Journal of Computer Science 7 (3): 421-426, 2011
ISSN 1549-3636
© 2011 Science Publications

A New Method of Generating Index Label for Dynamic XML Data

Jayanthi Paramasivam and Tamilarasi Angamuthu
Department of CSE, Kongu Engineering College,
Perundurai 638052, Tamilnadu, India

Abstract: Problem statement: The processing of the XML queries needs an efficient indexing method. The index generation includes the labeling of the nodes. But for the dynamic data, when the changes are made, the re-computation of the labeling is needed. **Approach:** Based on the structural indexing used the performance of the query processing system will be affected. The dynamic XML document allows inserting, deleting and updating operations. **Results:** Suppose for the frequently changed documents, the indexes will also be changed. i.e., it requires the re-computing the labels frequently. This leads to an inconvenient system. To avoid this problem, in the scheme New Labeling Scheme for XML (NLSX), only the small, capital letters and digits are used to generate persistent labels for the nodes in the document. In the proposed system New Labeling Scheme for XML using Unicode Characters, characters from Unicode Characters (NLSXU) are used. Thus it can provide more combinations of the characters for persistent labeling the nodes in the document so that it will very much reduce the space needed to store the labels. Using the proposed scheme NLSXU, the index size of the real world data sets will be greatly reduced by 81% of the existing scheme NLSX. The results shows that the proposed scheme NLSXU will reduce the size of the indexes of the synthetic data sets up to 26, 34, 71 and 95% than the NLSX, LSDX, GRP and SP schemes respectively. Also when compared to LSDX scheme, the NLSXU will reduce the time taken for generating the labels by 96 and 80% for the real world datasets and the synthetic data sets respectively. **Conclusion:** Finally when compared to NLSX scheme, the NLSXU will reduce the time taken for generating the labels by 66 and 15% for the real world datasets and the synthetic data sets respectively. Thus it will improve the performance of the query system.

Key words: Dynamic XML, persistent labeling, semi structured, synthetic data sets, indexing scheme, unique persistent label, generating labels, ASCII system, CJK unified ideographs

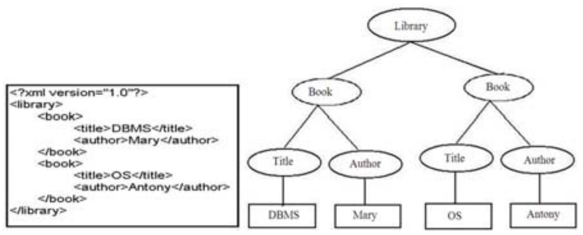
INTRODUCTION

In these wonderful and privileged times, none of this would be possible without data. XML (eXtensible Markup Language) is powering the revolution of the data. In the Internet, the documents used XML format is useful for representing and exchanging the information. The indexing methods of the XML documents involves with various numbering and labeling schemes. While processing the XML document, for each of the node, the unique persistent label value as an index will be assigned so that it gives the fast access to these nodes while querying the data. The dynamic nature of the database needs to the change labels of the existing nodes each time. For the frequent updating of the document, instead of evaluating the query the re-computation of the labels will occupy most of the time in the query processing. The parent-child relationship can be inferred from the label values of the nodes. It is useful for structural query processing (http://www.w3.org/TR/xquery/). The proposed method provides the ways for inserting, deleting and updating nodes without changing their label values. The experimental results show an improvement in the system.

MATERIALS AND METHODS

For processing the query over the XML document, various methods have been proposed. The different ways of labeling and numbering of the nodes are used in these schemes. A few examples are Path indexing, Node indexing and structural indexing. The strengths and weaknesses of several node labeling schemes have been discussed by Su-Cheng and Chien-Sing (2009). The comparison of various labeling schemes for ancestor queries has been discussed by Kaplan *et al*. (2002). The re-computation of the labeling problem present in the study of Tatarinov *et al*. (2002) and Li and Moon (2001). The methods proposed by Meuss and Strohmaier (1999); Grust (2002); Amato *et al*. (2003), Haw and Lee (2008a, 2008b), Mirabi *et al*. (2010) are using path indexing.

Corresponding author: P. Jayanthi, Kongu Engineering College, Department of CSE, Perundurai 638052, Tamilnadu, India



```
<?xml version="1.0"?>
<library>
      <book>
            <title>DBMS</title>
            <author>Mary</author>
      </book>
      <book>
            <title>OS</title>
            <author>Antony</author>
      </book>
</library>
```

Fig. 1: A Sample XML Document and its tree representation

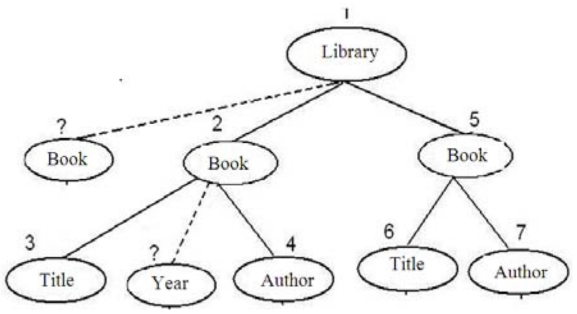


Fig. 2: The effects of updating nodes

These methods do not provide the update operation in the XML document. The performance of the query processing is increased by indexing methods (Kaelin, 2004). The XML document for a library system and its tree representation is shown in Fig. 1. In this sample document, library is the root node present at the level 0 and has two child nodes books in the next level 1 and so on.

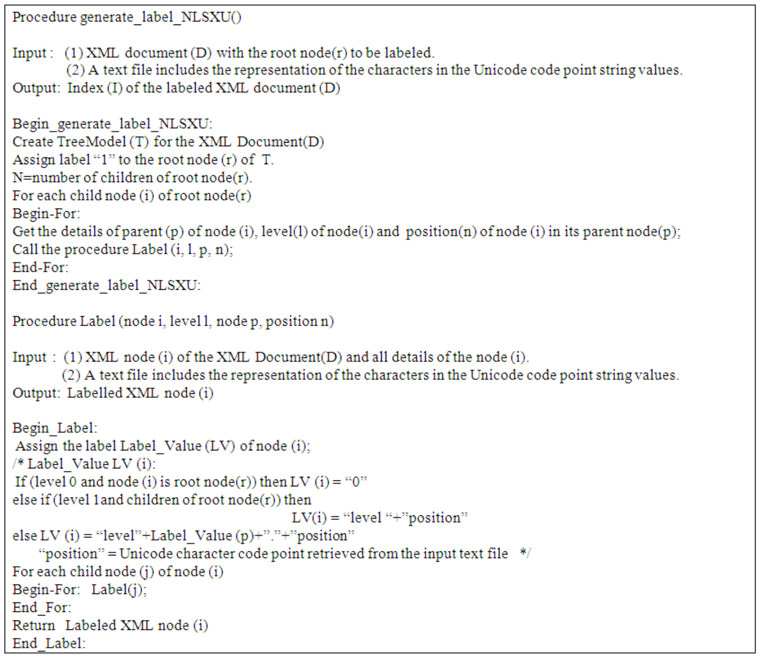
The numbering methods based on the prefix values were proposed by Cohen *et al*. (2002), in which a specific code will be assigned for each node in the document. The binary code will be doubled by adding sequence of zeros, in the double-bit growth approach. Tatarinov *et al*. (2002) used Dewey prefix-based numbering scheme. The structural information i.e., parent-child relationship can be found in the Li and Moon (2001) system which uses range information. The preserved codes are used in the prefix-based system PBi Tree proposed by Yu, Luo, Meng and Lu (Jeffrey *et al*., 2004). When the reserved code is not enough then renumbering is needed. While updating takes place, it will affect the label values of the existing nodes.

The problem due to updating is shown in the Fig. 2. The determinant way of adding or deleting nodes is not described in the XML Schema. This will leads to updating problem in the labeled document. The order in which the siblings to be maintained is not described in the schema.

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 title of the7th book in the library document" needs the information about the order of the siblings. The frequently changing document and large size document needs more time for re-computing the label values rather than processing the query. The proposed system NLSXU will provide a solution of this re-computation problem. The queries involved with structural pattern can be effectively processed by using the indexing scheme which uses persistent labels for the document nodes in the XML document.

NLSXU-proposed system: In the existing systems, Amato *et al*. (2003), Cohen *et al*. (2002), Tatarinov *et al*. (2002); Li and Moon (2001); Jeffrey *et al*. (2004); Lu et al.(2004), Grust (2002), Haw and Lee (2008a, 2008b), Meghdad Mirabi *et al*. (2010) have used numbers for generating the labels of the XML nodes. Some others like Wang *et al*. (2003) used letters for labeling scheme. Rakhmadi et al. (2010) proposed a labeling approach for images. Doung and Zhang (2005) proposed a system LSDX (Labeling Scheme for dynamically updating XML Data). In that, both the combination of letters and digits for generating the labels is used.

Unicode characters in labels: In the NLSXU, the labels will be generated using letters, digits and Unicode characters (Unicode Consortium, 2010). The combination of letters, digits, will be different from other methods. In NLSXU, the labels are generated from the set of characters. The set includes the digits (0-9), uppercase letters (A-Z), lowercase letters (a-z) and few of the characters in the Unicode character set. The Unicode Standard contains a set of unified Han ideographic characters. The term "CJK" refers to the Chinese, Japanese and Korean languages. The Unicode value of the characters in the set will be taken into account such that it maintains the order of the siblings of a node in the XML document. In this experimentation, for generating labels, the characters in the Unicode Character set are chosen arbitrarily. i.e., the characters in ASCII system (10 digits, 26 uppercase letters, 26 lowercase letters) and 20901 characters from CJK unified ideographs are used. Thus the size of the index will be greatly reduced.



```
Procedure generate_label_NLSXU()

Input :   (1) XML document (D) with the root node(r) to be labeled.
          (2) A text file includes the representation of the characters in the Unicode code point string values.
Output:  Index (I) of the labeled XML document (D)

Begin_generate_label_NLSXU:
Create TreeModel (T) for the XML Document(D)
Assign label "1" to the root node (r) of T.
N=number of children of root node(r).
For each child node (i) of root node(r)
Begin-For:
Get the details of parent (p) of node (i), level(l) of node(i) and position(n) of node (i) in its parent node(p);
Call the procedure Label (i, l, p, n);
End-For:
End_generate_label_NLSXU:

Procedure Label (node i, level l, node p, position n)

Input :  (1) XML node (i) of the XML Document(D) and all details of the node (i).
         (2) A text file includes the representation of the characters in the Unicode code point string values.
Output:  Labelled XML node (i)

Begin_Label:
 Assign the label Label_Value (LV) of node (i);
/* Label_Value LV (i):
 If (level 0 and node (i) is root node(r)) then LV (i) = "0"
else if (level 1and children of root node(r)) then
                                         LV(i) = "level "+"position"
else LV (i) = "level"+Label_Value (p)+"."+"position"
    "position" = Unicode character code point retrieved from the input text file   */
For each child node (j) of node (i)
Begin-For:   Label(j);
End_For:
Return   Labeled XML node (i)
End_Label:
```

Fig. 3: NLSXU-algorithm for generating labels

Proposed algorithm-NLSXU: As shown in the Fig. 3, the algorithm needs several steps for generating the persistent labels for the XML document in NLSXU. It is a recursive algorithm which uses the depth first traversal order and assigns the labels.

Maintaining the order of the siblings: The labeling sequence in the proposed system is different from the other existing systems such as LSDX by Doung and Zhang (2005). The previous study NLSX uses only the alphabets including both upper case and lower case and also numbers. The new idea for identifying the siblings of a node will be introduced in this proposed NLSXU Scheme. The combination of the ASCII characters and CJK Unicode Ideographs will be made using the increased order of the code point value of the character in Unicode Character Set. The UTF-8 version 5.2 will be used for this purpose. The Unicode code point value of the characters used are given below:

Digits (0-9):	"\u0030" to "\u0039"
Uppercase letters (A-Z):	"\u0041" to "\u005A"
Lowercase letters (a-z):	"\u0061" to "\u007A"
CJK Unified Ideographs (CJK letters):	"\u4E00" to "\u9FCB"

Thus the sequence of characters used for counting the siblings of a node will be given in the Eq. 1:

-(0,0,L)-(0,0,CJK)-(0,D)-(0,U)-(0,L)-(0,CJK)
-(D)-(U)-(L)-(CJK)-(LCJK,D)-
(LCJK,U)-(LCJK,L)-(LCJK,CJK)
-(LCJK,LCJK,D)- (LCJK,LCJK,U (1)

Where:

D	=	The range 0-9
U	=	The range a-z
L	=	The range A-Z
CJK	=	The CJK Unified Ideographs

LCJK refers to the last character in CJK Unified Ideographs. In CJK Unified Ideographs, only the first 20901 characters only are used. The code point of the last character in CJK set considered is "\u9FCB". The characters present in each pair of parenthesis will be varied depend on the notation used. For example, in a pair within a parenthesis (LCJK, D) will make the number of combinations using the last character of the CJK Unicode character set and the digits range from 0 to 9. The sequence shown in Eq. 1, is very carefully designed for such documents. i.e., at any point the labels can be expanded in both left and right directions by using this sequence; thus the problem of re-computation is avoided in dynamic XML documents.

Solution for dynamic change in the document: The problem of re-computation of labels exists in the labeling systems proposed by Tatarinov *et al*. (2002);

Li and Moon (2001); Meuss and Strohmaier (1999); Grust (2002), Amato *et al*. (2003), Haw and Lee (2008a, 2008b). According to NLSXU, when a new first child node or new left most child is added in the existing document, the previous count of the old first child node or old left most child will be retrieved from the Eq. 1. And it will be used as the position of the new first child node or new left most child. Thus, the persistent labels of the existing document need not be changed for the changes in any part of the document. i.e., the position of the child node of a node will be identified by the sequence mentioned in the Eq. 1. So, from the letters and digits of the label, the ancestors and descendants of a node will be identified easily.

RESULTS AND DISCUSSION

The proposed system NLSXU has been implemented in Java 2 JDK 5.0. For manipulating the nodes in the XML document, Sun Microsystems SAX parser is used. The XMark datasets (IBM Corporation, 1999; Schmidt *et al*., 2002) are used for the experiments. The scaling factors 0.01-0.5 are used for generating the different size XML documents. Additionally, the experiments were conducted with the various XML repositories obtained from the XML data repository (Miklau, 2001), Digital Bibliography Library Project (DBLP) (Ley, 2003) and the Protein Sequence Database), a bench mark dataset, namely, XMark. The dataset TreeBank, which has many more distinct elements and deeper structure, is used for experiments. The dataset SwissProt is also used for the same. On Genuine Intel CPU, 2140 @1.60GHz, 2.49 GB of RAM on Windows XP system with 75GB hard disk the experiments were conducted to analyze the performance of the proposed system.

Analysis of the index sizes: While comparing, it is clear that the proposed scheme NLSXU will reduce the space for the synthetic data sets up to 26, 34, 71 and 95% with NLSX, LSDX, GRP and SP schemes respectively as shown in the Fig. 4. The size of the index of the real world datasets using the methods NLSX and NLSXU can be compared and the results are shown as in the Fig. 5. Using the proposed scheme NLSXU, the index size is greatly reduced by 81% of the existing scheme NLSX.

Time taken for generating labels: From the analysis for the synthetic datasets as shown in the Fig. 6, it is found that 80 and 15% of the time is reduced by using NLSXU for generating the labels when compared with LSDX and NLSX schemes respectively. Similarly, for the real world datasets, the graphs as shown in the Fig. 7, it is found that a good amount of time (96 and 66%) is reduced by using NLSXU for generating the labels for the real world datasets when compared to the schemes LSDX and NLSX respectively.

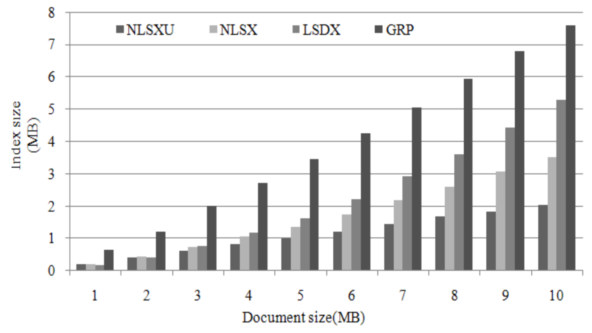


Fig. 4: Index size of the synthetic datasets

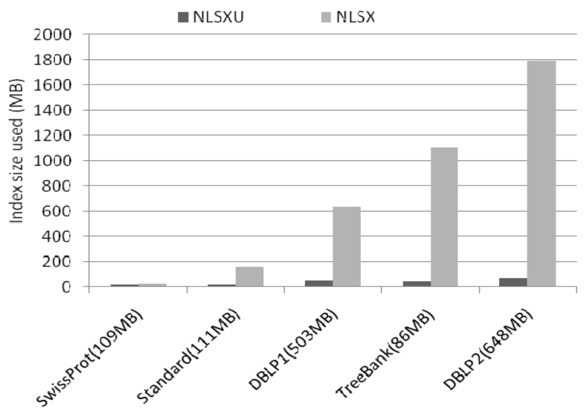


Fig. 5: Index size of the real world datasets

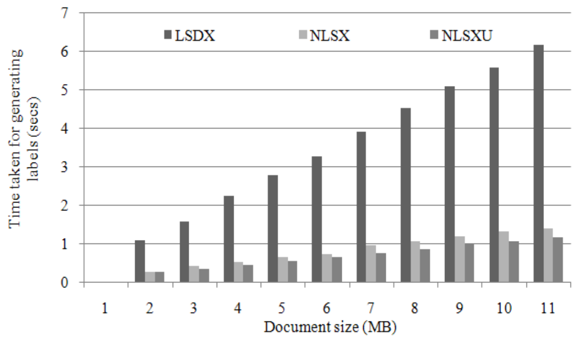
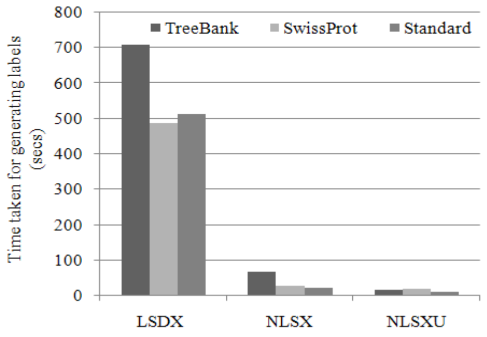


Fig. 6: Time taken for generating labels by various schemes for synthetic datasets



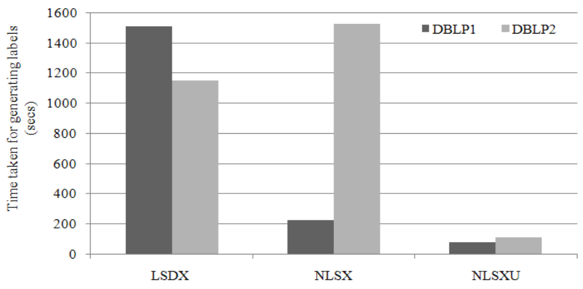


Fig. 7: Time taken for generating labels by various schemes for the real datasets

CONCLUSION

One of the issues in the dynamic XML is the need for re-computation of the labels. To avoid such problem, in the new proposed scheme NLSXU uses a different sequence using unicode characters for labeling the siblings or child nodes. Additionally, the label provides the information about the ancestor and descendant relationship in the document. The comparison of the results for the real-world and synthetic dataset shows that the size of the index generated and time taken for generating the labels will be reduced to a greater amount. Also the clustering of the results and ranking the elements may also be added for further improvement. Also the label compression techniques may be improving the performance.

REFERENCES

Amato, G., F. Debole, F. Rabitti and P. Zezula, 2003. Yet Another path index for xml searching. Proceeding of the 7th European Conference on ECDL, Research and Advanced Technology for Digital Libraries, Aug. 17-22, Trondheim, Norway, pp: 176-187.

Cohen, E., H. Kaplan and T. Milo, 2002. Labeling dynamic XML trees. Proceedings of the 21st ACM Sigmod-Sigact-Sigart Symposium on Principles of Database Systems, (PODS'02), ACM New York, NY, USA., pp: 271-281. DOI: 10.1145/543613.543648

Doung, M. and Y. Zhang, 2005. LSDX: A New Labelling scheme for dynamically updating XML data. Proceeding of the 16th Australasian Database Conference, (ADC'05), University of Newcastle, Newcastle, Australia, pp: 185-193.

Grust, T., 2002. Accelerating XPath location steps. Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data, (SIGMOD'02), ACM New York, NY, USA., pp: 109-120. DOI: 10.1145/564691.564705

Haw, S.C. and C.S. Lee, 2008a. TwigINLAB: A decomposition-matching-merging approach to improving XML query processing. Am. J. Applied Sci., 5: 1199-1205. DOI: 10.3844/ajassp.2008.1199.1205

Haw, S.C. and C.S. Lee, 2008b. TwigX-Guide: Twig query pattern matching for XML trees. Am. J. Applied Sci., 5: 1212-1218. DOI: 10.3844/ajassp.2008.1212.1218

IBM Corporation, 1999. XML data generator. Progress Software Corporation http://www.stylusstudio.com/xml_generator.html

Jeffrey, Y.X., D. Luo, X. Meng and H. Lu, 2004. Dynamically Updating XML Data: Numbering Scheme Revisited. World Wide Web, 8: 5-26. DOI: 10.1023/B:WWWJ.0000047377.03543.64

Kaelin, M., 2004. Database Optimization: Increase query performance with indexes and statistics. Tech Republic. http://www.techrepublic.com/forum/discussions/14-148063-2277326

Kaplan, H., T. Milo and R. Shabo, 2002. A comparison of labeling schemes for ancestor queries. Proceedings of the 13th annual ACM-SIAM symposium on Discrete algorithms, (SODA'02), Society for Industrial and Applied Mathematics Philadelphia, PA, USA., pp: 954-963.

Ley, M., 2003. The DBLP Computer Science Bibliography. University Trier. http://www.informatik.uni-trier.de/~ley/db/

Li, Q. and B. Moon, 2001. Indexing and Querying XML data for regular path expressions. Proceedings of the 27th VLDB Conference, (VLDB'01), San Francisco, CA, USA pp: 361-370.

Meuss, H. and M.C. Strohmaier, 1999. Improving index structures for structured document retrieval. Proceeding of the 21st BCS IRSG Colloquium on IR, Apr. 19-20, University of Munich, Germany, pp: 1-14.

Miklau, G., 2001. UW XML Repository, http://www.cs.washington.edu/research/xmldatasets

Mirabi, M., H. Ibrahim, A. Mamat, N.I. Udzir and L. Fathi, 2010. Controlling label size increment of efficient XML encoding and labeling scheme in dynamic XML update. J. Comput. Sci., 6: 1529-1534. DOI: 10.3844/jcssp.2010.1529.1534

Schmidt, A., F. Waas, M. Kersten, J. Carey and M. Manolescu *et al*., 2002. XMark: A Benchmark for XML Data Management. Proceedings of the 28th international conference on Very Large Data Bases, (VLDB'02), VLDB Endowment, Singapore, pp: 974-985.

Schmidt, A.R., F. Waas, M.L. Kersten, D. Florescu and I. Manolescu *et al*., 2001. The XML benchmark project. Centre for Mathematics and Computer Science-The Netherlands. http://portal.acm.org/citation.cfm?id=869004&pref layout=flat

Su-Cheng, H. and L. Chien-Sing, 2009. Node Labeling Schemes in XML Query Optimization: A Survey and Trends. IETE Tech. Rev., 26: 88-100. DOI: 10.4103/0256-4602.49086

Tatarinov, I., S. Viglas, K. Beyer, J. Shanmugasundaram and E. Shekita *et al*., 2002. Storing and querying ordered XML using a relational database system. Proceedings of the 2002 ACM SIGMOD international conference on Management of data, (SIGMOD'02), ACM New York, NY, USA., pp: 204-215. DOI: 10.1145/564691.564715

Wang, W., H. Jiang, H. Lu and X.J. Yu, 2003. PBiTree coding and efficient processing of containment joins. Proceeding of the 19th International Conference on Data Engineering, Mar. 05-08, Bangalore, India, pp: 391-391.