XRecursive, An Efficient Relational Method to Store XML Data

M.A. Ibrahim Fakharaldien ,K.F.Rabbi ,Jasni Mohamed Zain, Norrozila Sulaiman
Faculty of Computer System and Software Engineering,
University Malaysia Pahang,
Kuantan Malaysia
mohfakhrdeen@gmail.com,fazley.rabbi@ymail.com,*jasni@ump.edu*.my,*norrozila*
@ump.edu.my

***Abstract*—** Storing XML documents in a relational database is a promising solution because relational databases are mature and scale very well and they have the advantages that in a relational database XML data and structured data can coexist making it possible to build application that involve both kinds of data with little extra effort . In this paper, we propose an algorithm schema named XRecursive that translates XML documents to relational database according to the proposed storing structure. The steps and algorithm are given in details to describe how to use the storing structure to storage and query XML documents in relational database. Then we report our experimental results on a real database to show the performance of our method in some features.

Keywords: XML, Relational Database, SQl.

I. INTRODUCTION

Today's data exchange between organizations has become challenging because of the difference in data format and semantics of the meta-data which used to describe the data. Now day' XML emerged as a major standard for representing data on the World Wide Web while the dominant storage mechanism for structured data is the relational databases, which has been an efficient tool for storing, searching, retrieving data from different collection of data. The ability to map XML data in relational databases is difficult mission and challenging in the world of all IT organization so there is a need to develop an interfaces and tools for mapping and storing XML data in relational databases.

XML

The extensible Markup Language (XML) is quickly becoming the de facto standard for data exchange over the Internet [10] and now it plays a central role in data management, transformation, and exchange. Since it s introduction to industry in the

Late 1990s, XML [1] has achieved widespread support and adoption among all the leading software tools, server, and database vendor s. As importantly, XML has become the lingua franca for data by lowering the cost of processing, searching, exchanging, and re-using information. XML provides a standardized, self-describing means for expressing information in a way that is readable by humans and easily verified, transformed, and published, the hot topic is to seek the best way for storing XML documents in order to get high query processing efficiency[12]. In addition, data can be transmitted to remote services anywhere on the Internet using XML-based Web services to take advantage of the new ubiquity of connected software applications. The openness of XML [2] allows it to be exchanged between virtually any hardware, software, or operating system. Simply put, XML opens the door for information interchange without restriction.

Relational Databases

Today, the dominant storage mechanism for structured enterprise data is the relational database, which has proven itself an efficient tool for storing, searching for, and retrieving information from massive collections of data. Relational databases specialize in relating individual data records grouped by type in tables. Developers can join records together as needed using SQL (Structured Query

Language) and present one or more records to end-users as meaningful information. The relational database model revolutionized enterprise data storage with its simplicity, efficiency, and cost effectiveness. Relational databases have been prevalent in large corporations since the 1980s, and they will likely remain the dominant storage mechanism for enterprise data in the foreseeable future. Despite these strengths, relational databases lack the flexibility to seamlessly integrate with other systems, since this was not historically a requirement of the database model [3]. In addition, although relational databases share many similarities, there are enough differences between the major commercial implementations to make developing applications to integrate multiple products difficult. Among the challenges are differences in data types, varying levels of conformance to the SQL standard, proprietary extensions to SQL, and so on.

II. PROBLEM DEFINITION

For the storage of XML document, the key issue is transforming the tree structure of an XML document into tuples in relational tables [11].

Nowadays, there are more and more data presented as XML document, the need of storing them persistently in a database has increased rapidly while the native–XML databases usually have limited support for relational databases. In recent years, with the popularity of relational databases (RDB), approaches based on RDB[4,5,6,7,8,9,] to store and manipulate XML data as relational tables but still there is need to manage XML data and relational data seamlessly with similar storage and retrieval efficiencies simultaneously. XML and Relational databases cannot be kept separately because XML is becoming the universal standard data format for the representation and exchanging the information whereas most existing data lies in RDBMS and their power of data capabilities cannot be degraded so the solution to this problem a new efficient methods for storing XML documents in relational database is required.. A new efficient method for storing XML document in relational database is proposed in this paper to face these problems.

III. THE XRECURSIVE METHOD

A. XML Document

The data structure of XML document is hierarchical, consist of nested structures. The elements are strictly marked by the beginning and ending tags, for empty elements by empty-elements tags. Character data between tags are the content of the elements. It is an instance of XML document contains information about an employee as follows:

```
<? Xml version="1.0" encoding="UTF-
8"?>
<Personnel>
  <Employee type="permanent">
        <Name>Seagull</Name>
        <Id>3674</Id>
        <Age>34</Age>
   </Employee>
  <Employee type="contract">
        <Name>Robin</Name>
        <Id>3675</Id>
        <Age>25</Age>
    </Employee>
  <Employee type="permanent">
        <Name>Crow</Name>
        <Id>3676</Id>
        <Age>28</Age>
    </Employee>
</Personnel>
```

Fig.1 XML Document

B. The Tree Structure Representation Of XML Document

In this section, the structure independent mapping approach is explained with a sample XML document shown in Figure 1.

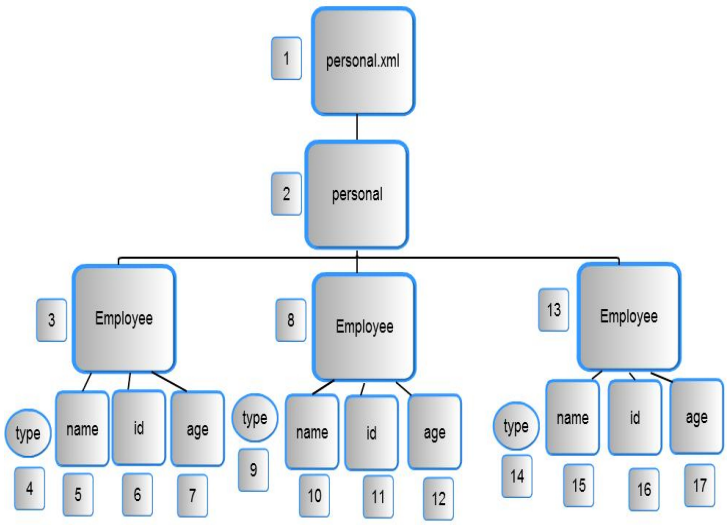

Figure.2 Tree structure of XML document with XRecursive labeling.

C. XRecursive Structure

Each and every XML can be describing as a XML tree. In this figure the squares are the elements and the ovals are the attributes of the elements. A generated XML tree has been shown in the figure. Every element or attributes are identified by a signature (number).

Definition: XRecursive Structure: XRecursive is a storage structure for storing XML documents where each path is identified by its parent from the root node in a recursive manner. In this structure when an element or type associates with its signature it also represents its parent element. We add document name in association with the id to be able to add multiple XML file in the storage. Figure 2 represents the storage of the XML file associated with its signature. For every element there will have a signature associated with it and there will also have a parent's signature associated with it. In table 1: tagName represents the name of the node; id represents the id of the node which is the PK. And finally pId represents the parent id of the node. As document name don't have any parent id so the id of the document name and parent id of the document name is same that has been shown in the figure 2.

tagName	id	pId
personnel.xml	1	1
Personnel	2	1
Employee	3	2
type	4	3
Name	5	3
Id	6	3
Age	7	3
Employee	8	2
type	9	8
Name	10	8
Id	11	8
Age	12	8
Employee	13	2
type	14	13
Name	15	13
Id	16	13
Age	17	13

Table 1: tag_structure

tagId	value	type
4	permanent	A
5	Seagull	E
6	3674	E
7	34	E
9	contract	A
10	Robin	E
11	3675	E
12	25	E
14	permanent	A
15	Crow	E
16	3676	E
17	28	E

Table 2:tag_value

In table 2, we represent the value associated with the elements or type. In XRecursive structure there is no need to store the path value or path structure as it will be determine recursively by its parent id. In Table 1: tagName is the name of the tag, where id is the parent key. In Table 2: tagId presents the Table 1 id thus tagId is the foreign key. In Table 2 tagId only represents the elements which contain a value and the value represents on the value column. And the type 'A' denoted to the attribute and 'E' denoted to the element.

IV. THE ANALYSIS OF EXPERIMENT

We ran experiments using our xml document in Fig.1. Our experiment was performed on 3.00

GHz Pentium 4 processor with 4GB RAM, 240 GB of hard disk running on windows XP system, we used MySQL v5.01 as the database for storing XML document and java language (Version jdk6) for parsing the XML document and then save it in MySQL . The experiment evaluates the efficiency of storing XML document in relational database based-on the XRecursive structure. . The experiment is made with respect to the following four factors:

1. Insertion Time
2. Database Size
3. Parse Time

Table 3 shows the final results of our experiment:

Insertion Time	1.194 Seconds
Database Size	0.001875 MB
Parse Time	0.064 Seconds

Table 3: Performance

A. Insertion Time

The insertion time of XML document in Fig.1 using XRecursive method is given in Table 3. AS seen in the Table 3, the XRecursive is fast and the reason could be that the data is stored in only two tables in this method.

B. Database Size

The database size for XML document in Fig. 1 using XRecursive method is given in Table 3 show that by this storage method we can reduce not only the size of database requirement of the labeling of node, but also the number of tables.

C. Parse Time

The time of parsing the XML document using XRecursive method is faster because it uses Document Object Model (DOM) parsing technique. Using DOM it read the nodes and corresponding child node.

D. XML Validation

XRecursive method also included with XML validation package. Before parsing the whole XML document, it checks that the XML file has valid DTD, grammar checking etc. It also shows the errors on the XML file with the specific line no.

V. CONCLUSION

XRecursive, a general storage method for XML document using relational database is proposed in this paper. XRecursive adopts the model-mapping method to store XML document in relational database, to decompose the tree structure into nodes and store all information of nodes in relational database according to the node types by recursive way. It can deal with any documents no matter whether it has fixed schema or not. By using this method we can reduce the database size require to store the XML document into relational database. The storing algorithm of XML document into relational database was also given in the paper, and examined the accuracy of it by using the XML document in performance section. Utilizing the actual Xml document evaluated the performance of storing XML document into relational database by using our method.

REFRENCE

1- Grandi, F., Mandreoli, F., Tiberio, P., and Bergonzini, M. (2003). A temporal data model and management system for normative texts in XML format. In Proceedings of the 5th ACM international Workshop on Web information and Data Management (New Orleans, Louisiana, USA, November 07 - 08, 2003).

2- Augeri, C. J., Bulutoglu, D. A., Mullins, B. E., Baldwin, R. O., and Baird, L. C. (2007). An analysis of XML compression efficiency. In Proceedings of the 2007 Workshop on Experimental Computer Science (San Diego, California, June 13 - 14, 2007).

3- Reed, D. (2006, February). Take a good look. Data Strategy, 2(4), 24-29. Retrieved November 17, 2008, from Business Source Complete database.
4- Sybase Corporation: Using xml with the Sybase adaptive server sol databases. Technical whitepaper, August 21 1999.
5- Xrel: a path-based approach to storage and retrieval of xml documents using relational databases. ACM Trans. Interet Technol., 1(1):110–141, 2001.
6- Xparent: An efficient rdbms-based xml database system. InICDE ’02: Proceedings of the 18th International Conference on Data Engineering, page 335, Washington, DC, USA, 2002.IEEE Computer Society.
7- H. Jiang, H. Lu, W. Wang, and J. X. Yu. Path materialization revisited: an efficient storage model for xml data. In ADC ’02: Proceedings of the 13th Australasian database conference, pages 85–94, Darlinghurst, Australia, Australia, 2002.Australian Computer Society, Inc.
8- J. Kyung-Soo. A design of middleware components for the connection between xml and rdb. In Proceeding of the IEEE International Symposium on Industrial Electronics, pages1753–1756, 2001.
9- M. Rys. Microsoft sol server 2000 xml enhancements. Microsoft Support Webcast, April 2000.
10- Hasan Zafari, Keramat Hasami, M.Ebrahim Shiri. Xlight, An Efficient Relational Schema To Store And Query XML Data. In Proceeding of the IEEE International conference in Data Store and Data Engineering. pages 254-257, 22 April 2010
11- Liew Yue, Jiadong Ren, Ying Qian, Storage Method of XML Documents Based-on Pri-order Labling Schema. In Proceeding of the IEEE International Workshop on Education Technology and Computer Science. Pages 50-53, 30 December 2008.
12- Liu Sainan, Liu Caifeng, Guan Liming, Storage Method for XML Document based on Relational Database, In Proceeding of the IEEE International Symposium on computer Science and Computational Technology. Pages 127-131, 2009.