CRITICAL ANALYSIS OF DESIGNING A GRAPHICAL QUERY LANGUAGE

E. Keramopoulos[1], D. Dervos[1], I. Deligiannis[1], G. Evangelidis[2], P.Pouyioutas[3]
[1] Alexander Technological Educational Institute of Thessaloniki (GREECE)
[2] University of Macedonia (GREECE)
[3] University of Nicosia (CYPRUS)
euclid@it.teithe.gr, dad@it.teithe.gr, ignatios@it.teithe.gr, gevan@uom.gr, pouyioutas.p@unic.ac.cy

Abstract

During the last two decades, information has been stored in a plethora of different databases, locations and data models. In many cases, companies and organizations use data, that is stored in a variety of data models and remote database servers, which can be correlated. A challenge would be to define queries joining data from all those different sources. Thus, data can be found in Relational, Object-Relational, Object-Oriented, XML databases or even in text files. For each model there is a corresponding query language, i.e. SQL, SQL3, OQL and XQuery. Moreover, it is very tricky to join data from different sources and we have not found in the literature a query language that has been designed to use data in the same query from Relational, Object-Relational, Object-Oriented and XML databases.

On the other hand, the syntax of all the above mentioned query languages, is complicated and demanding especially for novice users. 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. For this reason users prefer to use graphical query languages instead of a query language that has a textual syntax.

In this paper, we present our conclusions in designing graphical query languages (GQLs). We have already designed, developed and evaluated, by organizing a controlled experiment, two GQLs, namely GOQL for Object-Oriented Database System and KINISIS for XML databases. The most important lesson that we learned is that it is very important the design of a graphical query language to be “free” of the underlying query language. Finally, we introduce a solution to the problem of joining data from different sources which stored in different database models, based on a graphical query language.

Keywords: Graphical Query Languages, Relational Database Model, Object-Relational Database Model, Object-Oriented Database Model, Semistructured Database Model, XQuery.

1 INTRODUCTION

Over the past two decades, the phenomenal growth of the Internet and the use of new Database Models resulted in the spreading of storing the information in a plethora of different databases, locations and data models. In many cases, companies and organizations use data, that stored in a variety of data models (Relational, Object-Relational, Object-Oriented and Semistructured) and remote database servers, which can be correlated. Moreover, a different corresponding query language has to be used in order to search and retrieve data from databases of different models, i.e. SQL, SQL3, OQL and XQuery. Also, it is very tricky to join data from different sources and we have not found in the literature a query language that has been designed to use data in the same query from Relational, Object-Relational, Object-Oriented and XML databases.

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 novice) 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 [1]. In the database community the importance of graphical representation to conceptual design was recognized early and research into this field led to the

development of a number of semantic models such as the Entity Relationship Model [2]. Recently, there was considerable interest in the development of Graphical Query Languages (GQLs) [3-19]. GQLs are graphical user interfaces that allow users to query (retrieve, create, delete and update) their underlying database, i.e. GQLs are special purpose graphical user interfaces.

In this paper, in section 2 we discuss the principles and characteristics of Graphical Query Languages. In section 3 we present the basic characteristics of the two GQLs that we designed, implemented and evaluated, namely GOQL [20] for the ODMG 3.0 data model [21] and KINISIS [22] for the XML [23]. In Section 4, we introduce lessons learned from the process of designing a graphical query language. Finally, the paper concludes by discussing our current and future work.

2 GRAPHICAL QUERY LANGUAGE CHARACTERISTICS

In this section, we present briefly the features of an analysis methodology that is employed to evaluate/characterize a graphical query language. Our analysis methodology has adopted some characteristics from the analysis methodology on query languages proposed in [24] and from the surveys on graphical query languages on databases of [25, 26]. The analysis includes amongst others the languages of AMAZE [5], G-Log [9], GOMI [4], Kaleidoquery [7], PICASSO [10], Pasta-3 [11], QBD* [3], SUPER [12], Gql [8], OdeView [13], QUIVER [6], XML-GL [14], 'XQuery By Example' (XQBE) [15], Xing [16], visXcerpt [17] and Xcerpt [18].

In particular, the following features comprise our analysis methodology:

- The level of **expertise that users** of a GQL should have.
- The **underlying data model** supported by the graphical query language because it is the underlying data model that determines the querying mechanism.
- The way a **graphical scheme** is utilized in formulating a query.
- The **design characteristics**, which include functions that are supported by the query system, a language form and the expressivity of the language.
- The existence of a **formal definition**, i.e. whether a syntactic analysis, a semantic analysis and a proof of the expressive power have been proposed.
- The platform(s) used for the **implementation** of the language.

A detailed analysis of our research work presenting the support of the above features by the most representative GQLs and a comparison analysis of those GQLs based on the proposed methodology can be found in [25, 26].

3 THE DEVELOPMENT OF TWO GQLS

In this section we present briefly the two GQLs that we have designed, implemented and evaluated as a result of our research.

3.1 GOQL

GOQL [20] is a graphical query language for object-oriented database systems, fully compliant with the Object Database Management Group's standard (O.D.M.G. 3.0) OQL [21]. It provides a graphical user interface suitable for the novice users, by using desktop metaphors for the graphical illustration of the object-oriented features of the underlying database model. GOQL and User's View which is the graphical representation of the database schema, implemented using Tcl/Tk [27, 28] and o2 Object Oriented Database Management System [29] as the underlying database system.

In GOQL, the procedure for creating a query is straight forward and easy. The language provides the user with a copy of the User View (Fig. 1) [30]. The construction of a query starts as soon as the user selects an object of a class. Following this, GOQL opens a query window which is comprised of two parts, namely a query canvas and a toolbox (Fig. 2). The query canvas is the area where users can construct their queries. It contains all the object(s) that a user selects from the User View, whilst the toolbox is a facility that contains icons for the various tools that GOQL provides for the construction of a query; the tools of the toolbox operate on the objects that the query window contains. In Fig. 2 is represented a GOQL query that projects the *email* of those *male Authors* whose *age* is not greater than 40 and their *name* start from 'A'.

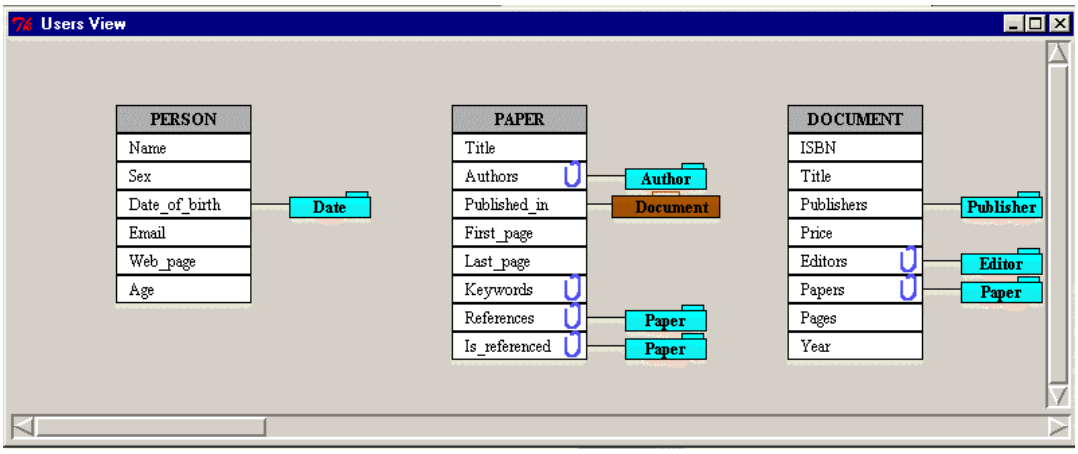

Figure 1: The Users' View window

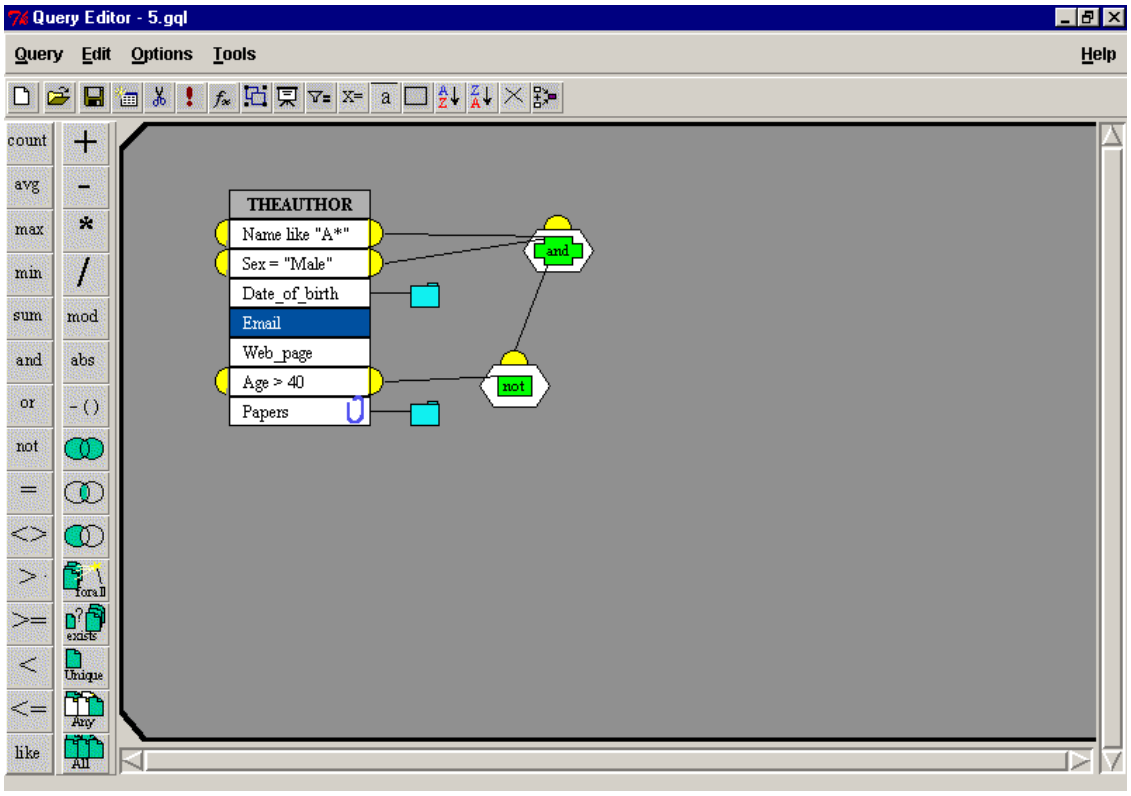

Figure 2: An example query in GOQL.

We also evaluated the design of the language through a controlled experiment [31]. In GOQL we used color in order to emphasize or differentiate important elements of a query such as operators, conditions, subqueries and complex types. The result of the experiment showed that the use of color improved the user's understandability of GOQL by 10%. Another hypothesis that we examined in that experiment was that "the use of non-symbolic names improves the understandability of a graphical query language". We evaluated that hypothesis on (a) the property names (attributes and methods) of the underlying database model (for example we check "Date of Birth" and "DOB") and (b) the operator names that a graphical query languages uses (for example we check the boolean operators with their names 'and', 'or', and 'not' and with corresponding well-known mathematical symbols $\wedge$, $\vee$, and $\neg$). The result of the experiment showed that (a) the use of meaningful names, for the representation of the properties of the underlying database model, instead of abbreviations did not make any difference to the understandability of a graphical query language and (b) the use of non-symbolic names for the representation of the operators improved the GOQL understability by 15%.

3.2 KINISIS

Kinisis is a graphical query language which is designed on top of XQuery as an end-tool using Java and Java Swing API and having as an underlying database management system IBM DB2, in order to execute the “produced” XQuery queries. The query construction using KINISIS is supported by a Graphical User Interface that has a user-friendly approach through a 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 a set of “road traffic act” metaphors that the user can choose from toolboxes. The KINISIS metaphors are analyzed in [22]. In fig. 3, an example of KINISIS is represented. This query finds all the *product names* that can be found in ‘Accessory’ (ACC) *departments*. The results are being sorted by the *product name*.

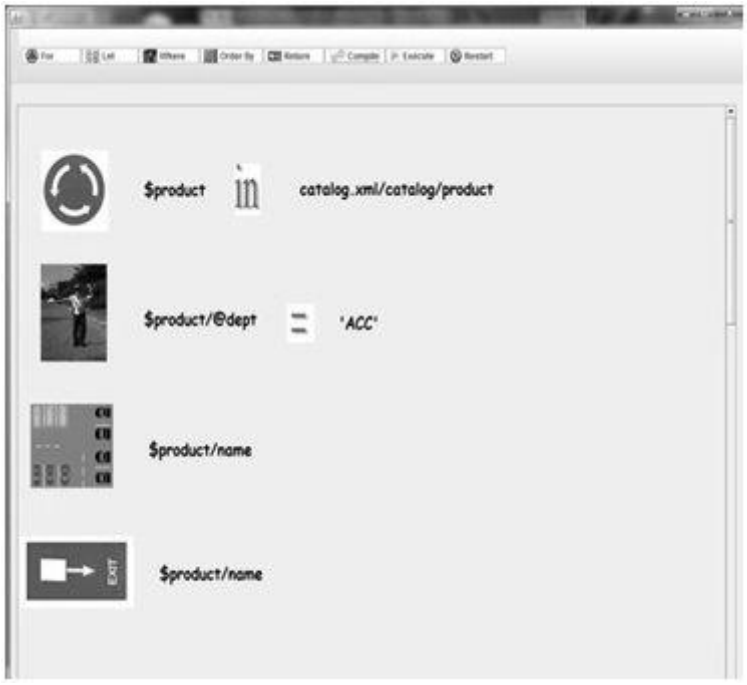

Figure 3: An example in KINISIS

The graphical query in fig. 3 corresponds to the following query:

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

Moreover, we conducted a controlled experiment study (the controlled experiment is analyzed in [32].) in order to compare the quality factor understandability between KINISIS and XQuery. The results of the experiment showed that KINISIS GUI improves user efficiency against XQuery textual syntax. The first result of experiment was that the subjects answered significant more correct answers using KINISIS application (80.91% correct answers) than IBM DB2 (51.82% correct answers). The subjects found out that the KINISIS application is friendlier to use since they had to draw the query instead of the classic textual way of IBM DB2. Due to some automated parts of KINISIS application, it was easier for the subjects to compose the queries, avoiding mistakes, as wrong path expressions, or other spelling mistakes. This gave the ability to the subjects to answer correct in more complex queries.

On the other hand, although the subjects found that the KINISIS interface is friendlier than the textual interface of IBM DB2 (which is the W3C XQuery syntax), they felt less confident with it. One explanation is that the subjects were not familiar enough with KINISIS interface, because it was the first time that they used it and thus they found that 68,2% of the KINISIS functions are non-predictable. On the other hand, the subjects graded the KINISIS GUI and interaction more than 60% successfully. This is a good result for KINISIS interface but it was not as good as we expected. This shows that KINISIS graphical query construction and representation improves the user efficiency but the user interface needs improvement. The main reason for that we believe that is because the KINISIS query construction in based on XQuery Flower philosophy. A user has to have a good knowledge of XQuery in order to use KINISIS.

4 DESIGN LESSONS LEARNED

Our research on GQL has two targets. The first one was to improve the comprehension of a textual query language (eg. OQL, XQuery, etc) for the novice users by designing a GQL with the same expressive power. The second one was to improve the performance of novice and expert users by using a GQL instead of a textual equivalent query language. We conducted two successful controlled experiments in order to investigate the above two targets regarding the two graphical query languages that we developed, i.e. GOQL and KINISIS.

The results of this research, regarding the design of a new graphical query language, are clear and consist of the following four features:

- The use of color is essential for a GQL despite a number of problems related to its use such as that color aesthetics are subjective, and that some people are color blind. In a GQL the color can be used in order to differentiate and emphasize an important element of the query.
- The use of non-symbolic names for operators is mandatory for the design of a GQL for novice users. All the users are not familiar with mathematic symbols and because of that the use of special or mathematical symbols restrict the use of such a GQL to expert users and even to users that have a prior knowledge of mathematical symbols or the special symbols that used for that specific GQL.
- The use of metaphors can improve the use of a GQL if the set of metaphors (e.g. road traffic act metaphors that used in KINISIS) used is familiar to users.
- The design of a graphical query language must be “free” of the “equivalent” textual query construction mechanism. Thus, a GQL that designed for XML data has to be different from the query construction and syntax of XQuery. First of all in such a case, users have to have prior knowledge of the “equivalent” textual query construction and syntax; i.e. XQuery FLOWR syntax. Thus, all novice users excluded from this specific GQL usage. Moreover, the GQL “inherits” all the “handicaps” of the underlying query language. On the other hand a novice user is familiar with the navigation process of browser tools and we believe that a new graphical query language has to be designed having as starting point, of the query construction, a graphical database schema, which could be a graph and on that to be defined the appropriate query operators by the illustration of metaphors.

Finally, novice users could have the opportunity to define queries in order to retrieve information by joining data from different data sources based on different data models. This can be achieved by the use of a GQL based on XQuery. XML technology is a standard way of data interchanging on the Web and it can be used in order to transform a database of any data model into XML files. Thus, a GQL based on XQuery and on a “free” design philosophy it would ideal for a novice user and also it could be applied in many different databases from different data models.

5 CONCLUSIONS

In this paper, we addressed the principles and characteristics that a Graphical Query Language should support and introduced design features that must be used for the development of a new Graphical Query Language for all types of users; i.e. novice and expert users. In our present and future research work we focus on the design of a new graphical query language having as design philosophy that the graphical illustration of a graphical query will not be based on a one-to-one representation of the XQuery constructs, but in a “browsing” method of the underlying semi structured metadata and data.

ACKNOWLEDGEMENTS

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

[2] Chen P. P. (1976). The entity-relationship Model: toward a unified view of data. Journal of ACM Transactions on Database Systems, 1(1), pp. 166 – 192.

[3] Angelaccio M., Catarci T., Santucci G., (1990). QBD*: A Graphical Query Language with Recursion. IEEE Transaction on Software Engineering, 16(10), pp. 1150-1163.

[4] Jun Y.S., Yoo S.I., (1995). GOMI: A Graphical User Interface for Object-Oriented Databases. International Conference on Object-Oriented Interface Systems (OOIS), pp. 238-251.

[5] Boyle J., Leishman S., (1996). Gray M.D., From WIMPS to 3D: The Development of AMAZE. Journal of Visual Languages and Computing, 7(3), pp. 291-319.

[6] Chavda M., Wood P. T., (1997). Combining Constraints and Data-Flow in A Visual Query Language. IEEE Symposium on Visual Languages, Capri (Italy), pp. 125-126.

[7] Murray N., Paton N., Goble C.(1998). Kaleidoquery: A Visual Query Language for Object Databases. In the 4th IFIP Working Conference on Visual Database Systems - VDB 4, L'Aquila, Italy, pp. 247-257.

[8] Papantonakis A. (1995). Gql, a Declarative Graphical Query Language Based on the Functional Data Model. PhD Thesis, Birkbeck College, University of London.

[9] Paredaens J., Peelman P., Tanca L. (1995). G-Log: a graph-based query language. IEEE Transactions on Knowledge and Data Engineering, 7(3), pp. 436-453.

[10] Kim H., Korth H.F., Silverschatz A. (1988). PICASSO: A Graphical Query Language. Software-Practice and Experience, 18 (3), pp. 169-203.

[11] Kuntz M., Melchert R. (1989). Pasta-3's Graphical Query Language: Direct Manipulation, Co-operative Queries, Full Expressive Power. In the 15th International Conference on Very Large Databases, Amsterdam, pp. 97-105.

[12] Dennebouy Y., Andersson M., Auddino A., Dupont Y., Fontana E., Gentile M., Spaccapietra S., (1995). SUPER: Visual Interfaces for Object + Relationship Data Models. Visual Languages and Computing, 6(1), pp. 73-99.

[13] Dar S., Gehani N.H., Jagadish H.V., Srinivasan J. (1995). Queries in an Object-Oriented Graphical Interface. Visual Languages and Computing, 6(1), pp. 27-52.

[14] Ceri S., Comai S., Damiani E., Fraternali P., Paraboschi S., Tanca L. (2005). XML-GL: a Graphical Language for Querying and Restructuring XML Documents. In Proceedings of 8th International World Wide Web Conference, pp. 151-165.

[15] 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), pp. 398 – 443.

[16] Erwig M. (2003). Xing a visual XML query language. In Journal of Visual Languages and Computing, 14(1), pp. 5–45.

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

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

[19] Berger S, Bry F, Bolzer O, Furche T, Wieser C., (2004). 'Xcerpt and visXcerpt: Twin query languages for the Semantic'. Proceedings of Web. 3rd International Semantic Web Conference (ISWC2004), Hiroshima, Japan.

[20] Keramopoulos, E., Pouyioutas P. & Ptohos T. (2008). The GOQL Language and its Formal Specifications. In International Journal of Computer Science & Applications (IJCSA), 5(2), pp. 23-51.

[21] Cattell R.G.G., Barry D.K. (2000). The Object Database Standard: ODMG 3.0. Morgan Kaufmann Publishers.

[22] Keramopoulos E., Pliakas A, Tsekos K., Deligiannis I. (2011). KINISIS, a Graphical XQuery Language. Proceedings of International Conference on Integrated Information (IC-ININFO'2011), Kos, Greece.

[23] XML 1.0, 2008. Extensible Markup Language (XML) 1.0 (Fifth Edition), [online] Available at: http://www.w3.org/TR/REC-xml/ (Accessed at 8 May 2012).

[24] Catarci T., Costabile M.F., Levialdi S., Batini C. (1997). Visual Query Systems for Databases: A Survey. Visual Languages and Computing, 8 (2), pp. 215-260.

[25] Keramopoulos E., Pouyioutas P., Ptohos T. (2002). Comparison Analysis of Graphical Models of Object-Oriented Databases and the GOQL Model. In the 6th WSEAS International Conference on Computers, Rethymno, Crete Island, Greece.

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

[27] Ousterhout J.K. (1995). Tcl and the Tk Toolkit. Addison-Wesley Professional Computing Press.

[28] Johnson E.F. (1996). Graphical Applications with Tcl & Tk. M&T Books.

[29] o2 Technology (1995). o2 User Manuals.

[30] Keramopoulos E., Pouyioutas P. & Ptohos T. (2002). A Formal Definition of the Users View (UV) of the Graphical Object Query Language (GOQL). In Proceedings of the International IEEE Conference on Information Visualisation 2002 (IV'2002). London, England.

[31] Georgiadou E., Keramopoulos E., (2001). Measuring the Understandability of a Graphical Query Language through a Controlled Experiment. In the BCS International Conference of Software Quality Management, Loughborough, pp. 295-307.

[32] Keramopoulos E., Tsekos K., Pliakas A., Deligiannis I, (2012). Assessing the Understandability of Kinisis against XQuery through a Controlled Experiment. In IADIS International Conference Information Systems 2012 Berlin, Germany