Issue11 vol 3 (June 18) ISSN: 2321-8134

IJFEAT

INTERNATIONAL JOURNAL FOR ENGINEERING APPLICATIONS AND TECHNOLOGY

An Efficient Path based Mapping of XML Document to Relational Database

Atul D. Raut

[1]*Prof. & HOD, IT Deptt. ,J.D.I.E.T. Yavatmal, , Maharashtra, India,* ***atuldraut@gmail.com***

Abstract

RDBMS is a well known and very extensively used data storage and querying format for flat unordered data. XML has gained prominence as data storage and exchange format for web applications, but XML data is hierarchical and ordered in nature. Hence mapping of XML to Relational data is very challenging and difficult, but is required to gain all the advantages of RDBMS. In this paper a path based approach for mapping XML document to Relation tables is proposed. This technique successfully maps an XML document using one important table and some other tables without the requirement of a schema or a DTD.

Index Terms: *Path table, path summary, count, sequence list.*

--- *** ---

1. INTRODUCTION

In many organizations it is required to map the data that is in one form to another form for reasons depending on the specifications of the projects. For example, the users may prefer to store all the data which is in XML files in the relational database or the relational data in the XML format. There may also a need to perform the operations which can be done on XML in relational database and vice versa such as run queries against the relational database and generate results in XML. Two organizations using different format of data storage, may need to exchange data. XML being platform independent is one main reason why people prefer data transfer in the XML file format [1]. One major issue in mapping XML to Relational table is the loss of information due to shredding XML documents and in lining the shreds to Relational tables.

To map the XML document to relational table the technique first creates a path summary. This path summary contains all the unique root to leaf paths of tree representation of a XML document. This path summary is first mapped to relational table pattab. This relational table pathtab contains four attributes or columns namely the pathid, path, count and path sequence list. The structure of path table appears as below.

Pathid	Path	count	Sequence list

Pathid is the sequence number of the path, path is a sequence of node names seprated by the character '/' from root node to one level above the leaf node in the tree representation of XML document. Count is the number of times the particular path appears and Sequence list is the list containing the sequence numbers for that path.

The other tables required for this technique depends on the number of entries in the path table pathtab. If pathtab contains six entries then this technique will require six other tables to completely map the given XML document to relational database. The main contributions of the proposed technique can be summarized as below

1. This technique can be used to fire an sql query on the given XML document.
2. It can also be used to fire an xpath query over XML document represented in the form of relational database efficiently
3. Does not require a DTD.
4. Query processing requires very less main memory as compared to document storage. Hence the query response time is very less.

2. RELATED WORK

Three major approaches have been proposed for mapping XML to relational database and querying XML data. The first approach develops a native XML databases that support the XML data model and XML query languages directly. This includes Software AG's Tamino XML Server, IXIA's TEXTML Server, Sonic Software's eXtensible Information Server and MODIS's Sedna Native XML DBMS. The second approach makes use of technologies, such as relational DBMSs , to store and query XML data [2–7]. This is a very challenging approach since XML is ordered and hierarchical where as Relation tables are flat and unordered. . This approach can be broadly classified into two categories. The first one is the schema less or structure centric approach, which uses the XML structure to guide the mapping process [2-4] [6-7]. The second is schema centric approach which uses the XML schema or a XML DTD to map XML to Relational storage. This approach has a major drawback of reconstructing the database whenever the XML schema changes which is very expensive [5]. The third approach makes use of XML support enabled by commercial database systems. Currently, most major databases, such as SQL Server, Oracle and DB2, provide mechanisms to store and query XML data by extending the existing data model with an additional XML data type (e.g.,XMLType in Oracle 10g) so that a column of this data type can be defined and used to store XML data. In addition, a set of methods is associated with this new XML data type to process, manipulate and query stored XML data.

3. PATH SUMARRY BASED MAPPING OF XML TO RELATIONAL STORAGE

Creating the path summary of the given XML document is the most important step of the proposed technique. The structural summary once created can be used for self indexed and compact storage of the entire XML document and also for the subsequent querying. The following section describes the creation of structural summary.
The structure index of an XML document is a tree, such that every path from the root to some leaf node in the document appears exactly once in the structure index. The structure index *SI(D)* of an XML document *D* is a tree, whose nodes are labeled with element and attribute names from the document. The relationship between *D* and *SI(D),,CI(D)*(Content Index)can be described based on a function φ: *(D)* → *SI(D),CI(D)* recursively defined as follows:

I. Φ maps the root of *D* which includes the element name and attributes names without its values into the root of *SI(D)*. The two nodes have the same label.
II. Let *children (n, l)* represent the set of all the *l*-labeled XML elements in *D*, consisting of elements names and attribute names without their values, which are children of the XML element *n*. If *children (n, l)* is not empty, then *φ(n)* has a unique *l*-labeled child n_l in *SI(D)* and for each n_i *€child(n, l)*, $\varphi(n_i)$ is n_l .

From implementation point of view the path summary is represented in the form of array of strings stored in path table pathtab. The path is constructed by traversing the path summary tree from root node to one particular leaf node and adding a / in front of each node name which may include the attribute names without their values. Mathematically this process can be represented as

$$/r/n1/n2/\ldots\ldots\ldots\ldots/n_{li} \longrightarrow SI[i]$$

where
r $\longrightarrow$ the root node of the path summary tree
n1 ,n2 $\longrightarrow$ node names on the path summary tree including the attribute names without their values, if the attributes are present in the node
and n_{li} $\longrightarrow$ the leaf node name on the path summary tree
The subscript of the array of string acts as an index id and provides a link to the attributes values and contents on this path.
III. The function φ(D) maps each of the text node T(D) in the XML document to a unique path in the path summary tree which is finally mapped to the structure index. This mapping can be represented as
φ(D) :T(D)→PathSummary(D)→S[I]

Using the index value of I a table pathtabI is created and the contents of text node on this path is stored in the pathtabI. This kind of path based storage leads to element less, self indexed storage of XML document.
Thus when the above rules are applied to a XML document, it maps the entire XML document to Relational tables. This storage then could be utilized directly for firing sql or xpath queries over the Relational representation of the given XML document. Consider the following partial fragment of sample XML document.

```
<table ID="lineitem">
        <T>
            <L_ORDERKEY>1</L_ORDERKEY>
            <L_PARTKEY>1552</L_PARTKEY>
            <L_SUPPKEY>93</L_SUPPKEY>
            <L_LINENUMBER>1</L_LINENUMBER>
            <L_QUANTITY>17</L_QUANTITY>
            <L_PRICE>24710.35</L_PRICE>
        </T>
</table
```

For convenience purpose in the above XML only one <T> element is shown. The path table pathtab for above XML will consists of following entries

Pathid	Path	count	Sequence list
1	table/T/L_ORDERKEY	1	<1>
2	table/T/L_PARTKEY	1	<2>
3	table/T/L_SUPERKEY	1	<3>
4	table/T/L_LINENUMBER	1	<4>
5	table/T/L_QUANTITY	1	<5>
6	table/T/L_PRICE	1	<5>

Since the path table pathtab has six entries following six additional tables will be required to completely map the XML document to Relational tables

Pathnum	Orderkey
1	1

Pathnum	Partkey
2	1552

Pathnum	Superkey
3	93

Pathnum	Linenumber
4	1

Pathnum	Quantity
5	17

Pathnum	Price
6	24710.35

4. QUERYING XML DOCUMENT

The function φ() which is used to map the XML document to Relational tables can also be used to specify a SQL or Xpath query as explained below.

III. The function φ(PQ) (where PQ indicates a path query containing parent-child, ancestor-descendant or mixed type of relationship or SQL query) maps the query to a unique path in the path summary tree which is then mapped to pathtab returning the index value. Using the index value corresponding data table is accessed to give the query results. This can be represented as

φ(PQ)→SI(D)→DI(i)

Consider the following xpath query

table/T/L_ORDERKEY

This query will be mapped to first row in the path table pathtab returning an index of 1. Using this index value of 1 first table is accessed to get the result of the query.

Now consider the following SQL query

Select L_ORDERKEY
From L_ORDERKEY
Where L_ORDERKEY>0

The names appearing in the form clause are table names and they are obtained from the path table pathtab as given below. Each entry in the path table is scanned. For every row in the path table, the contents are tokenized using / or // as the separator and the last token is obtained. This last token is compared with the names appearing in the form clause. If there is a match, then using the index value of the row (which is 1 in this case) corresponding table (which is table 1 in this case) is accessed to get the results of the given query.

5. EXPERIMENTAL RESULTS

The proposed technique for mapping XML to Relational tables is implemented on Intel Pentium IV dual core 3.0 GHz processor having 2 GB of DDR RAM with VB.NET running on Windows platform. The technique requires high main memory for data storage. This involves reading the XML document and creating the path table and other related tables. Query processing using the proposed technique on the other hand requires very less main memory.

5.1 Test Data Set

XML test data sets can be broadly classified as data centric or document centric. Data centric document are those which have regular structure, where as document centric XML documents have irregular structure and contains lot of textual data. Based on the references of several XML data sets, XML documents can be further classified as

1. XML documents with no or negligible attributes.
2. XML documents with sufficient number of attributes.

Table 1 provides the size, total number of root to leaf paths and number of unique root to leaf paths of the above data set.

Table 1: Data set and its characteristics for the proposed approach

Sr. No.	Document Name	Size(MB)	Number of unique root to leaf paths	Total number of root to leaf paths
1	Shakespeare	7.5	161	146888
2	Orders	5.12	9	135000
3	Lineitem	30.7	16	962800
4	Treebank	84	220893	132535800
5	Mondial	1.7	291	44698
6	Xmark	4.96	65	90167
7	DBLP	5.33	60	80345

5.2 Query Performance

Query performance can be measured by the query response time. This section presents the results of execution of different types of queries, on the test data set. Table 2-3 shows response time and number of blocks/files transfer for different types of xpath and SQL queries on Orders XML documents.

Table 2: Response Time for xpath on "Orders" XML document

Sr.No.	Query	Time in Milliseconds	No. of block transfer
1	/table@id=orders/T/O_ORDERKEY	41	2
2	/table@id=orders/T/O_CUSTKEY	29	1
3	/table@id=orders/T/O_ORDERSTATUS	24	1
4	/table@id=orders/T/O_ORDER-PRIORITY	30	1
5	/table@id=orders/T/O_CLERK	32	1
6	//T/O_SHIP-PRIORITY	19	1
7	//T/O_COMMENT	17	1

Table 3: Response Time for SQL queries on “Orders” XML document

Sr.No.	Query	Time in Milliseconds	No. of block transfer
1	Select L_ORDERKEY From L_ORDERKEY Where L_ORDERKEY>0	37	2
2	Select L_PARTKEY From L_PARTKEY Where L_PARTKEY >0	25	1
3	Select L_SUPPKEY From L_SUPPKEY Where L_SUPPKEY >0	24	1

6. CONCLUSION

In this paper a path based mapping of XML document to Relation database is proposed. The path based grouping and storage of the contents of XML document used in the proposed technique, results in reduction of the number of block transfer and hence this technique requires minimum query response time for xpath as well as SQL queries. The path table in combination with other tables represents parent-child and ancestor-descendant relationship among the nodes of the XML document.

REFERENCES

1. Jandhyala, Sandeep, "An Automated XPATH to SQL Transformation Methodology for XML Data." Thesis, Georgia State University, 2006.
http://scholarworks.gsu.edu/cs_theses/21

2. Zhuyan Chan et. al, “Index Structures for Matching XML Twigs using Relational Query Processor,” in *Proc Data engineering workshop ICDEW* ,pp. 5-8 April 2005.

3. Ibrahim Dweib, Ayman Awadi and Joan Lu. “MAXDOR: Mapping XML document into relational database,” *The Open Information System Journal.*, vol. 3, pp. 108-122, June 2009.

4. Igor Totarinov, Stratis D Vigals, Kevin Beyer et.al., “Storing and Querying Ordered XML using a Relational Database System,” in *Proc. ACM SIGMOD Int’l Conference on Management of Data, Madison Wisconsin USA*, pp. 204-215, 2002.

5. Sandeep Prakash, Sourav S Bhowmick and Sanjay Madria ,“Efficient recursive XML query processing using relational database systems,” *Data and Knowledge Engineering Journal*, vol.58 issue 3, pp 2007-242, 2006.

6. Jun-Ki Min, Chun-Hee Lee, and Chin-Wan Chung, “XTRON: An XML data management system using relational databases,” *Information and Software Technology Journal*, vol. 50, issue 5, pp. 462-479, 2008.

7. Shankar Pal, Istvan Cseri, Oliver Seeliger, Gideon Schaller, Leo Giakoumakis and Vasili Zolotov, “Indexing XML data stored in a relational database,” in *proc. 30th VLDB Conference*, Toronto, Canada, pp 1146-1157, 2004.

.