International Journal on Integrated Information Management
Vol.01 (2012) DOI: 10.15556/IJIIM.01.02.001

The design of kinisis

Euclid Keramopoulos[a], Konstantinos Tsekos[a], Achilleas Pliakas[a], Ignatios Deligiannis[a]

[a] *Department of Information Technology, Alexander Technology Educational Institute, Thessaloniki, Greece*

Abstract: XQuery is the standard query language for semistructured data and especially for XML documents. Based on XQuery, we designed and developed KINISIS, a graphical XQuery language. In KINISIS we use metaphors, extracted from the road traffic act, in order to define queries. In this paper we present the design of KINISIS, the metaphors used, the implementation of KINISIS and the results of a controlled experiment where we assess the usability of KINISIS against XQuery.

Keywords: Semistructured Data, Graphical XQue

1. Introduction

XML (Extensible Markup Language) (XML, XML 1.0 Recommendation 2008 fifth edition) is a markup language developed by the World Wide Web Consortium (W3C) to deliver structured content over the web and to provide a competitive way of storing data. XML query languages were mainly designed as a solution for retrieving information from XML documents. Traditional SQL applications can evolve to deal with XML data using extensions for XML to construct XML data from relational data, as well as store, query, and retrieve XML data. These extensions form the SQL/XML (Beza et al., 2007, Funderburk et. Al., 2002) and were mostly used in DBMSs; however there was a need for purely expression languages to perform this kind of applications created that led to XML Query languages. More than twenty different query languages for XML data were introduced. In (Bekiropoulos et al. 2010) an analytical review is presented for XML Query languages.

On the other hand, the majority of computer users need only to learn how to complete simple work tasks, whereas the problems they have to solve are usually expressed in non-computing terms. Nowadays, the main type of user has changed from the skilled professional to the computer literate (unskilled or naive) user, and thus the user interface has to be simpler and friendlier. The initiation of graphical user interfaces, which utilize users cognitive skills and harness, both advances in graphics technology and increased computing power, simplified and improved the way users interface with computers and made computer systems accessibly to an even larger number of users. Currently, graphical user interfaces have become an essential part of any computer system and system designers have come to accept that in order to improve users' productivity, it is essential for a user interface to address users' skills (Dix et. al., 2004).

Thus, a number of graphical interfaces for XML query languages were developed. XML-GL (Ceri et. al., 1999, Erwig, 2000) is a graphical query language that depicts documents and their related Document Type Definition (DTDs) with the use of graphical representations. Recently, a graphical query language called XQuery By Example (XQBE) (Braga et. al., 2005) was created, based on XML-GL. In fact, XQBE implements XQuery 1.0 queries (XQuery 1.0, 2007) with graphical representations. Xing (Erwig, 2003) is a graphical query language as well as visXcerpt (Fuhr and Grojohann, 2001, Bry and Berger, 2003, Berger et. al., 2004), which is an extension of Xcerpt (Bry and Berger, 2003, Berger et. al., 2004). In Bekiropoulos et. al., (2010), an analytical review of graphical interfaces for XML query languages is presented. Moreover, a list of features is introduced that a graphical XQuery language should support as a result of the review.

The structure of this paper is as follows. In section 2, we present KINISIS design and also we introduce analytically all the metaphors we used in order to represent the basic features of FLOWR XQuery expression. In section 3, we briefly analyze the implementation of KINISIS by presenting some examples, and in section 4, we present the experimental evaluation of KINISIS usability vs XQuery. Finally, in section V, we draw our conclusion and our future plans regarding KINISIS.

2. **Related Work**

KINISIS is a graphical query language which is designed on top of XQuery (Pliakas & Tsekos, 2010). It supports all the XQuery features and it is represented graphically by a set of "road traffic act" metaphors. XQuery is based on XPath and it "looks for" the data following a path. In a similar philosophy, we used metaphors in order to design a graphical XQuery by drawing the path to the requested information, through the respecting road traffic act signs. The metaphors we used come from road traffic. We chose this subject because the rules of the road traffic are well known so we can combine for example a sign of road traffic with a term of XQuery. Road signs are standard since 1968 when the European countries signed the Vienna Convention on Road Traffic treaty (Vienna Convention on Road Signs and Signals), aiming at standardizing traffic regulations in participating countries in order to facilitate international road traffic and to increase road safety. Part of the treaty was the Vienna Convention on Road Signs and Signals, which defined the traffic signs and signals. As a result, in Western Europe the traffic signs are well standardized, although there are still some country-specific exceptions, mostly dating from the pre-1968 era.

2.1. KINISIS Metaphors

In this section, we present the metaphors we used for the basic features of XQuery FLWOR expression.

2.1.1. For

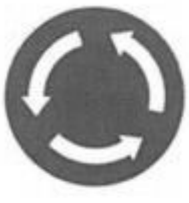


Figure 1. The “for” metaphor

A ‘for’ clause sets up an iteration that allows the rest of the FLWOR to be evaluated multiple times, once for each item in the sequence returned by the expression after the in keyword. Like the node above, we meet in a road, which oblige us to move in a circular way.

2.1.2. Let

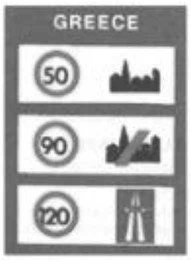


Figure 2. The “let” metaphor

A ‘let’ clause is a convenient way to bind a variable to a value. Unlike the for clause, a ‘let’ clause does not result in iteration, it binds the whole sequence to the variable rather than binding each item in turn. Like this sign which sets up the speed limit according to the travelling area.

2.1.3. Where

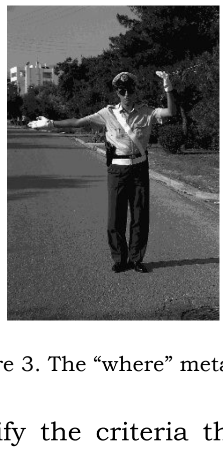


Figure 3. The “where” metaphor

The ‘where’ clause used to specify the criteria that filter the results of a FLWOR expression. The where clause can reference variables that were bound by a ‘for’ or ‘let’ clause. Like traffic policeman who controls the traffic and decides which cars can continue their way.

2.1.4. Return

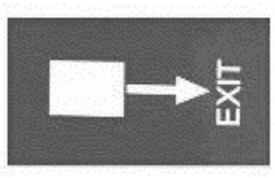


Figure 4. The “return” metaphor

The ‘return’ clause consists of the return keyword followed by the single expression that is to be returned. It is evaluated once for each iteration, assuming that the where expression evaluated to true. The result value of the entire FLWOR is a sequence of items returned by each evaluation of the return clause. Like the exit sign of a motorway which leads to our destination.

2.1.5. Order By

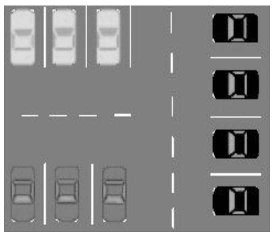


Figure 5. The “order by” metaphor

The ‘order by’ clause is used to sort the results and is made up of one or more ordering specifications, separated by commas, each of which consists of an expression and an optional modifier. Like a parking, which has the parked cars sorted according the color of the car, as we can see in this sign.

2.1.6. Count

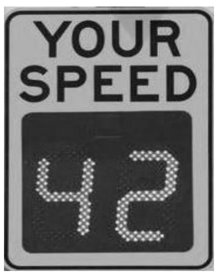


Figure 6. The “count” metaphor

This function is used to determine the number of items in the sequence, so this sign counts the current speed of a car, which is travelling in a motorway.

2.1.7. Min

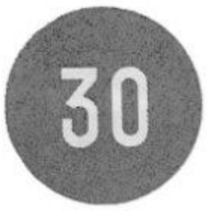


Figure 7. The “min” metaphor

This function is used to determine the minimum value of the items in the sequence, as this sign obligates the drivers to travel with the minimum speed of 30km/h.

2.1.8. Max

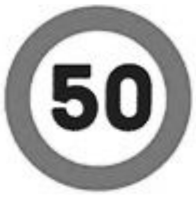


Figure 8. The “max” metaphor

This function is used to determine the maximum value of the items in the sequence, as this sign obligates the drivers to travel with the maximum speed of 50km/h.

2.1.9. Avg

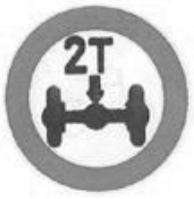


Figure 9. The “avg” metaphor

This function is used to determine the average value of the items in a sequence, like this sign, which inform us for the average weight, which should have per axis every truck.

2.1.10. Sum

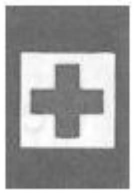


Figure 10. The “sum” metaphor

This function is used to determine the total value of the items in a sequence. The symbol of the summation is “+” so we chose this sign that shows us the “+” symbol.

2.1.11. And

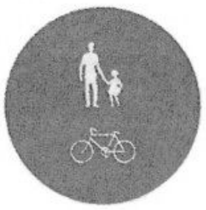


Figure 11. The boolean “and” metaphor

A conditional expression, which constructed with the operator “and” like “boolean1 and boolean2” returns true if both of its operands are true. Like in this sign allows only pedestrians and cyclists to pass across.

2.1.12. Or

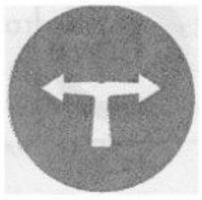


Figure 12. The boolean “or” metaphor

A conditional expression, which constructed with the operator “or” returns true if one or both of its operands is true. Like in this sign we can choose one of the two ways (left or right route).

2.1.13. If/Else

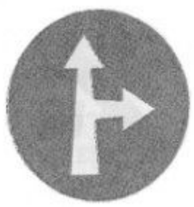


Figure 13. The “if” metaphor

The expression after the “if” keyword is known as the test expression. It must be enclosed in parentheses. If the test expression evaluates to true, the value of the entire conditional expression is the value of the “then” expression. Otherwise, it is the value of the “else” expression. Like in this sign we can follow the entire route or to go from an alternative route.

3. **Implementation**

We have completed the implementation of KINISIS as an end-tool using Java and Java Swing API. The query construction using KINISIS is supported by a Graphical User Interface that has a user-friendly approach through drag and drop mechanism. We designed the GUI in order the users to work on an XQuery Flower philosophy, with the difference that we replaced all the tricky syntax of XQuery by metaphors that the user can choose from toolboxes. The results of the formal experiment that we have conducted in order to evaluate our GUI (section 4) was quite satisfactory.

Moreover, we have developed a compiler, which transforms the graphs into equivalent XQuery structures and send them to the underlying IBM DB2, in order to execute the XQuery query. In order to present the function of KINISIS the following three example queries are used.

3.1. Examples of KINISIS

The example queries of KINISIS based on the following running example which has been taken from (Walmsley, 2007) and includes two XML files. The "catalog.xml" is a product catalog containing product details for every department of the catalog. The "prices.xml" keeps all the prices of the products respecting particular dates.

catalog.xml

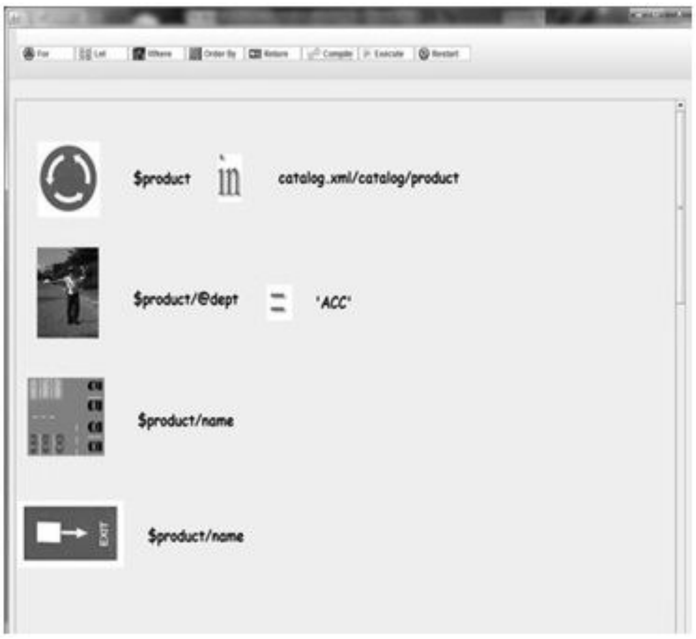
```
<catalog>
 <product dept="WMN">
  <number>557</number>
  <name language="en">Fleece Pullover</name>
  <colorChoices>navy black</colorChoices>
 </product>
 <product dept="ACC">
  <number>563</number>
  <name language="en">Floppy Sun Hat</name>
 </product>
 <product dept="ACC">
  <number>443</number>
  <name language="en">Deluxe Travel Bag</name>
 </product>
 <product dept="MEN">
  <number>784</number>
  <name language="en">Cotton Dress Shirt</name>
  <colorChoices>white gray</colorChoices>
  <desc>Our <i>favorite</i> shirt!</desc>
 </product>
</catalog>
```

prices.xml

```
<prices>
 <priceList effDate="2006-11-15">
  <prod num="557">
   <price currency="USD">29.99</price>
   <discount type="CLR">10.00</discount>
  </prod>
 <prod num="563">
  <price currency="USD">69.99</price>
 </prod>
 <prod num="443">
  <price currency="USD">39.99</price>
  <discount type="CLR">3.99</discount>
 </prod>
</priceList>
</prices>
```

3.1.1. Example 1.

This query finds all the product names that can be found in Accessory (ACC) departments. The results are being sorted by the product name.



$product in catalog.xml/catalog/product

$product/@dept = 'ACC'

$product/name

$product/name

Figure 14. Example1 in KINISIS

The graphical query in fig. 1 corresponds to the following query:

```
for $product in doc("catalog.xml")/catalog/product
where $product/@dept='ACC'
order by $product/name
return $product/name
```

The first example query is developed in four steps. Every step is represented by a metaphor:

1. At the beginning the user works on the 'for' metaphor and s/he choose the xml file that the query is based on (catalog.xml).
2. Then, the user apply the query 'filters' by applying the predicates as $product/@dept='ACC', in the WHERE metaphors.
3. The next step is to apply the order criteria in the corresponding metaphor.
4. Finally, the projection of the query is presented in the RETURN metaphor.

3.1.2. Example 2.

This query presents the product number with its price. The two nodes are in different files, i.e. the product/number is in "catalog.xml" and the price is in "prices.xml". Thus, in order to get the correct output the user has to join the two xml files on the clause: "$product/number = $price/@num"

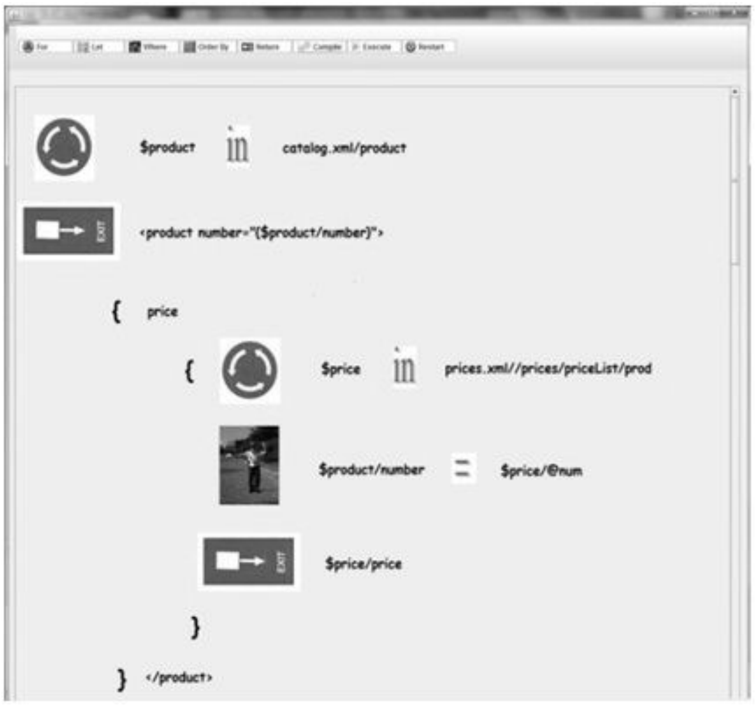


Figure 15. Example2 in KINISIS

The graphical query in fig. 2 represents the following query:

```
for $product in doc("catalog.xml")//product
return <product number="{$product/number}">
{ attribute price
 { for $price in doc("prices.xml")//prices/priceList/prod
  where $product/number = $price/@num
  return $price/price}
} </product>
```

In KINISIS, in order to represent the join operation, the XQUERY structure is followed by using two FLWORs, one embedded in the return clause of the other. The outer FLWOR returns the list of products, regardless of the availability of price information. The inner FLWOR selects the price, if it is available.

3.1.3. Example 3.

The query returns the average discount of all products

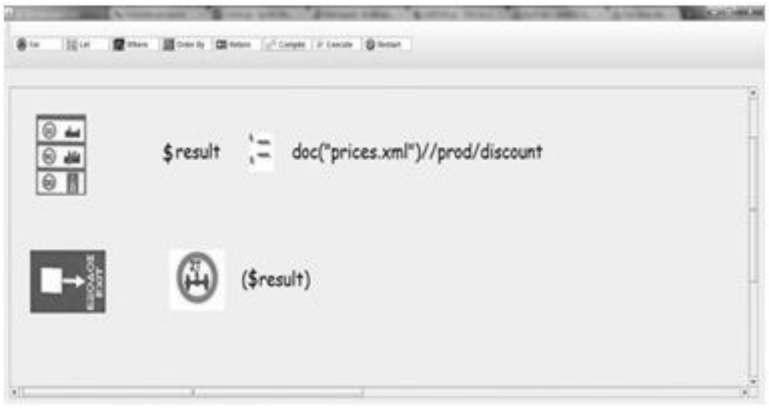


Figure 16. Example3 in KINISIS

The graphical query in fig. 3 represents the following query:

```
let $result := doc("prices.xml")//prod/discount
return avg($result)
```

It is often desirable to perform calculations on the groups. For example, suppose that a user wants to know the sum or the average of the quantities for a department. This type of aggregation can be performed using the aggregation functions and KINISIS support all the aggregate function that XQuery supports.

4. Experimental Evaluation of KINISIS usability

We carried out a controlled experiment in order to evaluate the metaphors that we use in KINISIS and also the language itself. Twenty two undergraduate students of the Department of Information Technology at Alexander Technology Educational Institute were participated to compose a few queries using both IBM DB2 and the KINISIS.

The students found out that the KINISIS application is friendlier to use since they have to draw the query instead of the classic way of IBM DB2. Due to some automated parts of KINISIS application, it was easier for the students to compose the queries avoiding mistakes as wrong path expressions or other spelling mistakes. This gave the ability to the students to answer in more complex queries. As a result, there were more correct answers using KINISIS application (80.91%, 89 correct answers from 110 queries in total) than IBM DB2 (51.82% correct answers). Moreover, the time needed from students to compose the queries was much less when they use KINISIS than IBM DB2.

Finally, students filled questionnaires shown that the metaphors of traffic signs were suitable to make them understand the construction of XQuery language.

5. Conclusions

In this paper we presented the design, implementation and evaluation of KINISIS, a new graphical query language for XML documents supporting the XQuery FLOWR structure. KINISIS uses the road traffic act metaphor in order to represent graphically the query structure in a common picture as the one when we drive our vehicle following the traffic signs in order to arrive at our destination. KINISIS addressed to naive XQuery users. In our future plans is to alter the design of KINISIS in order to follow a different design philosophy than XQuery FLOWR.

Acknowledgments

The work presented in this paper has been supported by the Research Committee of Alexander Technology Educational Institute of Thessaloniki under the Research Support Program 2009 (Π.Ε.Ε. 2009).

References

[1] Bekiropoulos K., Keramopoulos E., Beza O., Mouratidis P. (2010). "A list of features that a graphical XML Query language should support". International Journal of Computer Systems Science and Engineering (IJCSSE), 25(5).

[2] Berger S, Bry F, Bolzer O, Furche T, Wieser C., (2004). "Xcerpt and visXcerpt: Twin query languages for the Semantic". Proc of Web. 3rd International Semantic Web Conference (ISWC2004), Hiroshima, Japan.

[3] Beza O, Patsala M. & Keramopoulos E., (2007). "Comparison of XML Support in IBM DB2, MICROSOFT SQL SERVER and ORACLE". Proc 2nd International

Scientific Conference, eRA-2: The Contribution of Information Technology to Science, Economy, Society and Education, Athens, Greece, 22-23 September.

[4] Braga D., Campi A. and Ceri S., (2005). "XQBE (XQuery By Example): a visual interface to the standard XML query language". ACM Transactions on Database Systems, 30(2), 398 – 443.

[5] Bry F. and Berger S., (2003). "Xcerpt and visXcerpt: From Pattern-Based to Visual Querying of XML and Semistructured Data". Proc of 29th International Conference on Very Large Databases, 1053 – 1056.

[6] Ceri S., Comai S., Damiani E., Fraternali P., et al., (1999). "XML-GL: a Graphical Language for Querying and Restructuring XML Documents". Proc 8th International World Wide Web Conference, 151-165.

[7] Dix A, Finlay J, Abowd G, Beale R (2004). Human Computer Interaction, 3nd Edition. Europe: Prentice Hall.

[8] Erwig M., (2000). A Visual Language for XML. Proc of IEEE International Symposium on Visual Languages, 47-54.

[9] Erwig M., (2003). "Xing a visual XML query language". Visual Languages and Computing, 14(1), 5–45.

[10] Extensible Markup Language (XML) 1.0 (Fifth Edition), W3C Recommendation, (26 November 2008). Available through the internet: http://www.w3.org/TR/xml/ [accessed: 21/11/2011]

[11] Fuhr N. and Grojohann K., (2001). "XIRQL: A Query Language for Information Retrieval in XML Documents". Proc of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New Orleans, 172-180.

[12] Funderburk J. E., Malaika S., Reinwald B., (2002). "XML programming with SQL/XML and XQuery", IBM SYSTEMS Journal, 41(4), 642-665.

[13] Pliakas A. & Tsekos K. (2010). Υλοποίηση εργαλείου γραφικής απεικόνισης της XQuery (The development of a graphical XQuery tool), Final year project in Greek, Department of Information Technology, Alexander Technological Educational Institute of Thessaloniki, Greece.

[14] Priscilla Walmsley, (2007). XQuery. O'Reilly.

[15] Vienna Convention on Road Signs and Signals. Available through the internet: http://www.irrationalsigns.com/history-of-road-signs.htm [accessed: 21/11/2011]

[16] World Wide Web Consortium (W3C). Available through the internet: http://www.w3.org/ [accessed: 21/11/2011]

[17] XQuery 1.0: An XML Query Language (Second Edition), W3C Recommendation 14 December 2010. Available through the internet: http://www.w3.org/TR/xquery/ [accessed: 21/11/2011]