A Model Mapping Approach for storing XML documents in Relational databases

Dr. Mrs. Pushpa Suri[1], Divyesh Sharma[2]

[1] Department of Computer Science & Applications, Kurukshetra University, Kurukshetra, Haryana, India

[2] Department of Computer Science & Applications, Kurukshetra University, Kurukshetra, Haryana, India

Abstract

The Extensible Markup Language (XML) is used for representing data over the web. Storing XML documents in relational databases uses two kinds of approaches: Model mapping and Structured mapping. This paper explores a model mapping approach for storing XML data in relational database which use two tables in it: Node table and Data table. Node table stores all node id's along with node names. Data table stores corresponding node values in it. We also propose an algorithm that shows how the nodes of the XML document are stored in terms of tables in database.

Keywords *Extensible markup language, Document type descriptor, Relational database, Structured mapping approach, Model mapping approach.*

1. Introduction

XML (Extensible Markup Language) is emerging as a standard format for data and documents in the internet. Storing and managing XML documents using relational databases are an attractive area of research for the researchers since relational data bases are mature and handle XML queries efficiently. Two types of strategies arise when we want to store and query XML documents into relational data bases:-

Structured Mapping Storage: When the structure i.e. DTD(Document Type Descriptors) or XML schema is given with the document and used to translate XML to database schema. Many approaches in [1][2][3] uses DTD information to generate database schema from available XML schema. Several inlining techniques like basic, shared and hybrid in [1] comes which solves complex DTD associated with the XML document by simplification rules. After simplification of DTD, inlined DTD graph is drawn and then generate database schema according to the inlined DTD graph. In [2],[3] different inlining algorithms proposed to generate database schema.

Model Mapping Storage: When no DTD or no XML schema is associated with the XML document In [4],[5],[6][8][9][10] several Model mapping schemes are used to store and query XML documents. These approaches does not dependant on the complexity of the XML schemas or its structure.

This paper explores a model mapping schema which is DTD independent and requires two tables to store XML document in database. First XML document is decomposed in to tree and then the nodes of the tree are mapped to the relational schema in the form of the tables in relational database.

2. Related Work

Many researchers proposed different model mapping schemes Edge[4],XRel[5],XLight[8],XRecursive[9],X[PEV][6],LNV[10] .In Edge [4] approach XML document is srored as edges in database and has the following structure: Edge (source, ordinal, target, flag). Source stores the OID's of the elements, ordinal stores the order of the attribute and flag stores the inter object reference or object values. Values of the nodes can be stored in separate tables.

X[Rel] [5]approach used four tables to store XML documents in relational database. The schema is as follows:-

Attribute(doc id, path id, start, end, value)
Text (doc id, path id ,start, end, value)
Path(path id, pathexpr)
Element(doc id, path id, start, end, index, reindex)

The concept of region was given by this approach. Element table associates each path with a region (Start, end). There was no edge information explicitly maintained in this schema. A region is specified by start and end points of a node. Attribute table represents s attributes values, Text table stores text values, Table Path stores the distinct paths, Element table stores start and end values of a node. This approach requires large number of join operations in query processing of XML documents.

X[PEV] [6] approach used three tables.

Path(path id, pathexpr)

Edge(path id, doc id, source ,target, label, order, flag)
Value(path id ,doc id, source, target, label, order, value)
Path table is same as in X[Rel][4].Edge table stores the parent child relationship among the nodes. Value table stores the values of elements attributes and the leaf. This scheme stores the edges explicitly.
We don't use path concept. Instead of storing the distinct paths, we store parent node id along with every node id to maintain parent child relationships of the nodes. Firstly we assign a unique identifier to each node and then the corresponding node values along with each node id are stored in relational table.

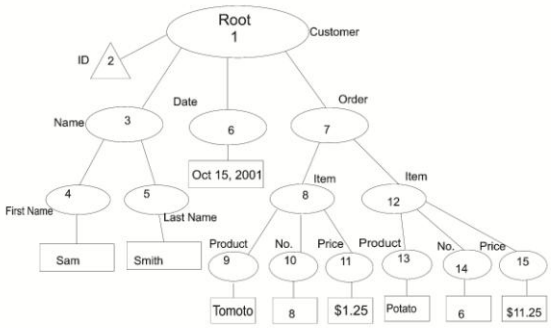

Fig. 2 : XML Data Graph of XML document in fig. 1.

3. XML Data Model

A well defined database system is based as a well defined data model. XML data model is often simplified to a labeled tree or a directed graph including elements with their character data and attributes with their values [7]. There are different data models like X path 1.0,DOM model are there to transform XML document as a tree. The X query 1.0 and X path 2.0 data models XML documents as are ordered graph using 7 types of nodes [4]. The description of these models are found in [7].
In this paper we consider four types of nodes i.e. root, element, text, attributes. Fig.2. is an example of an XML data graph which is used to represent the XML document as a tree. In this all the round nodes are element nodes. All the triangle nodes are attribute nodes. All the rectangles are the text values. Every node is represented by a unique identifier here.

```
<Customer id=1>
<Name>
     <first name>Sam </first name>
      <last name>Smith</last name>
</Name>
<Date>October 15,2001</date>
<Orders>
<Item>
       <Product>Tomato</Product>
       <Number>8</Number>
      <Price>$1.25</Price>
</Item>
<Item>
       <Product>Potato</Product>
       <Number>7</Number>
      <Price>$11.50</Price>
</Item>
</Order>
</Customer>
```

Fig. 1: An example XML document

3.1 Proposed approach

We describe a model mapping approach that does not require DTD or XML schema. The structure of this approach is as follows:-
Node (Node id, Node name)
Data (Doc id, Node id, Parent id, Node value, Node type, Node pos)
Table Node stores all node id's with their names in it. We assign a unique id to each node of the XML document. Node name attribute represents the name of the node. In data table , Doc id attribute specify the id of the particular XML document in this case say this is 10 i.e. doc id = 10, Node value attribute represents the value of the node i.e. it stores text values in it . Parent id is the id of the parent node of a node. Node type attribute is used to indicate whether the node is an element or an attribute or a text. Node Pos attribute is a position of the node among its siblings in the XML data graph. For Example in the case of the node 'date' its type is a text element and its position among its siblings is (1, 2) i.e. the position of the parent node which is the root and its position among its siblings. The Algorithm is as follows:-
// Transform XML document in to tree and tree is traversed according to the algorithm//
Algorithm 1.1.Begin
2. Read the XML document
3.Assign a unique id to the XML document say 10 here i.e. doc id = 10.
4.Create new object of the NODE table say node .
5. Create new object of DATA table say data.
6. While XML document is not null Repeat step 7 to 12.
7.Identify the root element of the XML document
(i) Node id=1
(ii) Node type=Element
(iii) Node pos =1
(iv) Parent id=null.
(v) Node name=root node name
8. Store Node id, Node name to NODE.

9. Store Node id , Parent id, Node type, Node pos to DATA.
10. Traverse the whole XML tree in the depth first manner from node to node.
11. If the node is an element
(i) Node id = Node id + 1.
(ii)Node type=element
(iii)Node pos= (parent Node pos, child Node pos)
(iv) Node name=node name
(v) Store Node id, Node name to NODE.
(vi) Store Node id , Node type , Node pos to DATA .
12. If the node is an attribute
(i) Node id = Node id + 1.
(ii). Node type = attribute.
(iii) Node pos= (parent Node pos, child Node pos).
(iv) Node value = text value.
(v) Node name=node name
(vi) Store Node id, Node name to NODE.
(vii) Store Node id, Node type, Node pos, Node value to DATA.
12. If the node element has a text value.
(i) Node id = Node id + 1.
(ii) Node type = text.
(iii) Node pos=(parent Node pos, child Node pos)
(iv) Node value = text value
(v) Node name=node name
(vi) Store Node id with Node Name to NODE.
(vii) Store Node id, Node type , Node pos , Node value to DATA.
13. End.

Table 1 : Node

Node ID	*Node Name*
1	Customer
2	Id
3	Name
4	First Name
5	Last Name
6	Date
7	Order
8	Item
9	Product
10	Number
11	Price
12	Item
13	Product
14	Number
15	Price

Table 2 : Data

Doc Id	*Node Id*	*Parent Id*	*Node Value*	*Node Type*	*Node Pos*
10	1	Null	Null	Element	1
10	2	1	1	Attribute	1,1
10	3	1	Null	Element	1,2
0	4	3	Sam	Text	1,2,1
10	5	3	Smith	Text	1,2,2
10	6	1	Oct 15,2001	Text	1,3
10	7	1	Null	Element	1,4
10	8	7	Null	Element	1,4,1
10	9	8	Tomato	Text	1,4,1,1
10	10	8	8	Text	1,4,1,2
10	11	8	$1.25	Text	1,4,1,3
10	12	7	Null	Element	1,4,2
10	13	12	Potato	Text	1,4,2,1
10	14	12	6	Text	1,4,2,2
10	15	12	$11.25	Text	1,4,2,3

3.2 Example Of Query Processing

The Example SQL query is as follows:-
For example the query is " find out the name of the node where Node pos =(1,2)".The corresponding SQL query is:-
Select N. Nodename from Node N, Data D where N.node id= D.node id AND D.Node pos like % 1,2%.
This query will join both Node and Data tables and retrieve name of the node from node table.

4. Performance Evaluation

We compare this approach with [4],[5] with respect to database size . After storing XML document in relational database the data base size of our approach is much less than X [REL] and X[PEV] .

Table 3 : Comparison Table

Method	*Database Size (In Bytes)*
XREL	1065
XPEV	979
Our Method	615

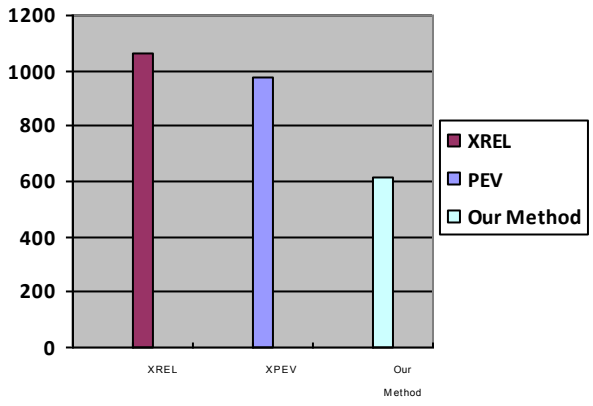

Fig. 3 : Comparison Graph

5. Conclusion

This approach stores nodes with its values, type, name and node position in a single table. So it represents a compact structure for storing the whole information of an XML document in relational databases and it requires less memory space for storage. In query processing it requires less number of join operations as compared to other approaches..

References

[1] J.Shanmugasundaram, K. Tufte, C. Zhang, G.He, D. Dewitt,J.Naughton, "Relational Databases for Querying XML Documents:Limitations and opportunities,"VLDB 1999 ,pp : 302-314.

[2] M. Atay, A Chebotko, D. L iu, S. Lu, F. Fotoubi , "Efficient schema based XML to relational data mapping", Information systems ,Elesevier 2005.

[3] S.Lu,Y. Sun, M.Atay,F. Fotouhi, " A New inlining algorithim for mapping XML DTDS to relational schema" In Proceedings of the First International Work-shop on XML Schema and Data Management , in conjuction with the 22nd ACM International Conference on Conceptual Modeling , Chicago, IL, October 2003.

[4]M. YoshiKawa.,T.Amagasa,T.Shtimura, "XREL: A path based approach to storage and retrieval of XML documents using relational databases".ACM Transactions on Internet Technology,1(1) ,pp:110-141, August 2001.

[5] J. Qin, S.Zhao, S. Yang, W. Dau, " XPEV : A Storage Approch for Well-Formed XML Documents", FKSD ,LNAI 3613,2005 pp.360-369.

[6] D.Florescu, D.Kossman, " A Performance Evauation of Alternative Mapping Schemes for storing XML Data in a Relational Database",Rapport de Recherche No. 3680 INRIA,Rocquencourt, France,1999.

[7] A.Salminen,F.Wm, "Requirements for XML Document Database Systems. First ACM Symposium on Document Engineering", Atlanta.2001 pp:85-94.

[8] H. Zafari, K. Hasami, M.Ebrahim Shiri, " Xlight,an Efficient relational schema to store and query XML data", In the proceedings of the IEEE International conference in Data Store and Data Engineering 2011, pp:254-257.

[9] M.Ibrahim Fakharaldien, J.Mohamed Zain, N. Sulaiman, "XRecursive: An efficient method to store and query XML documents",Australian Journal of basic and Applied Sciences,5(12) 2011 pp: 2910-2916.

[10] M.Sharkawi, N. Tazi, " LNV : Relational database Storage structure for XML documents", The 3rd ACS/IEEE International Conference On Copmputer Systems And Applications, 2005, PP:49-56.