An ID Based Algorithm for storing XML documents in Relational Datbases.

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
XML has emerged as a standard for representing data over the internet. Relational Databases provides a mature Way to Store and Query these documents. XML documents can be stored either schema based or schema less in relational databases. Query Translation of XML queries in to SQL Queries is found an interesting problem for researchers. In this paper we deal with translation of XPATH query in to SQL . We provide an Algorithm which will convert XPATH in to SQL without using advanced SQL operator. This Algorithm will handle the parent-child and ancestor-descendant relationship of XPATH expression and shows how translation of these expressions in to SQL expressions occurs.

Keywords: *Relational Database, XML, XPATH, SQL*

1. Introduction

XML is a markup language that defines a set of rules for encoding the documents in a format that defines a set of rules that is both human readable and machine readable [1]. It is widely used for data presentation over the web. Relational data bases provide a mature technology to store and query the XML documents.
There are mainly two categories.

1. Schema Based
2. Scheme Less.

In schema-based mappings, each XML schema is attached with the document itself. In scheme less mappings,, XML schema is missing. While, translating XML queries into SQL, there major techniques are identified.

- ID Based
- Path Based
- Internet Based

In ID Based technique, each element is associated with a unique-ID and tree structure XML document is preserved by maintaining a foreign key to the parent.
In Internet Based technique ,each element is associated with region representing the sub tree under it. In Path Based technique, each element is associated with a path-id representing the root to leaf path in addition to the internet based and id-based representation. In this paper, we present as ID-Based query translation algorithm which will translate XPATH expression in to SQL. This Algorithm will translate XML expression by using the parent-child relationships among the nodes without using the advanced SQL operators.

2. Related Work

XML to SQL query translation in schema-based and scheme less relational storage has been studied by many researchers. The query mapping Algorithm in [1] rewrites a, XPATH query in to the regular XPATH expression which is capable of handling the recursion both in DTD and in a X-Path query and then they provide an algorithm for translating regular XPATH. Expression in to relational query by using the LFP (Least Fix Point Operator).The query mapping algorithm of [2] first derives a query graph in to strongly connected components and generates an SQL query of each component. The query mapping algorithm of [3] first identify all matching paths in the XML document and grants a path prefix tree. Then in algorithm they use (parent-child). relationships to translate X-Path expression into SQL.
The query mapping Algorithm of [4] first unfolds the cycle in the document and then they generates the algorithm for translating x-path queries in to SQL.
The query mapping algorithm of [5] uses internet based approach to capture the ancestor descendant relationships in the XML document.

3. Query Mapping Algorithm

When we want to store XML documents in the relational database, we have to map these documents into relations, A Mapping scheme has to be defined. which will store XML elements and attributes in to relations. Let the mapping be \$ mapping. Given an XML schema S with element type set E and attribute type set A and database schema R, a \$ mapping is a mapping \$: (EVA) → R, such that given an attribute or element type e $(e) is the relation in which the instance of e will be stored.[3]

```
<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>
```

Figure 1 : An Example XML document

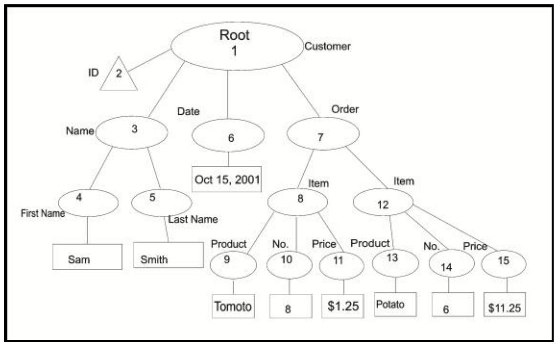

Fig. 2 : XML Data Graph of XML document in fig. 1.

Source id	Target id	Node Pos	Node values
1	Null	1	Null
2	1	1,1	1
3	1	1,2	Null
4	3	1,2,1	Sam
5	3	1,2,2	Smith
6	1	1,3	Oct 15,2001
7	1	1,4	Null
8	7	1,4,1	Null
9	8	1,4,1,1	Tomato
10	8	1,4,1,2	8
11	8	1,4,1,3	$1.25
12	7	1,4,2	Null
13	12	1,4,2,1	Potato
14	12	1,4,2,2	6
15	12	1,4,2,3	$11.25

Table 1: An Example of $ Mapping

Definition 1. Query Mapping
Suppose an XML Schema, Database, a Mapping, and a set of Relational Queries is given. Then translation of relational queries in to equivalent queries is known as QueryMapping.[3]

Definition 2. Path Expression
A Simple path expression is a simple expression of XPATH query in which "/" represents parent child relational ships among the nodes and "//" represents ancestor-descendant relationships of the nodes. [3].

An Example of XML document, its corresponding data graph and $ mapping is given in figure1 & 2 and table 1 respectively.

4. Proposed algorithm

```
Algorithm1.1.Input – Path expression P, $ Mapping .
2.Begin
3.Let P = a1n1, a2n2, …….. annn
4.from clause= "from"
5.where clause = "where"
6.for  i =1 to m do
7.from clause- + = " $ .$(ni) as Pi
8 .end for
9 .for i = 2 to m do
10. if ($a_i$ = "/") then
11.where clause + = $P_{i-1}$ ($n_{i-1}$ .ID) = $P_i$ ($n_i$.sourceID)
12.else if $a_i$ = " // " then
13.where clause + = $P_{i+1}$ ($N_{i+1}$ source ID) = Pi (ni..target ID) AND  $P_{i+1}$ ( $n_{i+1}$ .Node Pos) >$P_i$ .(ni .Node Pos)
15.end if
16.end for
17.sql = select ($P_k$ .(nk.ID)$^+$ from clause + where clause
18.End.
```

We propose a query mapping Algorithm in which the input is a path expression of XPATH query, and $-mapping .We construct from and where clause of the SQL query. If the path expression is child axis '/' we perform the following expression n_{i-1}.ID = n_i.sourceID. To test the Ancestor Descendant relationships we check the corresponding node positions and source and target id's of the nodes.

Example 1

If Path Expression A//B is given then according to the proposed algorithm the corresponding SQL query becomes:

Select P2.B.ID from A as P1, B as P2 Where P2.B.SourceID=P1.A.Target ID AND P2.B.NodePos>P1.A.NodePos.

5. Conclusion

We propose an ID based XML to SQL translation Algorithm. This algorithm handles the parent child and ancestor descendant relationships of XPATH expression.

In future research we will work on implementation of this algorithm by using Java platform.

References

[1] [1] Fan W., Yu, J.X.,Lu,H,Lu.j,Rastogi,R: "Query Translation from Xpath in to SQL in the presence of recursive DTD."In :Proc. Of the 31st VLDB Conference ,Trondheim,Norway(2005).

[2] Krishan murthy,R.,Kaushik,R.Naughton :"Recursive XML schemas ,recursive xml queries and relational storage : xml TO sql Query Translation,"In : Proc. Of the 20th International Confernece on Data Engeneering, Boston, USA(2004) pp: 42-53.

[3] AtayM, Fisher p,:" Optimizing XML to SQL translation for Analytical Databases using Intellegent Path deviation". ACM pp:15-17,(2010)

[4] M. Atay, A. Chebotko, D. Liu, S. Lu, and F. Fotouhi. "XML-to-SQL query mapping in the presence of multi-valued schema mappings and recursive XML schemas. In 18th International Conference on Database and Expert Systems Applications, "2007J

[5] M. Yoshikawa, T. Amagasa, T. Shimura, and S. Uemura. "XRel: A path-based approach to storage and retrieval of XML documents using relational databases". ACM Transactions on InternetTechnology (TOIT), vol1(1), no ,pp:110–141,2001

[6] R. Krishnamurthy, R. Kaushik, and J. F. Naughton. "XML-to-SQL query translation literature: The state of the art and open problems". In XML Database Symposium,2003.

[7] D. Florescu and D. Kossmann. Storing and querying XML data using an RDBMS. IEEE Data Engineering Bulletin, vol 22, no 3, pp: 27–34, 1999.