Available online at www.sciencedirect.com
ScienceDirect
Information and Software Technology 50 (2008) 462–479
INFORMATION AND SOFTWARE TECHNOLOGY
www.elsevier.com/locate/infsof

XTRON: An XML data management system using relational databases

Jun-Ki Min [a,*], Chun-Hee Lee [b], Chin-Wan Chung [b]

[a] School of Internet-Media Engineering, Korea University of Technology and Education, Byeongcheon-myeon, Cheonan, Chungnam 330-708, Republic of Korea
[b] Division of Computer Science, Department of Electrical Engineering & Computer Science Korea Advanced Institute of Science and Technology (KAIST), Daejon, Republic of Korea

Received 30 June 2006; received in revised form 1 May 2007; accepted 16 May 2007
Available online 29 May 2007

Abstract

Recently, there has been plenty of interest in XML. Since the amount of data in XML format has rapidly increased, the need for effective storage and retrieval of XML data has arisen. Many database researchers and vendors have proposed various techniques and tools for XML data storage and retrieval in recent years. In this paper, we present an XML data management system using a relational database as a repository. Our XML management system stores XML data in a schema independent manner, and translates a comprehensive subset of XQuery expressions into a single SQL statement. Also, our system does not modify the relational engine. In this paper, we also present the experimental results in order to demonstrate the efficiency and scalability of our system compared with well-known XML processing systems.

Keywords: XML; Relational database; Query

1. Introduction

As data are collected over diverse application areas, the requirement of inter-operability for sharing and integrating them has increased. Therefore, W3C has proposed the eXtensible Markup Language (XML) [4]. Due to its flexibility and self-describing nature, XML is considered as the *de facto* standard for data representation and exchange in the Internet.

To retrieve XML data, several query languages have been proposed. Among them, XQuery [2] is considered as the standard query language for XML data since it is broadly applicable across all types of XML data.

Since the amount of data in XML format has rapidly increased, the need for effective storage and retrieval of XML data has arisen. Many database researchers and vendors have proposed various techniques and tools for XML storage and retrieval [3,6,11,10,13,21,28,35,39].

Text file systems, native XML management systems (special-purpose systems), and traditional databases can be used as repositories for XML data. Using a text file system as an XML repository is the most convenient and prevalent approach. But, it is inefficient in retrieval since XML data is always parsed into an intermediate format such as a DOM tree.

An alternative for an XML repository is to use native XML database systems such as LORE [25] and Strudel [14]. LORE is designed for managing semi-structured data. Its data model is Object Exchange Model (OEM) which is a simple and nested object model. Strudel is a web-site management system and its data model is a labeled directed graph similar to OEM. Obviously, these works suggest valuable techniques and insights for the management of irregularly structured data. However, it is uncertain whether these approaches will be widely accepted in the real world since they are not mature enough to process queries on a large amount of data and in multi-user environments [17].

* Corresponding author.
E-mail addresses: jkmin@kut.ac.kr (J.-K. Min), leechun@islab.kaist.ac.kr (C.-H. Lee), chungcw@islab.kaist.ac.kr (C.-W. Chung).

0950-5849/$ - see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.infsof.2007.05.003

In addition, native XML management systems have two potential drawbacks as pointed in [33]. First, they do not use the existing mature storage and query capability. Second, they have difficulty in integrating the existing data, most of which is relational data. And, major DBMS vendors (SQL Server, Oracle, and DB2) have developed an XML management system using an RDBMS. Therefore, we focus an XML management system using an RDBMS.

1.1. The goals of XTRON

In this paper, we present an XML data management system using an RDBMS called XTRON. We have chosen to use an RDBMS due to its ability to behave as a stable repository as well as an efficient query optimizer and executor.

The design goals of XTRON are as follows:

- *No modification of the relational engine*: The modification of a relational engine may incur unintended side effects such as the consistency problem. Thus, the main goal of our design is to use the relational engine without modification.
- *schema independence*: Some works [23,36] ignore the schema information of XML data. In [39], the relational schema using DTD is different from that without DTD. Thus, a design goal of XTRON is to utilize schema information if it is available and to store XML data over identical relational tables whether DTD exists or not.
- *Efficient evaluation of path expressions*: To support an efficient evaluation of XML queries, some work uses path indexes which incur the modification of the engine. In contrast to the previous work, we represent a label path as an interval in $[0.0, 1.0)$. Using the containment relationships between intervals of label paths and an interval of the path expression, the path expression can be efficiently evaluated without the modification of a relational engine.

In addition, to demonstrate the efficiency and scalability of XTRON, we implemented XTRON and comparison systems: edge approach, region approach, and region with path table approach. Also, we show the effectiveness of XTRON compared with well-known XML processing systems: Galax and Berkeley DB XML.

1.2. Organization

The remainder of the paper is organized as follows. In Section 2, we present various methods for storing and retrieving XML data. After we show the architecture of XTRON in Section 3, we describe the details of storing XML data in XTRON in Section 4. In Section 5, we present mechanisms for XML data retrieval. Sections 6 and 7 show GUI of XTRON and the results of our experiments. Finally, in Section 8, we summarize our work and suggest some future studies.

2. Related work

Recently, in order to store XML documents using relational database systems, many XML storage systems and techniques using relational tables have been proposed.

With respect to mapping of the graph model to relational tables, these mapping schemes are basically classified into two groups: One is the *edge approach* [17] and the other is the *region approach* [20,23,36,37,41].

The edge approach stores the edges in the XML graph into the relational tables. In this approach, a unique node identifier (*nid*) is assigned to each node of the XML graph. An edge of the XML graph is represented as $<nid_1, label, nid_2>$ where nid_1 is a node identifier for the source of the edge, nid_2 is a node identifier for the target of the edge, and *label* is the label of the target node. Generally, the edge approach is efficient in computing parent–child relationships. However, the edge approach is inefficient in computing ancestor–descendant relationships since ancestor–descendant relationships are computed by the massive joins for parent–child relationships.

The edge approach has many alternatives according to the mapping rule from a set of edges to relational tables. Florescu and Kossman [17] provided three alternatives. The first one is to store all edges in a single table, called an edge table. The second one is to partition all edges with respect to the label. Then, each sub-group of edges is stored in distinct tables, called a binary table. The last one, a universal table approach, is to store all sequences of edges to leaf nodes of the XML graph in a single table which is equal to the result of a full outer join of binary tables.

Based on the edge approach, [35] suggested the inlining technique which utilizes the structural information contained in a DTD (Document Type Definition). The intuitive behavior of the inlining approach is that if one-to-many or many-to-many relationship between element nodes are defined in DTD, then sets of edges between the element nodes are mapped to different tables, otherwise (i.e., one-to-one relationship), sets of edges between the element nodes are mapped to the same table using the inlining technique. Thus, the inlining approach reduces the join overhead for evaluating queries. However, this inlining technique may lose the order information of child elements.

The region approach originated from the information retrieval (IR) field [7,29]. In this approach, XML data is considered as a tree structured data. As shown in Fig. 2 which is an example of the region approach corresponding to Fig. 1, this approach assigns a region to an element in XML data. Generally, a region is represented by a pair (start, end) consisting of the position of the start tag and the position of the end tag of an element.

The root node & 1 in Fig. 1(b) is represented by $(1, 2000)$ when XML data in Fig. 1(a) has 2000 words. In this case, a region satisfies the following property.