Database Systems CS6530 (Fall ’08) Project Proposal

Pramod R Sanaga
School of Computing,
pramod@cs.utah.edu,
u0504960

December 23, 2008

1 Overview

The overall goal of this research project is to add an XQuery to SQL translation module to the existing codebase of Shrex [**?**].

Shrex is a tool to map annotated XML schema into equivalent relational schema, with the help of user input during the mapping process. It then allows XML documents to be shredded into an RDBMS and stored in terms of tables with columns as the attributes of the corresponding XML structure. Shrex implements a query translation module that takes XPath queries(only a subset of the language spec is currently implemented), turns them into SQL queries on the stored tables and evaluates them.

XQuery and XPath are in many respects similar in their underlying data model. XPath is a subset of XQuery, which is more expressive and allows for complex queries other than path based predicates. As far as this project is concerned, the FLWOR construct of XQuery is of the most interest. We would also look at implementing as many of the built-in functions of XQuery as possible.

2 Implementation Details and the Design process

Shrex currently supports XPath queries without using a custom parser (it uses Apache JXPath) to handle XPath queries. There is a single method in ’XPath-Query.java’ to translate two kinds of XPath expressions into SQL - simple path expressions and expressions involving wild cards. Writing a custom parser for XQuery is not going to be enough to support a meaningful subset of XQuery, in the time available (including the FLWOR construct). The correct way to deal with the XQuery - SQL translation is to make use of an XQuery reference parser. The XQuery parser returns an intermediate representation (IR) of the XQuery expression, which can then be used along with Shrex annotations to generate a series of join expressions on the involved tables.

I looked at the grammar for XQuery on the W3C website - they do not have a full reference implementation available. They do have a parser that creates an abstract syntax tree (AST) representation of XQuery expressions, but it does not evaluate the queries - so this is not useful for our purpose. I then looked for a complete XQuery parser - one that can evaluate the expressions as well. There are two implementations that satisfy this criteria - Saxon and eXist Native XML Database. For the purpose of this project, I chose to work with the Saxon library. The following is a plausible path for writing an XQuery - SQL translation module.

1. Read the XQuery expression.
2. Use the Saxon parser to parse the query and obtain the AST.
3. Modify the nodes of the AST such that wherever there is an absolute path expression, we evaluate that path using the query translation already present in Shrex (ie. XPath -¿ SQL) and replace the XPathContext for that node in the AST with an in-memory representation of the XML returned by Shrex.
4. Evaluate this modified AST using Saxon and display the results.

Final Design:

Modifying the AST returned by Saxon proved to be too intricate for our purpose. So, instead I decided to still use the existing functionality offered by Shrex for translating XPath expressions. But, in place of creating in-memory XML structures to be used as input XPathContext for the AST nodes, I write the XML files to a temporary directory. The original XQuery expression is modified using XQuery's doc() function to refer to the corresponding file for every absolute XPath exression.

When a user enters an XQuery expression in the GUI, we look for absolute path expressions in the query. Each such expressions is evaluated using Shrex's XML-¿relational mapping resulting in an intermediate XML result. These results are placed in temporary files and we construct a new XQuery expression using these files. Evaulating the new query gives us the data being queried on the XML file by using Shrex's relational mapping and we display these results to the user.

Initially Proposed Approximate timeline:

October 19th: Would finish reading the related work publications and the shrex codebase. Would add the ability to translate and evaluate simple path based predicates written in the XQuery language.

November 19th: Adding the ability to translate FLWOR constructs, including nesting if feasible.

December 19th: Looking at adding other constructs such as If-else, in-built functions and non-path based predicates.

Tools:

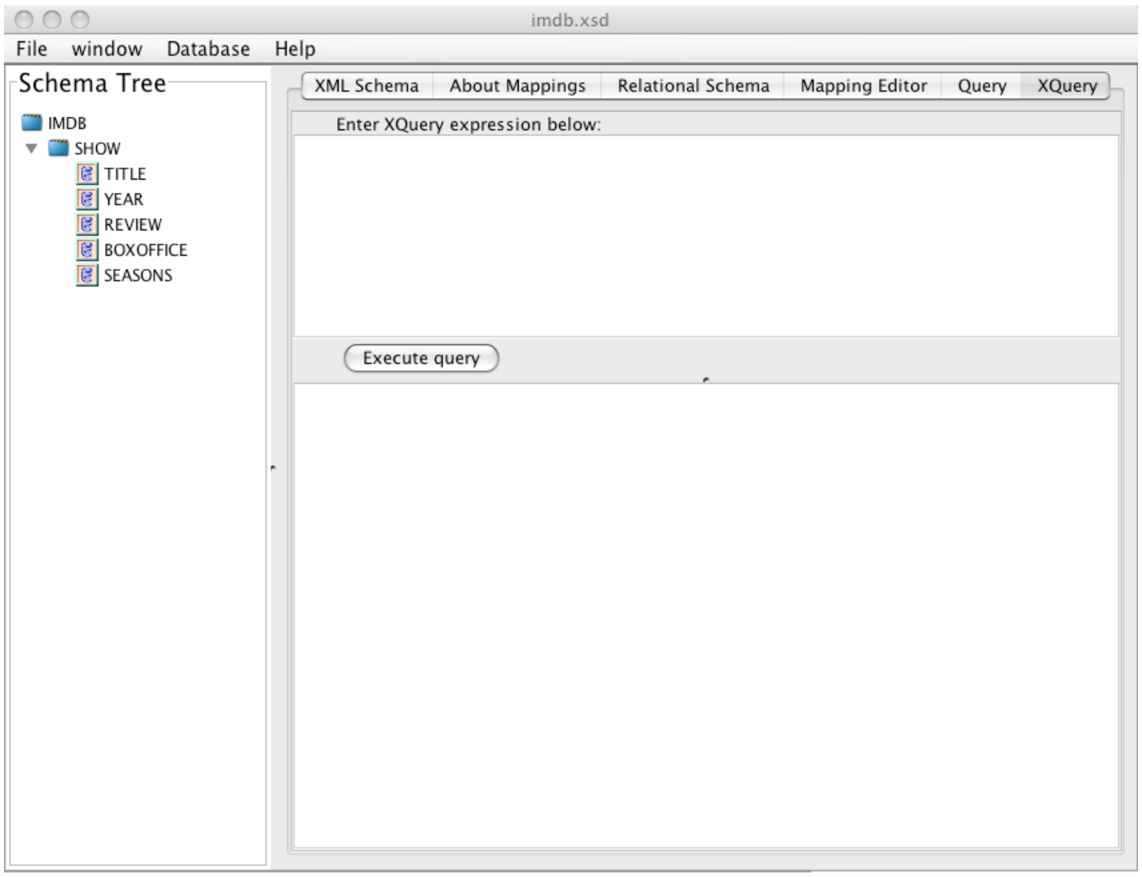
Shrex is written using Java. So, we used Java(including the regex package) for the XQuery to SQL module. We also use the open source Saxon platform for evaluating the modified XQuery expressions.

3 Results

The screenshot below shows the addition of the XQuery-¿SQL translation tool to the Shrex GUI. The user loads the schema, defines the relation-mapping and stores it in a database connection - just as before, there have been no changes in functionality. After the data file has been loaded, the new 'XQuery' tab shown below has a text box for entering the queries. The following types of queries are accepted.

1) All the path expressions in the query must refer to the data present in the relational database.
2) At the moment, the only path expressions supported are the ones that can be translated using Shrex's XPath -¿ SQL translation tool. As and when more functionality is added to XPath translation part of Shrex, it automatically becomes available to the XQuery -¿ SQL module.

Other than the above limitations, the queries can be written using the full extent of the XQuery language - there are no other restrictions.



4 Conclusion

We implemented an XQuery to SQL translation module specific to the Shrex XML-relational mapping tool. It accepts the entire XQuery language, with the path expressions being limited to a set accepted by Shrex's XPath translation module. It can be easily extended to the complete XQuery 1.0 specification by modifying the XPath expression evaluation in Shrex.

References

[1] D. DeHaan, D. Toman, M. P. Consens, and M. T. Özsu. A comprehensive xquery to sql translation using dynamic interval encoding. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 623–634, New York, NY, USA, 2003. ACM.

[2] T. Grust, M. Mayr, J. Rittinger, S. Sakr, and J. Teubner. A sql: 1999 code generator for the pathfinder xquery compiler. In *SIGMOD '07: Proceedings*

of the 2007 ACM SIGMOD international conference on Management of data, pages 1162–1164, New York, NY, USA, 2007. ACM.

[3] R. Krishnamurthy, R. Kaushik, and J. F. Naughton. XML-to-SQL query translation literature: The state of the art and open problems. In *Database and XML Technologies, First International XML Database Symposium (XSym)*, pages 1–18, 2003.