Proceeding of the International Conference on Advanced Science, Engineering and Information Technology 2011

Hotel Equatorial Bangi-Putrajaya, Malaysia, 14 - 15 January 2011

ISBN 978-983-42366-4-9

XML Labeling Schemes for Dynamic Updates: Strengths and Limitations

Samini Subramaniam[1], Su-Cheng Haw[2], Poo Kuan Hoong[3]

Faculty of Information Technology

Multimedia University

Cyberjaya, Malaysia

samini.subra@mmu.edu.my[1], sucheng@mmu.edu.my[2], khpoo@mmu.edu.my[3]

***Abstract*— The importance of XML processing has become a significant field at present days with the intention to support user queries in the most proficient way. In conjunction with this, many labeling schemes were proposed to identify the elements in XML document uniquely as well as preserve structural relationships among the nodes to cater queries with multiple combinations. On the other hand, due to the flexible structure of XML document, the data that is presented and communicated through this technology changes frequently. Therefore, labeling scheme must be able to support dynamic updates so that the existing labels do not require alteration. In this paper, we present some of the existing labeling techniques and their degree of support for structural relationship and dynamic updates.**

***Keywords*— XML, Numbering scheme, Labeling scheme, Structural Relationship, Dynamic update**

I. INTRODUCTION

Recently, RDBMS has become the most ultimate storage standard for data management in the World Wide Web (WWW). RDBMS production has evolved over time and its importance was comprehended and more sources are using this tool to store and query the XML document. Due to this, an efficient mapping method is certainly provision to ensure seamless data integration between XML and RDBMS. The process of shredding an XML document into relational database and constructing an XML document based on the information stored in the relational database requires the nodes or elements in the XML document to be identified uniquely so that data integrity can be sustained throughout the storing and querying processes. Hence, the effort of scrutinizing and proposing techniques to uniquely identify the nodes, to be exact, labeling methods in XML document is a great challenge to the researchers.

Generally, there are two types of user queries which are full-text and structural queries. Full-text query is a text based search where the queries contain keywords and the RDBMS needs to search the matching results based on the word(s) entered. For an example, 'Find all the hotel names which starts with Renaissance'. Thus, DBMS will return all the hotel names which contain the word Renaissance. This query is simpler compared to structural queries. This is because, the support for structural query requires the structural relationships among the nodes to be preserved and recorded. Structural relationship among the nodes in XML document can be classified into four categories which are Ancestor-Descendant (A-D), Parent-Child (P-C), Sibling and Level information. Users' queries can fall into any one or combination of these categories. Simultaneously, labeling method must be able to cater these queries adequately. For an example, 'Find all the hotels names which are situated in Klang Valley, price less than RM 200 per night and rated as three stars'. In order to cater this query, the relationships among the elements, in this case hotel name, price, location and rating need to be recorded to provide accurate answers which match all these conditions.

Apart from the support for both types of queries, a good labeling scheme must be able to support dynamic updates which can be adding, deleting or updating the data in the original source. This is in view of the fact that the data in XML document changes frequently according to the requirements by the people in an organization or users. A good labeling method should allow these to occur without re-computing the labels of the existing nodes.

The objective of this paper is to study the techniques used in the existing labeling methods which classifies the nodes distinctly and determine strengths and limitations of these approaches.

The rest of the paper is ordered as follows. In section II, we briefly present some preliminaries on XML labeling scheme. Section III is the main focus of our paper whereby we describe the existing labeling approaches. In section IV, we show comparisons of existing labeling techniques. Lastly, section V concludes the paper.

II. PRELIMINARIES

There are two types of labeling that can be used to in an XML document namely *edge labeling* and *node labeling*. Edge labeling labels edges in an XML tree using the element names that appear in an XML document. Fig. 1 illustrates edge labeling for simple XML document.

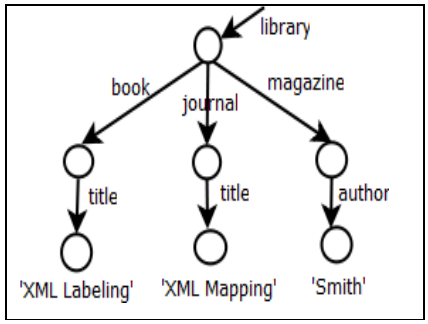


Fig. 1 XML tree with Edge labeling

Node labeling labels the nodes using element names that appear in an XML document. Most of the labeling schemes in present days use the node-labeling method as the root to produce unique labels for the elements in the XML document. Fig. 2 illustrates node labeling for a simple XML document.

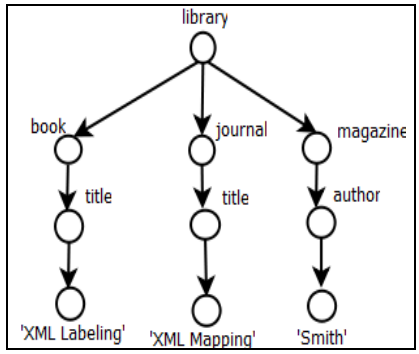


Fig. 2 XML tree with Node labeling

It is vital for a XML document to be conventional to a reliable labeling scheme which plays an important role to the efficient XML and query processing.

III. EXISTING LABELING SCHEMES AND THEIR TECHNIQUES

There are many labeling techniques proposed by many researchers with the intent to identify the nodes uniquely and preserve the structural information among the nodes to cater users' queries. In this section, we describe and present the ideas used by the existing approaches.

A. *Global Order Encoding*

Global Order Encoding [1] assigns each node in an XML tree with a digit which denotes the node's absolute position in an XML document. As Global Encoding Scheme uses the depth-first search to assign the unique number it is easier to determine parent child relationships among nodes. Fig. 3 illustrates the numbering technique used in Global Order Encoding scheme.

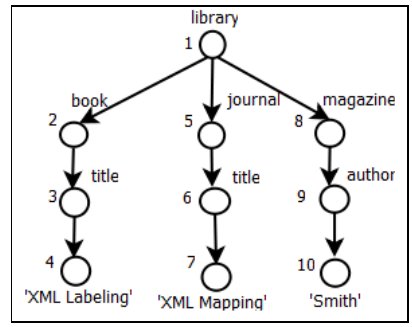


Fig. 3 Global Order Encoding

This scheme does not support dynamic updates because the labels of the existing node require renumbering in case if there is any insertion, deletion or update to the existing document. For an instance, if a new node is inserted into the existing document, all the nodes after the newly inserted node requires re-numbering. This shows the inefficiency of this method because re-computation of labels consumes memory and delays XML processing which degrades the performance of query handling.

B. *Local Order Encoding*

Local Order Encoding [1] assigns a number to the nodes based on their relative position among the siblings. This scheme can be used to generate a path vector by combining the ids of the parent and ancestors. Nevertheless, this path may grow larger depending on the complexity of an XML document. Fig. 4 illustrates labeling method using Local Order Encoding.

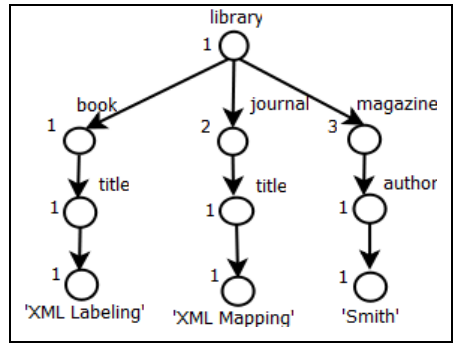


Fig 4: Local Order Encoding

In addition to this, Local Order Encoding does not support dynamic update because the siblings after the newly inserted node require re-calculation to generate new unique labels. Although this scheme results lower overhead during update process compared to Global Encoding method, certainly this scheme consumes time which delay the XML processing.

C. DeweyID

Dewey Encoding [1] is a better approach in determining the structural relationship among nodes compared to Global and Local Encoding. This is due to fact that Dewey Encoding assigns each node with a vector which denotes the path from the root node to the current node and these paths are separated by a dot divider. Thus, A-D relationship can be determined if the ancestor's label is the prefix of the descendant's label. However, a complex XML document may have longer paths and assigning extensive label to a node is absolutely not feasible. Fig. 5 depicts the numbering technique used in Dewey Order ID. Apart from that, since this scheme is a prefix-based scheme, the support for dynamic update is essentially complicated. This is because, if there is a change in the parent label, the child and descendant labels will be adjusted simply because ancestor's labels are being inherited throughout the document. Apparently, this scheme is not appropriate to support dynamic update.

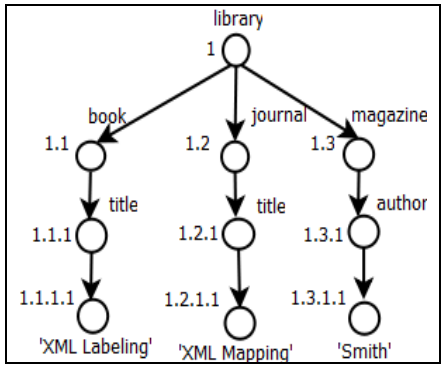


Fig. 5 Vector path in Dewey ID

D. Prime Number Labeling

There are two methods that can be used in prime number labeling scheme namely, bottom-up and top-down labeling schemes [2]. For bottom-up approach, leaf nodes will be assigned a unique prime number which represents the self-label of the node itself. The parent node will be the product of the child nodes. For an instance, if the labels for two leaf nodes are 3 and 5 respectively, the label of the parent node will be 15 (3 x 5). Parent-child relationship can be determined easily be calculating the factor for the number assigned for the parent node. Ancestor-descendant relationship can be calculated by calculating the modulus of the ancestor and descendant node. If the result is 0 then ancestor-descendant relationship among the two nodes exists. For an instance, if the self-label of the ancestor node is 77 and child node = 7, 77 mod 7 = 0; thus, node with the label 77 is the ancestor of node with the label 7.

On the other hand, top-down approach calculates the label of a node by multiplying parent label and self-label which is a unique prime number. For an instance, if the parent label is 2 and the self-label is 7 (prime number), the label assigned for this node is 14 (2x7). Parent-child relationship can be determined easily by dividing the child label and parent label. If these numbers are divisable, then parent-child relationship exists between these nodes. Ancestor–descendant relationship can be ascertained using the same method as in bottom-up approach. Though this approach supports dynamic update, prime number used in this approach may grow larger which produces huge value for the self label of a node. Since prime number that is assigned to a node can only be used once thus, larger amount of prime numbers are required for complex XML document.

E. Interval- Based Labeling Scheme

Interval-based labeling [3] method is based on the depth-first traversal which assigns an unique number to each label which can be referred as the start_position_number and followed by an end_position_number which is given when the node is traversed back from the same branch of the tree. An interval will be given between the start-position and end-position. Fig. 6 shows the technique used in Interval-based labeling.

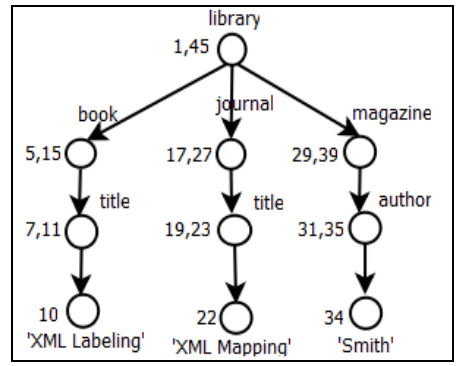


Fig. 6 Interval-based Labeling

Even though this technique allows determining ancestor-descendant and parent-child relationship, dynamic update is not supported if the inserted nodes are more than the interval allocated between the existing nodes. In the worst case scenario, re-labeling will be required which is certainly arduous.

F. Dynamic Interval Labeling Scheme

Interval [4] and Interval-based Labeling are comparable in terms of the reserved numbers allocated for newly inserted nodes. In order to fix the limitation the Interval-based Labeling, Dynamic Interval treats newly inserted nodes as a sub tree and only one numbers will be used from

the reserved number. This is an advantage for bulk loading as the nodes in single insertion will be considered as one tree. Clearly, this reduces the usage of the reserved numbers and more nodes can be inserted at a time.

G. ORDPATH

The labels of the nodes denote the path from the ancestor to the current node based on ORDPATH [5]. ORDPATH assigns only odd and positive integers as the label for existing nodes. Fig 7 illustrates an example of node labeling using ORDPATH. This scheme uses negative and even integers for new insertion of nodes. For an example, if an insertion happens on the right of the existing nodes, then the new label for the node can be generated by +2 will be added to the last ordinal. Likewise if a node is added on the left of the existing nodes, -2 will be added to the first ordinal from left.

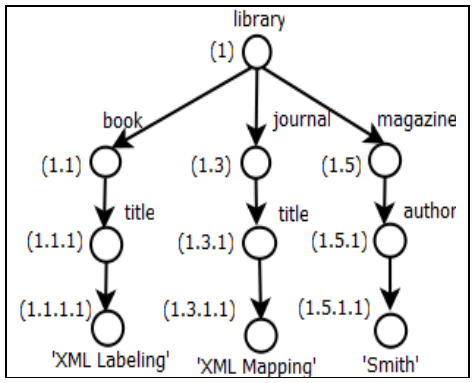


Fig. 7 Example of ORDPATH

The drawback of ORDPATH is the length of the labels which very much depending on the depth of the tree which exemplify complexity of XML document.

H. LSDX

LSDX [6] proposes a different labeling technique as compared to other techniques which combines integer and character to generate unique labels for the nodes. This method produces more unique labels which allow more insertions to happen to the existing document. Label of a node starts with level, followed by parent label, a dot separator and b for the first label and then subsequent alphabets in an alphabetical order. Example of LSDX labeling is shown in Fig. 8.

The disadvantage of this approach is collision may occur which is caused by the production of same labels for two nodes which is absolutely improper and inadequate.

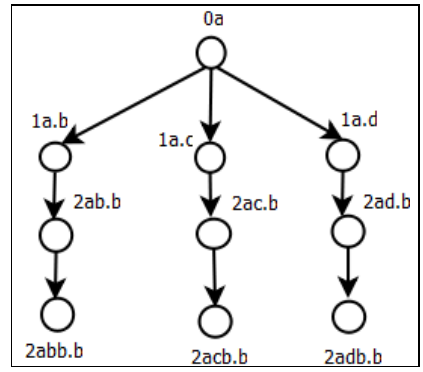
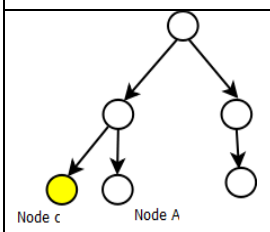


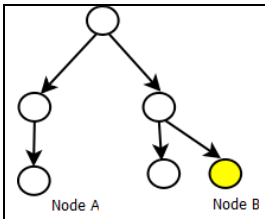
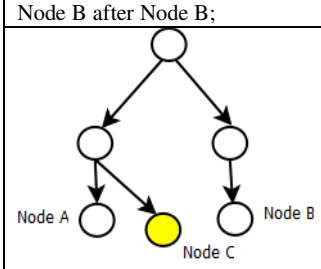
Fig. 8 Example of LSDX

I. Persistent Labeling Scheme

Persistent Labeling [7] is an example of an efficient labeling method which provides the ability to determine structural relationships among the nodes in the document. This is due to the schema proposed by this approach which is (level(l), [parent_label] (p_i,p_j), [self_label] (s_i,s_j)) with the aim to maintain the parent-child relationships among nodes throughout the document. Along with this, ancestor-descendant relationships can be ascertained by tracing the parent-label of the descendant node bottom-up until the ancestor node is reached. This scheme also caters dynamic updates efficiently without having the labels of the existing nodes to be re-calculated. Persistent labeling produces unique labels each time a node is being inserted and deletion or any updates to the existing nodes are certainly allowed without any changes to the existing labels consequently. There are three significant scenarios that need to be concerned to generate unique labels for newly inserted nodes. The scenarios are shown in the Table 1. Assume that the label Node C denotes the newly inserted node.

TABLE I
LABELING TECHNIQUE USING PERSISTENT LABELING SCHEME

Scenario	Technique to Generate Unique Label
 Node C is inserted before Node A provided that no nodes before Node A;	Label C = (c_i,c_j) :- (c_i,c_j) = (a_i -1,a_j)

 Node C is inserted after Node B provided that no nodes after Node B after Node B;	Label C = (c_i,c_j) :- $(c_i,c_j) = (b_i + 1, b_j)$
 Node C is inserted between Node A and Node B	Label C = (c_i,c_j) $\mathbf{c_i} = b_i . a_j + a_i . b_j / d$ $\mathbf{c_j} = 2 . a_j . b_j / d$: where d is Highest Common Factor for $(b_i.a_j + a_i.b_j)$ and $(2.a_j.b_j)$

Hence, Persistent Labeling method is absolutely efficient for dynamic mapping. However, the disadvantage of this approach is length of the labels which is long and this may degrade the performance of the labeling scheme to identify a node expeditiously.

IV. Comparisons Between Existing Approaches

There are many labeling approaches that were proposed in recent years but they need to be assessed if they support two most crucial issues which are: i) support for structural query; ii) support for dynamic updates. The assessment of each approaches are recorded in Table 2 as shown below:

TABLE 2
COMPARISONS BETWEEN LABELING SCHEMES

Approach	**Structural Relationship**			**Dynamic Update**	**Remarks**
	PC *	**AD#**	Sibling		
Global Order	√	×	×	×	All the nodes after newly requires re-labeling during dynamic updates
Local Order	×	×	×	×	Sibling nodes after newly inserted during dynamic updates
Dewey ID	√	√	√	×	Child nodes and sibling nodes requires re-labeling during dynamic updates
Bottom-Up Prime Number Labeling	√	√	×	√	All prime numbers might be used up for complex XML document
Top-Down Prime Number Labeling	√	√	×	√	All prime numbers might be used up for complex XML document
Interval-Based Labeling	√	√	√	√	Reserved number in the interval might be used up for complex XML document. If such, re-labeling is required
Dynamic-Interval Based	√	√	√	√	Reserved number in the interval might be used up for complex XML document. If such, re-labeling is required
ORDPATH	√	√	√	√	Extensive size of label length for complex XML document
LSDX	√	√	√	√	Collision of label which generates same IDs for nodes
Persistent Labeling	√	√	√	√	Length of label is rather long and should be simplified adequately

* PC: Parent-Child
\# AD: Ancestor-Descendant

Users' queries may vary according to their needs. It is indeed crucial for a labeling scheme to maintain the structural relationships among nodes to support for structural queries which contains either P-C or A-D relationships or combination of both P-C and A-D.

V. COMPARISONS AND FUTURE WORK

Labeling scheme plays significant role in supporting user queries regardless of full-text or structural query. Many labeling methods that were proposed support keyword-based search but be oblivious to structural queries which obstructs the processing of queries with multiple criteria. Simultaneously, it is also important for a labeling scheme to support dynamic update to avoid the labels of the existing nodes to be adjusted when there is a new insertion, deletion or updates to the nodes. Thus, an efficient and dynamic labeling scheme contributes to the competency and seamless XML query processing and maintenance of XML data can be done without any alteration to the existing nodes.

VI. REFERENCES

[1] I. Tatarinov, S. Viglas, K. Beyer, J. Shanmugasundaram, E. Shekita, and C. Zhang, "Storing and Querying Ordered XML Using a Relational Database System", *In Proceedings of ACM SIGMOD,* 2002, pp. 204-215.

[2] X. Wu, M.L. Lee and W. Hsu, "A Prime Number Labeling Scheme for Dynamic Ordered XML Trees," *In Proc. of ICDE,* 2004, pp. 66-78.

[3] C. Zhang, J. Naughton, D. DeWitty, Q. Luo, and G. Lohman, "On Supporting Containement Queries in Relational Dabase Management System", *In Proc. of ACM SIGMOD*, 2001, pp. 425-436.

[4] J-H. Yun and C.W. Chung, "Dynamic Interval-based Labeling Scheme for Efficient XML Query and Update Processing", *Journal of Systems and Software*, 2008, pp. 56-70.

[5] P. O'Neil, E. O'Neil, S. Pal, L.Cseri, G. Schaller, and N.Westbury, "ORDPATHS: Insert-Friendly XML Node Labels", *In Proc. of ACM SIGMOD,* 2004, pp. 903- 908.

[6] M. Duong, and Y. Zhang, "LSDX: New Labbeling Scheme for Dynamically Updating XML Data", *In Proc. of 16th Australian Database Conference*, 2005, pp.185- 193.

[7] A. Gabillon and M. Fansi, "A Persistrent Labeling Scheme for XML and tree Database,"In Proc. of ACI, 2006, pp. 110-115 .