Volume 4, No. 4, March-April 2013

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

# Join Queries Translation from SQL to XML

Bhargavi K.*
Department of Information Science and Engineering,
Siddaganga Institute of Technology
Tumkur-572 103, Karnataka, India
bhargavi.tumkur@gmail.com

Chaithra H. S.
Wipro InfoTech, Wipro Limited,
Bangalore-560 035, Karnataka, India
chaithrahs.be@gmail.com

***Abstract-*** SQL (Structured Query Language) was one of the standardized query languages for requesting information from a database. Over the past few years organizations are using SQL for storing and managing the organization specific information. At the same time there is an emerging trend towards XML (Extensible Markup Language), and it has become a standard for information exchange over the internet. Users/Developers are supposed to use two different kinds of languages for data reception and manipulation. This problem is overcome by converting the user's SQL queries into XML. In this paper, we have developed a converter tool to convert SQL join (Left, Right, and Full) queries into XML. As a result users can access XML database through SQL queries only.

***Index Terms—*** SQL; XML; XPath; RDBMS; Query Converter

## I. INTRODUCTION

SQL is a special purpose programming language designed for managing data in Relational Database Management Systems (RDBMS). SQL was one of the standardized query languages for requesting information from a database, but nowadays there is a growing trend towards XML. XML [1] [2] is a markup language that defines a set of rules for encoding documents in a format that is both human readable and machine readable [3]. The design goals of XML emphasize simplicity, generality, and usability over the Internet [4]. XPath [5] is used to traverse through elements and attributes in an XML document. Although the design of XML focuses on documents, it is widely used for the representation of arbitrary data structures like web services and it has become a standard for data (information) exchange over the web [6] [7] [8].

Scalability, reliability, performance regularity, etc. SQL is well understandable query language and it has been the standard language for manipulating data in RBDMS. Over the past few decades organizations are using RDBMS and SQL for storing and managing the data. It will become an overhead for the organizations to give up the existing RDBMS entirely, translate all of the documents into XML format.

Users are supposed to use two different kinds of languages for data reception and manipulation i.e., SQL for RDBMS, and XPath for XML. Users may not know various query languages syntax, and semantics; so in order to overcome this problem we have to convert SQL queries into XPath expressions [12], so that users can access and manipulate the data on RDBMS, and XML databases using SQL [13] [14] [15]. This in turn reduces the burden of learning different kinds of languages for users. In this paper we present a novel framework, where SQL join (Left, Right, and Full) queries are converted first to XPath expressions then to XML data. It also provides the detailed transformation process of conversion (SQL to XML) along with their relevant algorithms. We have also developed a converter tool to demonstrate the framework.

The rest of the paper is organized as follows: section 2 provides some related works on query conversion, section 3 provides translation framework, section 4 gives the transformation process, section 5 discusses algorithms developed for query conversion, section 6 gives the benefits of XML, section 7 provides snapshots of the developed converter tool, and finally section 8 draws the conclusion.

## II. RELATED WORKS

A relational database is a collection of data items organized as a set of formally described tables from which data can be accessed easily. The software used in relational database is called as RDBMS. RDBMS [9] is the basis for SQL, and for all modern database systems such as MSSQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access [10]. The data in RDBMS is stored in database objects called tables. A table is a collection of related data entries and it consists of columns and rows. RDBMS [11] is one of the successful database management systems, because of its features like

In [16], a framework is proposed to transform the SQL queries into XUpdate expressions. SQL has been used, as it has long been the standard query language. Users are allowed to access XML, and relational databases through the SQL query language by using the proposed framework. Thus only INSERT and UPDATE SQL queries are converted into XQuery's.

In [17], the BEA AquaLogic Data Services platform (DSP) provides a service-oriented, XML-based view of heterogeneous enterprise data sources. The AquaLogic DSP includes a JDBC driver that connects the SQL with the XML world through a SQL to XQuery translator. The method is confined to AquaLogic DSP and essentially focus on XQuery 1.0. XQuery 1.0 supports only reads and not updates. Thus only SQL SELECT statements are supported.

A framework is proposed in [18] to transform SQL

statements into XPath expressions. Users are allowed to access XML, and relational databases through the SQL query language by using the proposed framework. Thus only SELECT, DELETE, and RENAME SQL queries are converted into XPath expressions.

The [19] explores the feasibility of accessing XML data through SQL interfaces, called Relational over XML (ROX). It highlights the forces that are driving the industry to evolve towards ROX. It discusses the impact of denormalization of data in XML documents both from a semantic and performance perspective. The implications of ROX for manageability and query optimization is bulleted.

The [20], three semantics-based schema conversion methods are presented. First method converts an XML schema to a relational schema, second method derives a nested structured XML schema from a flat relational schema by repeatedly applying nest operator so that the resulting XML schema becomes hierarchical, and third method takes a relational schema as input and generates an equivalent XML schema as output.

## III. SQL TO XML TRANSLATION FRAMEWORK

In this section we present a novel framework for SQL to XML translation. The scheme of SQL to XML conversion is shown in Figure 1.

The framework operates through the following steps:

User will upload XML File to Server, which will be treated as XML Database (XML Repository). A sample XML file is shown in Fig 2

User gives SQL Join query as input to the Query Converter. In-side the converter there are 2 components for processing the input i.e., Query Analyzer, and Query Mapper.

In the Query Analyzer the SQL join query will be splitted and taken into different variables in order to form XPath, i.e., the parent and child node assumption made by the Query converter will be sent to Query Mapper to determine whether the assumption made is correct or wrong.

Query Mapper will send request to XML Database and gets the XML Data, Query Mapper analyses the data and determines the parent and child nodes. Then compares its result with the assumption sent by Query converter, and sends the result as true if the assumption made by Query Converter is correct, or else sends the result as false.

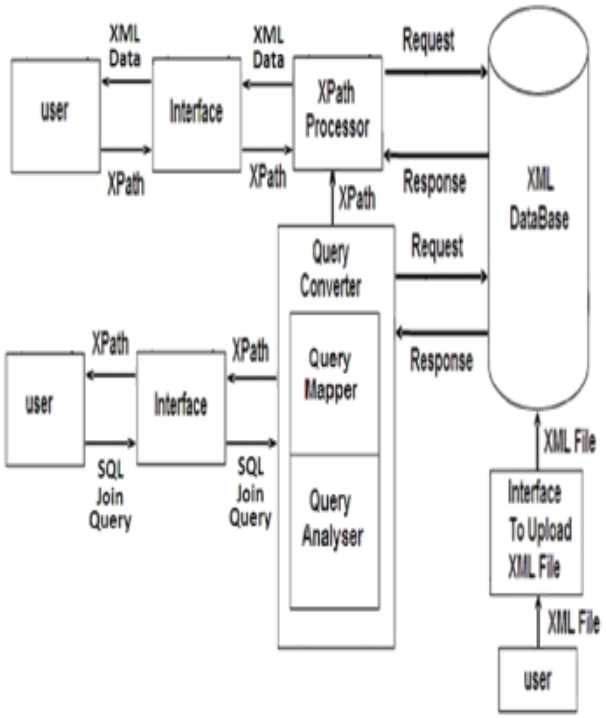


Figure 1. Framework of query translation

Depending on the result obtained by Query Analyzer, and Query Mapper; Query converter will do further processing and form the XPath Expression.

XPath Processor processes the received XPath expression to get the selected set of nodes. XPath Processor will then send the request to XML database, and in turn obtain the XML data corresponding to the selected set of nodes.


Figure. 2. Sample XML file

## IV. QUERY TRANSFORMATION

In this section we describe how to transform queries expressed in SQL to XML. The transformation process is shown in Figure 2.

Transformation process happens through following steps:

Collecting input from user: Query converter prompts user to provide the input (SQL Join Query).

Validating Syntax of SQL Query: Syntax Verification is done by Splitting the SQL Query. Few validations are done such as, keywords presence, keywords spelling, keywords positions, and braces positions and counts.

Extracting Query to variables and Forming Trees: SQL Join Query is splitted, outer query parts are extracted to appropriate variables, and inner select query is extracted to array variables as trees. Table names are identified based on appropriate query position in the tree structure. Parent and child table names are assumed from the identified table names.

Mapping of Tables to XML Tags: Assumed parent and child table names are compared with the parent and child XML Tags that are determined by analyzing the Xml data. If there exists a match mapping is done, else the assumption is changed.

Formation of XPath: From the appropriate variables, trees, mapped Variables; XPath is formed as follows: //ParentTableName[ParentTableWhereConditionClause] / ChildTableName[ChildTableWhereConditionClause] / OuterSelectColumnsClause

Presenting the output to user: XPath Processor processes the formulated XPath and generates the XML data.

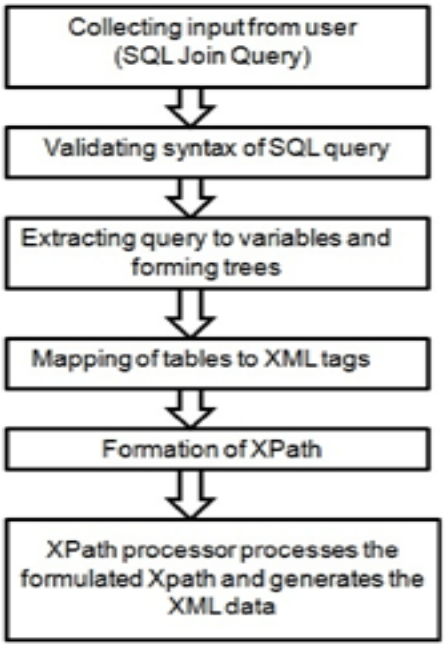


Figure. 3. Transformation process

## V. ALGORITHMS

In this section we discuss the algorithms that are developed for query conversion. Mainly we concentrate on Query Converter, Query Analyzer, and Query Mapper, XPath Processor modules of the translation framework.

Query Converter

Query Converter is the central component in the framework of query translation. It takes SQL join query as input and extracts its portions to appropriate variables; it also identifies the TableNames, SelectClauseColumns, and WhereClause conditions by calling the Subroutine Query Analyzer. Make assumptions of parent and child table names and related contents. Check whether the assumption is correct or not by calling Query Mapper subroutine. If Query Mapper return true then assumption is correct otherwise change the assumption, then proceed with further processing. OuterSelectColumnsClause Variable is splitted and each select column is taken into Array. For each column in the outer select columns clause looping is done. In each iteration; if column is present in parent table then Xpath is formed as follows://ParentTableName[ParentTableWhereConditionClause]/ ColumnName, else if column is present in child table, then XPath is formed as follows: XPath =//Parent-TableName[ParentWhereConditionsClause]/[ ChildTable-NameChildWhereConditionsClause]/ColumnName. The working of Query Converter is shown in algorithm 1.

Query Analyzer

Query Analyzer is one of the sub-component in the Query Converter component of the proposed query translation framework. It Processes the inner select query to get inner select query portions which is specified in the input parameter. Inner SQL query is splitted and extracted to appropriate positions in the tree. Requested portion of inner select query is then returned by extracting it from appropriate positions in the tree. The logic of Query Analyzer is given in algorithm 2.

Query Mapper

Query Mapper is one of the sub-component in the Query Converter component of the proposed query translation framework. Assumed parent and child table names will be passed as input to this sub-component. XPathDocument object is created, which is used to create XPath Navigator. Using navigator control is moved to root then to its child, there by identifying the parent and child tag names. Compare the identified parent and child tag names with their respective assumed table names; if there exists a match then return true else return false. The working of Query Mapper is given in algorithm 3.

XPath Processor

XPath Processor is one of the components in the proposed query translation framework. Generated XPath will be passed

**Algorithm 1** Working of Query Converter

1: Begin
2: Accept the SQL join query as input from user.
3: Extract the portions of the SQLJoinQuery into respective variables (OuterSelectColumnsClause, LeftSideInnerSe-lectQuery, TypeofJoin, RightSideInnerSelectQuery)
4: Depending on the type of join, Parent and Child Table-Names are assumed using Query Mapper.
5: **if** TypeofJoin is left or full **then**
6: CALL step 18 to make assumption
7: check assumption correctness using Query Mapper

8: **if** assumption = false **then**
9: CALL step 19
10: **end if**
11: **else if** TypeofJoin is right **then**
12: call step 19 to make assumption
13: check assumption correctness using Query Mapper
14: **if** assumption = false **then**
15: call step 18
16: **end if**
17: **end if**
18: Get inner select queries details and assign left part of inner query to parent and right part of inner query to child.
19: Get inner select queries details and assign right part of inner query to parent and left part of inner query to child
20: OuterSelectColumnsClause Variable is Splitted by "," (comma) and each select column is taken into string array.
21: **for** Each select column **do**
22: Extract ColumnName
23: **if** ColumnName belongs to ParentTable **then**
24: check ColumnName in ParentSelectColumnsClause then Form the XPath
25: **else if** ColumnName belongs to ChildTable **then**
26: check ColumnName in ChildSelectColumnsClause then Form the XPath
27: **end if**
28: **end for**
29: Pass the generated XPath to the XPath Processor, which processes it and presents the corresponding XML data as output to user.
30: End

As input to this component. XPathDocument object is created, which is used to create XPath Navigator. Using navigator, iterator will be created for selected set of nodes. Using iterator looping is done on the selected set of nodes, to present the XML data. The logic of XPath Processor is given in algorithm 4.

## VI. BENEFITS OF XML

It is platform and vendor independent.

Information coded in XML is easy to read and under-stand.

It is an extremely portable language to the extent that it can be used on large networks with multiple platforms like the internet, and it can also be used on handhelds, palmtops, and PDAs.

**Algorithm 2** Working of Query Analyzer

1: Begin
2: Inputs to this subroutine i)SelectQuery ii)Type-String specifying which portion of the SelectQuery should be returned. Possible values for Type are "SelectColumnsClause" or "TableName" or "WhereConditionsClause".
3: Extract the portion of the inner select query into respective variables (i.e., SelectColumnsClause, Table Name, Where-ConditionsClause)
Tree-Array[0] = "select" keyword
Tree-Array[1] = SelectColumnsClause
Tree-Array[2] = "from" keyword
Tree-Array[3] = TableName Tree-
Array[4] = "where" keyword Tree-
Array[5] = WhereConditionsClause
4: Depending on the Type, requested portion of SelectQuery is returned as output from the Subroutine.
5: **if** (Type == "SelectColumnsClause") **then**
Return Tree-Array[1];
6: **else if** (Type == "TableName") **then** Return Tree-Array[3];
7: **else if** (Type == "WhereConditionsClause") **then** Return Tree-Array[5];
8: **end if**
9: End

In XML there are no fixed set of tags and new tags can be created as they need.

XML documents can be stored without schemas because they contain meta data.

The look and feel of an XML document can be controlled by XSL (EXtensible Stylesheet Language), allowing the look of a document to be changed without modifying the content of the document.

It supports multilingual documents and unicode.

The tree structure of XML documents allows documents to be compared and aggregated element by element.

It can embed multiple data types (audio, video, java applets, etc.).

Mapping existing data structure i.e., relational databases to XML is simple.

It provides a one server view for distributed data.

It is being rapidly adopted by industries like IBM, Microsoft, NetScape, SAP, Software AG, DataChannel, etc.

It is currently the most sophisticated and popular format for distributed data over World Wide Web.

## VII. CONVERTER TOOL

We have developed a converter tool for converting the SQL queries into XML. First the user will browse the XML file and upload it, which is shown in Fig 4. When user clicks upload button, contents of the XML file will be shown in table form. This is shown in Fig 5. User enters the SQL query then clicks the convert button, which is shown in Fig 6. On conversion XPath will be displayed, as shown in Fig 7. An additional option for verifying the XPath is also provided; when the user

**Algorithm 3** Working of Query Mapper

1: Begin
2: Include NameSpace "System.Xml.XPath"
3: Create an XPathDocument object
XPathDocument xmlPathDoc = new XPathDocument(XMLFileName)
i.e., XMLFileName – Uploaded XML FileName Along with its Directory Location Path
4: Create a navigator for the xpath document
i.e., XPathNavigator p-xPathNav = xmlPath-

Doc.CreateNavigator()
5: Initially make assumption as false
6: Move to the Root Node in XML File i.e., p-xPathNav.MoveToRoot()
7: Move to the first element i.e., p-xPathNav.MoveToFirstChild()
8: Move to the next first element i.e., p-xPathNav.MoveToFirstChild()
9: DO
10: **if** (p-xPathNav.Name == ParentTableName) **then**
11: **if** (p-xPathNav.MoveToFirstChild()) **then**
12: WHILE (p-xPathNav.MoveToNext())
13: **if** (p-xPathNav.Name == ChildTableName) **then**
14: Make assumption as true
15: **end if**
16: ENDWHILE
17: p-xPathNav.MoveToParent()
18: **end if**
19: **end if**
20: END DO WHILE (p-xPathNav.MoveToNext())
21: Return ”true” if assumption is valid otherwise return ”false”
22: End

**Algorithm 4** Working of XPath Processor

1: Begin
2: Converter generated XPath or user specified XPath will be given as input.
3: Include NameSpace System.Xml.XPath
4: Create an XPathDocument object XPathDocument xmlPathDoc = new XPathDocument(XMLFileName)
5: Create a navigator for the xpath document XPathNavigator p-xPathNav = xmlPathDoc.CreateNavigator()
6: Create a iterator for selected set of nodes XPathNodeIterator xPathIt = p-xPathNav.Select(XPath)
7: WHILE
8: XMLData = XMLData + xPathIt.Current.Name + ” = ” + xPathIt.Current.Value
9: END WHILE
10: Presenting the XML Data as output to user
11: End
12: Clicks on verify XPath, XML data corresponding to generated XPath will be displayed, as shown in Fig 8.


Figure. 4. GUI of the converter

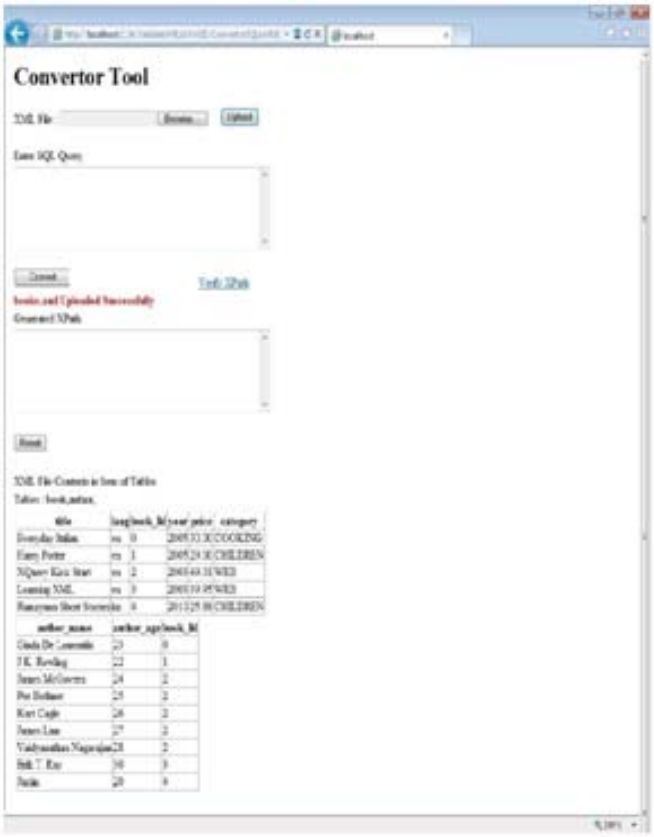


Figure. 5. XML file in table form


Figure. 6. Interface to enter SQL query


Figure. 7. Generated XPath


Figure. 8. Generated XML data

## VIII. CONCLUSION

We have developed a converter tool for converting SQL join queries into XML. The developed tool act as an automatic converter of SQL queries into XML data. The converter reduces the burden of learning syntax and semantics of different kinds of languages on developers/users. Thus the developed converter tool can be used in the organizations, where operating with different query languages is an essential requirement. As future work, we are planning to enhance the features of the developed converter tool by considering other complex SQL queries like nested left join, right join, full join, etc.

## IX. REFERENCES

[1] T. Bray, J. Paoli, C. Michael, F. Yergeau, E. Maler, ”XML 1.0 Specifica-tion”, W3C, 5th Edition, November 2008

[2] Pablo Berdaguer, Alcino Cunha, Hugo Pacheco, and Joost Visser,” Coupled Schema Transformation and Data Conversion for XML and SQL”, Work funded by Fundacao para a Ciencia e a Tecnologia - POSI/ICHS/44304/2002.

[3] Bonifati A., Ceri S., ”Comparitive Analysis of Five XML Query Lan-guages”, ACM SIGMOD Record 29(1), 2000

[4] http://www.mulberrytech.com

[5] J. Clark, and S. DeRose, ”XML Path Language (XPath) version 1.0”’, W3C Recommendation, November 1999

[6] Matthias Nicola, Tim Kiefer, ”Generating SQL/XML Query and Update Statements”, CIKM09, ACM-2009

[7] Tadeusz Pankowski, ”XML-SQL: An XML Query Language Based on SQL and Path Tables”, EDBT 2002 Workshops, 2002

[8] Zhen Hua Liu, Sivasankaran Chandrasekar, Thomas Baby, Hui J. Chang, ”Towards a Physical XML independent XQuery/SQL/XML Engine”, PVLDB ’08, ACM-2008

[9] Xixuan Feng, Arun Kumar, Benjamin Recht, and christopher Re, ”To-wards a Unified Architecture for in RDBMS Analytics”, SIGMOID12, ACM-2012

[10] Youssef Bassil, ” A comparative study on the performance of the Top DBMS systems, Journal of Computer Science and Research (ICSCR), 2012

[11] Florescu D., D. Kossman, ”Storing and Querying XML Data Using an RDBMS”, IEEE Data Engineering Bulletin, 1999

[12] Surajit Chaudhuri, Zhiyuan Chen, Kyuseok Shim, and Yuqing, ”Storing XML (with XSD) in SQL databases: Interplay of Logical and Physical Designs, IEEE Transactions on Knowledge and Data Engineering, 2005

[13] Jayavel Shanmugasundaram, Rajasekar Krishnamurthy, Igor Tatarinov, ” A General Technique for Querying XML Documents using a Relational Database System”, SIGMOD Record, Vol. 30, No. 3, September 2001

[14] Dongwon Lee, Murali Mani, and Wesley W. Chu, ” Nesting-based Relational-to-XML Schema Translation”, 2001

[15] Manolescu I., D. Florescu, D.Kossman,” Pushing XML queries inside Relational Databases, INRIA, Rapport de recherche, 1-41 2001

[16] Vidhya P.M, Philip Samuel, ”Insert Queries in XML database”, pages 9-13, IEEE-2010

[17] S. Jigyasu, et al., ”SQL to XQuery translation in the aqualogic data services platform”, proc. 22nd International Conference on Data Engi-neering, page 97, April 2006

[18] Vidhya P.M, Philip Samuel, ”Query Translation from SQL to XPath”, pages 1749-1752, IEEE-2009

[19] Alan Halverson, Vanja Josifovski, Guy Lohman, Hamid Pirahesh, and Mathias Morschel, ”ROX: Relational Over XML”, pages 264-275, Pro-ceedings of the 30th VLDB Conference, Toronto, Canada, 2004

[20] Dongwon Lee, Murali Mani, and Wesley W. Chu, ”Schema Conversion Methods between XML and Relational Models”, 2002