SCHEMA BASED STORAGE OF XML DOCUMENTS IN RELATIONAL DATABASES

Dr. Pushpa Suri[1] and Divyesh Sharma[2]

[1]Department of Computer Science and Applications, Kurukshetra University, Kurukshetra
pushpa.suri@yahoo.co.in
[2]Department of Computer Science and Applications, Kurukshetra University, Kurukshetra
divyeshsharma89@gmail.com

ABSTRACT

XML (Extensible Mark up language) is emerging as a tool for representing and exchanging data over the internet. When we want to store and query XML data, we can use two approaches either by using native databases or XML enabled databases. In this paper we deal with XML enabled databases. We use relational databases to store XML documents. In this paper we focus on mapping of XML DTD into relations. Mapping needs three steps: 1) Simplify Complex DTD's 2) Make DTD graph by using simplified DTD's 3) Generate Relational schema. We present an inlining algorithm for generating relational schemas from available DTD's. This algorithm also handles recursion in an XML document.

KEYWORDS

XML, Relational databases, Schema Based Storage

1. INTRODUCTION

As XML (extensible markup language) documents needs efficient storage. The concept of Relational data bases provides more mature way to store and query these documents. Research on storage of XML documents using relational databases are classified by two kinds : Schema oblivious storage, Schema Based storage .Schema oblivious storage does not use any DTD (document type definition) or XML schema to map XML document in to relations[4][5][6][7][8][9][10]. Schema based storage requires DTD attached with the XML documents itself . In this paper we discuss mapping of XML DTD into relations with handling of recursion.

2. RELATED WORK

Shanmugasundaram [1] , provides three basic Inlining algorithims for mapping XML DTD into relations :Basic, shared and Hybrid. In basic inlining, for each element in XML document, a relation is created. This will lead to excessive relations and when we process queries to these relations, this will lead to excessive joins. In shared inlining, the elements having single occurance are inlined to its parent elements. This will reduce the number of relations as compared to Basic. In Hybrid, shared element i.e. the element having single occurance but shared by many

parents are also inlined to its parent elements. Mustafa Atay [2] extends these algorithims by providing the concept of endID. endID is the maximum id of the descendants of an element . For every element that is to be inlined, an endID is stored. This is helpful when we reconstruct the document. .Lu [3] provides a seperate edge table for storing elements having multiple occurances.

3. PROPOSED WORK

We discuss the concept of Inlining in detail. Inlining is basically to store the information of the child element with its parent element in the same relation. Elements having indegree = 1 i.e. the element having one parent can be inlined into same relation of its parent element. Indegree is the number of incoming edges to a node where a node represents an element itself in XML document. The first step to generate the relational schema from available DTD is to simplify the complexities of DTD i.e. solve +, *, |,? operators of DTD.
+ Operator means one or many. i.e. one element can occur one or many times in XML document.

* Operator means zero or many i.e. one element can occur zero or more times in XML document.

| Operator means choice.

? Optional operator means whether element occurs or not

Suppose e, e1, e2 denotes elements itself. Then Resolve DTD by these rules :-

1. $e^{**} \rightarrow e^*$
2. $(e+)+ \rightarrow e^*$
3. $e+ \rightarrow e^*$
4. $e_1 \mid e2 \rightarrow (e_1, e_2)$
5. $(e, e_1) \mid (e, e_2) \rightarrow (e, e_1, e_2)$
6. $e? \rightarrow e$
7. e - - e - - $\rightarrow e^*$
8. e^*-- -- e $\rightarrow e^*$
9. e - --- $e^* \rightarrow e^*$
10. e^* - - - $e^* \rightarrow e^*$

Next step is to make the graph of the simplified DTD. A DTD graph is the graphical representation of the XML document type definitions. In this graph, nodes are represented by elements and attributes of DTD and edges represents their parent-child relationships. Cycles represents recursion between elements of the documents. We represent two edges in this graph.

Single edge $\rightarrow$

Multiple edge $\xrightarrow{*}$

Elements having single occurrences are denoted by single edge and having multiple occurrences are denoted by multiple *edges. After making simplified DTD graph, last step is to generate relational schema based on simplified DTD graph. This is the algorithm for generating relational schema:-

Algorithm1: XML DTD Mapping

Input: DTD graph

Output: Relational Schema

1. For the nodes in the DTD graph having indegree = 0 ,Create Relation.

1.1. In relation, introduce an attribute id which serves as primary key in the relation.

1.2. Introduce an attribute Rootelem for that element to specify whether the element is a root or not.

2. For the nodes having indegree = 1 Inline them with their parent elements.

3. For the nodes having indegree >1 with single occurrence, inline them with their parent elements.

4. For the nodes having multiple occurrence i.e. *edge in DTD graph having indegree =1, Create relation.

4.1. In Relation, introduce the attribute id.

4.2. Introduce attribute parent id to specify the id of parent element.

5. Nodes having multiple occurrences with indegree > 1 ,Create Relation.

5.1. In Relation, introduce an attribute id for identification for that node.

5.2. Introduce an attribute Parent type to specify the parent element.

5.3. Introduce an attribute Parent id to specify the id of the parent.

5.4. Introduce an attribute Node type to specify the name of node.

6. If the nodes having cycles in the DTD graph then

6.1. If cycle contains only single edges create relation for one of the element in the cycle.

6.2. If the cycle contains one *edge and one single edge. Make Relations for both of the elements with an id reference of single edge element stored in *edge element.

6.3. If the cycle contains both *edges then create Relations for both of the elements.

4. A COMPLETE EXAMPLE

```
<! Doctype Document [
<! ELEMENT Document (company)>
<! ELEMENT company (dept*, employee+, projects+, cname)>
<! ELEMENT dept (dname?, employee+, company)>
<! ELEMENT employee (id, name, address, designation, project+)>
<! ELEMENT project (ptitle, pno | location)>
<! ELEMENT name (firstname?, lastname?)>
<! ELEMENT ptitle (# PCDATA) >.
<! ELEMENT dname (# PCDATA) >.
<! ELEMENT pno (# PCDATA>)
<! ELEMENT location (# PCDATA) >
<! ELEMENT address (# PCDATA) >
<! ELEMENT designation (# PCDATA)>
<! ELEMENT cname (# PCDATA)>
<! ELEMENT firstname (# PCDATA)>
<! ELEMENT lastname (# PCDATA)>
]
```

After simplification of DTD's by applying the rules given above, the DTD becomes:

```
<! ELEMENT company (dept*, employee*, project*, cname)>
<! ELEMENT dept (dname, employee*, company)>
<! ELEMENT employee (id, name, designation, project*)>
<! ELEMENT project(ptitle, pno, location)>
<! ELEMENT name (first name, last name)>
```

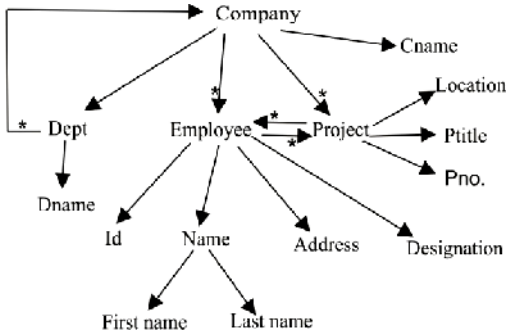

Figure 1: DTD graph

In this case, we are having cycles at root element. Two kinds of cycles exist here. One cycle exists between employee and project element. Both elements have multiple occurances and recursive to each other. Second cycle exists between company and dept. Company is the root element of the document. Dept has multiple occurances. Both cycles will be handled according to the algorithm given above.

The corresponding relational schema. becomes:-

Company (id, root elem : true, cname)

Dept (id, parent id, dname ,company. id)

Employee (id, parent type, parent id, node type, name.firstname, name.lastname, address, designation)
Project (id, parent id, parent type, node type, pno, ptitle, location)

5. CONCLUSIONS

In this paper we have given the inlining algorithm which will generate relational schema of the given XML DTD. Although recursion in XML documents is not very common but we have discussed it and also elaborate that how it will be handled while generating the relations. In future research will extend this concept in query processing of XML documents and translation of XML queries in to SQL queries.

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 algorithm 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 Approach for Well-Formed XML Documents, FKSD, LNAI 3613, 2005, pp.360-369.

[6] D.Florescu, D. Kossman, A Performance Evaluation 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 Computer Systems and Applications, 2005, pp:49-56.