AN APPLICATION TO INTEGRATE RELATIONAL AND XML DATA SOURCES

Ana Mª Fermoso García, Roberto Berjón Gallinas
Computer Science Faculty, Pontificia University of Salamanca, C/ Compañía, 5, 37002 Salamanca (Spain)

Mª José Gil Larrea
Engineering Faculty (ESIDE), Deusto University, Avda. Universidades 24, 48014 Bilbao (Spain)

Keywords: Relational databases, XML, data integration, heterogeneous data sources.

Abstract: Nowadays, special with the Internet explosion, enterprises have to work with data from heterogeneous sources, such as data from conventional databases, or from new sources of Internet world like XML or HTML documents. Organizations have to work with these different data sources at the same time, so, it's necessary to find some way to integrate this heterogeneous information.In this paper we are going to centre in two main types of data, conventional data from relational databases, and the new web data format XML. Traditional relational database continues being the main data store and XML has become the main format to exchange and representation data on the web. At the end our purpose would be that the necessary data in each moment were in the same and common format, in XML, because this is the most used format on the web.This paper proposes an efficient environment for accessing relational databases from a web perspective using XML. Our environment defines a query system based on XML for relational databases, called XBD. XBD has a full XML appearance, query language and query results are in XML format. For the end user it is similar to query a XML document. This system includes a model to adapt any relational database in order it could be queried in two new query languages, derived from XSL and XQuery languages, and a software tool to implement the functionality of the XBD environment.

1 INTRODUCTION

Nowadays business information systems are suffering numerous changes. Business is transforming in e-business and it has to manage large data volumes and besides, from heterogeneous sources.

When business gets into the web it has to manage new types of web data as XML or HTML, for example to exchange information with other business. However, business information continues also stored in the traditional databases as the relational one.

At the same time, the volume of business information is growing a lot. The information is the power and data are transformed in information when they are used to make business decision. This is happening, for example, in the new data warehouse store systems, where data are stored and analyzed before using them like knowledge. Previously this storing, it would be useful to integrate this information from different formats.

In this context, where organizations have to manage a lot of data but from heterogeneous sources, like conventional database systems or the new web data sources, it could be necessary to use tools as mediators to integrate these different data to a common format like XML, the main format language on the web.

As we just say Internet has change the business world and has also made to appear new data formats as XML. The XML history has always been connected to the Web evolution (W3C, 2004a). In fact XML has reached the status of standard of data exchange and representation in Internet.

In order to manage large amounts of information, Database Management Systems (DBMS) continue to be one of the most used tools,

Mª Fermoso García A., Berjón Gallinas R. and José Gil Larrea M. (2005).
AN APPLICATION TO INTEGRATE RELATIONAL AND XML DATA SOURCES.
In *Proceedings of the Seventh International Conference on Enterprise Information Systems*, pages 313-319
DOI: 10.5220/0002555403130319
Copyright © SciTePress

and the most extended model is the relational one, although the object oriented model or data warehouse is being used more and more.

Databases world, in order to manage data from all these heterogeneous sources and formats, have had to adapt also to the new market, allowing for example to store these new types of web data, or changing the way to work with the data using them as knowledge, as in the data warehouse.

Therefore, because almost all organizations eventually get into the web, where XML is emerging as a more effective means of describing the semantic content of WWW documents, and because the main amount of business information continues stored in relational databases, it would be useful to manage a relational database from a web perspective using XML. That is, not only to use XML language as interface to pass information between systems that interact with each other on the web, also to employ a XML based language to access to the databases of these systems.

In this paper we are going to centre only in these two main data sources, the conventional relational database system, because as we just say most of the business data continuing store in them, and XML data, because it is the most extended data format to exchange and represent information on the web.

The idea would be how to integrate these two types of data in order for example to use them later, to store them in a data warehouse or exchange their information in the web.

Besides, one of the main requirements of the query system, as we just said, is its XML based appearance.

Another important requirement would be that data maintains its original format, the information remains stored in the database and it does not have to go through any migration process.

Our goal will be to permit to query a relational database, without taking into account its complexity, size or subject of its information, in a similar way that if it was a XML document and obtaining XML data after query execution too. The end user will not need to know the real storage of the data, for him it will appear as if he were querying a XML document. The main contribution of this paper is to explain the foundations of this new query system called XBD (XML for Databases) which implements this functionality. XBD is a query system based on XML for querying relational databases.

2 OBTAINING XML FROM RELATIONAL SOURCES

The purpose of this research is to help to integrate relational database and XML data, obtaining XML data from relational database data. One time all the information, from databases or XML documents, was in the same format, XML, will be easier to manage it together.

Last versions of the traditional Database Management Systems (DBMS) as Oracle (Oracle, 2002a) (Oracle, 2002b), Microsoft SQL Server (Microsoft, 2002) or IBM DB2 (IBM, 2002), have had to be updated to work with XML data. According to this, they already can store XML data together with relational information, carrying new types of data like XMLType in Oracle and SQL Server, or XML column and XML collection in DB2. But in the present research we are not interested in the storing of XML information in databases systems, we are only interested in how to obtain XML data from relational sources. In this sense, these new versions of the most used DBMS, allow to query relational data in the database query language SQL, but returning the query results in XML format.

Using active server pages designed with Java Server Pages, Microsoft Active Server Pages or PHP web pages, is also possible to attain the same effect. The query to the database in SQL is included inside the web page code, and they will return the database query results in XML. These results are added when the web page is showed in a explorer.

In both kinds of tools, DBMS tools or active server pages, the query language is SQL. In our query system we want not only the query results have a XML appearance, also the query language have to be derived from XML, so, we will not use SQL as query language.

Other systems that could be in relation with our research are the ones that transforming the data model of a relational database model, to a special XML View called *virtual XML view.* This view show to the user the database content in order he can make later, queries over this database using XML languages and obtaining the results in XML format, too. The virtual XML view is similar to a XML document where would be stored the database records content, but it is “virtual” because the database data are not really translated physically to XML.

The two main systems of this type are XTABLES (Funderburk, 2002) and SilkRoute (Fernández, 2002).

In XTABLES the virtual XML view are called “default view”. This system use XQuery as query language to obtain the default view and to make the following queries over this view.

In SilkRoute the virtual view is called "query view". The first query to obtain this virtual view is implemented in the RXL language, a special language characteristic of this system. In the following queries over this initial view, is used XML-QL as query language.

To obtain XML data from databases is also our purpose, but the problem is that in these systems it is necessary a first query to create the Virtual XML View, and only the database side affected by this initial query, will be able to be queried in the following queries. In our system each query will have to be able to access any part of the database in any moment.

One time we have made a short study about some methods to obtain XML data from relational sources, and studied their disadvantages compared to our aims, we are going to explain our proposal, called XBD system, to get the same goal. In (Fermoso, 2004) we can find a detailed description of this new query system.

3 XBD: A NEW QUERY SYSTEM

If we could access relational databases via XML, this would make easy to integrate XML data sources, not only with each other, also with structured data like relational databases, and this is the purpose of our new XBD system.

To develop our XBD query system, it would be necessary a method to make any relational database can be queried in this XML environment.

In the previous study, some analyzed systems create first a virtual XML view allowing the following queries over the database. In our case we are going to implement other solution, we are going to create the "database XML schema" document as exit of the adaptation process. Only when we just obtain this XML document, will be possible to make queries in XML over the database. Later we will explain the meaning of this document.

One time we describe the model to adapt a database to the XBD query system, we have to define the syntax of the language we will have to use to query the database.

Finally, we are not going to use SQL as query language because it is not XML based, but at the end the query over the database will have to be executed in SQL, therefore, it would be necessary to describe the method to translate the query in our XML language, to a query over database in SQL, in order to execute it and return the query results in XML.

As we just say, the first step to get our goal, would be to adapt any relational database to our query system, this is, to establish, the model that allows rewriting the relational database structure in a special format, XML accessible, but without any kind of physical translation to XML from the database records content, because this would be inefficient (we would need additional store size and queried data would not be always updated). This step is called "adaptation database system".

The second step would be to establish the query XML based language, to access the databases in our environment.

The last step would be to define the model to translate the query from this query language into a SQL SELECT clause, to make the final access to the database using the own Database Management System (DBMS), because it will execute the query in a more efficient way and will already return the query results in XML. This step is known as "query evaluation system"

We can conclude that the architecture of the proposed system will be made up of two independent components. The first one, translates the structured data model of the database to a special XMLSchema, it is the adaptation database system. The second component, the query evaluation system, receives a query in a XML language over the database, which should have already been adapted to the system with the preceding component, producing query results in XML. Figure 1. shows this architecture.

We are going to analyze first the new languages we have created to make queries in our XBD environment and later, we will describe the tasks and functions of the two main components of this system, that is, how do the adaptation database and query evaluation systems work.

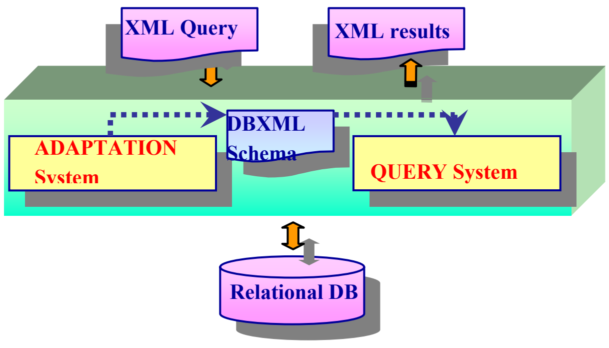


Figure 1: XBD System Architecture

4 XBD QUERY LANGUAGES

If we want our system to have a XML appearance, we should select a language from this environment. On the other hand, we could have decided to create a new language from the beginning, or use a language that already exists. We believe it is better the second solution because it is easier for the user to make the query if he already knows the language.

The two main languages belonging to the group of XML languages are XSL (W3C, 2004d) and XQuery (W3C, 2004e). XSL (Extensible Stylesheet Language) is XML fully defined, widely implemented, and nowadays an industry de facto standard. Actually, it is not a query language, but a presentation language, although by its final results could also be considered like a query language. XQuery is not XML defined, but will be sure the future query language for XML information. So, we have decided our query language was based in these languages and we will have to study how to adapt the two languages to the database query.

At the end we have adapted both languages to be used in our system and we have created two new languages derived from them called Adapted XSL and Adapted XQuery. It will be possible to use any of these two languages to query databases.

We have added new characteristics and/or modified others over their standard versions, to adapt these languages to querying relational databases. In every moment we have tried to keep the similarity between the meaning of the standard language constructions of these languages and the equivalent SELECT clause in SQL, since at the end the query in Adapted XSL or Adapted XQuery, will have to be translated to SQL.

In both new languages, every query document is translated to a single SELECT clause in SQL language, although there could be other SELECT clauses inside one (nested SELECT). The SELECT clause obtained after the translation process will be executed over the database.

4.1 Adapted XSL Language

To adapt the standard XSL language to XBD system we only need to use some of the standard XSL constructions: `stylesheet`, `apply-templates`, `template`, `value-of`, `if`, `for-each`, `when and short`.

We are going to explain the mining of these constructions in relation with the database query equivalent in SQL:

`xsl_adapt:stylesheet`: shows the file where is the Database XML Schema obtained in the adaptation process. In this document we find the information about how to connect with the database that we are going to query. The name of the Database XML Schema document will appear in a new attribute of this construction, called "`schema_db`".

`xsl_adapt:apply-templates/xsl:templates`: every "template" is translated to a single SELECT sentence in SQL.

The value of the `match/select` attributes of these constructions show the main table of the query. This table will appear in the FROM clause in the equivalent SQL SELECT sentence. If other constructions in the query, `value-of`, `if`, … reference fields from other tables in the database, these tables will also be added to the SELECT sentence.

`xsl_adapt:value-of`: shows the database fields queried. These fields will appear in the SELECT clause when the query was translated to SQL.

`xsl_adapt:if`: adds a condition to the query. This construction will be translated to a WHERE clause of the SELECT sentence in SQL.

`xsl_adapt:for-each`: allows grouping the query. In SQL this construction is translated to the GROUP BY clause of the SELECT sentence.

`xsl_adapt:when`: adds a condition to a GROUP BY clause in SQL, the "HAVING" part of the GROUP BY in SQL.

`xsl_adapt:sort`: allows to sort the query. It is equivalent in SQL to the ORDER BY clause.

All these constructions have attributes (`select`, `match`, `test`, …) where you can use path expression. These path expressions refer to database tables and fields in the query. To express these path values the XPath language (W3C, 2004b) syntax is used, where the symbol "/" separates the names of tables and fields of the database referenced in the query.

Besides, it is also possible to nest SELECT sentences to permit the "joins" in the queries. Since a "template" and the corresponding "apply-template" are translated to a single SELECT sentence, if inside a "template" appears an "apply-template" construction, which will correspond to other SELECT sentence, it means that this second SELECT will be nested inside the SELECT of the first "template", and in the same way, inside of the "template" associated to the "apply-template" could be other apply-template equivalent to other nested SELECT, and so on.

For example, the following select sentence, which query the `"name_prod"` field from a table `"Product"`: `SELECT name_prod FROM Product`, in Adapted XSL language would be equivalent to:

```
<?xml versión="1.">
<!DOCTYPE xsl_adapt:stylesheet system
"xsl_adapt.dtd">
<xsl_adapt:stylesheet
version = "1.0"
schema_db= "schema_db_produc.xml">
  <xsl_adapt:template match=/">
     <xsl_adapt:apply_template
          select="Product">
  </xsl_adapt:template>
  <xsl_adapt:template
         match=//Product">
     <xsl_adapt:value_of
         select="name_prod">
  </xsl_adapt:template>
</xsl:stylesheet>
```

In the "schema_db" attribute of the stylesheet construction appears the name of the XML document that contains the Database XML Schema.

4.2 Adapted XQuery language

To adapt XQuery language to database query we only need to use the FROM-LET-WHERE RETURN (FLWR) XQuery expression, because it is the most similar to the SELECT clause in SQL. The mapping to obtain the equivalent SQL clause would be:

FROM, RETURN WHERE: show the tables of the query. This tables will appear in the FROM clause of the equivalent SELECT sentence.

RETURN: shows the fields of the query. These fields will be the database fields of the SELECT clause in the SQL sentence.

WHERE: allows the addition of a condition to the query. It is translated to a WHERE clause in the SELECT sentence of SQL.

LET: allows grouping the result. It will be the GROUP BY of the SELECT sentence in SQL.

SORTBY: this sentence appears inside a FROM clause and allows the result to be sorted. It is equivalent to the ORDER-BY clause in the translation to SQL.

The path expressions which appear in the FLWR sentences show the database fields and tables in the query. These path expressions use the XPath syntax.

In the same FLWR expression it is also possible to reference different tables and later, in the RETURN clause, join their results. In this way it will be possible to make nested SELECT when the query is translated to SQL.

For example, the same query as in the last section (SELECT nom_prod FROM products) and being "schema_db_produc.xml" the XML document with the Database XML Schema of the queried database, which contains database structure information once it has been adapted to XBD system, in Adapted XQuery, would be:

```
FOR $p IN
document("schema_db_produc.xml")
RETURN Product/name_prod
```

5 XBD SYSTEM COMPONENTS

XBD system has two main functions, implemented in a new software tool, that are described in the next sections. The first is used to adapt any relational database so that it could be queried in XBD and the second, once the database has been adapted to our system, it allows it may be queried in our Adapted XSL or Adapted XQuery languages.

5.1 Adaptation Database System

This system allows any relational database to be queried in XBD. To do this it is necessary to obtain a special XML document called Database XML Schema which will have all the necessary information about the database structure: tables, relations and fields and will be used as input for the next query component. The appearance of this document will be very similar to a XML Schema document (W3C, 2004c). Besides, we can obtain two types of Database XML Schema documents, one with the appearance of a XSD schema document of the W3C and another one with the XDR schema of Microsoft, since these are the two types of schemas used to validate XML documents.

The elements of these documents depend on the format used, element and ElementType in XDR schema and SympleType, ComplexType, sequence and element in XSD. These elements will contain all the information about the database: tables, their relationships with other tables and cardinality, fields of every table with their data types and information on whether they are a key in this table or not.

In this way, if we use a Database XML Schema in XSD format, the "SympleType" element will describe every database field or column, and the "name" and "type" attributes of this element, will specify the name and data type of the field. To describe every database table it will have to be used the "ComplexType" element. Nested with every "ComplexType", the "sequence" and "element" elements will describe all the fields and linked tables to the table of the "ComplexType". For every relationship the cardinality is showed with the "minoccurs" and "maxoccurs" attributes. For every field it is specified if the column is or not a key of the table and the type of key, primary or foreign key (pk or fk).

Using the XDR schema format, we make a parallelism to represent the database structure, but now using "element" to define the name and type (table or field) of every component of the database, and "ElementType" to describe the content of every table, similar to "Complextype" in the XSD Schema.

For example, if we have a database where we have information about products and orders: `PRODUCTS (cod_prod, stock)`, `ORDERS (cod_ord, date_ord, cod_prod, cant_ord)` and we know a product can appear in many orders, the representation of the `PRODUCTS` table in the Database XML schema in XSD format could be the following.

```
<SympleType name="cod_prod"
type="NUMBER">
<SympleType name="name_prod"
type="VARCHAR2">
<ComplexType name="PRODUCTS">
 <sequence minOccurs="1"
   maxoccurs="1">
    <element type= "cod_prod"
     pk="true">
    <element type= "stock_prod>
  </sequence>
  <sequence maxoccurs="unbounded">
    <element type ORDERS>
  </sequence>
</ComplexType>
```

We have added to our both types of Database XML Schema documents a new element called `conexion`, which contains the necessary information to connect the database queried. In this element, the values of the "owner" and "cad_conex" attributes, specify the necessary information to make the connection to the database (driver, user and password), although the details of this string depend of the DBMS system of the database, Oracle in our example:

```
<conexion type="basedatos" owner="pp"
cad_conex="jdbc:oracle:thin:pp/pword@
srvdes.upsa.es:1521:bd"/>
```

5.2 Query Evaluation System

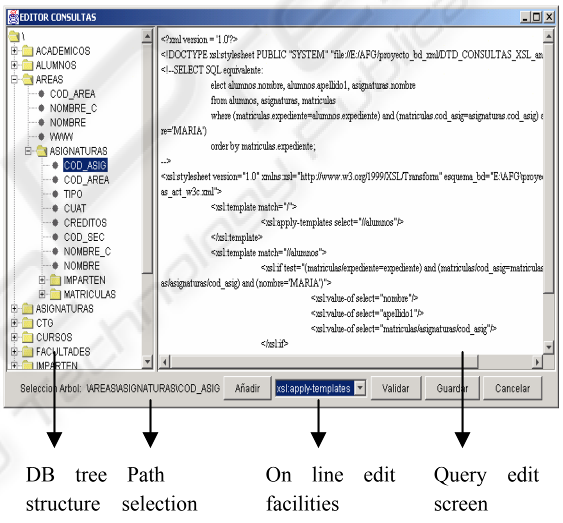
This component allows querying a relational database, adapted with the previous component, in our XML languages. In practice our software tool only implements queries in Adapted XSL language, but the steps would be similar if we use our Adapted XQuery language.

Thanks to the interface of the software component which implements this query function, the end user always believes he is querying a XML document; he makes a query in a XML language and obtains the results in a XML format too.

Query evaluation system allows, first to edit or modify a query in Adapted XSL language, then to test it, and finally to execute it, so, it has three components.

The editor component allows us to edit or modify a database query in Adapted XSL language. To help the user to edit the query document, the database structure is shown as if it were an XML document, in a tree structure. The nodes and branches of this tree will be the tables and fields of the database.

Every table is represented in a node and expanding the branch associated, you can find the fields of this table and the tables in relation with it. In this way, the user thinks he is accessing to a XML document, the appearance is the same. Besides, this component also provides editing on line facilities to write the Adapted XSL constructions in the query. In the figure 2 we can see the appearance of the editor component.



```
<?xml version = '1.0'?>
<!DOCTYPE xsl:stylesheet PUBLIC "SYSTEM" "file://E:/AFG/proyecto_bd_xml/DTD_CONSULTAS_XSL_an
<!--SELECT SQL equivalente:
          elect alumnos.nombre, alumnos.apellido1, asignaturas.nombre
          from alumnos, asignaturas, matriculas
          where (matriculas.expediente=alumnos.expediente) and (matriculas.cod_asig=asignaturas.cod_asig) a
re='MARIA')
          order by matriculas.expediente;
-->
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform" esquema_bd="E:\AFG\proye
as_act_w3c.xml">
          <xsl:template match="/">
                    <xsl:apply-templates select="//alumnos"/>
          </xsl:template>
          <xsl:template match="//alumnos">
                    <xsl:if test="(matriculas/expediente=expediente) and (matriculas/cod_asig=matriculas
as/asignaturas/cod_asig) and (nombre='MARIA')">
                              <xsl:value-of select="nombre"/>
                              <xsl:value-of select="apellido1"/>
                              <xsl:value-of select="matriculas/asignaturas/cod_asig"/>
                    </xsl:if>
```

Figure 2: XBD Query editor interface

The tester component allows us to test if the query is or not correct in two senses. It tests if the query syntax is correct applying the Adapted XSL rules, and it also tests if all the elements referenced in the query with the path attribute values, belong to the database structure components, tables, fields and relations.

Once the user has proved the query is correct, he can execute it over the database with the evaluator query component. This component executes the query over database and returns the query results in XML format. To execute it, first the component translates the Adapted XSL query to the equivalent SELECT clause in SQL, applying the rules of the 4.1 section. After that, the DBMS will execute this SQL query, and thanks to the new DBMS tools, will return the result in XML.

6 CONCLUSIONS

In this paper we have studied the need of a system for querying relational databases in a web environment using a XML appearance. XML is the most useful standard to data exchange and integration on the web and as such, can serve like a link between heterogeneous data sources that have to work together on the web, like structured databases and XML documents.

Our new XBD system gives user the possibility to query a relational database like if it was a XML document. This makes easy the possibility to integrate XML and relational data in a transparent way for the user, that always thinks he is queried XML data.

We have described the main characteristics of the XBD system. These features, as we can see in the next paragraphs, show that it is efficient and transparent for the user, and this fulfils our aims.

We have also defined the language that will be used for querying. Finally, we explain the main components of this system that permit the adaptation to any relational database and then to query it.

To show the efficiency of our system, we can see that to adapt the database, we do not have to translate the database data or records to any format, XML or others, and this implies saving storage space and that the queried data will always be updated. Besides, with this adaptation process, user can query any side of the database, because there isn't any initial query or XML view that restricts the rest of queries.

Second, we get the query execution were very fast because it is executed by the self DBMS where the database is located, we have only added the translation process from Adapted XSL to SQL, and this extra time is minimum compared to the complete query execution time. Finally, the whole environment has an XML appearance, in this way the user believes he is querying a XML document, not a database, and this is very important when we want to work in a web environment managing different data sources, XML and databases, but in a transparent way for the user.

In addition to all this advantages and to prove them again, our XBD system has been satisfactory tested in real production environments. For example, it is being used with success in an application where university students query the marks of their courses in the web. XBD allows query the database where teachers introduce the student data and shows the query results in a XML format on the web. Besides, if it is necessary, these results could be merged later with other XML data, for example with other XML information about the students that we have obtained from other university databases.

REFERENCES

Fermoso, A., Berjón, R., 2004. Acceso a datos relacionales en entornos Web. Un caso práctico. Publicaciones Universidad Pontificia de Salamanca

Fernández, M., Kadiyska, Y., Morishima, A., Suciu, D., Tan., W.C., 2002. SilkRoute: a framework for publishing relational data in XML. ACM Transactions on Database Systems (TODS), Vol. 27, Nª 4, December, pp. 438-493.

Funderburk, J. E., Kiernan, G., Shanmugasundaram, J. , Shekita, E., Wei, C., 2002. XTABLES: Bridging relational technology and XML. IBM Systems Journal, Vol. 41 , Nª 4.

IBM, 2002. IBM DB2 Universal Database. XML Extender Administration and Programming. Version 8. IBM Corporation.

Microsoft, SQL Server 2000. XML and Internet Support. Microsoft Corp.

Oracle, 2002. Oracle 9i Release 2. Database Concepts. Oracle Corp., March.

Oracle, 2002. Oracle 9i Release 2. XML Database Developers's Guide-Oracle XML DB. Oracle Corp., October.

World Wide Web Consortium (W3C), 2004. Extensible Markup Languaje (XML). http://www.w3c.org/xml.

World Wide Web Consortium (W3C), 2004. XML Path Language (XPath). http://www.w3c.org/TR/xpath.

World Wide Web Consortium (W3C), 2004. XML Schema. http://www.w3c.org/2001/XMLSchema.

World Wide Web Consortium (W3C), 2004. Extensible Style Language (XSL). http://www.w3c.org/Style/XSL.

World Wide Web Consortium (W3C), 2004. XQuery: A Query Language for XML. http://www.w3c.org/TR/xquery.