Software Engineering
ISSN: 2229-4007 & ISSN: 2229-4015, Volume 3, Issue 1, 2012, pp.-14-16.
Available online at http://www.bioinfo.in/contents.php?id=76

QUERY TRANSLATION FROM SQL-XML ENVIRONMENT

JADHAO P.P. AND BAMNOTE G.R.

Department of Computer Sc. & Engg., Prof. Ram Meghe Institute of Technology & Research, Badnera, MS, India.
*Corresponding Author: Email- priyankajadhao@gmail.com, grbamnote@rediffmail.com

Received: February 21, 2012; Accepted: March 15, 2012

Abstract- Developing techniques for managing and querying the growing body of XML data is becoming increasingly important. A popular approach to evaluating XML queries is to translate them to relational queries and then to use a relational database system to evaluate the result. The XML and relational data models are significantly different, and as a result, the corresponding query languages (XQuery and SQL respectively) also differ significantly. This mismatch raises some interesting questions: (i) From a functionality perspective, is it possible to handle all XML data sets using this approach or are there any fundamental limitations in SQL that create problems? (ii) From a performance perspective, are there any implications on the quality of the SQL queries produced due to this mismatch between the two data models? This paper focuses on query translation, which lies at the core of managing XML data using relational database systems.
Keywords- XML data, SQL.

Citation: Jadhao P.P. and Bamnote G.R. (2012) Query Translation from SQL-XML ENVIRONMENT. Software Engineering, ISSN: 2229-4007 & ISSN: 2229-4015, Volume 3, Issue 1, pp.-14-16.

Introduction

XML is emerging as the universal standard format for data exchange, and as a result, XML data management is becoming increasingly important. Relational database systems (RDBMSs) have dominated the commercial data management space for several decades. So, using RDBMSs to manage XML data is an attractive option. Unfortunately, the XML data model differs substantially from the relational data model, so using RDBMSs to support XML data poses a number of interesting and challenging problems.

The need for interoperation and data exchange through the Internet has made Extensible Markup Language (XML) a dominant standard language. Much work has already been done on translating relational data into XML documents and vice versa. However, there is not an integrated method to combine them together as a unifying technology for database interoperability on the Internet. Users may not be familiar with various query language syntax.

A methodology is developed to allow users to interoperate RDB and XMLDB through query translation. The database gateways receive the input queries before they are sent to the underlying databases. Query translation will be done through the gateways. The translated query will be sent to the appropriate database. Users can rely on one data model and his or her familiar query language to access data in both the RDB and the XMLDB.

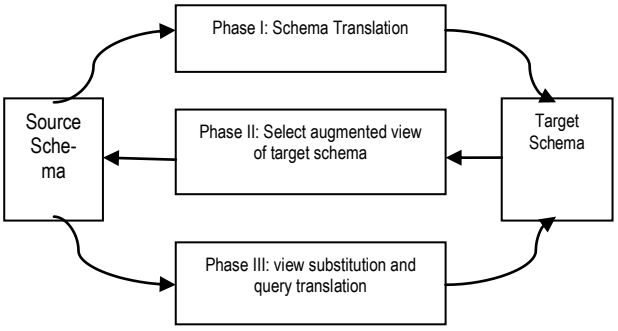

Fig. 1- The three steps for database reengineering query translation

The query translation process consists of two main steps of schema translation and query translation. The system allows users to

input SQL query which is translated into XPath for selecting the data on XMLDB. The architecture composes of two gateways, the XML Gateway and the Relational Gateway, as shown in Figure 2. There is a common interface between these gateways, which connect to a XMLDB server and a RDB server.

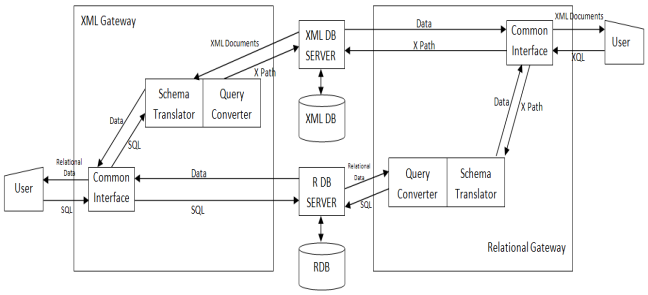

Fig. 2- XML and Relational Gateways

Methodology of Query Translation between SQL and XML

As shown in Figure 1, a need to abstract an augmented view of the target XML tree structures into a relational schema in phases I and II. A compatible tree structure in a partitioned relational schema and a mapped target XML schema. Then in phase III, translate an SQL to XQL according to the mapped XML schema. Similarly, translate an XQL to SQL according to the partitioned relational schema

A. Phase I: Schema Translation between Relational and XML Data

We add a sequence number into a relational table for data position in XML document. For any table that is used for query translation, an extra column - *seqno* is required. This column is used by the XML gateway described as follows:

For each table, the last column is *seqno*. This *seqno* column is used to ensure that the records returned from database are in the right order. The column is also used for translation of XQL location index functions (e.g. *position()*). The *seqno* column is incremented by one for each new record of the same key value and maintained by using the insert trigger. The records in the repository table *node_tablecolumn_mapping* is used for mapping the column of the table that is used for maintaining the *seqno* value.

node_tablecolumn_mapping Table

Table name	Node key column
CLIENT	ClientID
CLIENTACCOUNTEXECUTIVE	ClientID
ACCOUNTEXECUTIVE	AEID
BALANCE	ClientID

CLIENTACCOUNTEXECUTIVE Table

ClientID	AEID	*Seqno*
600001	AE0001	1
600001	AE0002	2
600002	AE0001	1
600003	AE0003	1

On inserting a new record into a table, the insert trigger first locates that the column is used for counting *seqno* from the *node_tablecolumn_mapping* table. Then, the trigger selects the maximum *seqno* value for the new record. The maximum *seqno* value plus one is assigned as the *seqno* value of the new record. There is no need to update the *seqno* value in case the record is deleted. In XQL, the location index function (e.g. *position()*) counts the order of the record relative to the parent node.

B. Phase II: Select Augmented View of Target XML Schema in Mapping RDB to XML Schema

To convert a relational database into an XML document tree, we integrate the translated XML document trees into an XML document tree, select an element as root and put its relevant information into a document. We load the relational database into the object instances of the XML documents. Each XML document focuses on a root class upon user requirements. The selection is driven by some business requirements. Relevance concerns elements that are related to a root selected by the users among the integrated XML document tree (DOM tree).

C. Phase III: Query Translation between XQL and SQL

i. Query Translation from SQL to XQL

Positioning-Based Query Translation between SQL and XQL with Location Counter 5.

In query transformation, a syntax-directed parser converts the SQL into multi-way trees. The transformation process is performed, based on the subtree matching and replacement technique. The process of SQL query transformation is given in Figure 5.

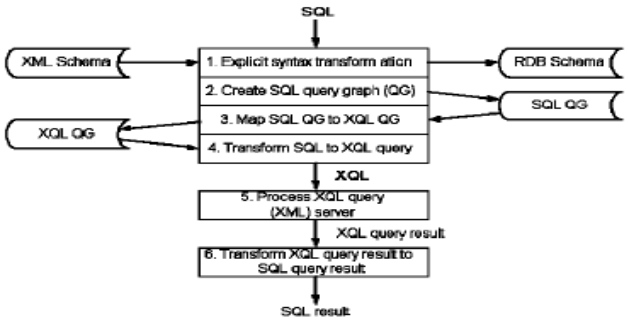

Fig. 3- Process for SQL to XQL Transformation

D. Translation of SQL Query to XPath Query

After the schema is done, SQL query can be translated to XPath query by the following steps:

Step 1 Decompose SQL Query Transaction- The basic syntax SQL SELECT statement is given as follows:

- SELECT {attribute-list-1} FROM {relation-list}
- WHERE {join-condition} AND / OR {search-condition-1}
- ORDER BY {attribute-list-2} GROUP BY {attribute-list-3}
- HAVING {search-condition-2}

Step 2 Create the SQL Query Graph- Based on the relation-list and the join-condition in the SQL query transaction, the SQL query graph is created. The join condition is based on the natural join or based on the search condition specified in the SQL query.

Step 3 Map the SQL Query Graph to XPath Query Graph- The SQL query graph is mapped to the XPath query graph. The table joins from the SQL query graph forms the XPath location path, which are the steps for navigating down the document tree from root node.

Step 4 Transform SQL to XPath Query- In this step, the SQL query is transformed into XPath syntax as follows:

/root/node1[@attribute1=condition]/...[@attribute2=condition]/@attribute3

The attribute-list in the SQL query is mapped to the leaf attribute node at the bottom of the document tree. If all the attributes of the element node are selected, "@*" is mapped to select all the attributes from the leaf element node. If more than one attributes are selected, the union operator is used to get the result. For example /root/node1/@attribute1 | /root/node1/@attribute2

Step 5 Transform XPath Query Data into SQL Query Data- The XML document returned from XMLDB is formatted into tables before the document returning to user. The format of the result is based on the data stored in the table *table_column_seq* (prepared in pre-processed schema translation).

Related work

Shanmugasundaram [2] presents three inlining algorithms that focus on the table level of the schema while Florescu and Kossmann [3] investigate various performance issues among five algorithms that focus on the attribute and value level of the schema. They all transform the given XML DTD to a relational schema. Collins [4] describes two algorithms for mapping relational and network models schemas into XML schema using the relational mapping algorithm. Such an approach allows the data in the relational and network database system. Tatarinov [5] studies how XML's ordered data model can be supported using an unordered RDB system. They propose three order encoding methods (Global Order, Local Order and Dewey Order) for representing XML in the relational model. Tseng and Hwung [6] developed a system called XML meta-generator (XMG) that is an extraction scheme to store and retrieve XML documents through object-relational databases. WebReader [7] is a middleware for automating the search and collecting information and manipulation in XSL, WebReader also provides the users with a centralized, structured, and categorized means to specify for querying web information.

An SQL to XQuery translation algorithm to access XML data in a federated environment is presented in [8]. Based on a global relational schema, queries are written in SQL and translated to XQuery before execution on an XML data source. This postulates that a mapping from a global relational schema to a local XML schema exists. They describes metadata necessary to translate SQL to SQL/XML, and presents a method to automatically build the mapping tables. It uses a translation from SQL to XQuery to provide applications access to data sources in an integrated Data Service Platform. XQuery functions are wrapped around data services and produce *flat* XML structures that resemble relational tables. This flat XML can be queried with SQL through an SQL to XQuery conversion. Similar to [9], the query generation depends on the known flat structure of the XML data and it is not applicable to arbitrarily nested XML.

Conclusion

The need for interoperation and data exchange through the Internet has made Extensible Markup Language (XML) a dominant standard language. Much work has already been done on translating relational data into XML documents and vice versa. However, there is not an integrated method to combine them together as a unifying technology for database interoperability on the Internet. Users may not be familiar with various query language syntax. This paper describes a methodology that translates SQL query to XML query. The mechanism has a distinct feature that XDB and RDB are able to co-exist, and XQL and SQL can be employed to access both systems. For the ease of the user there exists a mechanism that will translate SQL query into XML and vice-versa. The paper only introduces a methodology behind the translation.

References

[1] Rajasekar Krishnamurthy, *XML-to-SQL Query Translation*.
[2] Jayavel Shanmugasundaram et al. (1999) *25th VLDB Conference*, 302-314.
[3] Florescu D. and Kossman D. (1999) *IEEE Data Engineering Bulletin*, 22(3), 27-34.
[4] Samuel Robert Collins et al. (2002) *Information and Software Technology*, 44 (4), 251-257.
[5] Igor Tatarinov et al (2002) *ACM SIGMOD int'l conference on Management of data*.
[6] Frank S., Tseng C. and Wen-Jong Hwung (2002) *Journal of Systems and Software*, 64 (3), 207-218.
[7] Chan J. and Li Q. (2000) *1st International Conference on Web Information Systems Engineering*, 2, 47-56.
[8] Jahnkuhn H. et al. (2006) *QLQP*, *EDBT Workshop*, *LNCS 4254*.
[9] Jigyasu S. et al. (2006) *ICDE.*