# XSLT By Example

Daniele Braga Alessandro Campi Roberto Cappa Damiano Salvi

Politecnico di Milano - Dipartimento di Elettronica e Informazione
piazza Leonardo da Vinci, 32 - 20133 Milano - Italy

braga|campi@elet.polimi.it

## ABSTRACT

XQBE (XQuery By Example, [1]), a visual dialect of XQuery, uses hierarchical structures to express transformations between XML documents. XSLT, the standard transformation language for XML, is increasingly popular among programmers and Web developers for separating the application and presentation layers of Web applications. However, its syntax and its rule-based execution paradigm are rather intricate, and the number of XSLT experts is limited; the availability of easier "dialects" could be extremely valuable and may contribute to the adoption of XML for developing data-centered Web applications and services. With this motivation in mind, we adapted XQBE to serve as a visual interface for expressing XML-to-XML transformations and generate the XSLT code that performs such transformations.

## Categories and Subject Descriptors

D.2.2 [**Design Tools and Techniques**]: User interfaces; H.2.3 [**Languages**]: Data manipulation languages (DML), Query languages; D.1.7 [**Visual Programming**]

## General Terms

XML, visual query languages

## Keywords

Semi-structured data, XQuery, visual query languages

## 1. MOTIVATION AND DESIGN ISSUES

The diffusion of XML in most applicative fields is accompanied by the increasing success of XSLT for generating different views of the same Web site, in order to allow its fruition to different users with different kinds of devices. This diffusion poses a pressing need for providing a wide spectrum of professionals with the capability to transform XML data, including those with limited computer programming skills (like many HTML designers). We describe here a user friendly interface, based on an intuitive visual paradigm, developed for this purpose aiming at both the unskilled users and the experienced ones, who may want to rapidly draft their transformations and refine them later on.

The W3C (World Wide Web Consortium) promotes two textual languages to express XML document transformations and to query XML data, XSLT [3] and XQuery [2] respectively. These languages, however, are far too complicated for occasional or unskilled users, who might need to specify document mappings or transformations only being aware of the basics of the XML data model and approximately familiar with the schema of the documents they have to manage. Nevertheless, even this basic knowledge of the task should be enough to allow such users to express their queries and transformations with the core primitives of a simple manipulation language; otherwise, XML will never step up to the status of a universally and successfully adopted data representation format.

XQBE (XQuery By Example [1]) was initially designed with both the objectives of being intuitive (according to the aforementioned principles) and of being directly mappable to XQuery, so as to work as a GUI capable of running on top of any existing XQuery engine. The XQBE prototype implementation has now been enriched with the capability of generating XSLT stylesheets performing XML-to-XML transformations, including XML-to-HTML transformations, as HTML can be regarded as one particular XML language. We kept in special consideration the case of HTML: we defined and implemented some abbreviated constructs that allow to compactly introduce the typical components of HTML pages, such as lists, tables, etc.


*WWW 2005,* May 10–14, 2005, Chiba, Japan.
ACM 1-59593-051-5/05/0005.

## 2. XQBE AND XSLT

Query in Figure 1 reads "*Build an HTML table with each book in a different row and with a column for the title and a column for the price*". In XSLT:

```
<xsl:template match="/">
 <table> <xsl:for-each select="bib/book">
   <tr> <td> <xsl:value-of select="title"/> </td>
        <td> <xsl:value-of select="price"/> </td> </tr>
  </xsl:for-each> </table>
</xsl:template>
```

The basic interpretation of a visual transformation is that the XML data matching the description on the left are transformed into the data described on the right. All transformations have a vertical line in the middle, that separates the *source* part (on the left) from the *construct* part (on the right); both parts contain labelled graphs that represent XML fragments and express structural properties of such fragments and conditions upon values. Tags are represented by rectangles, attributes and PCDATA content by black and white circles respectively. The source part describes the XML data to be matched in order to construct the result, while the construct part specifies which parts are to be retained in the result and (optionally) which newly generated XML items are to be inserted (represented as trapezia).

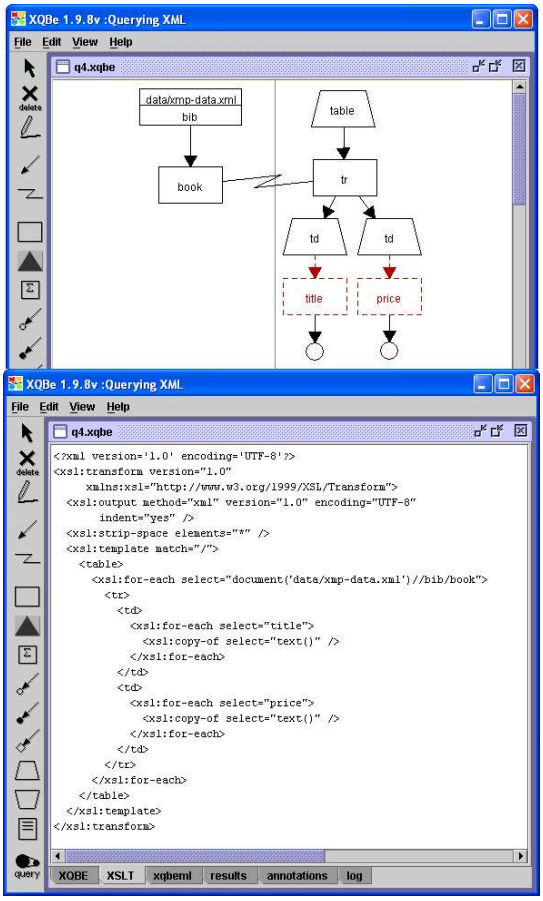


**Figure 1: Snapshot of the XQBE interface**

The correspondences are expressed by explicit binding edges that connect the nodes of the source part to the nodes that will take their place in the output document.

In Figure 1 *one table row* is constructed *for each book*, according to the interpretation of the binding edge that connects the `book` in the source part to the `tr` in the construct part. In each row two `td`s are inserted, each one in turn containing an element projected from the matched books. `table` and `td` are trapezoids as they are new w.r.t. the data source, while `title` and `price` are dashed as they specify how to access data but are not included in the result.

# 3. IMPLEMENTATION

Two snapshots of the tool at work are shown in Figure 1. Users draw queries in windows composed of two parts, corresponding to the source and construct parts. Graphs are built choosing the graphical constructs from a toolbar on the left. The graphical constructs and the graphs themselves are internally represented as XML data. XQBE queries can also be saved and exported as XML data. Once the users complete their queries they can compile them and execute the corresponding XQuery or XSLT statements. The tool assists the user during the editing process and provides syntactic feedback in several forms, to facilitate the drawing of correct queries. Many incorrect configurations are prevented "on line" by not allowing to connect two nodes or to draw a component in a place where it makes no sense. The syntactic feedback is not limited to "topological" errors, but

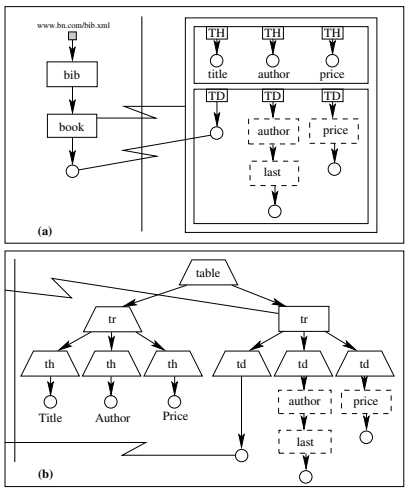


**Figure 2: A shortcut for constructing HTML tables**

makes default automatic and semi-automatic corrections to typical frequent errors, both during the editing process and at compile time. Another interesting application of XQBE as a language for specifying mappings is that of porting the content of a message from one schema to another, which is often necessary whenever different Web Services have to cooperate exchanging XML data. This can be regarded as a case of special interest within the general problem of porting the content of a document from one schema to another. Our XQBE visual editor allows to load XML Schema specifications and accordingly construct the query graphs with few mouse clicks. Users can load a DTD or XML Schema definition for the target data, thus enabling the tool to suggest the possible subelements of each selected item by showing its first-level expansion.

## 3.1 Macros for generating HTML documents

XML-to-HTML transformations are so frequent that we defined a set of "macros" that allow to save time (and query space) when generating HTML code in the construct part. These macros allow to compactly denote with one node several recurrent HTML constructs such as headers, lists, links, images, tables, etc., which are atomic concepts but require many tags and attributes to be specified. A plain XQBE transformation would force to fill the query graph with many trapezoidal nodes that are almost irrelevant to the semantics of the transformation, while the shortcuts allow to focus on the data-centric aspects of the XML-to-HTML mapping. These macros are expanded into regular XQBE nodes at compile time. Figure 2(a) shows the most complex macro, the one for generating tables. The external binding associates the table to a node in the source part (to impose the cardinality of rows); an arbitrary number of columns can be defined and filled by projecting the implicit context or importing new values by means of additional binding edges. Figure 2(b) shows the equivalent in "plain" XQBE.

# 4. REFERENCES

[1] D. Braga, A. Campi, and S. Ceri. XQBE (XQuery By Example): a Visual Interface to the Standard XML Query Language. *ACM-TODS*, June 2005, in print.
[2] W3C. XQuery. http://www.w3.org/XML/Query.
[3] W3C. XSLT. http://www.w3c.org/Style/XSL/.