Knowledge-Based Systems 106 (2016) 135–149

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

Dynamic labeling scheme for XML updates

Jian Liu[a,*], X.X. Zhang[b]

[a] School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China
[b] Northeastern University, Shenyang, 110004, China

ARTICLE INFO

Article history:
Received 14 November 2015
Revised 17 May 2016
Accepted 18 May 2016
Available online 19 May 2016

Keywords:
Labeling scheme
XML
Query processing
Structural relationships
Node indexing
Updates

ABSTRACT

Nowadays several labeling schemes are proposed to facilitate XML query processing, in which structural relationships among nodes could be quickly determined without accessing original XML documents. However, previous node indexing often encounters some troublesome problems when updates take place, such as a large amount of labels requiring re-labeling, huge space requirements for the updated labels, and inefficient determination of structural relationships. In this paper, we propose a novel labeling scheme that not only completely avoids re-labeling but also improves the performance of determining the structural relationships when XML documents are frequently updated at arbitrary positions. The fundamental difference between our scheme and previous ones is that, the gain in update performance of our labeling scheme does not come at the expense of the label size and the query performance. In particular, instead of completely assigning new labels for inserted nodes, the deleted labels are reused in our labeling scheme for encoding newly inserted nodes, which could effectively lower the label size. Moreover, we formally analyze the effectiveness of our proposed labeling scheme. Finally, we complement our analysis with experimental results on a range of real XML data.

1. Introduction

With the prompt development of XML, there have been many researches about the problems of efficiently storing, managing and querying XML data [1,2,4]. Queries languages such as XPath [5] and XQuery [6] are proposed to process XML data. The core operation of these query languages is to efficiently determine the structural relationships [25,27,45] such as the ancestor-descendant, parent-child, sibling, ordering, lowest common ancestor (LCA), etc. To facilitate the determination of these structural relationships, several labeling schemes [7,23] have been proposed. In [32], Subramaniam et al. introduced a static labeling scheme called Relab which generates labels using depth-first traversal. In [7], Bruno et al. developed a technique of a chain of linked stacks to match twigs based on the interval-based labeling scheme. In [25], Lu et al. designed an algorithm to process the twig query based on the Dewey scheme. Using these labeling schemes, the structural relationships among elements/attributes can be directly determined by comparing their labels.

The increasing popularity of XML as a data exchange format has led to a huge amount of XML data updates, which have become increasingly important due to the fact that the usage of XML has shifted from being merely a flexible publishing language to a data format [11,14,34,38]. Many works have investigated the importance of efficient XML storage, indexing and query processing [13,22,23,35,40]. However, it is widely known that update remains the weakness in many XML databases [37]. Generally speaking, updates in XML mainly involve two things: (i) physically updating the XML data and indexes associated with, and (ii) renumbering the labels assigned to nodes, in order to maintain structural relationships for facilitating the query processing. As illustrated in paper [46], the cost of the former is similar under the same storage model, but the cost of the latter can vary significantly, when a different labeling scheme is used. Therefore, in this paper we study XML updates and focus ourselves on the latter issue–XML label updates.

Current labeling schemes can be divided into four main families [15], that is, subtree labeling (interval based scheme), prefix (Dewey) based scheme, multiplicative labelling and hybrid labeling. Since the labels in prefix based scheme contain the path information, it has become the main choice for XML keyword query processing [13,33,35,45] which has a great interest in evaluating the LCA of (a set of) nodes from their labels. While previous labeling schemes [7,25,32] work well for static XML documents, it may suffer from high cost for dynamic XML queries where nodes are arbitrarily inserted and deleted [8]. The major problem of dynamic queries [16,17] is that, in order to maintain the effectiveness of the

* Corresponding author.
E-mail address: lyot@163.com (J. Liu).

http://dx.doi.org/10.1016/j.knosys.2016.05.039
0950-7051/© 2016 Elsevier B.V. All rights reserved.

labeling scheme, the labels of a large number of nodes need update possibly when the original XML document has been updated, which makes the dynamic queries very inefficient. Some researches [3,9,10,18,19,21,24,29,31,43,44,48] have been dedicated to maintaining the effectiveness of labeling scheme in dynamic XML queries. In [31], O'Neil et al design a labeling scheme called ORDPATH which is used in the latest versions of Microsoft® SQL Server^{TM} for dynamic XML documents. ORDPATH uses positive, odd integers for initial labeling, and even, negative integers are reserved for later 'careting in' insertions. However, skipping even integers makes ORDPATH less compact, and the 'careting in' mechanism introduces additional complexity for its label insertions. Xu et al [43] present two labeling schemes for encoding dynamic XML trees. In their labeling schemes, the labels of updating nodes are assigned based on the mathematical operations of each Dewey component. Compared with the previous works, their labeling schemes have shown better performance when XML nodes are inserted. However, the labels in their labeling schemes are incompact and often need extra storing cost. Recently, the work [24] proposes an insert-friendly labeling scheme for dealing with dynamic queries in XML documents. Although this labeling scheme has demonstrated its superior power in the single insertion processing, it is suboptimal when insertion and deletion take place alternately. More specifically, it completely assigns new labels for the inserted nodes without considering the reuse of the deleted labels, which causes the increase of storage.

Updating XML data is a research topic that is still in its infancy, and the update costs of the previous approaches are still expensive. This fact is the motivation of our work. Without an efficient labeling scheme, re-labeling large amount of nodes caused by updates becomes a performance bottleneck if XML data are updated. The problem we are facing is challenging since it requires us to find such a labeling scheme that can minimize the query cost, completely avoid re-labeling when updates occur at any positions, and require minimal space for encoding.

As introduced in [31], the length of a node label is an important criterion in the quality of any dynamic labeling scheme and the larger the label size, the more significant is the negative impact on query and update performance. In this paper, we tackle the problem that efficiently supports update and query processing over XML data from a labeling scheme perspective. In particular, based on the prefix based scheme, we propose a novel labeling scheme called DPLS (Dynamic Prefix-based Labeling Scheme) for dealing with updates and queries in XML documents. DPLS is designed to address three specific goals: i) to facilitate the reduction of the costs of the query and the update services; ii) to completely avoid the need to relabel nodes under various updated scenarios and iii) to minimize the storage space costs associated with the label encoding. A distinguishing characteristic of the DPLS is the ability to reuse deleted node labels in a dynamic context, and consequently constrain the growth rate of the label size under frequent node insertions and deletions. Our labeling scheme yields high query performance as well as supporting the incremental updates. Furthermore, when updates take place, the structural relationships among nodes, such as the parent-child, ancestor-descendent, document order and LCA, can be effectively maintained without requiring re-labeling the existing labels. A theoretical analysis and an experimental evaluation of the DPLS labeling scheme are also provided. Both qualitative and experimental comparisons demonstrate the advantage of our proposed labeling scheme over previous approaches.

The rest of the paper is organized as follows. We firstly give the literature review and our observations in Section 2. The basic operators to support XML update is presented in Section 3. Section 4 describes the proposed labeling scheme in details. Section 5 is dedicated to the analysis of the DPLS. The experimental results are reported in Section 6, and we conclude the paper in Section 7.

2. Literture review and observations

2.1. Interval based scheme

Interval based scheme is derived from the inverted index data structure in information retrieval (IR). Similar to the technique dealing with text words in traditional IR systems, interval based scheme maps elements with the same tag name to an inverted list. The most representative work related to the interval based scheme family is the work introduced by Zhang et al [47]. In their work, the authors propose a technique whereby identifiers are represented as intervals. This labeling scheme aims to determine structural relationships among nodes by using the related containment information. In particular, one record is represented as a 3-tuple (*start, end, depth*) to identify the position of an element exclusively. While *start* which is generated by a preorder traversal of the document trees exactly finds the occurrence position. *end* is the maximal start of elements in the sub-tree of current element, and *depth* gives additional information to determine the parent-child relationship of tree elements. Given two elements *u, v*, the positional relationship between them is defined as follows:

- if $u.end < v.start$, u is a preorder node of v.
- if $u.start < v.start$, and $u.end > v.end$, u is an ancestor of v.
- if $u.start < v.start$, $u.end > v.end$, and $u.depth + 1 = v.depth$, u is a parent of v.
- if $u.start > v.start$, and $u.end < v.end$, u is a descendant of v.
- if $u.start > v.start$, $u.end < v.end$, and $u.depth - 1 = v.depth$, u is a child of v.
- if $u.start > v.end$, u is a postorder node of v.

Although interval based scheme is efficient to determine parent-child and ancestor-descendant relationships, it is difficult to provide information about the LCAs of a set of nodes. And most importantly, an insertion may incur a re-labeling of numerous nodes. For example, in Fig. 1, the insertion of node *a* incurs the re-labeling of the dotted box part.

The re-labeling problem may be alleviated by leaving gaps at the initial labeling [22], however, the labels still need to be relabeled after the gaps are filled. Moreover, this initial interval size wastes a lot of numbers and causes the increase of storage. To solve the re-labeling problem, Amagasa et al [3] introduce float-point values for extending the "*start*" and "*end*" of the intervals. But their approach confronts with a problem, that is, a fixed place can only execute 18 insertions when the initial labels are consecutive integer values. This makes their approach provide limited benefits for the updates [36]. In [18], Li et al introduce a labeling scheme, which supports that nodes can be inserted between any two consecutive sibling nodes without re-labeling the existing labels. In their following work [21], by reusing the deleted labels, a compact dynamic binary string encoding approaches called CDBS and an improved compact dynamic quaternary string encoding approach called CDQS are proposed to avoid re-labeling when XML data updates. This labeling scheme seems to be effective, but it is more suitable to dynamic XML trees [43]. In fact, it is common for an XML repository to have XML trees that are frequently updated and those that are not. As a result, the system has to bear the burden of deciding whether trees are dynamic or not. This is usually difficult because the updating frequency of an XML tree may vary as time goes by.

2.2. Prefix based scheme

Prefix (Dewey) based scheme encodes the father of a node in a tree as a prefix of its label using a depth-first tree traversal. Dewey encoding is illustrated in Fig. 2. From Fig. 2, we can see that a