

The User as Navigator

Mariusz Trzaska^{*}, Kazimierz Subieta^{##}

[#]Institute of Computer Science, Ordonia 21, Warsaw, Poland

^{*}Polish-Japanese Institute of IT, Koszykowa 86, Warsaw, Poland

Abstract. The old idea of navigation in a database is revisited with two essential changes: we address Web users rather than programmers and the navigation is accomplished as a visual metaphor. The Structural Knowledge Graph Navigator (SKGN) is a prototype implementation of this metaphor. It 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. The paper describes the motivation and main ideas of SKGN.

1. Introduction

More than 30 years ago Charles W. Bachman worked on the network database model during the development of the first commercial DBMS, IDS. For this work in 1973 he won the Turing Award, ACM's highest honor. His Turing Award lecture entitled "The Programmer as Navigator" [1] presents the view of a database as a primary resource and the programmer as a navigator through the database. Bachman proposed a new way of programmer's thinking, in which the programmer moves (in mind) from a record to a record, according to predefined pointer links among the records. The idea was accomplished in the DBTG CODASYL proposal, in particular, in its Data Manipulation Language (DML). Due to low-level programming style assumed in DML ("a-record-at-a-time") and dependence on physical data access details the idea met severe criticism. After the hot debate between proponents of network and relational models, the Bachman's idea has lost. The relational model has offered more promising programming style ("many-records-at-a-time") based on abstract mathematical concepts and free from physical data access details. This style has been accomplished in high-level query languages, notably SQL.

Despite the success of the relational systems and SQL, we have good reasons to return to the Bachman's idea of navigation. Navigation is the basic search paradigm of the Web, a world-wide (unstructured) database. The question is whether the navigation would equally be good as a querying tool for Web users addressing *structured* (object-oriented, XML, etc.) databases.

In the old debate two aspects of navigation were unfortunately mixed up. The first one concerns conceptual navigation, where the user moves through the network of abstract objects according to conceptual links (relationships, associations) among

them. The second aspect concerns physical implementation of the links. In the case of DBTG databases the programmer has to be aware of both aspects. This is not the case of object-oriented and XML-oriented database, where links are conceptual as well, thus the programmer has no need and no possibility to refer to physical details of data storage and access.

High-level navigational facilities retain the original Bachman's idea, but address it to logical databases, abstracting from physical access details. Such facilities are proposed by many authors, usually as so-called *path expressions*, see e.g. [2], [3], [4]. Path expressions are sequences of data names (plus other options) that reflect a path in some graph representing database structures. By putting a next name n at the end of a path expression the programmer moves from already selected vertex of the graph to another vertex connected with the previous one by an edge named n . Path expressions are also possible for relational databases [5] as an (unexplored) alternative to the SQL querying style (providing some kind of automatic "reverse engineering" from relational structures to entity-relationship diagrams). Conceptual path expressions for navigation in object database are available in ODMG OQL, in a limited version (no "many-paths-at-a-time"). In SBQL [6] the user can use path expressions with no limitations to navigate down a complex object structure and to navigate from objects to objects via association links. From the user point of view the navigation is quasi-parallel: there as many navigational paths as objects selected as starting points for the path expression. XML-oriented query languages (XQL, Xpath and XQuery) use path expressions as a tool for selecting parts and subparts of XML documents. Also the newest SQL-99 standard uses path expressions to navigate down nested tables structures and/or to navigate according to logical references.

Many Web applications (especially professional ones) are based on structured data, where information has the form of non-flat data structures with tags or attribute names bearing semantic meaning (ontology) 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. Our experience based on real applications has shown that there is a need to support the user by a generic, ad hoc 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.

In this paper we present a new incarnation of the Bachman's idea and path expressions. As a part of the ICONS project (EU 5-th Framework) we have implemented a graphical querying interface for Web users. The interface, named Structural Knowledge Graph Navigator (SKGN), allows the user to navigate through two types of graphs that can be displayed within a Web browser. One type of navigation, called *extensional*, concerns an object graph, with objects as vertices and named links among objects as edges. Another type of navigation, called *intentional*, concerns a schema graph *a la* UML class diagram. The user can dynamically switch between these two types of navigation. Another important feature of SKGN concerns persistent named *baskets* that can be used by the user for collection temporary and final results of querying and manual browsing. SKGN contains many other options that we have proposed to support usability and user awareness during querying and

browsing. We hope that the combination of features implemented in SKGN presents new quality for Web users, especially in context of Web Services and Semantic Web, which may require friendly graphical querying interface within structured Web resources. SKGN has been tested for a real Web applications (Structural Funds Portal for the Polish government) and the results are promising.

The rest of the paper is organized as follows. Section 2 discusses utilization Bachman's ideas. In Section 3 we present basic assumptions of SKGN. Section 4 presents main concepts that we have employed in this interface, real life examples and corresponding screenshots. Section 5 gives an overview of our future plans concerning SKGN. Section 6 concludes.

2. From Programmer to Web User

Bachman has defined a role of a programmer as navigator in several points. Below is a summary:

- A programmer can start from any known object (or from the database root) and sequentially access next objects until he/she reaches a desired object.
- He/she can reach objects, which have a particular value of a selected attribute.
- He/she can reach objects, which are connected to a given object via selected links.

This functionality we have implemented in SKGN. The essential difference that we have introduced concerns kinds of users: Bachman addressed programmers while SKGN addresses Web users. Thus navigation is the feature of a graphical querying interface rather than a programming language.

Note that end-user oriented graphical querying interfaces present a quite new quality in comparison to textual, keyboard oriented query languages (like SQL and OQL). Currently there is rather common opinion that the latter are too hard for casual Web users. Alternatives to them include graphical querying tools that can be roughly subdivided into two groups:

- Graphical query languages. Conceptually they are equivalent to keyboard-oriented query languages, but address end-users and use graphical metaphors rather than traditional syntax. There are many such languages, for instance Query by Example [7], VOODOO [8] and graphically sophisticated three-dimensional Kaleidoquery [9]. In some cases graphical queries are first translated to their textual counterpart (e.g. OQL) and then processed by an already implemented query engine. The result of the querying can be visual (visual metaphors) or textual (tables). In these query languages the user must know what he/she wants to find before formulating a corresponding query.
- Graphical browsing interfaces. The most important difference to the previous group is the manner of user work. Browsing systems rely on manual navigation from one object to the next one. During browsing the user can read the content of selected objects. Browsing is the obligatory option in situation when the user cannot define formally and precisely the criteria concerning the search goal. Browsing is typical for Web search engines such as Google.

In SKGN we have implemented both paradigms. In this way we offer a variety of querying options that (we hope) will be adequate to real-life situations that the user may have during his/her work with Web databases.

One of the most important issues behind end-user graphical querying interfaces is *usability* (see [10]). 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 a user search into many simple steps of interaction.

In our case the idea of navigation plays the chief role in both presented modes of querying. We believe that the idea implemented in SKGN as a graphical metaphor is quite natural for the users and sufficiently powerful. First acceptance tests have shown that the idea may support the appropriate level of usability for some kinds of tasks related to real-life Web applications.

3. Basic Assumptions of SKGN

SKGN has been designed for casual users who are not interested in long training and frequent access to documentation. In contrast to QBE [7], which employs the tabular view of data stored in a relational database, SKGN employs the 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, because we believe this is a key to usability. In the future we are going to extend SKGN to more advanced object models (see Chapter 5), but keeping in mind that more sophisticated features might be hard for users.

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 support 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 [11] a part of the IBM Garlic project. Our tool - SKGN presents a combination of features that can be found in other visual end-user querying tools, but to our best knowledge we didn't find a similar tool addressing Web users.

SKGN gets data from the database via a wrapper, which is a self-contained replaceable module with a well defined front-end. Thus SKGN can be adopted (with moderate effort) to many types of DBMS, repositories or file formats (e.g. XML) by writing a new wrapper for a particular kind of information resource. Detailed description of the SKGN architecture can be found in [12].

4. SKGN Concepts

The subdivision of graphical querying to "intentional" and "extensional" can be found in [13]. 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 [9] or QBE [7] 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) [14] and the already implemented query language SBQL (Stack-Based Query Language) [6]. 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 1 shows a screenshot presenting some state of the intentional navigation. Colors play an important part as 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,
- 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).

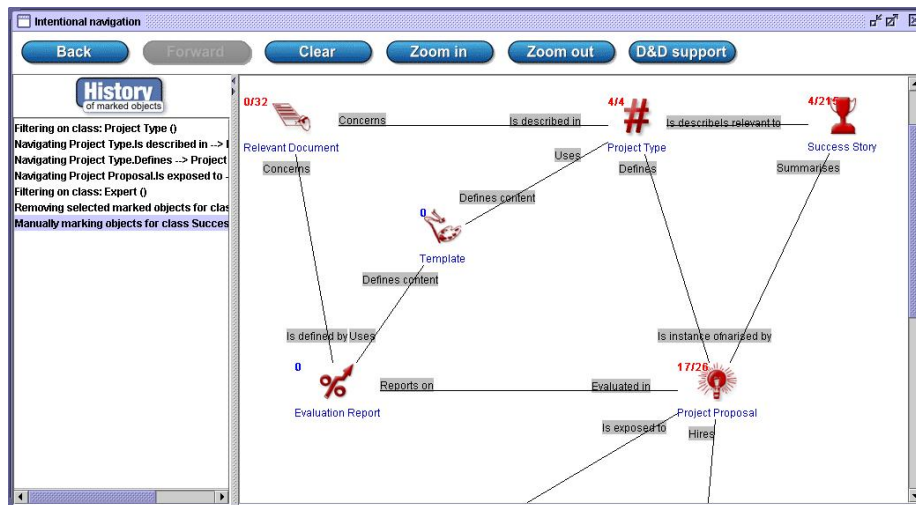


Figure 1. Intentional navigation window

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

- Filtering through a predicate based on objects' attributes (Figure 2). 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. 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 3). 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.

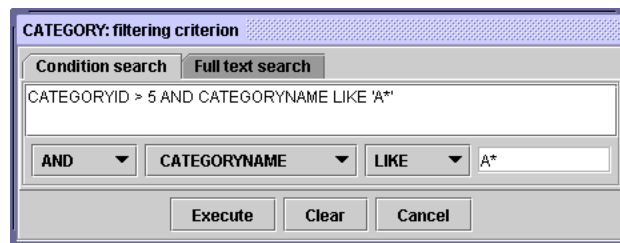


Figure 2. Marking objects - filtering

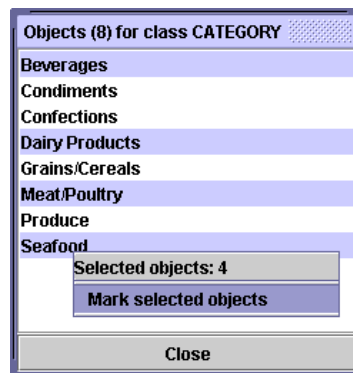


Figure 3. Marking objects manually

- 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.
- 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 OR, AND).

All activities, which involve marked objects could be undone or redone. This is similar to browser back/forward buttons. Each step is stored and shown in the list (see left side of Figure 1). 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 functionality is 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 4). 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.

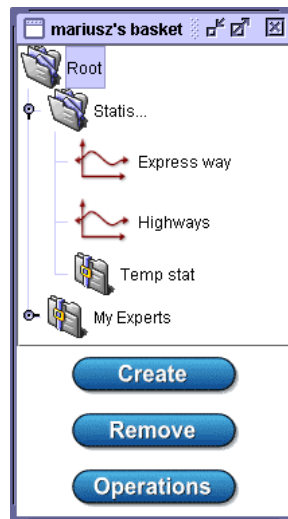


Figure 4. Main basket

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 storing 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.

4.3 Extensional Navigation

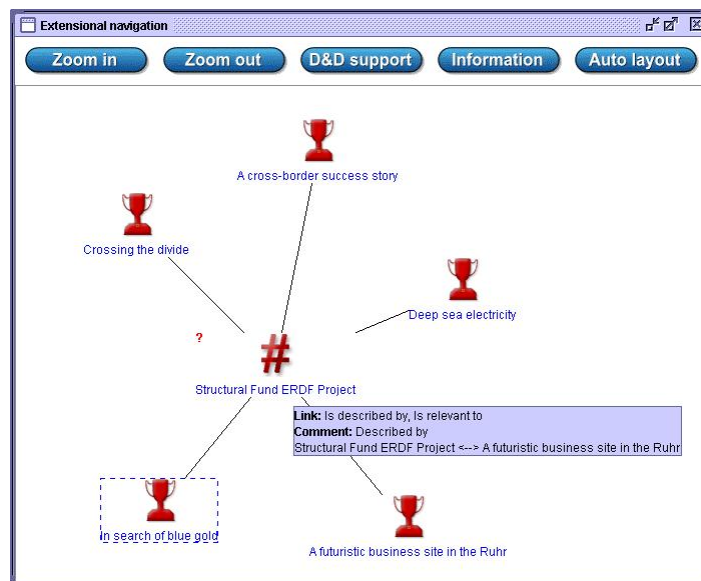


Figure 5. Extensional navigation window

Extensional navigation takes place inside extensions of classes. Figure 5 shows a corresponding graph. Graph's vertices represent objects, and graph's edges represent links. When the user double clicks 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, ontology, auto ontology and specialized tool tips 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 hard to define) among required objects. In such a situation the user can start navigation from any related object, and then follows the links. During the session it is possible to use basket for storing temporary objects or as starting points for new navigation.

4.4 Real Life Examples

This sub chapter shows possibilities of using SKGN. We will use typical cases based on class diagram (see Figure 6) for Structural Fund Portal.

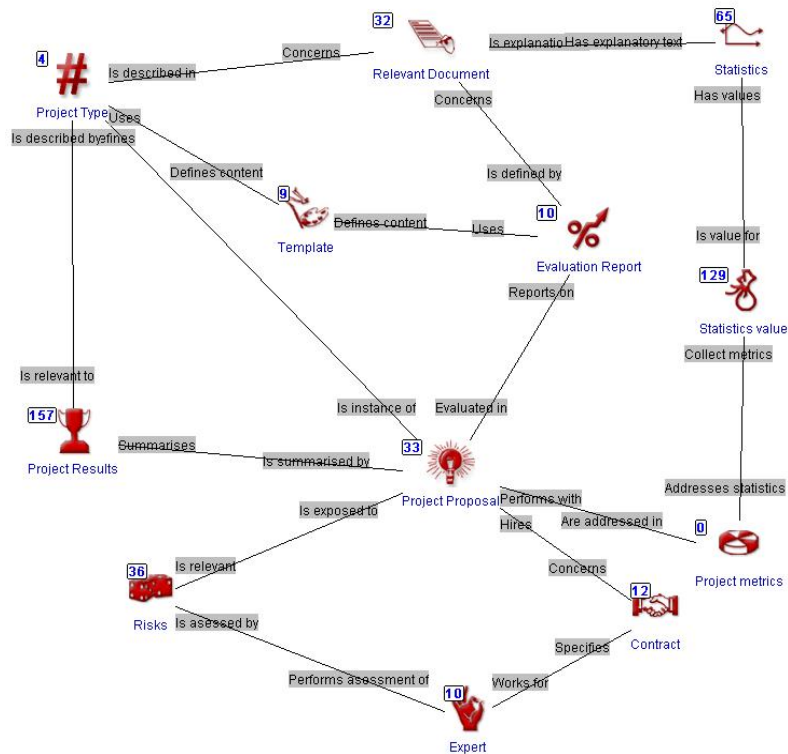


Figure 6. Schema diagram for Structural Fund Portal

Assume we want to know details of the contract for the expert called Smith. To accomplish that we will start intentional navigation. As a filtering predicate we use (on appropriate attribute) the name Smith. After a while SKGN marks one object representing our expert. Now we can navigate through role *Works for* to the appropriate Contract object. Using objects viewer, we can check all the details concerning the contract. Any object, which is important to us for longer time, can be stored in a basket.

Another task is to find experts who are experienced in particular project's type. We can start from a specific object from the class Project Type. To find that object we

use manual marking or filtering. After a while we have one marked object. Now we navigate from Project Type to Project Proposal via the *Defines* role. That action has marked some objects in the Project Proposal class. Next we navigate from Project Proposal to Contract and then from Contract to Expert. As a result, some objects from Expert class have been marked. Now we can check the details using Object Viewer (see Figure 7).

The screenshot shows a window titled "Object: Adrian Sho...". Below the title bar, it says "Object's class: Expert". The main content is a table with two columns: "Attribute" and "Value".

Attribute	Value
Name	Adrian Short
Number of EC Projec...	4
Years of Employmen...	8
MZU user id	JamesS
Minimum rate per hour	45
Maximum number of ...	60
Available from	2003-11-01 00:00:00.000
Available to	2003-11-30 00:00:00.000
Role	Project Eligibility Evaluator
Language of Contrib...	English
Language of Contrib...	Polish
Area of Expertise	Civil Engineering
Area of Expertise	Construction manageme...

At the bottom of the window, there is a "Close" button.

Figure 7. Example of using Object Viewer

It is also possible to find an expert who worked with particular project's type and, for instance, assessed particular risks. Let's assume that marked objects for Expert class contain all experts who worked with particular project (result of previous case). We start with marking objects, which describe interesting risk(s). Next we navigate from Risks to Expert. On the contrary to the previous case, we perform the intersection of sets (already marked experts and newly created). Then, as previously, we can record the results of our navigations in a basket.

5. Future Plans

The current version of SKGN is considered a prototype, which helps us in making investigations concerning various approaches to graphical information retrieval. We have some next ideas how to improve the navigation. Below we list some of them:

- Additional activities for marked objects. New functions, such as average, maximum, minimum, sum, etc., allow one to perform other kinds of queries, or to make reports from objects stored in the baskets.
- Stickers. A user may want to annotate particular object(s). To do this we would like to introduce little stickers, which can hold any kind of information, which are important to users. Stickers will be persistent and assigned to a particular user.
- Advanced 3D environment. It is especially important in case of extensional navigation, where we work with visualizations of particular objects. Current version of SKGN utilizes quite simple 2D approach. By adding next dimension, we

hope to achieve possibility to show a few hundreds objects in a legible way. We also consider if this approach would be better for intentional navigation too.

- Projections of marked objects. This functionality allows visualizing a set of marked objects where position (in terms of x, y (and maybe z in case of 3d environment) coordinates) of each of them will be based on value of particular attribute. That approach makes it possible to identify some groups of marked objects. For instance it will be easy to see the group of employees who are paid, let's say, less than \$2000 and have some specified experiences.
- Wrappers to Web Services and to CORBA objects. Creating such wrappers will enable the users to navigate within resources based on these well-established technologies.
- Facilities to attach to objects some "trade information", in the spirit of CORBA Trade Service, CORBA Property Services and Web Services (WSDL and UDDI), and search facilities according to this information. These facilities allow one to create "yellow pages" presenting business properties of particular objects. Current facilities of SKGN will be combined with these new one.
- Extending the utilized data model by generalization. We realize, however, that excessive growing of data model could have impact on user's easy-of-use. Thus we plan to perform some investigation concerning if this kind of improvements would be acceptable for common users.
- Modification of the existing user interface. For a non-professional user, some typical search operators are not obvious, for instance, logical operators AND and OR. Thus, friendlier graphical interface could be introduced. A candidate interface, based on a flow model, has been described in [15].
- Visual joins. They should allow to group objects form two or more classes, for instance, an employee object with a company object. A group of objects would be treated as an ordinary object, with possibilities of filtering, navigating, marking, storing in a basket, etc.

6. Conclusion

This paper describes the SKGN tool, 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 avoiding 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. Currently we consider extending SKGN with much new functionality. We also plan to integrate SKGN with advanced querying of structural knowledge through indices based on SDDS and the query language SBQL.

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] Bachman Ch. W.: The Programmer as Navigator. Communications of the ACM, (1973) 16 11 653 – 658
- [2] Zaniolo C.: The Database Language GEM. [SIGMOD Conference 1983](#): 207-218
- [3] Subieta K.: High-Level Navigational Facilities for Network and Relational Databases. [VLDB 1983](#): 380-386
- [4] Kifer M., Kim W., Sagiv Y.: Querying Object-Oriented Databases. Proc. SIGMOD Conf., (1992) 393-402
- [5] Subieta K.: Navigational facilities for relational data base. [Inf. Syst. 8](#)(1): (1983) 29-36
- [6] Subieta K., Beeri C., Matthes F., Schmidt J.W.: A Stack-Based Approach to Query Languages. Proc. 2nd East-West Database Workshop, 1994, Springer Workshops in Computing, 1995, 159-180
- [7] Zloof M. M.: Query-by-Example: A Database Language. IBM Syst. Journal, 16(4), 1977, 324-343
- [8] Fegaras L.: VOODOO: A Visual Object-Oriented Database Language For ODMG OQL. ECOOP Workshop on Object-Oriented Databases 1999, 61-72
- [9] Murray N., Paton N.W., Goble C.A., Bryce J.: Kaleidoquery - A Flow-based Visual Language and its Evaluation. Journal of Visual Languages and Computing 11(2), 2000, 151-189
- [10] Catarci T.. What Happened When Database Researchers Met Usability. Information Systems 25(3), 2000, 177-212.
- [11] Carey M.J., Haas L.M., Maganty V., Williams J.H.: PESTO: An Integrated Query/Browser for Object Databases. Proc. VLDB (1996) 203-214
- [12] Trzaska M., Subieta K.: Structural Knowledge Graph Navigator For The Icons Prototype. Proc. of the IASTED International Conference on Databases and Applications (DBA 2004)
- [13] Derthick M., Kolojejchick J., Roth S.F.. An Interactive Visual Query Environment for Exploring Data, Proceedings of the ACM Symposium on User Interface Software and Technology (UIST '97), ACM Press, (1997) 189-198
- [14] Litwin W., Risch T.: LH*G: A High-Availability Scalable Distributed Data Structure By Record Grouping. TKDE 14(4), 2002, 923-927
- [15] Shneiderman B.: Visual user interfaces for information exploration. In Proceedings of the 54th Annual Meeting of the American Society for Information Science, Medford. NJ, 1991. Learned Information Inc. 379–384