http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 4 (4): 439-444, 2005
ISSN 1812-5638

Comparative Analysis of Three Promisimg XML Query Languages and Some Recommendations

[1]Uttam Kumar Deb Nath and [2]Dencho N. Batanov
[1]Department of Computer Science and Engineering,
Chittagong University of Engineering and Technology, Chittagong-4349, Bangladesh
[2]Computer Science and Information Management Program, Asian Institute of Technology,
P.O. Box 04, Klong Luang, Pathumthani 12120, Bangkok, Thailand

Abstract: XML has become the standard tool for the next generation web document format. A query language for extracting information from the XML documents is worthmentioning. Query languages from different communities and individuals have been proposed. The World Wide Web Consortium(W3C) is a leading contributor in this field. We have discussed the different aspects of some selected XML query languages and reviewed the limitations of the selected query languages. As an outcome of our work, we have tried to recommend the necessary functionalities and requirements for a possible new query language.

Key words: Query language, XQuery1.0, YAXQL, XML-QL, XML

INTRODUCTION

XML[1] data are widely used in web information systems. Such applications usually use a large number of XML data. So, the web must allow users to retrieve only necessary portions of XML data by specifying search conditions to flexibly describe such applications[2]. Also the Web must allow users to combine XML data from different sources. This is what led to the need of designing the query languages for XML documents. The aim of this study was divided into four parts:

- Analyzing some selected presently existing query languages carefully and picking up their main functionalities.
- Evaluating some selected query languages against some collected, necessary and important requirements and functionalities.
- Analyzing the functionalities based on comparative analysis among three selected query languages.
- Proposing a set of recommendations for the core requirements and functionalities of a possible new query language.

The query languages mainly considered in this study are XML-QL[3], XQuery[4] and YAXQL[5]. There are lots of other query languages[6-9] used by the different communities. But we found out that the three query languages above are the most promising in user flexibilities and functionalities[10-12]. So, we took them as our matter of interest. Something important in other query languages may be missed, which may contribute to the better understanding of the query language requirements and functionalities.

As W3C (World Wide Web Consortium) is the standardization organization for the internet related topics, we took W3C proposed query language, XQuery 1.0, as the main focus of our evaluation task.

COMPARATIVE ANALYSIS OF XML-QL, XQUERY AND YAXQL

Introduction to the three query languages:

XML-QL: XML-QL was designed at AT and T Labs and found at the url: http://www.w3.org/TR/NOTE-xml-ql/. The XML-QL language uses a WHERE-CONSTRUCT structure to make the queries and result construction. It was aimed at solving the problems of data extraction, conversion, transformation and integration.

XQuery 1.0: XQuery was designed by W3C. It can be found at the URL: http://www.w3.org/TR/2003/WD-xquery-20031112/t is claimed to be broadly applicable across all types of XML data sources. The latest version of XQuery has been taken as working draft, which has been published on 12th of November'2003 and named XQuery 1.0.

Corresponding Author: Mr. Uttam Kumar Deb Nath, Department of Computer Science and Engineering,
Chittagong University of Engineering and Technology, Chittagong-4349, Bangladesh
Tel: 880 31 723336 Fax: 880 31 714910

YAXQL: YAXQL was designed at University of Mannheim, Germany by Guido Moerkotte. It can be found at his personal homepage of University of Mannheim. The URL is: http://pi3.informatik.uni-mannheim.de/staff/ mitarbeiter /moer/myself.html. It has been claimed as a web-aware and most powerful query language proposed so far with its some of the unique features missing in all other languages.

To make a comparison between these three query languages, almost all the query features are selected that should be present in a full-fledged XML query language and we tried to test the possibility of expressing these query features by all three languages with explicit examples.

Requirements considered for a full-fledged query language:

- Query syntax and the result of the query should be in XML syntax.
- Supports basic and important query functionalities
- Support for explicit type casting
- Distribution of queries
- Pattern matching
- Should be web-aware
- Sufficiently expressive
- Provide support for maximal reuse of queries
- Allow queries to return not only values but also references
- Based and related to other XML standards.

XML query language should be compatible with all other XML standards like XPointer, XSLT etc. So that any changes in the current standards can be accompanied with a change in the query language also.

Some example queries in YAXQL: Queries in XML-QL and XQuery are well known by the database community and thus are not shown here due to space limitation. Queries in YAXQL are a bit less spreaded and thus some unique features in YAXQL are as follows:

1. Query Identification, accessing and reuse

XML-QL

Does not have something like this.

Xquery

Does not have something like this.

YAXQL

YAXQL has this unique feature of reusing queries in other queries. For doing this, it can either reference to another query and bind the variable in that query to another value defined in the main query and use the result of that query by dereferencing the result variable of the previous query into the main query. Or it may also reuse the previous query by directly dereferencing the result variable of the previous query. Identification of query is done by giving the query a unique name with the name attribute.

The following example shows that the main query named "edb:q4" reuses the query "edb:q2" by first identifying the query with the "href" attribute of the <xql:query-reference> element. At the same time it is referring to the query "edb:q2" and assigning a new value for the variable "ec-name" in that query. Then the result of the query "edb:q2" is dereferenced in the "select" attribute of the <xql:match> element of the main query.

```
<xql:query name="edb:q4">
 <xql:declare var ="enzyme-name" maybe-unbound="NO"
/>
<xql:query-reference href = "edb:q2">
<xql:bind var = "ec-name" select = "enzyme-name"/>
</xql:query-reference>
<xql:match var = "inhibitor" select = "//$(edb:q2).result-
>/edb:inhibitor/" bind-to="text"/>
<xql:group-by var="$ec-name"/>
<xql:project>
<xql:bind var = "result">
<xql:value var = "inhibitor"/>
</xql:bind>
</xql:project>
</xql:query>
```

2. Unbound Parameters at Query Evaluation Time

XML-QL

Does not have this provision.

XQuery

Does not have this provision.

YAXQL

This is another unique feature of YAXQL. First of all, we want to describe the importance of this feature in

queries. One of the most important uses of this feature is in maximizing query reuse. Consider a service that is provided with a form-based interface. Not necessarily all the entries in the forms have to be filled by the user. Nevertheless, the service provider are wanted to be able to specify only a single query instead of one query for every possible combination of filled entries. This kind of reuse requires that the query contains several free variables (parameters/arguments) and only some of them are bound prior to query evaluation. This implies to modify the underlying logic of query languages seen so far. Such a new four-valued logic has been introduced in YAXQL to specifically deal with the semantics of queries containing free variables. No other languages proposed so far considers this point.

The following query demonstrates the use of unbound parameter at query evaluation time. It retrieves enzymes with certain properties. The properties must not all be specified.

```
<xql:query name =”edb:q6”>
 <xql:declare var = “name” maybe-unbound = “YES”/>
 <xql:declare var = “temp” maybe-unbound = “YES”/>
 <xql:declare var = “inib” maybe-unbound = “YES”/>
  <xql:match var = “enzyme” href = “edb.xml” select
=”/edb:enzymes/edb:enzyme”/>
      <xql:and>  <xql:eq>  <xql:source  select
=”$enzyme/edb:name/text()”/>
 <xql:value var =”name”/>
 </xql:eq>
   <xql:eq>  <xql:source  select  =”$enzyme/edb:temp-
opt/text()”/>
 <xql:value var =”temp”/>
 </xql:eq>
  <xql:eq>  <xql:source  select  =”$enzyme/edb:inhibitor
/text()”/>
 <xql:value var =”inib”/>
 </xql:eq>
 </xql:and>
 <xql:project>
 <xql:bind var =”result” select =”$enzyme”/>
 </xql:project>
</xql:query>
```

In this query three free variables have been declared e.g.,“name”,“temp” and “inib” which are not bound at the query evaluation time.

3. User Interaction using form-based interfaces

XML-QL

Does not have any such provision.

XQuery

Does not have any such provision.

YAXQL

This is another unique feature of YAXQL. It can define form-based interfaces with attached queries. This part of YAXQL is the second possibility (besides xql:evaluate element) to integrate queries into XML documents. A special element <xql:fill-form> allows to query websites with a form-based interface. A child element of <xql:fill-form>, <xql:form-entry> specifies the name of the field invoved in the form and declares a variable to be bound for this field.

Now, given a document, the <xql:ask-for-variable-binding> element allows to specify a field that can be filled by the user. The user’s input is then bound to a variable specified in the <xql:ask-for-variable-binding> element. Another attribute allows specifying a title for the form. All the variables bound explicitly or by filling in forms constitute the global environment for queries contained in the XML document. Query evaluation can be triggered by clicking on panels defined by <xql:eval-qc-on-click>, which references a query-construct-pair.

The following query portion demonstrates the use of this form-based user interface in YAXQL.

```
……
……
<xql:fill-form href=” “ select = “ “ var =”enzyme” >
 <xql:form-entry name =”enzyme-name”>
 <xql:value var =”enzyme”/>
 </xql:form-entry>
</xql:fill-form>
<xql:ask-for-variable-binding var =”enzyme”
 datatype=”string”
 maybe-null=”NO”
                                  title =”NEW FORM” />
<xql:eval-qc-on-click name=”edb:q2” href =”edb.xml” title
=”NEW FORM”/>
……
……
```

Summary of result of the comparative analysis of the three query languages: A total of (22) functionalities have been considered against each of the three query languages. All the example queries are not shown here. Only the unique features of the YAXQL have been

Table 1: Results of the three query languages against the twenty two query language functionalities

Query Funtionalities	XML-QL	XQuery	YAXQL
Selection and extraction	Y	Y	Y
Variable binding	N	N	Y
Element constructor	Y	Y	Y
Changing structure	Y	Y	Y
Ordering or Sorting	Y	Y	Y
Grouping	Y	Y(NE)	Y
Join	Y	Y	Y
Pattern matching	Y(NE)	N	Y
Universal and existential quantifier	N	Y	Y
Conditional expression	N	Y	Y
Tag variables	Y	N	Y(NE)
Operators	N	Y	Y
Aggregate functions	N	Y	Y
User defined functions	N	Y	Y
User defined predicates	N	N	Y
Integrating queries into documents	Y	N	Y
Embedded SQL queries	N	N	Y
Query Identification, Referenciation and Reuse	N	N	Y
Unbound parameters at query evaluation time	N	N	Y
Recursive query and dealing with IDREFS list	N	N	Y
User interaction using form based interfaces	N	N	Y
User defined data types and operations on data types	N	Y	N

mentioned. Table 1 summarizes the results found after examining each of the three query languages against the twenty two query language functionalities.

Table 1 represents the three query language functionalities. Y stands for YES meaning that the particular functionality is available in that language. N stands for NO meaning that the particular functionality is not available in that language. NE stands for NOT EXPLICIT meaning that the particular functionality might be available but not explicit in that language.

RECOMMENDATIONS FOR A NEW QUERY LANGUAGE FOR XML

Almost all the promising query languages for XML have been examined. From the very early concept of Object Query Language (OQL) upto the very strong query languages like QUILT, YAXQL and XQuery we have gone through all the functionalities the different query languages offer. It is seen that most of the languages are very good at some particular field of applications, which they were really intended for, but are not so good for other purposes. Recently, QUILT, YAXQL and XQuery demand themselves as universal query languages. They are very good query languages. But some lack of functionalities exist in all of these languages and hence the requirements for a new, universal query language have been reviewed, which will be equally powerful, versatile, user friendly and web-aware. The followings are the requirements we think that should be met by a new query language for the language to be equally powerful, versatile, user friendly and web-aware.

Query syntax and the result of the query should be in XML syntax. Multiple syntax bindings may be welcomed: An XML query language should be representable in XML itself so that no other special mechanisms will be needed to manipulate the queries. The result of the query should also be in XML so that the user can find the view of the usable format of the result.

According to the W3C requirements, that the XML Query Language may have more than one syntax binding provided that the transformation from one syntax to another does not deteriorate the benefit of having different syntax of the query. Also in that case, the way to switch from one syntax to another should be clearly defined with the possible cost-optimized transformation techniques.

Support for basic and important query functionalities: Query language should support the important and necessary functionalities like support for operation on all datatypes defined by the data model, sorting, user-defined functions, structural and hierarchical preservation and transformation of the document, joining, aggregation, universal and existential quantification, indexing, operations on names, literals and schemas etc. These functionalities are necessary for manipulating the document according the user's need.

Support for explicit type casting with clear semantics: Type casting is a very important feature for a query language. The language should be able to caste types from one to another for being able to handle different types of data in different situations and for different purposes.

In performing the type casting, the query language should define the semantics of the type casting very clearly. The castings should be logically correct so that these are easily understandable by the users. Type casting should not make the query complicated, if not easier.

Distribution of queries with bound variables to different sites: XML query language should not be dependent only on the resources in its creation context for evaluating the queries. It should have the ability to remotely execute the queries with bound variables without needing to communicate with its point of origin. This feature implies the distribution of queries into different websites where the queries will be partially evaluated, freeing the local query processor from doing this extra work. After doing the partial evaluation of the different query portion, they

will be collected to the original machine where the query was intended to be operated and the final result will be produced there.

Pattern matching: Query language should be able to be used against an XML source with limited knowledge of its documents' precise structure. This eventually implies pattern matching. This is a very powerful technique for querying against a source without knowing much about it.

Should be web-aware: Query language should be web-aware. This has several implications. First, it should allow to query documents which are not only semistructured data, but also any documents contained anywhere on the web. Also, if data services are provided by a form-based interface, the query language must support querying these interfaces also.

Declarative and sufficiently expressive: The query language should be a declarative language so that several different strategies can be followed for evaluating a query. In addition it should be sufficiently expressive. It implies supports of some functionality like joins, nested queries, recursive queries, full-text search etc. It also includes query support for all XML features.

Provide support for maximal reuse of queries: Query language should have support for reusing some interesting and valuable queries. It has several implications. Identification and accessing of queries to maximize possible reuse of queries are two core implications. Reuses of queries are important for reducing the cost of evaluating the same query again and again in the same or different site.

Allow queries to return not only values but also references: This is a pre-requisite to the query reuse. For allowing the queries to be reused, they should be identifiable and the result should be able to be sent to another identification number of queries so that it can be reused there or anywhere else. Query language should be able to produce the variable bindings bound to some other pointers and then in some other queries it should be able to return the reference to the previous query result.

Support for defining own datatypes and operation on those datatypes: It has a very good impact on making the queries interesting, easier and more useful. If somebody can define their own datatypes basing on some standards, they will be able to manipulate those data according to their needs and so queries can be made using those defined datatypes in a very easier way than it can be done from other primitive datatypes.

Selecting the most efficient way of querying: This is a very much important issue that has been discussed by the researchers in the database field. It is always possible to make some query constructs to query for some special things with even some amazing specialities than the other competitive languages. But the complexity of computation and the cost of query should also be taken into account so that some special but costlier queries will not reduce the performance of the query processor. This is even more important in querying from very large and different websites at the same time.

Based and related to other XML standards: XML query language should be compatible with all other XML standards like XPointer, XSLT etc. so that any changes in the current standards can be accompanied with a change in the query language also.

CONCLUSIONS AND FUTURE WORKS

XML has been accepted as the next format for data representation on the web. A suitable query language for XML data has thus become necessary for retrieving and interchanging electronic data on the web. The present status of the XML query language is really promising, even within this short interval from the development of the XML technology. It has been tried to concentrate on the functionalities of the different query languages, which have been claimed to be very much powerful for different types of data sources. Although some of the query languages developed until now are very much useful for several types of data sources, they are not completely universal to all data sources. That is what inspired us to review the query languages individually. Detailed comparison of three very good query languages have been made in the light of some very common and special requirements. It is consider as an evaluation of the present query languages. Then we made an analytical translation with examples between two selected languages to make it believe that both of these languages can be used for some common functionality. Then we evaluated the developments of the XML query languages upto the present status. At last, we described our point of view about the power and functionalities of the present query languages and recommended a list of requirements for a complete query language, which will really become a universal query language.

The future extension to this study can be developing the constructs of the new query language, which will

fulfill all recommended requirements. The future research can also review the semantics of the constructs for the new query language.

If studies go on, XML query language will get its final, full-fledged appearance and electronic data transaction on the web will be more flexible in the very near future.

REFERENCES

1. Yergeau, F., T. Bray, J. Paoli, C.M. Sperberg-McQueen and E. Maler, 2004. Extensible Markup Language (XML) 1.0, 3rd Edn., W3C Recommendation. 4th February
2. Abiteboul, S., 1997. Querying semi-structured data. Proceeding of International Conference on Database Theory, Deplhi, Greece, Springer-Verlag, pp: 1-18.
3. Deutsch, A., F.F. Mary, D. Florescu, A.Y Levy and D. Suciu, 1998. XML-QL: A query language for XML. Proceeding of Query Languages Workshop, Boston, Massachussets.
4. Boag, S., D. Chamberlin, M.F. Fernandez, D. Florescu, J. Robie and J. Simeon, 2003. XQuery 1.0: An XML query language. Technical Report W3C. 12th November.
5. Moerkotte, G., 2000. YAXQL: A powerful and web-aware query language supporting query reuse. Technical Report, University of Mannheim, Germany.
6. Abiteboul, S., D. Quass, J. McHugh, J. Widom and J. Wiener, 1997. The Lorel Query Language for Semistructured Data. Intl. J. Digital Libraries, 1: 68-88.
7. Bonifati, A. and S. Ceri, 2000. Comparative analysis of five XML Query languages. SIGMOD Record, 29: 68-79.
8. Robie, J., D. Chamberlin and D. Florescu, 2000. Quilt: An XML query language for heterogeneous data sources. 3rd International Workshop on the Web and Databases, Dallas, Texas. May 18-19.
9. Robie, J., J. Lapp and D. Schach, 1998. XML Query Language (XQL). Proceeding of Query Languages Workshop, Boston, Massachussets.
10. McHugh, J. and J. Widom, 1999. Query optimization for XML. In: Proceeding of VLDB, Edinburgh, UK., September 1999.
11. Christophides, V., S. Cluet and G. Moerkotte, 1996. Evaluating Queries with Generalized Path Expressions. Proceeding of ACM-SIGMOD International Conference Management Data.
12. Buneman, P.S., S. Davidson, M. Fernandez and D. Suciu, 1997. Adding structure to unstructured data. Proceeding of International Conference Database Theory, Deplhi, Greece, Springer Verlag, pp: 336-350.