Extendible Range-Based Numbering Scheme for XML Document

Guangming Xing
Department of Computer Science
Western Kentucky University
Bowling Green, KY 42101
Guangming.Xing@wku.edu

Bill Tseng
Department of Architectural & Manufacturing Sciences
Western Kentucky University
Bowling Green, KY 42101
Bill.Tseng@wku.edu

Abstract

Range-based labeling scheme allows determining the ancestor relation between two nodes in constant time. One disadvantage is that relabeling is unavoidable when arbitrary insertions are allowed. In this paper, one practical algorithm is presented to extend range-based scheme to accommodate arbitrary insertions without relabeling by combining with prefix-based labeling scheme. Range allocation methods to improve the performance of the labeling method are also considered.

1 Introduction

XML becomes a standard format for data exchange and document presentation [1, 2] over the Internet. To facilitate XML document processing or query processing in XML databases, several methods were proposed to index XML data, which can be classified into two broad categories: range-based labeling and prefix-based labeling schemes.

One example of range-based labeling scheme is extended preorder labeling proposed in [3]. Each node x in the document is labeled with a pair $(order, size)$, where $order$ denotes the preorder of x and $size$ denotes the size of the subtree rooted at x. For two nodes u and v in the tree, u is an ancestor of v if and only if $order(u) < order(v) \leq (order(u) + size(u))$.

One disadvantage of using range-based scheme is that it is difficult to accommodate arbitrary insertions. Relabeling is unavoidable even though gaps can be reserved for future insertions, as gaps can be filled by a sequence of insertions. Although relabeling may be infrequent, very short sequences do exist for which relabeling is unavoidable [4].

To get the conditions to necessitate the relabeling, we consider insertion of a node y as the child of node x with label (p_x, s_x), whose children are c_1, c_2, ... c_n with labels (p_1, s_1), (p_2, s_2),... (p_n, s_n) from the left to the right, respectively.

As the new node y will be appended as the last child of x, the label of y only depends on the label of the last child (right-most) of x, which is (p_n, s_n), and the label of x itself, which is (p_x, s_x).

It is not hard to verify that the label of y, (p_y, s_y) must satisfy the following two conditions: 1. $p_y > p_n + s_n$; and 2. $p_y + s_y \leq s_x$. (*)

The other labeling scheme is prefix-based. In prefix labeling scheme, two nodes u and v are labeled with $L(u)$ and $L(v)$, we say u is an ancestor of v if and only if $L(u)$ is a proper prefix of $L(v)$.

For extended preorder labeling, $size$ of node v is an upper bound for the number of descendents rooted at node v, and for prefix labeling, the *level* of a node in an XML tree is defined as the length of the label. In the next section, these two definition will be generalized to accommodate the new labeling scheme.

2 Extending Range-based Labeling

In our new scheme, each node v is labeled with an integer pair prefixed with a sequence of integers: $p_1...p_n(o, s)$. We call n as the level of the label for node v. As will be explained later, the s (stands for $size$) is the number of descendents whose label is at the same level as v. In the above label $p_n = o_p$, and o_p is from $p_1..p_{i-1}(o_p, s)$, the label of v's nearest ancestor whose level is $n - 1$.

Extended preorder labeling scheme works perfectly except in the case of insertions, when there is no range available. Let's consider the scenario when node x is inserted as a child of node p. (p is labeled with $(p_1...p_{n-1}(o_n, s_n))$, it may have children or not):

1. Based on (*), range is available. The inserted node will be labeled the same way as in [3].

2. Based on (*), no range is available. The inserted node will be labeled with $p_1...p_n(o_{n+1}, s_{n+1})$. The prefix $p_1...p_n$ is used to record the ancestor information the same way as the prefix labeling, and (o_{n+1}, s_{n+1}) is used to create a new pseudo-root of a new subtree, making insertions to

Proceedings of the International Conference on Information Technology: Coding and Computing (ITCC'04)
0-7695-2108-8/04 $ 20.00 © 2004 IEEE

this newly created node the same way as insertions handled by extended preorder labeling.

It should be noted that the second branch will not be executed very frequently. Each time the second branch is entered, it means that the label for the newly inserted node is made one longer than its parent. Let's illustrate the algo-

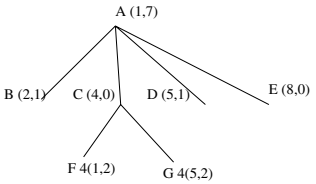

Figure 1. Illustration of our scheme

rithm using the above tree by the following insertions:

A, B, C, D, E, F, G.

When we try to insert D, we know that range is available as node C $(4, 0)$ is the right most descendent of A, and the tree rooted at A can have label up to 8 (calculated by $order + size$). But when node F is inserted as child of C, there is no range available ($size$ of C is 0, so it may not take any nodes as children in the original extended preorder labeling). We use the order of C (4) as the prefix, and append a range $(1, 2)$ in the next level.

In Figure 1., we know that nodes F and G can not be inserted as the children of node C without relabeling the whole tree, under the original extended preorder labeling scheme presented in [3], as the size of C is 0, which means the maximum number of children C has is 0.

It is easy to see that this new numbering scheme combines the advantages of range-based numbering scheme (determine the ancestor relation in constant time) and prefix-based numbering scheme (allow arbitrary insertions).

It should be noted that the $size$ here is different from the size used in the original paper [3]. The $size$ here refers to the upper limit of the descendents that are numbered in the same level (this node can still have descendents in the next level). Take Figure 1 as an example: the $size$ of node A is 7, so 7 is the upper bound of the number of descendents at the same level (nodes B, C, D, E, and the reserved range for one child of B, one child of D and one child of A between D and E, which are totaled to 7, but these do not include node F and G).

The main contribution of this paper is stated as the following theorem.

Theorem 1 *For any two nodes u and v in an XML document with label $o_{u,1}...o_{u,n-1}(o_{u,n}, s_{u,n})$ and $o_{v,1}...o_{v,m-1}(o_{v,m}, s_{v,m})$, we claim that u is ancestor of v iff $o_{u,1}...o_{u,n}$ is a proper prefix of $o_{v,1}...o_{v,m}$ or $o_{u,1}...o_{u,n-1} = o_{v,1}...o_{v,n-1}$, and $o_{u,n} < o_{v,m} < o_{u,n} + s_{u,n}$.*

It is easy to see that the prefix-based scheme can be viewed as a special case of our method: Inserted node will get $(c, 0)$ prefixed with its parent's label, where c is the order used in prefix-based scheme.

3 Range Allocation and Performance

Range allocation plays an important role for practical use of the above scheme. As illustrated in the previous section, if we restrict that the size of the inserted node be 1, the resulting labeling of the tree becomes a prefix labeling based on the new labeling scheme. Fortunately, DTD and the statistics about the document could be used to optimize range allocation.

4 Conclusions and Future Work

We presented a new labeling scheme to extend the extended preorder indexing method used in XISS [3] to accommodate arbitrary insertions by combining with labeling scheme. Based on the DTD and statistics about the document, we could achieve the same performance as extended preorder labeling while allows arbitrary insertions. Several potential applications of this methods are:

1. Native XML database: It would be interesting to integrate this work with the existing system developed by Li and Moon [3] and study how to use the statistics about the document to make range allocation more efficiently.

2. DOM: Range-based labeling is a very good alternative to the current method used for most DOM parsers when arbitrary update operations are allowed.

3. Temporal XML database, and version management for XML document.

References

[1] Abiteboul, S., Buneman, P., Suciu, D., *Data on the Web: From Relations tto Semistructured Data and XML*, Morgan-Kaufmann, CA, 1999.

[2] W3C. *Extensible Markup Language*, http://www.w3.org/TR/REC-xml.

[3] Li, Q., Moon, B., *Indexing and Querying XML Data for Regular Path Expressions*, Proceeding of the 27th VLDB, Roma, Italy, 2001.

[4] Cohen, E., Kaplan, H., Milo, T., *Labeling Dymanic XML Trees* PODS 02, June 2002.

[5] Bruno, N., Koudas, N., Srivastava, D., *Holistic Twig Joins: Optimal XML Pattern Matching*, Proc. of SIGMOD, 2002.