# Efficient Mapping XML DTD to Relational Database

Mohammed Adam Ibrahim Fakharaldien[1], Khalid Edris[2], Jasni Mohamed Zain[3], Norrozila Sulaiman[4]

*Faculty of Computer System and Software Engineering,University Malaysia Pahang,*
*Kuantan, Malaysia*
*mohammedfakeraldeen@yahoo.com, Khalod07@yahoo.com,*
*jasni@ump.edu, norrozila@ump.edu*

**Abstract.** Mapping XML documents into 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 two algorithms (XRD, XRS) that maps XML documents into relational database. The algorithm steps are given in details to describe how to map the XML documents into 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,XRD,XRS

## 1. 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.

The extensible Markup Language (XML) is quickly becoming the de facto standard for data exchange over the Internet (1) and now it plays a central role in data management, transformation, and exchange. Since it s introduction to industry in the Late 1990s, XML (2) 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(3). 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 (4) allows it to be exchanged between virtually any hardware, software, or operating system. Simply put, XML opens the door for information interchange without restriction.

## 2. 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 (5). 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 (6). 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.

## 3. Problem definition

Nowadays, there are more and more data presented as XML document, the need of mapping and storing them persistently in a database has increased rapidly. In recent years, with the popularity of relational databases (RDB), many researchers and vendors (including Oracle, Sybase, IBM, and Microsoft) have proposed their approaches based on RDB to mapping XML data into 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 is an efficient way to map XML documents into relational database.

## 4. The Proposed Algorithms

The process of shredding XML data into the flat tuples to be stored into the target database looks like a straightforward process once relational schema and schema mapping information is generated. However we see that there are several challenging issues which make this data mapping process nontrivial. Although schema mapping is done only once, data mapping is an on-going process. We usually deal with large size of XML documents in the data mapping phase unlike schema mapping phase where the input DTD sizes are very small compared to that of document sizes. This deference makes the efficiency a critical issue in data mapping where it can be tolerated in the schema mapping.

Data mapping takes a valid XML document and the output of a schema mapping as input, shreds the XML document into relational tuples, and inserts them into the relational database. In the following, we formalize the notion of data mapping:

**Definition.1** (Data Mapping) A data mapping DM is a function that assigns to each triple (X, R, σ) a set of relational tuples T, where X is a valid XML document, R is a database schema, σ is a σ-mapping over R, and T is the result of shredding X into relational tuples according to the layout described by R and σ.

As the target database schema might be complex and its corresponding XML-to-Relational schema mapping is non-trivial, it is challenging to design an efficient schema-based data mapping algorithm. The main challenging issues include the following:

Varying document structure. XML documents have varying structures due to the optional occurrence operators '?', '*', and choice operator '|' used in the underlying DTD, unlike relational tables which always have a fixed structure. A data mapping algorithm should keep track of the missing child nodes and handle structural deference between the same types of element nodes due to the optional operators using efficient data structures.

- Scalability. In an online environment, where new XML documents might be inserted into the database on-the-fly, a data mapping algorithm will be used frequently. Thus, it is critical that a data mapping algorithm is efficient and scales well with the size of XML documents. It is obvious that a linear data mapping algorithm will fulfill this requirement the best.

In the following sections, we present two data mapping algorithms, DOM-based XRD and SAX-based XRS to address these issues. An appropriate ordering technique is needed to keep the ordered XML documents in the unordered structure of relational tables. Several order encoding methods are proposed in [16]. Their experimental results show that the global order encoding performs the best on query intensive workloads. How-ever, our data mapping algorithms can be easily adapted to other order encoding schemes proposed in [16]

```
<? 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

### 4.1. SAX-Based Approach

SAX-based [17] data mapping approach only needs to process the document in one run. Our SAX-based data mapping algorithm, called XRS, is given in Figure 2. The data mapping algorithm SDM takes an XML document X, a database schema R and a σ-mapping σ as input as described in Definition 1. Event driven XRS algorithm makes a sequential scan of the whole document from top to bottom. It triggers procedures startElement(), characters(), and endElement() for start tags, character data and endtags respectively.

```
Algorithm XRS
00 procedure StartElement(Element e)
01 Input: XML Tree T, Database schema R
02 Output: elements in T are inserted into the relational database
03 Begin
04 for each Element do
05 Begin
06 if e is root then
07 Begin
08 id=1; pid=1;
09 End
10 Else
11 Begin
12 pid=e.parent.id; id=id+1;
13 End
14  End if
15 tagstructure(Element e)
16 End for
17 End Algorithm
00 Procedure tagstructure(Element e)
01 Begin
02 create a tuple tp of table tag_structure
03  tp.tagname=e.name
04 tp.id=e.id
05 tp.pid=e.pid
```

Figure 2: SAX-based data mapping algorithm XRS

### 4.2. DOM-Based Approach

DOM-based algorithms are popular because W3C adopts DOM as its standard for XML description. For a big XML file, or multiple XML files processed in multi-tasking environment, creating DOM trees is expensive. A DOM-based data mapping algorithm processes a document in two runs: in the first run, the parser browses the document and creates an XML tree in the main memory. In the second run, the data mapping algorithm accesses to this DOM tree and processes it. We use a tree data model to represent the XML documents since each valid XML document is rooted at a unique element which is specified by DOCTYPE declaration in the DTD.

```
00 Algorithms XRD
01 Input: XML Tree T, Database schema R
02 Output: elements in T are inserted into the relational database
03 Begin
04 create a tuple tp of table       tag_structure
05 create a tuple tp of table tag_value
06 Queue q:=EmptyQueue( )
07 q.enqueue (T.root)
08 while q is notEmpty( ) do
09 Begin
10 e:=q.dequeue( )
11 if e is root then
12 begin
13 id=1
14 pid=1
15 end
16 else
17 begin
18 pid=e.parent.id
19 id=id+1
20 end
21 end if
22 tagstructure(node n, tuple tp)
23 End while
24 End Algorithm
00 Procedure tagstructure(node n, tuple tp)
01 Begin
02 tp.tagname=n.name
03 tp.id=n.id
04 tp.pid=n.pid
05 insert tuple tp into table tag_structure
06 if n is leaf then do tagvalue(node n,tuple tp) End if
07 End
00 Procedure tagvalue(node n,tuple tp)
01 Begin
```

Figure 3: DOM-based data mapping algorithm XRD

## 5. Result and Discussion

We used a Pentium IV computer with 2.4 GHz processor and 1 GB main memory for the experiments. The experiments were run using Java 1.4.2 software development kit. We minimized the usage of system resources during the experiments to get more realistic results. We ran the programs 6 times and got the average value, excluding the first run, to have more accurate results. We chose auction.xml of the XMark benchmark [18] as our data set to compare the performance of the DOM-based algorithm XRD with the SAX-based algorithm XRS. We generated the test documents in 6 different sizes ranging from 25 MB to 125 MB. We constructed the XML Tree for each document using W3C's DOM specification. Our performance metric is the time to map the input XML document to the target relational data. While loading data to the database is not included in this time, the time for parsing the input XML documents is included in the measurement. The chart given in Figure 4. Shows the average time spent for each document using the two data mapping algorithms.

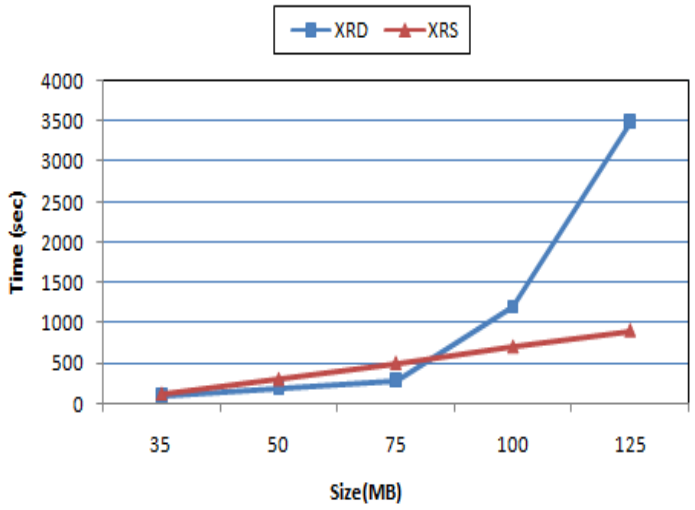


Figure 4: The performance of XRD versus XRS

As shown in Figure 4. XRS shows linear performance and scales very well with the size of the input XML documents while XRD shows linear performance up to the 75 MB document. DOM-based data mapping algorithm XRD has much better performance than the SAX-based algorithm XRS up to the 75 MB XML document. However, after 75 MB, the SAX-based algorithm starts to outperform as XML documents beyond 75 MB could no longer be represented as a DOM tree in the main memory in our experiments. For a large XML document whose XML tree does not fit in the main memory, part of the tree will be

swapped between the disk and the main memory, causing a considerable time on60 I/O operations and degrading the performance of the DOM-based approach. In this case, the event-driven SAX-based approach does not suffer.

We observed from our experiments that, as long as the document tree can fit in the main memory, the DOM-based approach for data mapping should be chosen. Otherwise, the SAX-based approach should be the choice for data mapping.

## 6. Conclusion

In this paper, we identified the challenging issues for the data mapping problem and proposed two linear data mapping algorithms, XRD and XRS, based on two well- known XML parsers DOM and SAX, respectively. While XRD is a generic model for a main memory data mapping algorithm, SDM represents a generic streaming data mapping algorithm.

We compared the performance of these algorithms. Experimental studies showed that these algorithms are efficient and well scalable with respect to the size of input documents, which is an important issue in the data mapping process. We justified with the experiments that our data mapping algorithms are generic enough to work with the classical schema mapping algorithms introduced in [19].

## 7. References


1- HasanZafari, Keramat Hasami, M .Ebrahim Shiri 2010. *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.

2- M.A. Ibrahim Fakharaldien, Siti Normazial Ihsan, Jasni Mohamed Zain, 2010. *A Middleware Prototype for Storing and Quering XML DocumentsIn RDB Using XParent Model Mapping Schema*. In proceeding of the IEEE International conference in Electronic and Information Engineering, Pages V2-560 - V2-563, DOI: 10.1109/ICEIE.2010.5559745.

3- Liu Sainan, Liu Caifeng, Guan Liming, 2009 *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.

4- 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 Workshop on Experimental Computer Science, San Diego, California.

5- M.A. Ibrahim Fakharaldien, Jasni Mohamed Zain, , Norrozila Sulaiman 2011. An Efficient Middleware for Storing and Querying XML Data in Relational Database

Management System. Journal of Computer Science, Volume7,Issue2,Pages 314-319,DOI: 10.3844/JCSSP.2011.314.319.

6- Reed, D 2008. Take a good look. Data Strategy, from Business Source Complete database. , 2(4), 24-29.
LiewYue, JiadongRen, YingQian 2008, Storage Method of XML Documents Based-on Pri-order Labeling Schema. In Proceeding of the IEEE International Workshop on Education Technology and Computer Science.

7- R. Goldman, J. McHugh, and J. Widom 1999. From semi structured data to XML: Immigrating the lore data model and query language, in: Proceedings of WebDB 99/, pp. 25–30.

8- 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.

9- I. Tatarinov, S 2002. Viglas, K. Beyer, et al., Storing and querying ordered XML using a relational database system, in Proceedings of the ACM SIGMOD.

10- M. Ramanath, J. Freire, J. Haritsa, P. Roy 2003.Searching for efficient XML-to-relational mappings, in: Proceedings of the International XML Database Symposium.

11- J. Shanmugasundaram, K. Tufte 1999. Relational databases for querying XML documents: limitations and opportunities, in: VLDB.

12- P. Bohannon, J. Freire, P. Roy, J. Simeon 2002 From XML schema to relations: a cost-based approach to XML storage, in Proceedings of IEEE ICDE, 2002.

13- D. Florescu, D. Kossman 1999. Storing and querying XML data using an RDBMS, IEEE Data Engineering Bulletin.

14- M. Yoshikawa, T. Amagasa, T. Shimura, S. Uemura 2001, XRel: a path-based approach to storage and retrieval of xml documents using relational databases, ACM TOIT 1 (1) 110–141.

15- F. Tian, D. DeWitt, J. Chen, C. Zhang 2002.The design and performance evaluation of alternative XML storage strategies, ACM Sigmod Record.

16- Tatarinov, I., Viglas, S., Beyer, K. S., Shanmugasundaram, J., Shekita, E. J.,and Zhang, C. (2002). Storing and querying ordered XML using a relational database system. In SIGMOD Conference, pages 204–215.

17- Megginson, D. (2004). Simple API for XML. SAX. http://www.saxproject.org/.

18- Schmidt, A., Waas, F., Kersten, M. L., Carey, M. J., Manolescu, I., and Busse,R. (2002). XMark: a benchmark for XML data management. In VLDB, pages 974–985.

19- Shanmugasundaram, J., Tufte, K., Zhang, C., He, G., DeWitt, D. J., and Naughton, J. F. (1999). Relational databases for querying XML documents: Limitations and support unities. In VLDB, pages 302–314.