Chapter 6
Labeling XML Documents

Jiaheng Lu
School of Information and DEKE, MOE, Renmin University of China

Liang Xu
School of Computing, National University of Singapore

Tok Wang Ling
School of Computing, National University of Singapore

Changqing Li
Duke University, USA

ABSTRACT

XML labeling schemes play an important role in XML query processing. Containment and Prefix labeling schemes are two of the most popular labeling schemes. In order to perform efficient XML query processing, this chapter shows how to extend the traditional prefix labeling scheme to speedup query processing. In addition, for XML documents that are updated frequently, many labeling schemes require relabeling which can be very expensive. A lot of research interest has been generated on designing dynamic XML labeling schemes. Making labeling schemes dynamic turns out to be a challenging problem and many of the approaches proposed only partially avoid relabeling. This chapter describes some recently emerged dynamic labeling schemes that can completely avoid relabeling, making efficient update processing in XML database management systems possible.

INTRODUCTION

XML has become a de facto standard for data exchange and representation on the World Wide Web and elsewhere. To facilitate query processing over XML data that conforms to an ordered tree-structured data model, two main techniques have been proposed including structural index and labeling approach. Compared with the traditional methods that performs hierarchical traversal of the XML tree, the labeling approach benefits from smaller storage size and efficient establishment of various relationships such as Ancestor-Descendant (AD) and Parent-Child (PC). As a result, we consider labeling as the preferred approach for XML query processing.

We classify existing labeling schemes into two main bodies: **static labeling schemes** and **dynamic labeling schemes** which are designed for static and

DOI: 10.4018/978-1-61520-727-5.ch006

dynamic XML data respectively. **Containment labeling scheme** (Li&Moon, 2001; Zhang, 2001) and Prefix labeling scheme (Abiteboul, 2006; Tatarinov, 2002) are two of the most popular static labeling schemes. Many variants of them also have emerged that serve different purposes. For example, **extended Dewey labeling scheme** (Lu&Ling, 2005) is prefix-based and designed to further enhance the performance of twig pattern matching queries. The significance of extended dewey is that it provides not only structural information, but also the information about the names of the corresponding nodes. Twig pattern matching can benefit from extended dewey labels because only the leaf node labels need to be scanned, significantly reducing I/O cost during query processing. However, the static XML labeling schemes can suffer from high cost of re-labeling when the XML tree structure is subjected to change. To efficiently process updates for dynamic XML documents, a lot of works have been done on designing dynamic XML labeling schemes.

In this chapter, we present a survey on the existing labeling schemes. We begin with the introduction of static labeling schemes, describing how they support XML queries. Next we introduce dynamic XML labeling schemes which are developed to facilitate XML query processing as well as efficient updating. Finally we introduce the encoding schemes (Li&Ling, 2005; Li&Ling 2006; Xu&Ling, 2007) which can be applied to static labeling schemes to generate dynamic XML labels.

STATIC XML LABELING SCHEMES

In order to perform XML query processing efficiently, one way is to develop a labeling scheme to capture the structural information of XML documents to facilitate query processing without traversing the original XML documents.

The existing labeling scheme use a tree-traversal order (e.g. extended preorder [Li&Moon, 2001]) or textual positions of start and end tags (e.g. containment [Bruno, Koudas, & Srivastava 2005]) or path expressions (e.g. Dewey ID [Tatarinov et al. 2002]) or prime numbers (e.g. [Wu&Lee, 2004]). By applying these labeling schemes, one can determine the relationship (e.g. ancestor-descendent and parent-child) between two elements in XML documents from their labels alone. We introduce the labeling schemes for static documents as follows.

Containment Labeling Schemes

In the containment labeling scheme (or called region encoding) [Bruno et al. 2005, Shanmugasundaram 2001], each label includes 3-tuple (*start, end, level*). Based on the strictly nested property of labels, we can use them to evaluate the PC and AD relationships between element pairs in a data tree. Formally, element *u* is an ancestor of another element *v* if and only if

```
u.start < v.start and v.end > u.end
```

That is, the region of *v* is contained by that of *u*. To check the PC relationship, we additionally test whether element *u* is exactly one level above element *v* in the data tree (i.e., *u.level = v.level-1*). For example, Figure 1 shows an example XML tree with containment labels.

Dewey ID Labeling Schemes

In the Dewey ID labeling scheme [Tatarinov et al. 2002, Lu&Ling 2004] (or called prefix scheme), each element is presented by a vector:

1. The root is labeled by a empty string ε; and
2. For a non-root element *u*, label(*u*)=label(*s*).x, where u is the x-th child of *s*.