FoXQ – XQuery by Forms

Robin Abraham
Department of Computer Science
Oregon State University
Corvallis, OR 97331, *USA*
abraharo@cs.orst.edu

Abstract

We introduce FoXQ, a visual language that would enable end users to query XML. Our aim is to develop a language that would bring a lot of the functionality of XQuery within the reach of the end users without getting them embroiled in the intricacies of XQuery syntax. The query interface is form-based and the query model is based on a document metaphor *in which the users formulate queries by filling out forms.*

1 Introduction

In a short span of time, XML has gained wide acceptance as the document and data standard on the web. As more and more XML data gets generated everyday, a lot of research focus has been on query languages for XML. The World-Wide Web Consortium (W3C) has chosen XQuery as the standard language for querying XML. From an end-user point of view, XQuery sacrifices usability for expressiveness. There is a lot of effort being invested in the development of query engines for XQuery. Most of them have simple, textual interfaces, and some of them provide help with the formulation of the queries by recognizing XQuery keywords and syntax. Attempts have also been made to help the user formulate textual queries by showing the DTD tree so that the user can click on the tree nodes to specify the data to be displayed. For end users, it is probably worse to have to edit abstract syntax than concrete syntax. The main reason for this imbalance in the development of interfaces is that a lot of the ongoing work is motivated by commercial applications of XML. Many organizations are using XML for data integration. In this context, the people using XQuery are database administrators or programmers who do not have much trouble picking up a new language. This cannot be said of the average end user who might not even have the benefit of a formal education or a rigorous introduction to computing. In contrast, our system is specifically targeted at end users. Our goal is not to invent a visualization for every part of XQuery. Our goal is to make a significant part of XQuery usable by end users. This requires careful consideration of what end users are capable of doing and what they cannot do. We believe that to be successful, research in end-user programming has to be taken into account. The FoXQ system [1] is based on our work on visual query language for XML [2, 3].

2 Basic Querying and Editing

The FoXQ query-builder interface, shown in Figure 1, puts a lot of the power of XQuery within the grasp of the end user. In the query-builder interface, the options on the panel on the left allow the users to create their own forms, after specifying an XML data source. The forms created are of two types:

1. For queries that involve simple selection of data, the users only need to specify one pattern (called a *selection pattern*), as shown in Figure 2.
2. For queries that involve restructuring or reformatting of the selected data, the user would specify a *rule*, which basically consists of two patterns – the selection pattern for extraction of the data and the *projection pattern* for the restructuring or reformatting of the extracted data. This is demonstrated by the query in Figure 1.

The FoXQ query in Figure 2 returns a list of all the titles in a bibliography database. The results of the query execution are shown in Figure 3. The equivalent XQuery is as follows.

```
for $b in
 document("http://www.bn.com/bib.xml")/bib/book/title
return
   $b
```

The FoXQ query in Figure 1 selects the author and title information from the bibliography database and returns the

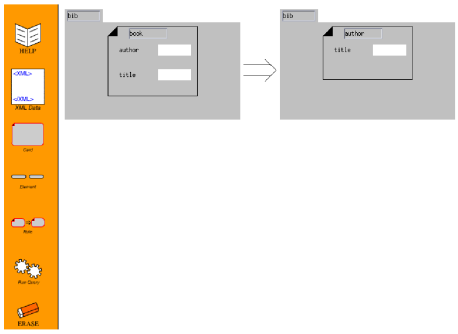

Figure 1. FoXQ query-builder interface.

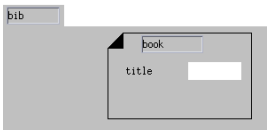

Figure 2. Selecting titles from bibliography.

titles grouped under the corresponding author informtaion. The equivalent XQuery is as follows.

```
for $a in distinct-values(
document("http://www.bn.com/bib.xml")/bib/book/author
    )
return
 <result>
   {$a}
   {
     for $b in
     document("http://www.bn.com/bib.xml")/bib/book
     where some $ba in
           $b/author satisfies deep-equal($ba,$a)
     return $b/title
   }
 </result>
```

FoXQ gives the end user many advantages over existing XQuery interfaces. The most important advantage is that the users are shielded from XQuery syntax. We are working toward providing the users more help in formulating FoXQ queries in cases where the XML data source has a DTD

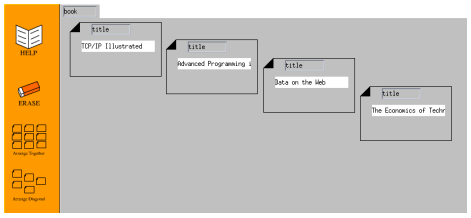

Figure 3. FoXQ result-vewier interface.

available. In these cases, the users could right-click on the form at any stage and they would be given a list of possible elements they could pick. This would help the users build correct queries.

Another advantage of the system is that the users are shielded from textual representation of XML data since the FoXQ result-viewer interface displays XML data using the document metaphor as shown in Figure 3. The result-viewer interface is actually a simplified XML editor as well. The users can delete *cards* in the returned data and this would result in the deletion of the corresponding XML elements. The users can also move the cards around on the workspace and this does reordering of the underlying XML elements. When the system is asked to save the results to a file, it saves the cards in a left–right, top–bottom sequence.

3 Advanced FoXQ features

The users can also generate new queries on saved result data. This *query-on-query* feature gives the user some of the power of fully compositional semantics that is offered by XQuery. This also allows the users to maintain local copies of data of interest to them. They do not have to connect to the web for subsequent data requests once they build a local repository. They would only need to connect to the web for periodic refreshes of their local repository. The system also supports the formulation of queries that involve a *join* of the data from two or more data sources.

4 Impact

Further research in this area would enable us to include more XQuery features in FoXQ. This would make more parts of XQuery usable by end users. By developing the interface in Java, we ensure portability of the system. We are working on an applet interface that would enable end users to use the system from a web browser. We translate the FoXQ queries to XQuery so that the user has the freedom to chose any query engine for the actual processing of XML data.

References

[1] R. Abraham and M. Erwig. FoXQ – Toward A Form-Based Visual XQuery. 2003. Submitted to the *XML Database Symposium (XSym 2003)*.

[2] M. Erwig. XML Queries and Transformations for End Users. In *XML 2000*, pages 259–269, 2000.

[3] M. Erwig. Xing: A Visual XML Query Language. *Journal of Visual Languages and Computing*, 14(1):5–45, 2003.