Labeling Schemes to Support Dynamic Updates on XML Trees: A Technical Review

Aisyah Amin, Su-Cheng Haw, Samini Subramaniam and Emyliana Soong
Multimedia University, Malaysia, {aisyah.amin@gmail.com, suucheng@mmu.edu.my, samini.subra@mmu.edu.my, emyliana.soong@gmail.com}

ABSTRACT

eXtensible Markup Language (XML) are widely use on World Wide Web (WWW) for data exchange purpose due to its expressivity and extensible nature. With the fast growing rate of data, especially with high updates, it is important to ensure that the XML is able to cope with frequent changes with very least affect on the existing structure. To ensure the structural relationships are preserved, XML tree is commonly annotated with labeling scheme. Various labeling schemes emerged with the intention to ensure that it is persistent, robust and durable enough to sustain the re-labeling due to updates. They can be grouped into four major groups, namely, region encoding, prefix-based, multiplicative and hybrid. In this paper, we review on some existing labeling scheme based on each grouping. Through the review, we observed that each labeling scheme assign the node based on their unique identifier, thus, has its strengths and weaknesses. Finally, we provide some discussions based on the labeling grouping.

Keywords: XML database, labeling scheme, dynamic updates, node indexing, structural relationship.

I INTRODUCTION

Extensible Markup Language (XML) is used to define data and designated to be self-descriptive. It is a tag-based syntax; similar to Hypertext Markup Language (HTML). XML is readable by human and machine as it uses natural language (Mohammad et al., 2011). At the same time, relational database is commonly being used as back-end in various industry. Nevertheless, due to the data is process independently of its context, relational database could not fulfill the market demand specifically in electronic business. To put it another way, relational database is simply unsuitable for semi structured data. As such, it is critical to store and retrieve XML structure (hierarchical model) via relational database (tables with rows and columns). The key criterion for a good mapper is to ensure that the four main structural relationships, i.e., ancestor-descendant (AD), parent-child (P-C), sibling and order are preserved (Dietz et al., 1982; Haw & Lee, 2009; Subramaniam & Haw, 2014). In order to do so, a good and effective labeling scheme employed on the node (also known as node indexing) is essential.

There are numbers of researches done on labeling scheme (Liu et al., 2013; Fraigniaud & Korman, 2016; Liu & Zhang, 2016; Qin et al., 2017). The purpose of this paper is to analyze and discuss on some recent labeling scheme, especially in terms of the support during dynamic updates (insertion operation). The main types of insertion happen: (i) left-most insertion, (ii) right-most insertion, and (iii) in-between insertion (Xu et al., 2012; Liu et al., 2013; Liu & Zhang, 2016). Left-most and right-most insertion are quite straight forward in many existing approaches. Most of the right-most insertion does not require relabeling in the nodes insertion. Nevertheless, due to space constrains, this paper only reviews the in-between insertion as the right-most and left-most insertions are straightforward.

The rest of this paper is organized as follows. Section 2 review on the four selected labeling schemes and also some recent works on the research area, while Section 3 summarizes and discusses on the advantages and disadvantages of these labeling schemes.

II REVIEW ON EXISTING LABELING SCHEME

The four main categories of labeling scheme are region encoding, prefix-based, multiplicative and hybrid (Haw & Lee, 2009). A region-based labeling scheme utilise the tree traversal navigation to assign label on the nodes to preserve the ordering while ensuring the structural relationships are preserved among nodes V-Containment (Xu et al., 2012). Tree traversal is the process of sequentially visiting each node in a tree data structure, and can proceed in different directions: depth-first traversal or breath-first traversal (Tahraoui et al., 2013). A prefix-based labeling schemes (Haw & Lee, 2009; Ghaleb and Mohammed, 2015) is usually the most simple scheme as it directly encode a node’s parent label as the prefix of its label.

On the other hand, multiplicative labeling scheme usually assign label based on some arithmetic computation to identify the structural relationships among nodes. A hybrid labeling scheme, however,
is composed of some combinations of existing scheme grouping to balance between one weakness with the strength of the other group (Aisyah & Haw, 2015).

We have selected to review on one example for each grouping. These are V-Containment (Xu et al., 2012), ME (Subramaniam & Haw, 2014), LLS (Mohammad & Martin, 2010), and DPLS (Liu & Zhang, 2016). Subsequent section also review some recent trends on labeling scheme.

SigmodRecord dataset is used as an example of the review throughout this paper. The partial view of SigmodRecord Dataset is depicted in Figure 1.

Figure 1. A Sample of SigmodRecord XML Document.

A. V-Containment (Region Encoding)

Xu et al. (2012) proposed V-Containment labels for region encoding which is based on containment labeling scheme (Zhang et al., 2001). Fundamentally, the labeling structure contains startV, endV, level), where by startV, endV are two vectors representing the one-time assignment of initial labeling pre/post labeling scheme (Dietz et al., 1982), and level denotes the number of edges between root nodes to current node. The initial labeling schemes are assigned based on depth first traversal to assign the initial node labeling. Figure 2 depicts the V-Containment (and also ME labeling scheme, which will be covered in Section B).

The authors introduced the idea of granularity sum (GS) as shown in Algorithm 1 to conduct an insertion.

Algorithm 1. InsertTwoVectorCodes(A, B)

**Data:** A and B which are two vector codes satisfying $A \prec B$

**Result:** C and D such that $A \prec C \prec D \prec B$

1. if $GS(A) > GS(B)$ then
2. return $(A+B)$ and $(A+2 \cdot B)$
3. else
4. return $(2 \cdot A+B)$ and $(A+B)$
5. end

Figure 3 shows an insertion of in-between node label as node D, node E and node F of V-Containment labeling. In this case, Node D will be inserted in-between the two nodes of nodes Node X and Node Y. The start and end should be between the end of its preceding sibling and the start of its following siblings. From Algorithm 1, the start and end of Node D should be $3, 118$ (= $(2 \cdot 1 + 1, 2 \cdot 39 + 40)$) and $2, 79$ (= $(1 + 1, 39 + 40)$). The start and end of E should be $3, 119$ (= $(2 \cdot 3 + 4, 2 \cdot 119 + 159)$) and $4, 159$ (= $(3 + 4, 2 \cdot 39 + 40)$). The range of node D is confined by its parent’s range. Node F is inserted after node E using Algorithm 1, the start and end of F should be $(10, 397)$ (= $(2 \cdot 3 + 4, 2 \cdot 119 + 159)$) and $(7, 278)$ (= $(3 + 4, 2 \cdot 119 + 159)$).

Therefore, V-Containment does support dynamic updates as it does not require to re-labeling the nodes.
C. Level-based Labeling Scheme (LLS) (Hybrid)

Level-based labeling scheme (LLS) is a hybrid labeling scheme based on interval and prefix-based labeling scheme (Mohammad & Martin, 2010). LLS labeling structure are assigned as $d,p,s$ whereby $d$ denote the depth of level, $p$ (indicate as PerL) is the number of node across d level and $s$ is the instance serial number that recognize nodes between the same node from the same class. Figure 4 shows the LLS labeling scheme on the summary tree.

On dynamic update, the relabeling of nodes only effect on the labels next to inserted nodes. For instance, Figure 5 shows an insertion summary tree of LLS labeling scheme (and also, an insertion of DPLS which will be covered in Section D). Whereas, Figure 7 shows an insertion of node label ‘Year Publish’. Node label ‘Year Publish’ is inserted in-between node labels ‘issue’ on the right and node labels ‘issue’ on the left. Node label ‘Year Publish’ is 2.21.1 as it is not from the same class of node labels ‘issue’. Follow by the insertion of label node ‘Publish Date’ is 3.41.1 and label node ‘Location (pagination)’ is 3.51.1. However, in this case, these three inserted nodes are not from among the same class. Thus, the relabeling of nodes does not require. An illustration of LLS labeling scheme (and also DPLS, which will be covered in Section D) is shown in Figure 6.

The labeling structures of LLS are explained in the following two definitions.

**Definition 1:** A tag path $t$ for a node $v$ is a sequence of tags, $l_1, l_2, \ldots, l_i (i \geq 1)$, of the nodes on the path from the root node to node $v$, separated by dots.
For instance, the tag path of node <3.31.2> is SigmodRecord.issue.articles.

**Definition 2:** A serial path r for a node v is a sequence of serial numbers, s₁, s₂, …, sᵢ (i ≥ 1), of the nodes on the path from the root node to v. For instance, the serial path of node <3.31.2> is (1,2,3), which contains the third part of the labels of the nodes, in the path from the root node to this node. Note that the d values (the levels) of the components of a serial path r of a node v, where r = (s₁, s₂, …, sᵢ), is d = (1,2, … , i), respectively, where i is the level of v.”

For instance, for node <3.31.2>, the levels of the component of the serial path (1,2,3) are (1,2, and 3), respectively.

LLS labeling tree structure can be summarized as in a group whereby all the same tag path will be shown at least once. For instance, in this case, node label issues are <2.11> only will be shown once. Similarly, node label volume is <3.11>, node label <3.21>, node label <3.31> and so on. Figure 4 shows how the summary of LLS tree.

However, there is a need in relabeling the node for dynamic updates. The relabeling of node effect in a level of the next node inserted. Figure 8 shows in theoretically nodes that need relabeling. On the other side, the advantage of LLS labeling is the labeling tree can be summarized.

**D. Dynamic Prefix-based Labeling Scheme (DPLS) (Prefix-based)**

Dynamic Prefix-based Labeling Scheme (DPLS) (Liu & Zhang, 2016) is an example of prefix-based labeling by extending on Dewey scheme (Tatarinov et al., 2002). This approach has two stages, whereby
the first stage includes constructing the initial DPLS labeling is assigned based on Dewey scheme, followed by the next stage is to handle any updates. The diagram of DPLS initial labeling is shown in Figure 6.

The second stages support the XML update such as insertion of labeling nodes. The first insertion of DPLS is representing as Node A follow by Node B and Node C. The dashed line denotes the new inserted nodes. For instance, Node A is inserted between two nodes with labels 1.1 and 1.2 and its label is 1.3(2) which equals to 1.((1+2)/(1 + 1)). Similarly, it is inserted after node B whereas node C is inserted in-between two nodes with labels 1.1. (4/2) and 1.2.1. Then, the label of node C is 1.1(5/3) which equals to 1.1. (4+1)/(2+1)). Figure 7 shows an insertion of DPLS in-between the nodes.

As a result, DPLS approaches avoid relabeling nodes during a dynamic updates occurs.

E. More Recent Related Works on Various Labeling Scheme

This section reviews some recent related works to highlight the trends of labeling schemes.

Fu & Meng (2013) proposed Triple-code which consists of <start, end, parent-id>. Their approach adopts the interval-based labeling scheme proposed by Li & Moon (2001) by replacing the ‘level’ tag with node’s ‘parent-id’, making it straightforward to obtain parent/child and sibling relationships.

He (2015) proposed prefix-based scheme using fractions, which he named it as DPESF Encoding. This labeling scheme is stored in Numeric-Character format based on the mapping rules as follows: “to map each digit \( n \in N = \{0,1,2,3,4,5,6,7,8,9\} \) in the numerator to a matching character \( c \in C = \{A,B,C,D,E,F,G,H,I,J\} \)”. As such, label with \((12514)\) is expressed as \(BCF14\).

On the other hand, Ghaleb & Mohammad (2015) proposed Dynamic XDAS as an example of hybrid labeling scheme. Dynamic XDAS uses binary digits (0 and 1) to indicate the labeling scheme. This approach support the lack of Level-based labeling scheme (LLS) that require node relabeling. The concept of Dynamic XDAS is the extend approach of Improved Binary String Labeling (IBLS) (Chemiavsky & Smith, 2010). IBLS uses Dewey ID techniques of lexical order in labeling the nodes. However, the size of data took a large storage.

More recently, Gopinathan & Asawa (2017) proposed an extended Dewey labeling scheme, which consists of [prefix.ordinal] label to support Content and Structure Query (CAS) effectively. In addition, they also proposed new path based indexing namely, path index (p_index) and path combined index (pc_index). These indexes were constructed using B+Tree and HashMap respectively.

Ahn et al. (2017) proposed to implement repetitive prime number (Sun & Hwang, 2014) labeling in a Map Reduce-based algorithm to overcome the problem of memory insufficient should a massive XML data is loaded in a single machine. Being in parallel environment, this allows multiple machines to compute labels independently.

III DISCUSSION

There are two aspects of labeling schemes, i.e. to ensure the structural relationships maintained (static XML) and to be persistent to any changes incurred during updates (dynamic updates). Thus, selecting the suitable labeling scheme is crucial. Some factors to be considered while doing so include (1) to ensure that the structural relationship is maintained at all time. In addition to that, (2) the labeling scheme should be persistent enough to avoid re-labeling during any updates.

Selecting the most appropriate labeling scheme is very important. For instance, region encoding and multiplicative labeling scheme require massive calculation as the size of XML document growth. In addition, multiplicative labeling scheme also suffers from large label sizes because it leaves big gaps, which may lead to overflow problems (Ahn et al., 2017). Prefix-based labeling scheme appears to be the most straightforward scheme as to determine the relationship of one node to the other can be done by checking on the prefix. Nevertheless, the growth size of some prefix-based labeling schemes are uncontrollable especially for XML tree with many levels down. On the other hand, hybrid labeling schemes have the potential to support faster query processing by combining the advantages of two or more labeling schemes (Haw and Amin, 2015).

Table 1 summarizes the advantages and disadvantages of reviewed labeling schemes.

<table>
<thead>
<tr>
<th>Labeling Scheme</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
<tbody>
<tr>
<td>V-Containment (Xu et al., 2012).</td>
<td>Supports dynamic updates without relabeling the nodes.</td>
<td>The value of inserted node becomes bigger.</td>
</tr>
</tbody>
</table>
IV CONCLUSION AND FUTURE WORK

In this paper, we have reviewed on the four groups of labeling schemes by showing how some of the technique works, and highlighted the pros and cons of the technique employed. To sum up, the multiplicative scheme may not be a good choice if it involves huge dataset due to costly computation time. In most cases, region encoding usually uses start, and end to labels the nodes which requires relabeling in nodes insertions. On the other hand, DPLS and LLS may be a good candidate for situation where frequent dynamic updates happen.

In our future direction, we intend to propose a hybrid labeling schemes by extending region encoding and prefix-based scheme. Using region encoding, one can easily determine the structural relationship among the nodes. Nevertheless, this scheme is not robust enough to support dynamic updates. On the other hand, prefix-based scheme appears to be the most straightforward, and some technique such as ORDERPATH (O’Neil et al., 2004) and DPLS (Liu & Zhang, 2016) have proven to be scalable to support dynamic updates. Thus, by combining the beautiful features of both schemes, the limitation may be overcome.

REFERENCES