How to cite this paper:

Xuan-Thuan Nguyen, Su-Cheng Haw, Samini Subramaniam, & Cong-Kha Pham. (2017). Dynamic node labelling schemes for XML updates in Zulikha, J. & N. H. Zakaria (Eds.), Proceedings of the 6th International Conference on Computing & Informatics (pp 505-510). Sintok: School of Computing.

DYNAMIC NODE LABELING SCHEMES FOR XML UPDATES

Xuan-Thuan Nguyen[1], Su-Cheng Haw[2], Samini Subramaniam[3], and Cong-Kha Pham[4]

[1,4] The University of Electro-Communications, Japan, xuanthuan@vlsilab.ee.uec.ac.jp, pham@ee.uec.ac.jp
[2, 3]Multimedia University, Malaysia, sucheng@mmu.edu.my, samini.subra@mmu.edu.my

ABSTRACT. Recent years have witnessed the rapid development of XML labeling schemes for the facilitation of XML query processing. Nonetheless, relabeling faces the daunting challenge due to space and time consumption whenever labels are inserted or deleted. In this paper, we review three XML labeling schemes that completely avoid relabeling and can re-use the deleted labels for encoding the new nodes. Afterwards, we also discuss the current trends in labeling schemes.

Keywords: node indexing, labeling scheme, dynamic updates, XML database, XML query

INTRODUCTION

With the massive growth of semi-structured and unstructured data recently, relational databases have gradually lost their domination. Non-relational databases such as native XML and NoSQL rapidly gain the attraction due to their high- level flexibility with newer data models (Zhang et al., 2016). Querying information within those databases within the reasonable time also attracts many researchers' attention. To reduce the query time effectively, incoming data should be indexed in advance using, for example, Hash or Tree-based indexing. Furthermore, updating the indexes, whenever new data are inserted or old data are removed, has become increasingly important corresponding to database size.

Native XML databases become increasingly attractive due to its support of semi-structured data. XML query is categorized into full-text query and structural query. Structural indexing is composed of three primary groups namely path indexing, node indexing, and sequence-based indexing. Among them, node indexing or also known as labeling schemes are widely employed in XML databases because queries can be determine easily based on the assigned label. In general, labeling scheme is categorized into four main categories, namely, interval-based scheme (subtree labeling), multiplicative labeling scheme, hybrid labeling scheme, and prefix-based scheme (Haw and Lee, 2011).

When it comes to database process, query and update (insert/delete) are the two most crucial tasks. When a part of a database is modified, the relative indexes must be updated correspondingly and the labels assigned to nodes must be relabeled to maintain structural relationships for facilitating the query processing. Those indexing methods are collectively referred to as static labeling schemes, e.g. Dewey (Tatarinovet et al., 2002), ORDPATH (O'Neil et al., 2004). On the other hand, dynamic labeling schemes indicate the methods whose node relabeling and recalculating are minimized or completely unnecessary. As a result, the cost of

labeling storage space is minimized, and the increase in latency is eliminated. Furthermore, in some approaches, it also opens the ability to reuse deleted node labels, where by the growth rate of the label size under frequent node insertions and deletions is a main concern.

Many prefix-based dynamic labeling schemes have been presented so far such as TDE (Liu and Chen, 2007), DDE/CDDE (Xu et al., 2009), QED (Cohen and Milo, 2010), IBSL (Chemiavsky and Smith, 2010), DFPF (Liu et al., 2013), DPLS (Liu and Zhang, 2016), XDAS (Ghaleb and Mohammed, 2015), DPESF (He, 2015). In this paper, three recent dynamic node indexing methods named DPLS, dynamic XDAS, and DPESF are examined in greater details based on the XML example as shown in Fig. 1. In addition, we also provide the insights on the current trends in indexing methods.

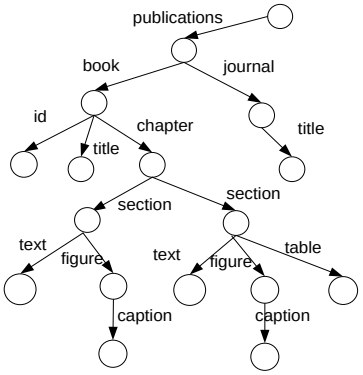

Figure 1. Running Example of XML

LITERATURE REVIEW

Dynamic Labeling Scheme for XML Updates

Liu and Zhang (2016) proposed a novel labeling scheme named Dynamic Prefix-based Labeling Scheme (DPLS), which the initial labeling of DPLS is based on Dewey labeling scheme (Tatarinovet et al., 2002), where by each label is a sequence of components used to represent a unique path from root labeled with a non-zero number (in this case, it is represented as 1) to a node. DPLS scheme is able to: (i) reduce the query costs, (ii) avoid the relabeling process under various scenarios, and (iii) ability to reuse deleted node labels.

For insertion operations, the authors proposed four methods of insertion: insert leftmost (*NodeC*), insert rightmost (*NodeD*), insert leaf (*NodeE*), insert sibling (*NodeA*, *NodeB*). An example of insertion process is shown in Figure 2. The dashed circles and lines represent the new nodes inserted into XML trees.

NodeA is inserted between two nodes with labels 1.1.2.1 and 1.1.2.2 and its label is 1.1.2.((1+2)/(1+1)) = 1.1.2.(3/2). Similarly, the label of *NodeB* is 1.1.2.((3+2)/(2+1)) = 1.1.2.(5/3). *NodeC* is inserted to the leftmost (before first child of the root), thus, it is encoded as 1.0. *NodeD* is inserted to the rightmost, and thus, its label is 1.3 generated by adding 1 to the local order of 1.2. Subsequently, *NodeE* is inserted as the child of the node labeled 1.3 and its label is 1.3.1.

The authors compared the performance regarding label size and insert time between DPLS and four well-known schemes (ORDPATH (O'Neil et al., 2004), DDE/CDDE (Xu et al., 2009), QED (Cohen and Milo, 2010) and DFPD (Liu et al., 2013)) in four real XML datasets. In terms of the label size, in the initial stage of DPLS, the size is nearly equivalent to that of DFPD. However, DPLS surpassed other works when insertions and deletions based on the following observations. DDE, CDDE and DFPD assign new labels for each newly insert-

ed nodes without reusing the deleted labels, while DPLS has the beauty to reuse the deleted labels for newly inserted nodes. The reusability is considered important especially on a frequently updatable condition. Reusability reduces the label size as label grows as the depth of the data tree do.

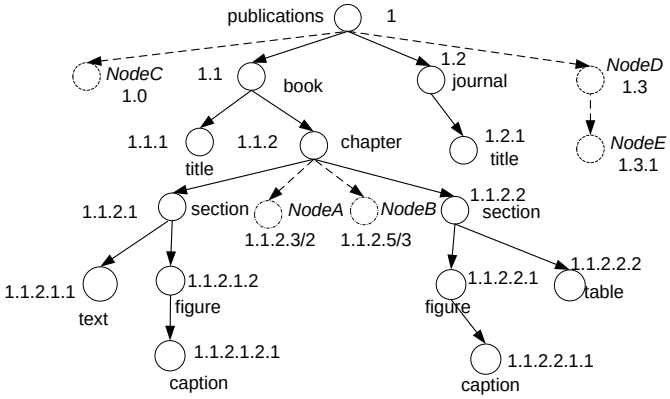

Figure 2. An Example of Insertion Process of DPLS.

Dynamic Prefix Encoding Scheme based on Fraction

He (2015) proposed a so-called Dynamic Prefix Encoding Scheme based on Fraction (DPESF). This technique achieves better time and space performance, while supports dynamic updating operation. In general, DPESF label scheme is similar to DPLS label, except that the author expresses the numerator by a set of alphabet characters. To begin with, the author introduces a set of numbers {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} and a set of alphabet characters {A, B, C, D, E, F, G, H, I, J}. Subsequently, the corresponding rule function f is defined as {(0, A), (1, B), (2, C), (3, D), (4, E), (5, F), (6, G), (7, H), (8, I), (9, J)}. According to f, a label such as 123/11 is expressed as BCD11, in which BCD is 123.

Figure 3 shows the XML tree labeled with DPESF. Similar to DPLS scheme, for insertion operations, the authors proposed four methods of insertion: insert leftmost (*NodeC*), insert rightmost (*NodeD*), insert leaf (*NodeE*), insert sibling (*NodeA*, *NodeB*). *NodeA* is inserted between two nodes with labels 1.1.2.1 and 1.1.2.2 and its label is 1.1.2.((1+2)/(1+1)) = 1.1.2.(3/2). Thus, the '3' is replaced with 'D' to generate the label as 1.1.2.D2. The label of *nodeB* is 1.1.2.((3+2)/(2+1)) = 1.1.2.(5/3), equivalent to be represented as 1.1.2.F3. On the other hand, *NodeC, NodeD* and *NodeE* maintain the same label as DPLS as there is no fractional number generated.

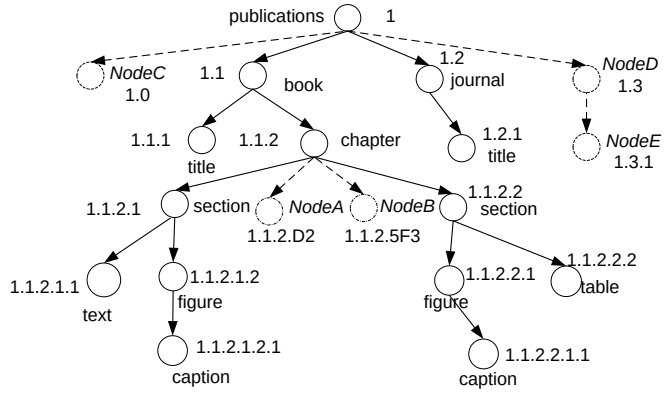

Figure 3. An Example of Insertion Process of DPESF.

The author compared the performance regarding label size and update process between DPESF and two well-known schemes, namely ORDPATH (O'Neil et al., 2004) and TDE (Liu and Chen, 2007) by five XML datasets. The advantages between DPESF over ORDPATH and TDE are summarized in Table 1.

Table 1. The Improvement of DPESF.

	Label size	Update process (static)	Update process (dynamic)
ORDPATH	Up to 5%	Up to 5%	Up to 5%
TDE	Up to 200%	Up to 20%	Up to 35%

Dynamic Labeling Scheme Based On Logical Operators

Ghaleb and Mohammed (2015) proposed a so-called dynamic XDAS, which is a combination of their original XDAS (Ghaleb and Mohammed, 2013) with another labeling scheme called Improved Binary String Labeling (IBSL) (Chemiavsky and Smith, 2010). This labeling scheme is group under hybrid labeling scheme (Haw and Lee, 2011), where by it combines the beautiful features from both labeling schemes.

IBSL was selected to completely avoid node relabeling and recalculation. However, IBSL generates labels based on lexical order, while XDAS generates labels based on masking techniques. The 'book' has the label 1,001, whereby the first part indicates the level, followed by the sequential ID 001 as 'book' is the first child of 'publication'. Next, the 'journal', which is the sibling of 'book' will have label 1,010. On the other hand, the label of node 'chapter' is 2,10001, where 2 is the level, 10 is the sequential ID, and 001 is the label from its parent node.

IBSL supports four types of insertion: insert leftmost (*NodeC*), insert rightmost (*NodeD*), insert subtree (*NodeE*), insert node between any two nodes (*NodeA*, *NodeB*). Figure 4 depicts an example of insertion operation.

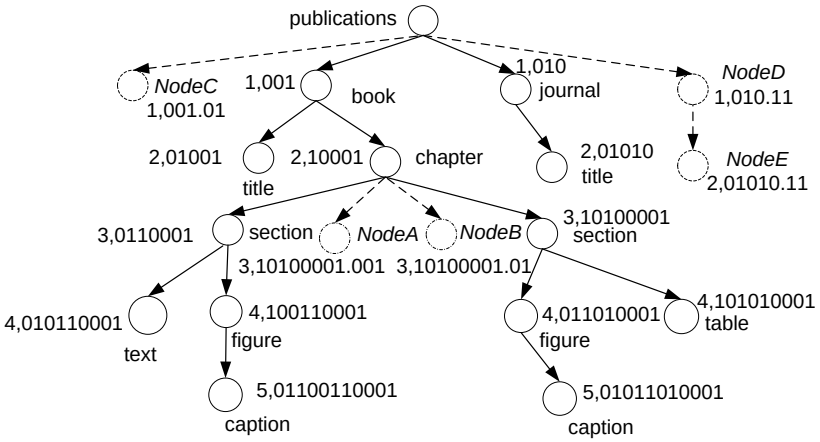

Figure 4. An Example of Insertion Process of Dynamic XDAS.

The authors compared the performance between dynamic XDAS and two well-known schemes, Dewey (Tatarinovet al., 2002) and IBSL (Chemiavsky and Smith, 2010) with respect to labeling size in three real XML datasets. The comparison is summarized as follows:

- Label size and space overhead: dynamic XDAS outperformed IBSL in both typical and worst cases. However, the improvements were not remarkable in comparison with Dewey.
- Update process: the improvements were not remarkable as compared with ISBL.

The advantages between dynamic XDAS over two previous works are summarized in Table 2.

Table 2. The Improvement of Dynamic XDAS.

	Label size	Update process
Dewey	Up to 200%	–
IBSL	Up to 750%	Up to 5%

SUMMARY AND DISCUSSION

Labeling schemes are essential to enable quick determination of structural relationships between any two nodes. Choosing the correct index, which fits one's needs, is critical. On top of this, the other factors for consideration are robustness, labeling size, computation cost, query retrieval speed, reusability and support for dynamic updates.

Based on the review done, it is learnt that DPLS outperformed existing techniques like ORDPATH, DDE/CDDE, QED and DFPD by reusing deleted labels for newly inserted nodes. This minimizes the storage cost and promotes the updating performance. On the other hand, DPESF made justified improvements in terms of update time as compared TDE and significant improvement in terms of label growth rate as compared to TDE. Although XDAS performed ideally in terms of the growth rate of the label size as well as space consumption, its improvements were not significant as compared to Dewey. Similarly, XDAS's performance on update process was not notable as compared to ISBL.

Having said that, Table 3 summarizes the advantages and disadvantages of the labeling schemes reviewed above.

Table 3. Summary of Reviewed Labeling Schemes.

Labeling scheme	Advantages	Disadvantages
DPLS	DPLS outperforms many state-of-art labeling scheme regarding to label size and update time. This is because the deleted labels are reused for encoding new inserted nodes.	As compared to DFPD, label size and update time of DPLS were slightly improved.
DPESF	As compared to TDE, DPESF clearly improved the label size and update time.	As compared to ORDPATH, the improvement, regarding label size and update time, is not clear.
XDAS	As compared to IBSL, XDAS significantly reduced the label size.	As compared to IBSL, the update time did not improve.

CONCLUSION

In this paper, three labeling techniques were reviewed in terms of its mechanisms in defining labels for newly inserted nodes, performance comparisons against the existing techniques in terms of updating performance as well as the growth of labeling size. From the studies conducted, we learnt the importance of a labeling technique that completely avoids re-labeling to manage dynamically changing data. It will be even more creditable when a labeling technique could re-use the deleted labels as this would improve the storage and update cost. In the future, we would propose a novel technique that supports dynamic updates seamlessly while optimizing storage and update performance.

ACKNOWLEDGMENTS

This work was partially supported by funding from FRGS, Ministry of Education, Malaysia.

REFERENCES

Cohen, E., Kaplan, H., & Milo, T. (2010). Labeling dynamic XML trees. *SIAM Journal on Computing*, 39 (5), 2048-2074. doi: 10.1137/070687633.

Chemiavsky, J. C., & Smith, C. H. (2010). A Binary String Approach for Updates in Dynamic Ordered XML Data. *IEEE Transactions on Knowledge and Data Engineering*, 22, 602-607. doi: 10.1109/TKDE.2009.87.

Ghaleb, T. A., & Mohammed S. (2013). Novel scheme for labeling XML trees based on bits-masking and logical matching. *World Congress on Computer and Information Technology (WCCIT), IEEE*, 1–5.

Ghaleb, T. A., & Mohammed S. (2015). A Dynamic Labeling Scheme Based on Logical Operators: A Support for Order-Sensitive XML Updates. *3rd International Conference on Recent Trends in Computing (ICRTC)*, 1211-1218. doi: 10.1016/j.procs.2015.07.416.

Haw, S. C., & Lee C. S. (2011). Data storage practices and query processing in XML databases: A survey. *Knowledge-Based System Journal*, 24(8), 1317-1340. doi: 10.1016/j.knosys.2011.06.006.

He, Y. (2015). A Novel Encoding Scheme for XML Document Update-supporting. *International Conference on Advances in Mechanical Engineering and Industrial Informatics (AMEII)*, 1844-1849. doi: 10.2991/ameii-15.2015.342.

Liu, J., Ma, Z. M., & Yan L. (2013). Efficient labeling scheme for dynamic XML trees. *Information Sciences: an International Journal*, 221, 338-354. doi: 10.1016/j.ins.2012.09.036.

Liu J., & Zhang, X. X. (2016). Dynamic labeling scheme for XML updates. *Knowledge-Based System Journal*, 106, 135-149. doi: 10.1016/j.knosys.2016.05.039.

Liu, Z. Y., Chen, Y. (2007). Identifying Meaningful Return Information for XML keyword Search, *Proceedings of the 2007 ACM SIGMOD International conference on Management of data*, 19-30. doi: 10.1145/1247480.1247518.

O'Neil, P., O'Neil, E., Pal, S., Cseri, I., Schaller, G., & Westbury N. (2004). ORDPATHs: insert-friendly xml node labels, *Proceedings of the 2004 ACM SIGMOD International conference on Management of data*, 903-908. doi: 10.1145/1007568.1007686.

Tatarinov, I., Viglas, S. D., Beyer, K., Shanmugasundaram, J., Shekita, E., Zhang, C. (2002). Storing and querying ordered XML using a relational database system, *Proceedings of the 2002 ACM SIGMOD International conference on Management of data*, 204-215. doi: 10.1145/564691.564715.

Xu, L., Ling, T. W., Wu, H., & Bao, Z. (2009). DDE: from dewey to a fully dynamic XML labeling scheme. *Proceedings of the 2009 ACM SIGMOD International conference on Management of data*, 719-730. doi: 10.1145/1559845.1559846.

Zhang, H., Chen, G., Ooi, B., Tan, K., & Ooi C. (2016). In-memory big data management and processing: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 27(7), 1920-1948. doi: 10.1109/TKDE.2015.2427795.