IASTED International Conference on Databases Applications ~DBA 2004~ Innsbruck, Austria, February 17-19, 2004

STRUCTURAL KNOWLEDGE GRAPH NAVIGATOR FOR THE ICONS PROTOTYPE

Mariusz Trzaska (mtrzaska@pjwstk.edu.pl)[*], Kazimierz Subieta (subieta@pjwstk.edu.pl)[#*]
[#]Institute of Computer Science, Ordona 21, Warsaw, Poland
[*]Polish-Japanese Institute of IT, Koszykowa 86, Warsaw, Poland

Abstract

Structural Knowledge Graph Navigator (SKGN) supports end users of Web applications by simple means for ad hoc querying and browsing in an object-oriented database. The interface has been implemented within the European project ICONS. SKGN utilizes three core concepts: intentional navigation (in a schema graph), extensional navigation (in an object graph) and annotated user baskets for storing intermediate and final results of querying. SKGN is organized as an applet on the client side and a wrapper to a database on the server side. The paper describes the main ideas of SKGN.

Key Words

graphical query interface, information browsing, intentional navigation, extensional navigation

1. Introduction

Among many methods of end-user information retrieval in large databases special attention is devoted to graphical interfaces. Typical query languages, such as SQL and OQL, address the needs of programmers rather than end-users. The latter do not accept a keyboard-oriented query language with complex, recursive syntax, sophisticated semantics and complex rules of use. They expect mouse-oriented interfaces, with friendly, easy to use and self-explained graphical metaphors based on choices from menu, clicking, dragging and dropping, etc. Examples of end-user-oriented interfaces are Web search engines (such as Google), which perform retrieval of Web pages through words and phrases that the user types down to determine his/her interest. There are (quite sophisticated) automatic matching rules to determine the pages that are the most pertinent to such a query. The interface is very simple and straightforward for the average, non-professional user.

This type of interfaces, however, performs full-text retrieval within an unstructured text. Many Web applications are based on structured data, where information has the form of data structures with tags or attribute names bearing semantic meaning of stored values. Examples are XML/RDF repositories and relational, object-relational and object databases that are used to store structured information and to make it available for Web users. Usually data structures behind a given Web application are hidden for the users and used only by application programmers to generate HTML pages. In some cases, however, there is a need to support the user by a generic querying tool, which assumes that the user is aware of the logical organization (i.e. a schema) of data stored in the database and consciously uses the schema and other metadata to formulate requests to the database and to interpret their results.

Currently there is a strong requirement to base such a tool on a graphical interface. In the database community this requirement resulted in research on visual query languages, which perform visualization of a schema, database instances, queries and query results (for a survey see [1]). An example of a successful graphical interface is Query-By-Example (QBE) [2] and various simpler tools based on the same paradigm. Database applications that access large data repositories, such as data mining and data warehousing, and the enormous quantity of Web resources also require information querying visualization; see for instance the ACM report on Strategic Directions in Human Computer Interaction [3].

As addressed in [4], the key issue behind end-user graphical query interfaces is *usability*. It is a measure of the acceptance of an interface by a common user. Non-obvious visual elements, hidden features and non-trivial graphical syntax are difficult for a common user; thus decrease usability. Usually usability requires sacrificing expressive power for simplicity. Alternatively, the power can be achieved by subdividing user search into many simple steps of user interaction.

Therefore there are two tendencies in the development of such interfaces. One tendency emphasizes the power and this concerns languages such as Kaleidoquery [5] or VOODOO [6] claimed to be as universal as ODMG OQL. Such tools are targeted towards very professional users. The second tendency avoids non-self-explanatory vidgets and visual tricks in order to decrease the learning effort and possible errors of users. Additional power can be achieved through storing intermediate search results and many interaction steps. For sophisticated users the unlimited power can be also available through gateways to typical query/programming languages. There are tradeoffs between these tendencies, e.g. [7], [8], [9].

SKGN follows the second tendency, i.e. it assumes casual users who are not interested in long training and frequent access to documentation. In contrast to QBE, which employs the tabular view of data stored in a relational database, SKGN employs navigation in a schema graph and in an object graph, where classes/objects are vertices and UML-like associations/links are edges. Currently SKGN does not utilize more sophisticated features of typical object models, such as inheritance, complex attributes and method invocation. This is caused mainly by our assumption to simplify the interface as much as possible.

The rest of the paper is organized as follows. Section 2 discusses basic assumptions of SKGN. In Section 3 we present the architecture of SKGN. Section 4 presents main concepts that we have employed in this interface, a typical search scenario and corresponding screenshots. Section 5 concludes and presents future plans.

2. Basic assumptions of SKGN

An important aspect that must be considered during the development of graphical information retrieval interfaces concerns the user model of querying. In the idealistic scenario the user realizes what he/she wants from the database, asks a query that expresses his/her need, quietly waits for the answer, utilizes the answer in his/her job, and then forgets on the query. Our experiments on real applications have shown that this scenario does not follow the way that the humans usually work. Most frequently, the user does not realize precisely from the very beginning what he/she wants to find. The search goal becomes clear after some interaction with the database. The user is frequently unable to ask precisely a query that expresses his/her need, because rough expression of the need, insufficient knowledge concerning the searching tool, and lack of options to express the need precisely. If the answer is not sufficiently quick, the user does not want to wait for it. He/she wants to try other options, or to use other information sources. The answer can be irrelevant, non-pertinent (not interesting for the user), incomplete, and/or with a lot of information noise. Thus the user may need further options to repeat the search with other criteria or to filter the result. In many cases the user is unable to search the entire database in one go because it is very huge. The retrieval can be performed in many days and sessions. The search result can also be used as input to another search.

To cover in SKGN this more realistic scenario we have assumed a simple graphical interface that combines several techniques, in particular:

- Selecting objects via a schema graph and simple manipulations on the schema (e.g. choices from menus, typing query conditions).
- Navigation in a *schema graph* from already selected objects via association roles to other objects; for example, from selected orders to products. This is called *intentional* navigation.
- Navigation in an *object graph*, similar as above. This is called *extensional* navigation.
- Smooth transition from intentional navigation to extensional navigation, and v/v.
- Manual filtering of non-pertinent objects through browsing along selected objects.
- Storing temporary and final query results (references to objects) within personalized data structures called *user baskets*. The user can create as many baskets as he/she wants. Baskets can be nested. The user can determine their role by naming them. Baskets can be shared among many user sessions.
- Objects from baskets can be used as starting points for next search.
- There are typical set operations on baskets (union, intersection, difference).

In addition to the above features there are several options to support user awareness, for instance, undo operation, showing user navigation path, showing quantities of selected/all objects, and others. A similar idea has been assumed in PESTO [8] a part of the IBM Garlic project. Although our idea is a combination of features that can be found in other visual end-user query tools, we hope that it presents new quality in the Internet context and has chances to be very useful for Web users.

3. Architecture of the SKGN

SKGN is a browser: it gets some data from the database and shows it to the user. The data are taken from the database via a wrapper. It has been developed according to the predefined AbstractDatabase interface, which presents a consistent, abstract and universal set of operations on an object-oriented database. This solution guarantees that SKGN is able to work with various repositories or database management systems, providing a wrapper for a particular repository kind. Currently SKGN works with Rodan Office Object Portal via a wrapper called Structural Knowledge Navigator (SKN).

SKGN has been designed and developed as an easy-to-use graphical interface to a data repository. SKGN architecture (Figure 1) consists of the following elements:

- HTML page – refers to a HTML page having URL of the Java applet containing the SKGN front-end.
- Application program (HTML page generator) – the program written by the application developer that generates HTML pages (possibly with URL of the SKGN applet).
- SKN API – can be used by the application programmer to manage the entire SKGN application on the server side. SKN API is also used by the SKGN applet.

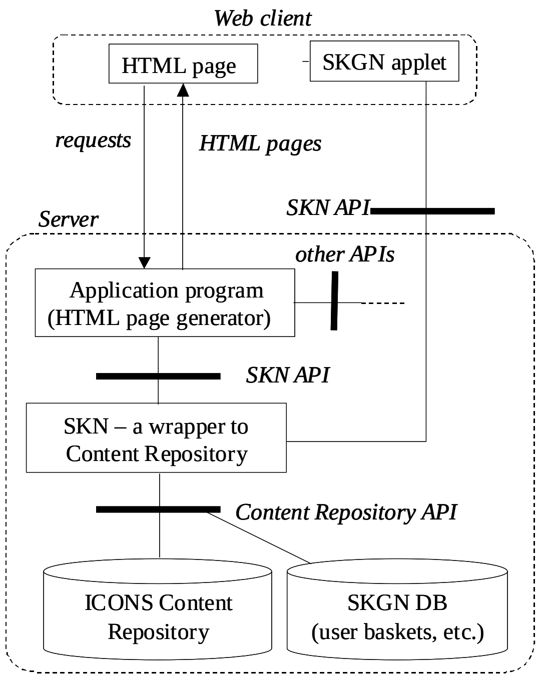


Figure 1. SKGN architecture

- other APIs – other application programming interfaces within the ICONS application development environment that can be used by the application programmer (e.g. management of the database, other graphical interfaces, a query language, etc.)
- SKGN DB – a SKGN local database storing persistent information on the state of processing, on the current user, personalization, parameterization and customization information, user baskets, etc.
- SKN – a wrapper to the content repository. It isolates the browser and a particular repository/database management system.
- ICONS Content Repository – a database management system that stores the entire content of an ICONS application.

4. SKGN Concepts

The subdivision of graphical querying to “intentional” and “extensional” can be found in [10]. We have adopted these terms for the paradigm based on navigation in a graph. The intentional navigation accomplishes a visual querying style that is distinguished from the Kaleidoquery [5] or QBE [2] styles. The user can start querying from previously received search results stored in persistent baskets. The results can be obtained in different ways, by mix of intentional navigations, extensional navigations, manual browsing and manual selecting. We plan to integrate other methods, in particular, very fast indices based on SDDS (Scalable Distributed Data Structure) [11] and the already implemented query language SBQL (Stack-Based Query Language) [12]. SDDS and SBQL are other parts of the ICONS project.

Intentional and extensional navigation are based on navigation in a graph according to semantic associations among objects. Because a schema graph (usually dozens of nodes) is much smaller than a corresponding object graph (possibly millions of nodes), we anticipate that the intentional navigation will be used as a basic retrieval method, while extensional navigation will be auxiliary and used primarily to refine the results. Next subchapters contain description of the methods.

4.1 Intentional navigation

Figure 2 shows a screenshot presenting some state of the intentional navigation. Colors (not shown here) play an important part as an information for the user.

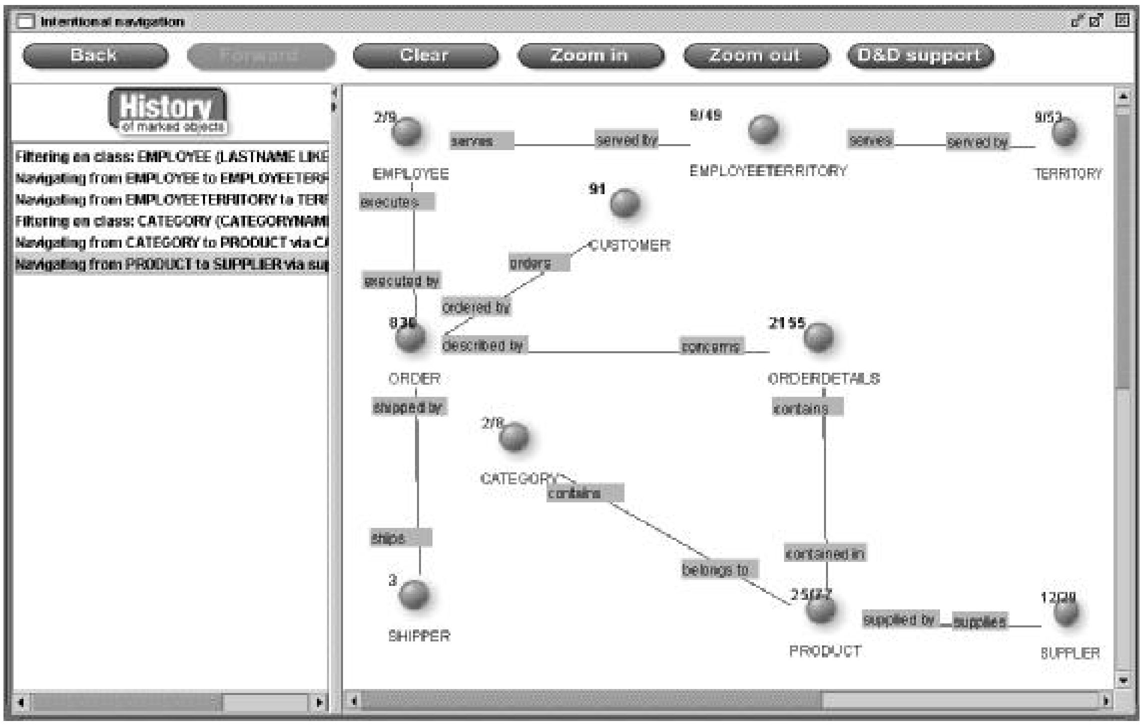
The main part of the intentional navigation window contains a database schema graph, which consists of:

- Vertices, which represent classes or groups of objects (see next),
- Edges, which represent semantic associations among objects (in UML terms),
- Labels with names of association roles. They are understood as pointers from objects to objects (like in the ODMG standard).

With each vertex (i.e. a visual representation of a class) we associate two numbers: the number of all objects in the class and the number of objects that are marked by the user. An object can be marked in the result of the following actions:

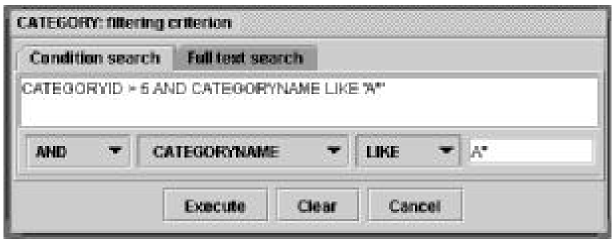
- Filtering through a predicate based on objects' attributes (Figure 3). The action cause marking those objects for which the corresponding predicate is true. There are two options: objects to mark are taken from a set of already marked objects or from entire extension of the class. In this way the user can refine the search. A user-friendly menu-based GUI allows the user to set the predicate by minimal operation on the keyboard. Filtering objects through a predicate follows the SQL *select...from...where* statement (because the ICONS object database is built on top of a relational database).
- Full text search (Figure 4). Similarly to filtering through a predicate, objects to mark are taken from a set of already marked objects or from entire extension of the class. Marks concerns objects that contain all of given keywords or phrases in any of their attributes,
- Manual selection (Figure 5). Using values of special attributes from objects (identifying objects by comprehensive phrases) it is possible to mark particular objects manually. It is especially useful when the number of objects is not too large and there are no common properties among them.
- Navigation from marked objects of one class, through a selected association role, to objects of another class. An object from a target class became marked if there

is an association link to the object from a marked object in the source class. This activity is similar to using path expressions in query languages. A new set of marked objects (a result of navigation), could replace existing one or it is possible to perform an union or intersection.



Filtering on class: EMPLOYEE (LASTNAME LIKE
Navigating from EMPLOYEE to EMPLOYEETERR
Navigating from EMPLOYEETERRITORY to TERR
Filtering on class: CATEGORY (CATEGORYNAMI
Navigating from CATEGORY to PRODUCT via CA
Navigating from PRODUCT to SUPPLIER via sup

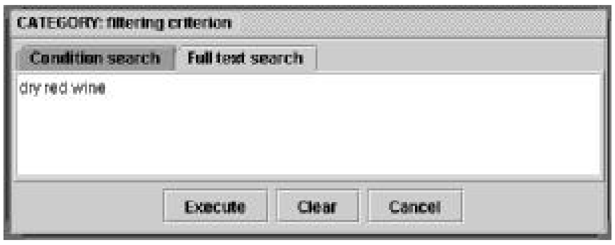
Figure 2. Intentional navigation window



CATEGORYID > 5 AND CATEGORYNAME LIKE 'A*'

AND | CATEGORYNAME | LIKE | A*

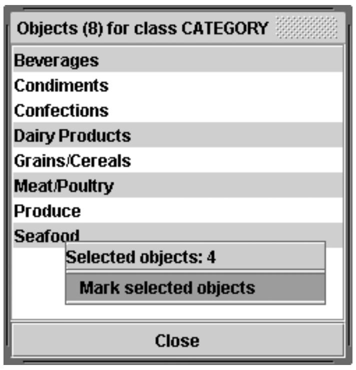
Figure 3. Marking objects - filtering



dry red wine

Figure 4. Marking objects - full text search

- Basket activities. Dragging and dropping the content of a basket on a class icon causes some operation on the marked objects of the class. New set of marked objects taken from basket can replace the existing set of marked objects, can be summed with it, can be intersected with it, or can be subtracted from it (equivalent to AND, OR).



Beverages
Condiments
Confections
Dairy Products
Grains/Cereals
Meat/Poultry
Produce
Seafood

Selected objects: 4

Figure 5. Marking objects manually

All activities which involves marked objects could be undoned or redoned. This is similar to browser back/forward buttons. Each step is stored and shown in the list (see left side of Figure 2). There is also a possibility to go back to any given step.

Intentional navigation and baskets allow the user to receive (in many steps but in a simple way) the same effects as through complex, nested queries. Integrating these methods with extensional navigation, browsing, manual selection and other options supports the user with the power, which is not available in typical query languages. Some functionalities are not available, e.g. queries involving joins and aggregate functions. This is not caused by any technical problem: we persistently want to avoid options, which will not be confirmed by needs of users working with real Web applications.

4.2 Baskets

Baskets are storages of search results. They store unique identifiers of the objects and other baskets (Figure 6). It is especially useful for information categorization and keeping order. During both kinds of navigation it is possible to drag an object (or a set of marked objects) and to drop them onto a basket. The main basket (holding all objects and sub-baskets) is assigned to a particular user. At the end of a user session all baskets are stored in the database. There is no limitation concerning the number of baskets or the number of objects stored in a basket. Below we list all basket activities of SKGN:

- Create a new basket.
- Change basket properties (name, description).
- Remove selected items (sub-baskets or objects).
- Perform operations on two baskets: sum of baskets, intersection of baskets, and set-theoretic difference of baskets. The operation result can be stored in one of the participating baskets or in a new one.
- Drag object and drop it onto extensional navigation frame. As the result, the neighborhood (other objects and links) of a dropped object will be downloaded from the repository.
- Drag a basket and drop it onto class's visualization in the intentional navigation frame. As the result a new set of marked object can be created (replace, add, intersect, subtract). Only objects of that class are considered.

Baskets allow to store selected objects in a very intuitive and structured way. Navigation could be stopped at any time and temporary results (currently selected/marked objects) can be named, stored and accessed at any time.

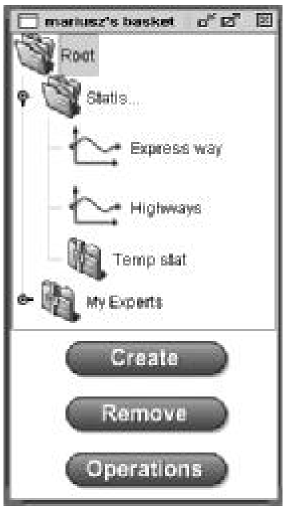
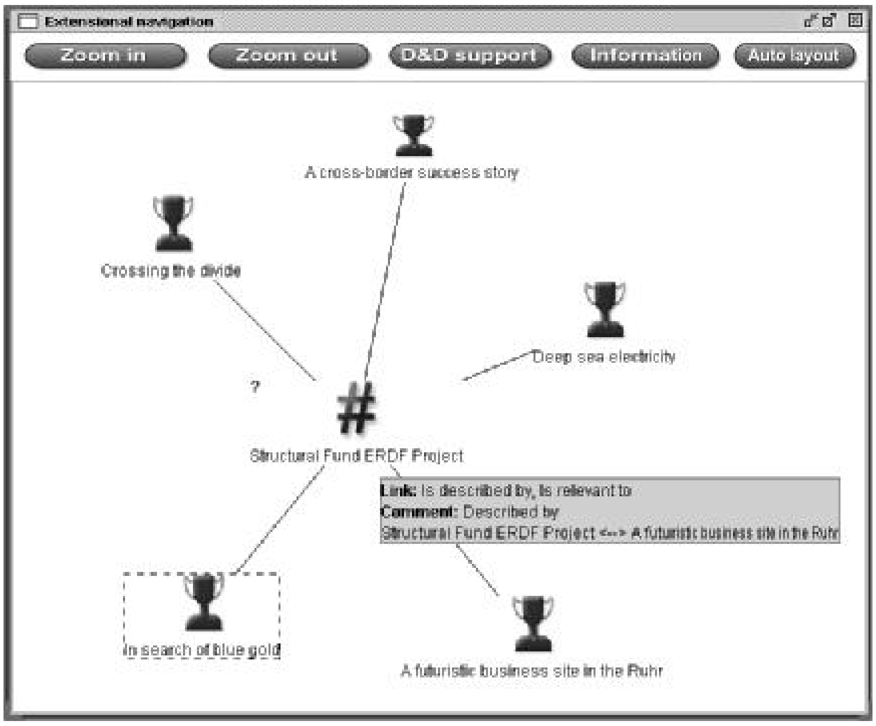


Figure 6. Main basket



Link: Is described by, Is relevant to
Comment: Described by
Structural Fund ERDF Project <--> A futuristic business site in the Ruhr

Figure 7. Extensional navigation window

4.3 Extensional navigation

Extensional navigation takes place inside extensions of classes. Figure 7 shows a graph during typical extensional navigation. Graph's vertices represent objects, and graph's edges represent links. When the user double click on a vertex, an appropriate neighborhood (objects and links) is downloaded from the repository. There are facilities such as zoom in/out, and scroll bars to improve navigation and legibility of the graph. Because of legibility, permanent showing of link labels is avoided (however it is possible to turn it on in user's settings). Instead, an ontology, an auto ontology and specialized tooltips features were introduced. Ontology functionality means that when an object is selected, its class is also selected (on a classes graph – the intentional navigation window). Auto ontology is similar to ontology with one difference – selecting an object's class proceeds when object's tooltip is shown (without selecting of the object). A similar functionality has been implemented for links.

Extensional navigation is useful when there are no common rules (or they are very hard to define) among required objects. In such a situation the user starts navigation from any related object, and then follows the links. During the session it is possible to use basket as a place for storing temporary objects or as starting points for new navigation.

5. Conclusion and Future Plans

This paper describes a graphical querying tool SKGN which offers some new quality for visual querying of large databases in the Web context. We hope that the presented concepts are powerful and easy to understand for non-professional users. We would like persistently avoid the situation when our visual querying tool will be perceived as yet another academic exercise. We address it to real users and real Web applications. A polish company Rodan Systems is using SKGN as a part of the currently developed Structural Fund Project Knowledge Portal. We also plan to use it in other Web applications based on the ICONS application development environment.

Our further plans include investigations on usability of visual query interfaces, aiming at introducing to this type of interfaces visual functionalities that are powerful, useful and natural for users. We plan to integrate SKGN with advanced querying of structural knowledge through indices based on SDDS and the query language SBQL. We also started preparation of a next project aiming at a full visual application development environment.

Acknowledgement

This work is supported by the EU 5th Framework project ICONS (Intelligent Content Management System), IST-2001-32429. The Rodan Systems company is our partner in the project, the main contributor to SKGN and our software verification body. We especially thank to Zbyszek Wróblewski and Bartosz Nowicki, who actively and efficiently participated in the development of SKGN.

References

[1] T.Catarci, M.F.Costabile, S.Levialdi, C.Batini. Visual Query systems for Databases: A Survey. *Journal of Visual Languages and Computing* 8(2), 1997, 215-260.

[2] M.M.Zloof. Query-by-Example: A Database Language, *IBM Syst. Journal*, 16(4), 1977, 324-343.

[3] B.A.Myers, J. D.Hollan, I.F.Cruz, eds. Strategic Directions in Human Computer Interaction. *ACM Computing Surveys*, 28(4), 1996.

[4] T.Catarci. What Happened When Database Researchers Met Usability. *Information Systems 25*(3), 2000, 177-212.

[5] N.Murray, N.W.Paton, C.A.Goble, J.Bryce. Kaleidoquery - A Flow-based Visual Language and its Evaluation. *Journal of Visual Languages and Computing* 11(2), 2000, 151-189.

[6] L.Fegaras. VOODOO: A Visual Object-Oriented Database Language For ODMG OQL. *ECOOP Workshop on Object-Oriented Databases* 1999, 61-72.

[7] N.H.Balkir, G.Özsoyoglu, Z.M.Özsoyoglu. A Graphical Query Language: VISUAL and Its Query Processing. *TKDE* 14(5), 2002, 955-978.

[8] M.J.Carey, L.M.Haas, V.Maganty, J.H.Williams. PESTO: An Integrated Query/Browser for Object Databases. *Proc. VLDB* 1996, 203-214

[9] S.Ceri, S.Comai, E.Damiani, P.Fraternali, S.Paraboschi, L.Tanca. XML-GL: A Graphical Language for Querying and Restructuring XML Documents, *WWW8 / Computer Networks* 31(11-16), 1999, 1171-1187.

[10] M.Derthick, J.Kolojejchick, S.F.Roth. An Interactive Visual Query Environment for Exploring Data, *Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '97)*, ACM Press, October 1997, 189-198.

[11] W.Litwin, T.Risch. LH*G: A High-Availability Scalable Distributed Data Structure By Record Grouping. *TKDE* 14(4), 2002, 923-927.

[12] K.Subieta, C.Beeri, F.Matthes, J.W.Schmidt. A Stack-Based Approach to Query Languages. *Proc. 2nd East-West Database Workshop*, 1994, Springer Workshops in Computing, 1995, 159-180.