Rapid Development of Web-Based Query Interfaces for XML Datasets with QURSED

Abhijith Kashyap
Dept. of Computer Science and Engineering,
SUNY at Buffalo
rk39@buffalo.edu

Michalis Petropoulos
Dept. of Computer Science and Engineering,
SUNY at Buffalo
mpetropo@buffalo.edu

ABSTRACT
We present QURSED, a system that automates the development of web-based query forms and reports (QFRs) for semi-structured XML data. Whereas many tools for automated development of QFRs have been proposed for relational datasets, QURSED- to the best of our knowledge, is the first tool that facilitates development of web-based QFRs over XML data. The QURSED system is available online at http://db.cse.buffalo.edu/qursed.

Categories and Subject Descriptors
D.2.2 [**Software Engineering**]: Design Tools and Techniques – *user interfaces, modules and interfaces.*

General Terms
Performance, Design, Experimentation, Languages.

Keywords
User Interfaces, XML, Query Interface.

1. INTRODUCTION
Web-based query forms and reports (QFRs) allow users to selectively query and retrieve information from data sources in a convenient and flexible fashion, without being exposed to the intricacies of the underlying query system. A number of tools [1-3] are available that greatly assist in the development of web-based forms and reports for relational data. However, these tools are not suitable for semi-structured (XML) data, since they are unable to capture the complexity that characterizes such datasets.

QURSED is, to the best of our knowledge, the first tool that facilitates the development of web-based QFRs over semi-structured data. QURSED is capable of handling intricacies that are inherently present in XML datasets, such as nesting, optional and repeating elements, and high structural variability. QURSED consists of a design-time component, the QURSED Editor that assists developers in specifying QFRs and a run-time component, which automatically formulates and executes XQuery expressions based on a *QFR Specification* and the input provided by a user.

When building web-based query interfaces on top of data sources, the developer needs to perform numerous tasks:

1. A developer uses a query language to declaratively specify (possibly parameterized) filtering conditions on the dataset, and the data to be displayed on the reports.
2. Query form pages display various form controls that a web user utilizes to instantiate parameters of filtering conditions, and report pages render the query results.
3. A developer performs an *association* (also known as *binding*) between the form controls on the query form page (visual aspects) and the parameters in filtering conditions (query aspects). A developer also performs an association between the query result and the html elements on the report page, so that data are displays appropriately.
4. During run-time, the parameterized filtering conditions for which the user has provided instantiations need to be included in the query, and the rest should be *excluded*.

To illustrate the above points we consider the XML version of the DBLP dataset [4]. Figure 1 shows part of the dataset's XML Schema as the QURSED Editor graphically presents it to the developer in order to formulate meaningful queries. The sample schema exposes the challenges a developer faces when it comes to specifying web-based query form and reports. Each `publication` element can be classified into one of three types based on its structure, that is, an `article` (journal), a `book` or an `inproceedings` (conference). While all types of publications have some common elements, such as `title` and publication `year`, an `article` has sub-elements `journal` and `volume` in which it was published, a `book` has an `isbn`, and an `inproceedings` has a `booktitle` of the conference it appeared. Each `publication` has a nested collection of author, where repeating `author` element can occur.

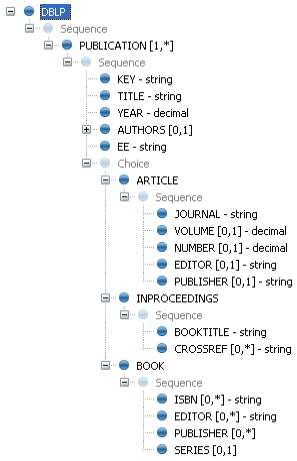


Figure 1 XML Schema of DBLP Dataset

A developer now wants to construct a QFR that allows a web user to query and report across all types of publications. Such a QFR is shown in Figure 2, where the title, author and publication year can be specified by using the appropriate form controls. The developer intention for the "Venue" form control is to refer to the `journal` element if the publication is an article, to `booktitle` if it is an inproceedings, and to `series` if it is a `book`. Figure 3 lists the

SIGMOD'11, June 12–16, 2011, Athens, Greece.

XQuery expression that the developer has to formulate in order to support the query form page of Figure 2.

2. QURSED ARCHITECTURE

Figure 2 presents the QURSED system architecture and depicts the process and actions involved in the generation and deployment of a web-based QFR. Also shown in the figure, is the interaction with a web user, which involves submission of query parameters on the form page, the formulation and execution of the XQuery expression and the display of the query results on the report.

QURSED consists of a design-time component, the *QURSED Editor* and the *QURSED Runtime*. The QURSED Editor is a tool used by the developers to create *QFR*s. It takes as input an XML Schema (XSD) describing the structure of the XML dataset to be queried, an XHTML query form page, and a template report page, which is an XHTML document describing the structure of the report. The Editor displays the schema and XHTML pages to the developer, who uses them to build the *QFR Specification.*

The *QFR Specification* and the query/visual associations generated by the editor form the input to the QURSED Compiler. The compiler then produces a set of dynamic server pages that controls the interactions with the user. The set of queries that can be generated by a QFR are parameterized by the options in the query form page associated with the QFR. A user initiates the query evaluation process by providing values to the parameters and submitting the form. QURSED Runtime then constructs XQuery expression based on instantiation provided by the user.

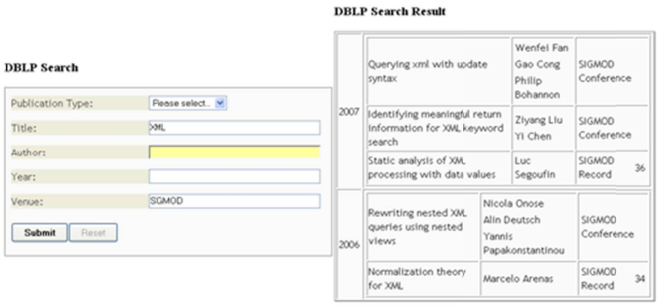


Figure 2 QFR for All Types of Publications

```
1    FOR $root IN fn:doc('dblp.xml'),
2        $DBLP IN $root/dblp,
3        $PUB IN $DBLP/publication,
4        $TITLE IN $PUB/title,
5        $AUTHORS IN $PUB/authors
6        $AUTHOR IN $AUTHORS/author,
7        $YEAR IN $PUB/year
8    WHERE
9        fn:contains(#title,$TITLE)
10   AND fn:contains($AUTHOR, #author)
11   AND $YEAR = #year
12   AND ((SOME $ARTICLE IN $PUB/article,
13             $JOURNAL IN $ARTICLE/journal
14         SATISFIES fn:contains($JOURNAL,#venue)
15       OR
16        (SOME $INPROC IN $PUB/inproceedings,
17             $BOOKTITLE IN $INPROC/booktitle
18         SATISFIES fn:contains($BOOKTITLE,#venue)
19       OR
20        (SOME $BOOK IN $PUB/book,
21             $SERIES IN $BOOK/series
22         SATISFIES fn:contains($SERIES,#venue))
23   RETURN
24       (: generate the report :)
```

Figure 3 Parameterized XQuery Expression for Figure 2

2.1 QFR Specification

It is particularly hard to encode multiple XQuery expressions when working directly with XQuery syntax, as shown Figure 3. QFR Specifications are based on the Tree Query Language (TQL) [5] which is designed to handle structural variance, nesting and optional elements present in XML datasets. TQL's captures the schema-driven generation of filtering conditions by the QURSED Editor, and also maps well to the model of nested tables used by the reports. During run-time, and once parameter instantiations are provided by a web user, the QURSED run-time component translates TQL queries to XQuery expression. Figure 5 shows the QFR Specification corresponding to the QFR of Figure 2.

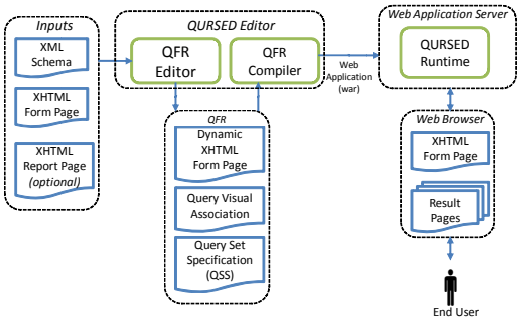
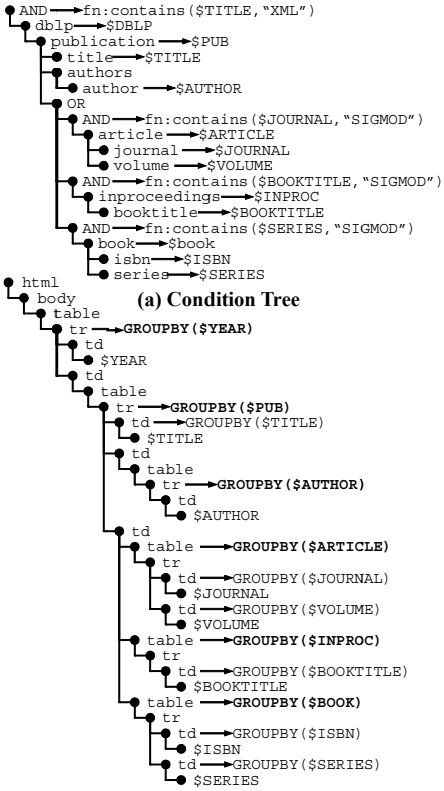


Figure 4. QURSED Architecture



(a) Condition Tree

(b) Result Tree

Figure 5. QFR Specification

A TQL query consists of a *condition tree* containing filtering conditions, and a *result tree* that defines the structure and the content of the report. The condition tree is essentially a *tree pattern* [6] following the structure of the schema in Figure 1. Element nodes are labeled with variables, if used in conditions or in the result tree. AND nodes are labeled with filtering conditions using variables labeling descendant element nodes. For example, the filtering condition `fn:contains($TITLE,"XML")` labels the

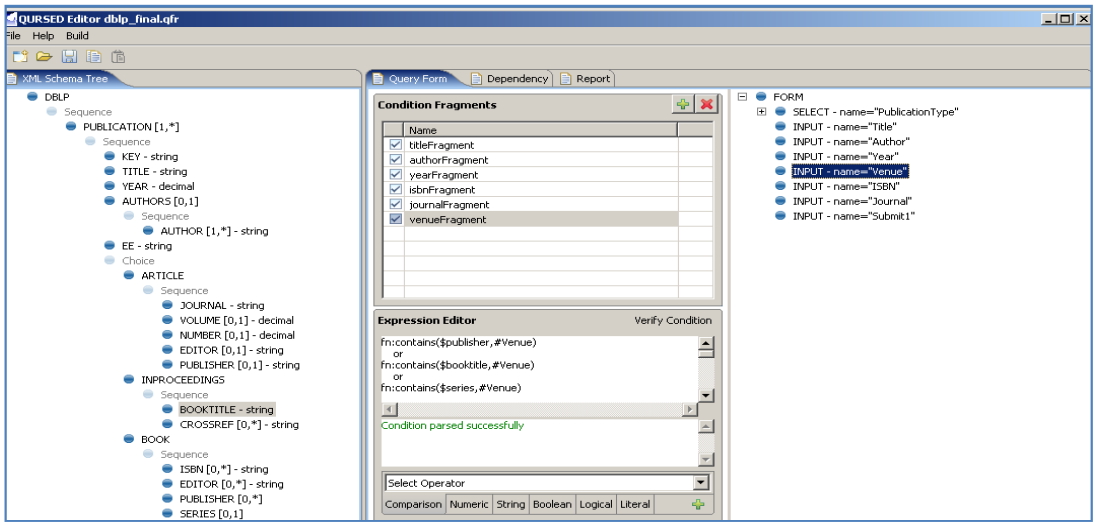


Figure 6. The QURSED Editor

root AND node and restricts the values of its `title` descendent node. The OR nodes encodes the structural variance represented by the CHOICE element group in the schema of Figure 1.

The result tree encodes an XHTML tree representing a template for the report page. Variables appear as children of table data cell elements (`td`) that will display actual data values after query evaluation. GROUPBY labels control the generation of table rows (`tr`) during query evaluation. For example, the `tr` node labeled with `GROUPBY($YEAR)` appears in the report of Figure 2 as many times as the number of distinct year values found in the dataset.

2.2 QURSED Editor

A developer uses the QURSED Editor shown in Figure 6 to develop QFRs. She does this by a series of visual actions which the Editor interprets in order to construct the QFR specification. The developer opens an XSD file, an XHTML form page and optionally an XHTML template report page. The developer then proceeds to define condition tree *fragments*.The fragment parameterizes the query that the publications venue provided as input by the user matches one of article's journal, book title of `inproceedings` or a book series. The schema tree is shown on the left panel and the right panel contains the XHTML document tree of the query form page.

To define a fragment, the developer first creates and names a fragment by clicking on the "New Fragment" button. Then, she builds the condition of the fragment, either by typing it into the expression editor or by drag-and-dropping nodes from the schema and query form document trees.

On the report side, the developer can use the Editor to create the result tree, such as the one shown in Figure 5b, by mapping the nodes in the schema tree to the document tree of the report template page, thereby binding the data elements in the query results to the presentation nodes in the result tree

2.3 QURSED Runtime

The runtime component instantiates a TQL query based on the form parameter values provided by the user. The condition tree of instantiated TQL query consists of the sections for which the user has provided values, thereby precluding their exclusion. For example, the sub-tree under the OR node of Figure 3 is included only if the user provides a value for the "Venue" field in the form.

The XQuery generation algorithm of QURSED works on TQL queries. The algorithm takes as input the condition tree *CT* and the result tree *RT* of a TQL query. The main routine of the algorithm traverses the result-tree top-down and builds an XQuery expression that consists of FLWGOR expressions [7].

3. DEMONSTRATION PLAN

In this demonstration we will show the effectiveness of QURSED in creating and deploying web-based forms and reports. Using the sample DBLP schema shown in Figure 1 and a prepared form page and report templates, the specification for the QFR shown in Figure 2 will be created. The entire specification can be created in under 5 minutes, demonstrating the power of QURSED.

QURSED has several additional features, whose description was omitted due to space considerations including:

- Automatic Report Template generation.
- *Dependent filtering conditions* to provide a fine-grained control over what XQuery expressions. For example, if the specification had a condition fragment <$isbn=#ISBN>, we would want to instantiate it only if the user chooses the *Publication Type* Book.

4. REFERENCES

[1] *Adobe ColdFusion*. Available: http://www.adobe.com/products/coldfusion/.

[2] *Altova MapForce*. Available: http://www.altova.com/MapForce.

[3] *Adobe Dreamweaver*. Available: http://www.adobe.com/products/dreamweaver/

[4] *DBLP XML Dataset*. Available: http://dblp.uni-trier.de/xml/.

[5] M. Petropoulos, Y. Papakonstantinou, and V. Vassalos, *Graphical query interfaces for semistructured data: the QURSED system.* ACM Trans. Internet Technol., 2005. 5(2): p. 390-438.

[6] Amer-Yahia, S., et al., *Tree pattern query minimization.* The VLDB Journal, 2002. 11(4): p. 315-331.

[7] XQuery 1.1 Use Cases. Available: http://www.w3.org/TR/xquery-11-use-cases/#dataproducts