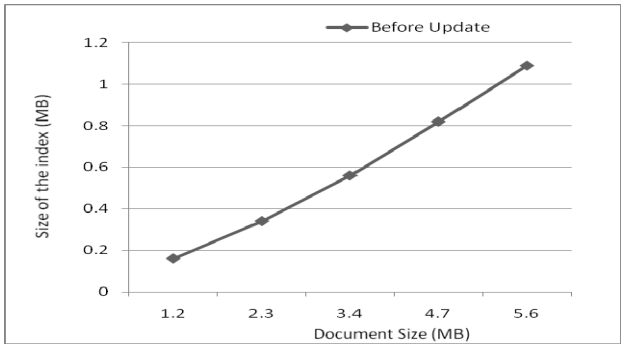

Fig. 6.a: Comparison of the index using NSAX before updation

For each time, 10000 labels are generated by the deletion operation. After insertions and deletions the size of the index of the documents were tested. The proposed system does not increase the size of the indexes in any of the cases.

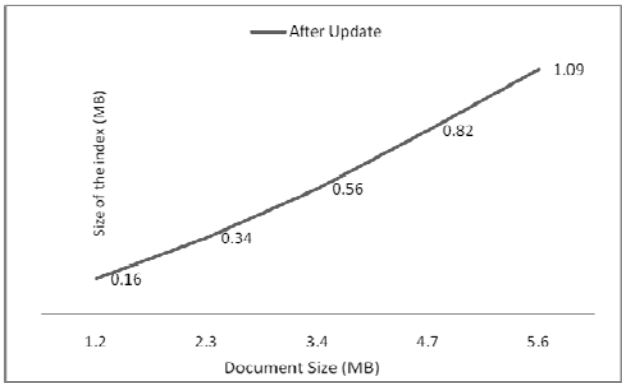

Fig. 6.b: Comparison of the index using NSAX after updation

D. Access Control Scheme

The accessibility of the node will be checked with the help of two matrices namely, Role Matrix and Access Control Label Matrix. The level of a node is explicitly described in the existing scheme [17]. But in NSAX, a single number is enough for describing the roles as well as the levels the user can access the nodes in the XML document. And also the changes in the roles can be done easily with NSAX. To query the XML document, the content of the nodes and the path expression is stored in Oracle database format. The corresponding label value is also stored along with them. From that the system first gets the role of the user. The query will be given interms of XPath expression. If the access control label matrix permits, then the content of the nodes against the given XPath expression is retrieved by searching the label values of the nodes.

V. CONCLUSION AND FUTURE WORK

The persistent labeling schemes do not have the re-computation of the existing nodes when the changes take place. To avoid this problem, for generating the labels the digits, lowercase and uppercase alphabets were used. Moreover, the results shows that the size of the index generated and time taken for generating the labels will be reduced when compared to the existing system. Also the access control scheme is introduced which is based on the Role Matrix and the Access Control Label Matrix of the document. Thus the performance of the system will be improved. In future, accessibility of the clustering of the nodes and ranking the elements may also be added for further improvement.

REFERENCES

[1] Amato, G., Debole, F., Rabitti, F. and Zezula, P. , "Yet Another Path Index for XML Searching", in Proceedings of ECDL 2003 Research and Advanced Technology for Digital Libraries, 7th European Conference, Trondheim, Norway, 2003.

[2] Haw Su-Cheng and Lee Chien-Sing, "Efficient Preprocesses for Fast Storage and Query Retrieval in Native XML Database", IETE Technical Review, Vol. 26, Issue 1, Feb. 2009.

[3] Cohen, E., Kaplan, H. and Milo, T., " Labelling dynamic XML trees", in Proceedings of PODS 2002.

[4] Grust, T., "Accelerating XPath Location Steps", in Proceedings of the 2002 ACM SIGMOD

[5] http://www.w3.org/TR/xquery/.

[6] IBM Corporation XML data generator http://www.alphaworks.ibm.com/tech/xmlgenerator, International Conference on Management of Data, Madison, Wisconsin, ACM 2002.

[7] Kaelin, M. , "Database Optimization: Increase query performance with indexes and statistics", Tech Republic, 2004.

[8] Kaplan, H., Milo, T. and Shabo, R, "A Comparison of Labelling Schemes for Ancestor Queries", http://www.math.tau.ac.il/~haimk/papers/comparison.ps

[9] Li, Q. and Moon, B., "Indexing and Querying XML Data for Regular Path Expressions", in Proceedings of VLDB 2001.

[10] Meuss, H. and Strohmaier, M. C., "Improving Index Structures for Structured Document Retrieval" in 21st BCS IRSG Colloquium on IR, Glasgow, 1999.

[11] Michael Ley. DBLP database web site. http://www.informatik.uni-trier.de/~ley/db/, 2003.

[12] Schmidt, A., Waas, F., Kersten, M., Carey, J. M.,Manolescu, I. and Busse, R., "XMark: A Benchmark for XML Data Management", in Proceedings of VLDB 2002.

[13] Tatarinov, I., Viglas, S., Beyer, K., Shanmugasundaram, J., Shekita, E. and Zhang, C., "Storing and Querying Ordered XML Using a Relational Database System", in Proceedings of SIGMOD 2002.

[14] Wang, W., Jiang, H., Lu, H. and Yu, X. J., "PBiTree Coding and Efficient Processing of Containment Joins", in 19th International Conference on Data Engineering, 2003 Bangalore, India.

[15] XMARK: The XML-benchmark project. http://monetdb.cwi.nl/xml 2002.

[16] Yu, X. J., Luo, D., Meng, X. and Lu, H., "Dynamically Updating XML Data: Numbering Scheme Revisited", in World Wide Web: Internet and Web Information System, 7, 2004

[17] Dong Chan An, Seog Park (2008): Access Control Labeling Scheme for Efficient Secure XML Query Processing, in KES 2008, Part II, pp 346-353, Springer –Verlag Berlin Heidelberg, 2008.

[18] Maggie Duong and Yanchun Zhang, "LSDX: A new labelling scheme for dynamically updating XML data", in the Proceedings of 16th Australasian Database Conference, Volume 39, 2005

[19] Jiaheng Lu, Tok Wang Ling, "Labeling and Querying Dynamic XML Trees", APWeb 2004