In: Zimmermann, Harald H.; Schramm, Volker (Hg.): Knowledge Management und Kommunikationssysteme, Workflow Management, Multimedia, Knowledge Transfer. Proceedings des 6. Internationalen Symposiums für Informationswissenschaft (ISI 1998), Prag, 3. – 7. November 1998. Konstanz: UVK Verlagsgesellschaft mbH, 1998. S. 273 – 282

Interface for Navigation and Database - Querying via WWW

A.L. Heuer, K. Siemonsen, T. Engel and C. Meinel

{Heuer, Siemonsen, Engel, Meinel}@ti.fhg.de
Institut für Telematik
Bahnhofstraße 30-32, 54292 Trier

Contents

Summary

We describe the concept and the implementation of a Navigation and Database Interface (NDI) on the World Wide Web. The prototype combines the storage of database queries and navigational structures, both made more valuable by comprehensive meta information. The multilingual design of the meta information set prepares the system for use in an international environment.

1. Introduction

As described in (Benn 1998) there is an enormous number of systems, scientific or commercial, trying to make databases available on the WWW. The most common way is using forms and CGI-scripts. Although this is a very easy solution, it

becomes more and more unmanageable if the number of queries has to be extended or modified frequently. The number of scripts will increase with the number of queries. The performance is rather poor and problems occur as the number of accesses rises.
Another common approach uses macro-HTML. Documents include special tags that are preprocessed by the server before sending the document to the client, so that the tables are generated on the fly. The results are put into predefined HTML-pages either by CGI scripts or by an added server functionality. This method is implemented easily and allows better maintenance than using CGI-scripts that mix executable code with database queries and layout. Some database vendors implement HTTP interfaces in their database engines. HTTP requests to such databases are processed directly. However, this method will only work on a proprietary basis. Each database vendor will create his own interface.
Finally there is the more complex approach of using distributed services. Using IIOP or other protocols, applications - distributed by application servers to the clients - connect to objects implementing business logic in order to manipulate data. The client only displays data, while database interaction uses appropriate services provided by database applications.
Our approach is similar to a database engine implementing HTTP. We built a server that delivers navigational information enriched with meta information. Such meta information are, as described below, a speaking name, a comment, a number of keywords and an additional HTML page containing detailed information. Meta information is stored in different languages. Therefore the same navigational structure can be used in multilingual environments. On the client side users receive dynamically generated HTML pages containing links. Each link leads either to another navigational page lower in the hierarchy or represents a database query on a relational database. Meta information is provided in either case. If users choose a database query link, the according query will be executed. The result set will be returned as a table.

2. The Design of the System

The system architecture is quite simple. A relational (Codd 1990) database, the so called metadatabase, is used to store information about other relational databases. An interface based on this metadatabase provides access to the databases known to the system.
The complete system, as shown in Fig. 1, actually consists of four components on the server side. Firstly there is the metadatabase itself. For example, it stores information about relational databases (DB1, DB2, ...). Furthermore there is the HTTP-server-like program (NDI) that provides the interface. It displays the navigational structures and executes the select-statements. Finally there are two tools for administration. One (Admin 1) is being used to maintain the information about the managed databases. The other one (Admin 2) allows to generate queries and navigation paths. Platform independence is aspired by doing all coding with Java. Therefore the server component and the administration tools are expected to run on most systems. Database connections are managed by JDBC drivers (Reese 1997). Because of our decision to use simple HTML pages submitted to the client via HTTP, every common WWW-Browser can be used as a front-end to the system.

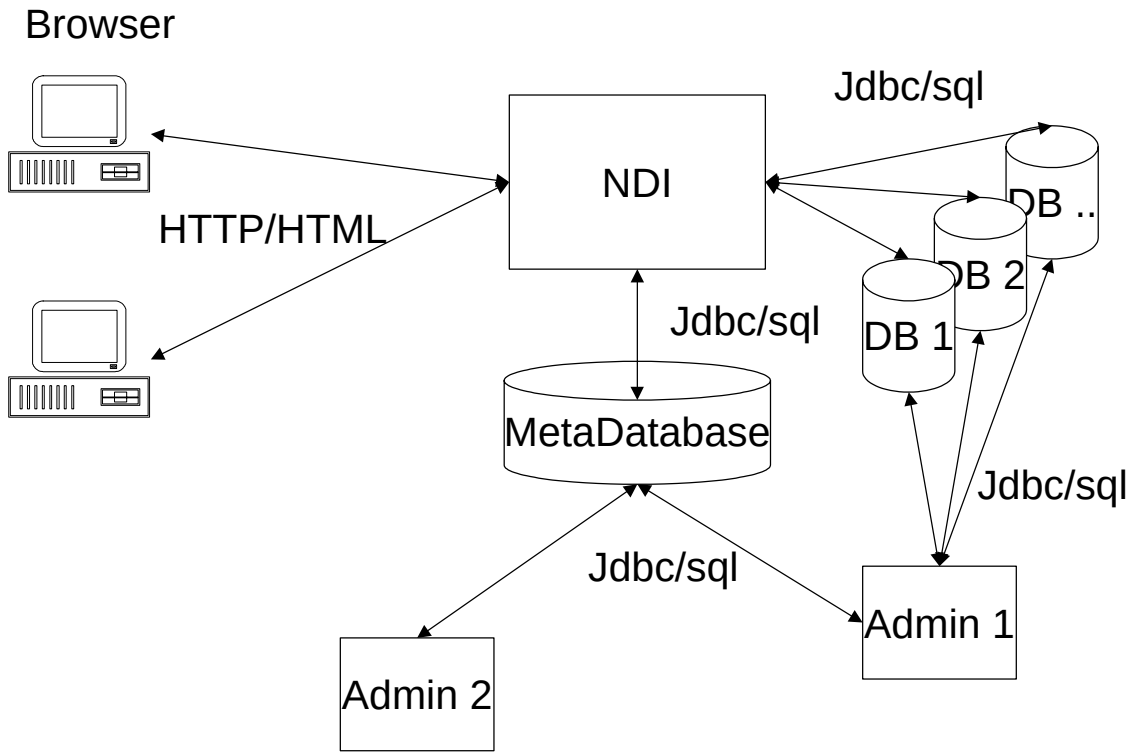

Fig. 1 The system architecture of the Navigation and Database Interface (NDI). The Browser as a client, and the NDI as Server that connects to databases using information from the metadatabase. Furthermore the two Admintools to manage the alignment between the metadatabase and the maintained databases (Admin 1) as well as the navigational structure(Admin 2)

2.1 The Metadatabase

The data needed to provide a navigational interface to the client is stored in a relational database. This database contains different kinds of information. Firstly there is the information about the databases known to the system. Secondly there are navigational structures. Furthermore the metadatabase retains the queries to be executed on the known databases. Finally it stores metadata to enable a user-friendly informational interface to the databases with their contents.
The metadatabase consists of over thirty tables managing the mentioned information. The following paragraphs describes the different kinds of information in detail.

2.1.1 The Meta Information

In our system meta information (Fig. 2) is an additional description for all objects stored in the metadatabase. This means there is description maintainable for single columns as well as for complete queries. Such meta information is at first a speaking name that is presented to the user for each object. For example, the system allows administrators to give each column a name describing the content of the column to the common user. This name can be defined simultaneously in any language maintained by the system. In addition to the speaking names it is possible to enter multilingual comments. They allow a short description of the objects they are linked to. For example, they may be displayed in the status line of the Browser with JavaScript, if the user moves the mouse pointer over the

corresponding link. Finally the administrators may use HTML pages served by a common webserver for detailed description.

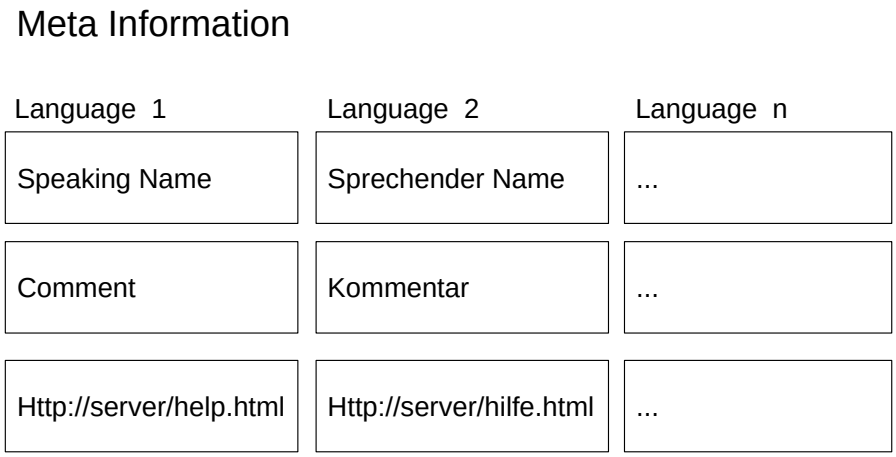

Fig. 2 The metadata managed for each object stored in the metadatabase.

The system stores only the URLs of these detailed help-pages. Therefore common techniques can be used to create and store the pages. Of course such detailed pages can also be differentiated by language. Furthermore such description pages may contain links to other relevant database contents. This adds an additional navigation layer, that is not maintained by the NDI anymore. In order to ease finding of an object a set of multilingual keywords can be attached to the objects.

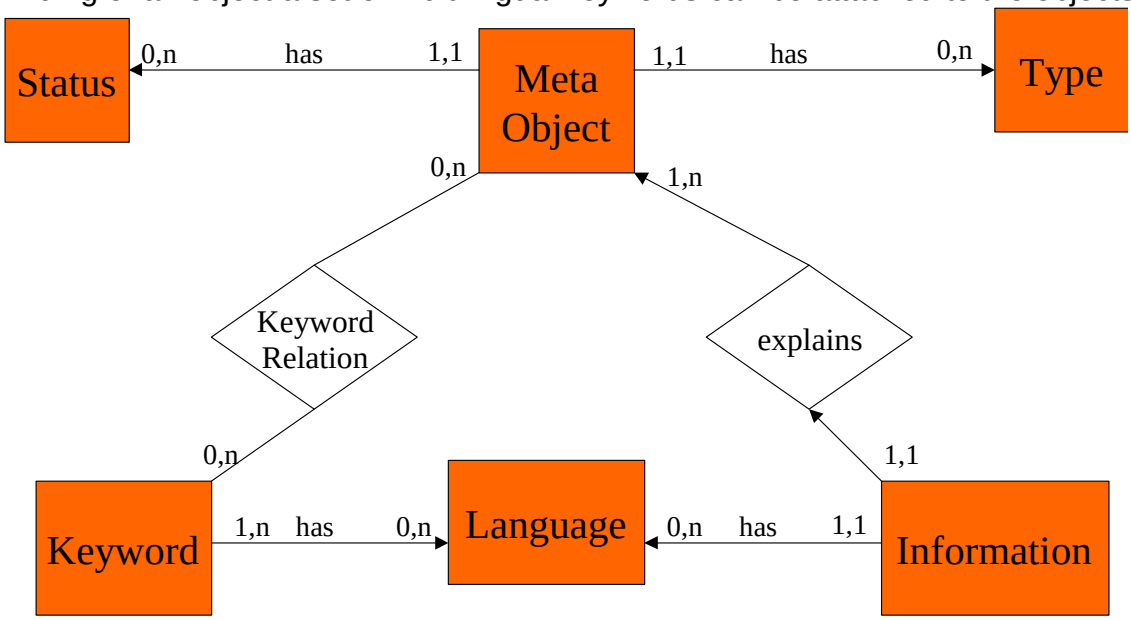

Fig. 3 The meta object with its relations. Other objects that need meta information inherit from this object.

In the data model a meta object (Fig. 3) was designed that contains the set of meta information. Each other object that needs access to metadata inherits from this object .

2.1.2 The Database Information

In order to access tables or views of any other relational database with the NDI,

the metadatabase has to store information about their structures. This means that each table or view with its columns and its data formats has to be referenced in the metadatabase. Parameters to access those databases are stored as well as their type, the corresponding hostname and the port of the database server.

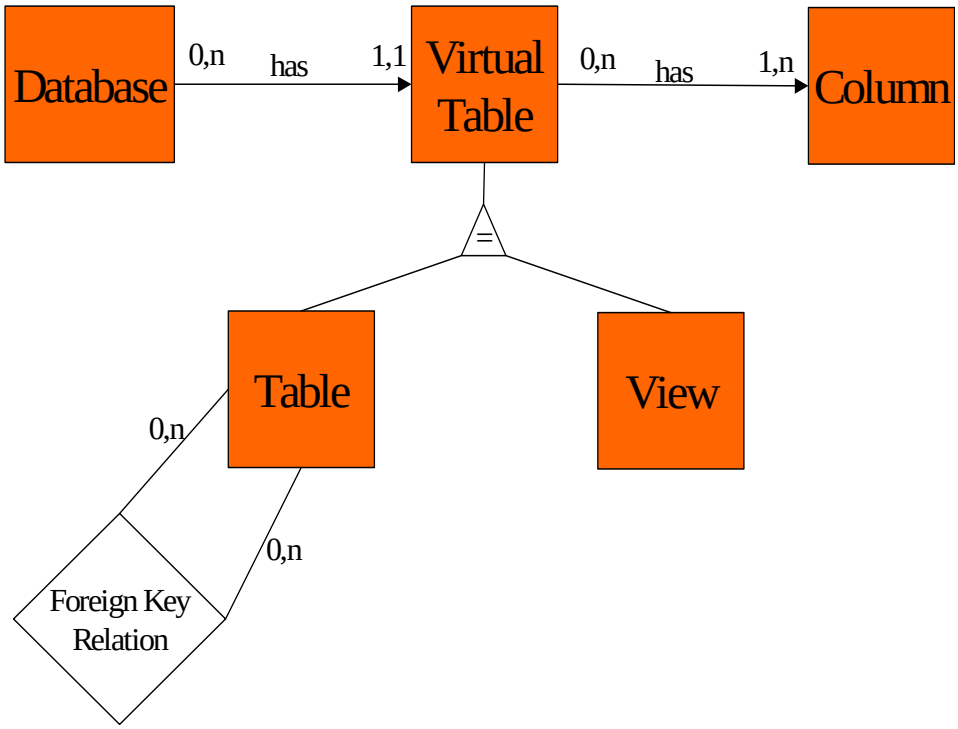

Fig. 4 The relationship of the entities relevant for the access of the maintained databases. Because of their similarities, tables and views are described in the model as virtual tables. Tables and views only extend those virtual table objects.

While this approach makes it difficult to handle constantly modified database structures, it is satisfying for databases that are set up once and modified only very seldom. Of course a modification in the sense of extension is easier to handle than removing columns or tables.

2.1.3 Atomic Storage of Query-Components

A common SQL (Date 1997) statement consists of several components. The NDI system does not store queries as strings. Each component is stored by reference separately. This means a select statement is broken down on column and table / view basis. Because the metadatabase does not use strings to store queries, it is very easy to modify a query afterwards. Furthermore, if some table is modified or a column is removed, there is no need to parse query strings. One can directly search in the sets of query components. Since each query consists of a set of components, the system creates query strings dynamically. If one wants to use the same query on a different database system, the syntax can differ. The dynamic creation of query strings allows to adapt the syntax.
Furthermore the atomic storage of the query components can be used for query inheritance in case of a further drill down. For each inheritance level the administrator can define which component he wants to add to or remove from the

actual basic query. So such a basic query can be extended until the resultset is the desired one.
Restriction of queries can be done by three different kinds of values. Firstly there are static values. They are set at the time of query creation. Furthermore the system provides two kinds of dynamic values. Date and time periods, e.g. today, yesterday or last 20 days are generated for restriction dynamically before the query is sent to the database. Finally there are text fields that have to be filled out by users at runtime and lists, from which users may select entries. In such a case of interaction with users a form has to be shown before the execution of the query.

2.1.4 The Navigation Structure

Besides the database dependent information the system has to store navigational information (Fig. 5). Therefore a hierarchical structure was chosen. The system stores nodes and edges.

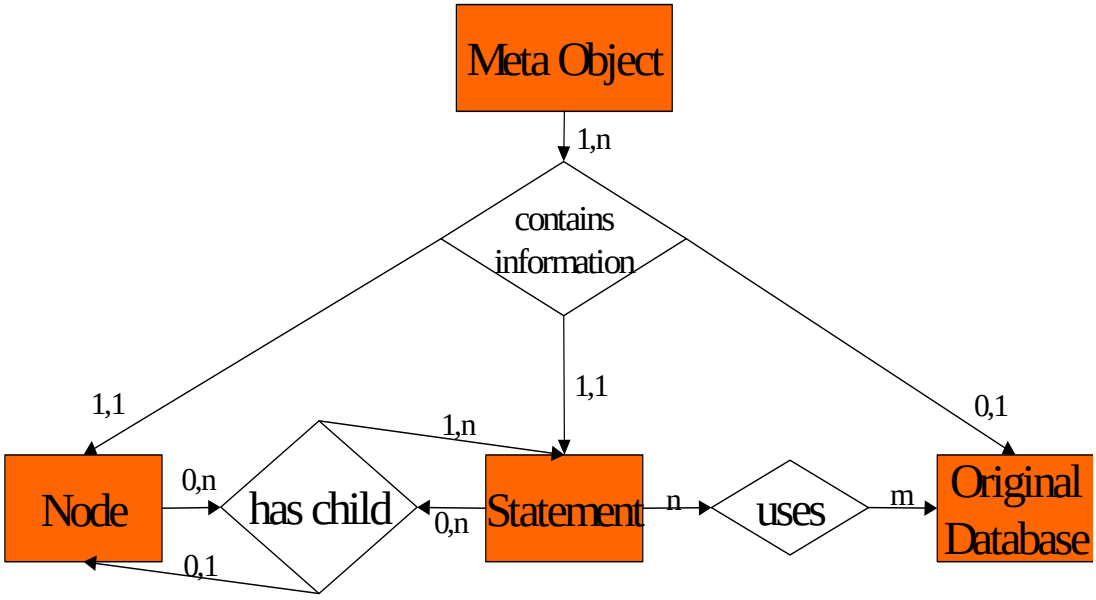

Fig. 5 The relationship between nodes and statements. Each node can have one parent node. A node may have child nodes and statements. Each statement can be linked to an arbitrary number of nodes. A statement is executed by the database it was built for. For nodes as well as statements and the original databases meta information is available.

For ease of maintenance each node may have several child nodes but only one parent node. A node contains a set of meta information, as described above. Connecting the nodes with edges creates a hierarchy which allows an upward and downward navigation in the structure. Equivalent to the directory structure in file systems, nodes may have child nodes that would represent subdirectories. To each node a number of queries (queries could be associated with the files in file systems) can be attached. Those queries may aim at different databases. So in the navigational structure it is possible to group queries by the content they deliver, independent of the database that stores the data. Users will not need to know which database stores the data they require. They will follow the content dependent paths.

One query object can be linked to many nodes, eventually in different contexts. So it is possible to build more than one path to the same set of queries. Therefore different views on the data can be generated. This allows to adapt the system for different groups of users. The system does not build a content dependent navigational structure automatically. Administrators have to develop such structures manually. In the node and query space the search by keywords that can be attached to them would be a kind of free navigation.

2.2 The Server

Using Java the NDI server was implemented platform-independent. The server program (Sridharan 1997) listens for HTTP requests on a specified port. It parses the requests and extracts the required information from the metadatabase. Either the system generates then a new navigation page that is presented to the user, or it executes a query on a database. In this case the received resultset will be shown to the user in a HTML-Table.
In the navigation pages each node object is represented by its speaking name. The name is connected via a hyperlink to the child nodes and the attached queries.
The server program (Fig. 1, NDI) runs multithreaded and is capable to connect to several different database engines (from different vendors) at once. The connection to all databases is done by JDBC. Therefore the server can only connect to databases which provide JDBC drivers that make them accessible via internet. Since nearly each bigger vendor provides a driver to his databases, this is no real restriction. In order to adapt to user requests, the server logs each database query. This may help to specify existing and further needed queries.

2.3 The Administration Tools

As mentioned above (Fig. 1) there are two tools that were designed in connection with the Navigation and Database Interface. Both tools are quite platform independent, since their coding was done with Java. They connect to the databases with JDBC in the same manner as the server.
The first one (Fig. 1 ,Admin 1) is needed to import the structures of the databases to be maintained by the system. It reads the system tables of the databases and extracts the required information from them. Since the system tables differ for each database type, this process requires some adaptational work, if a new database is to be imported. When JDBC drivers provide a complete set of database metadata the import process can be standardized. As described above each table, view and column of the maintained databases is referenced in the metadatabase. Furthermore meta information may be added during the import process. Besides this import functionality, certain consistency checks have to be implemented. The complete system will only work, if the information about the original databases concerning tables and columns is correct.
After an import the available information can be managed with a second tool (Fig. 1, Admin 2). This tool allows to create the hierarchical structures for navigation described above.

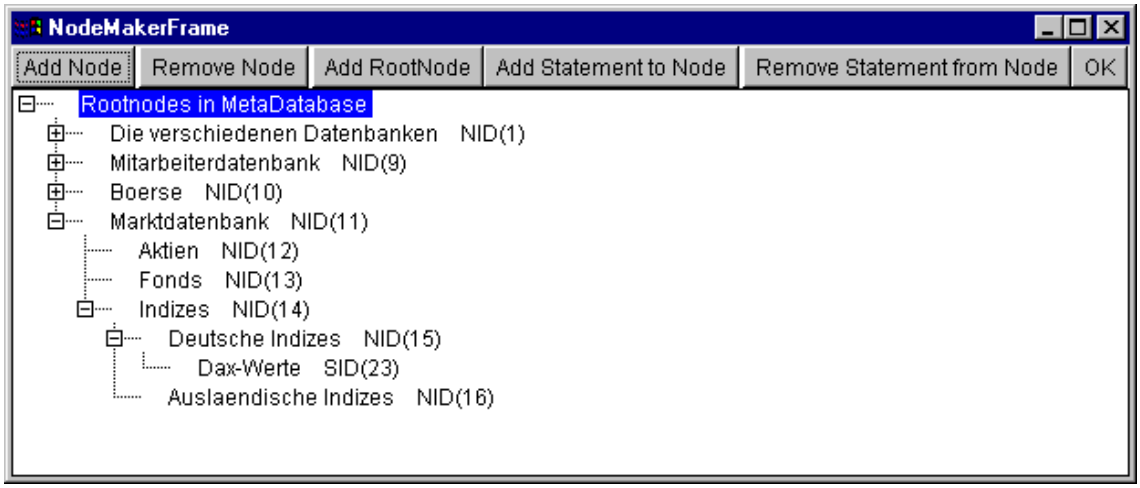

Fig. 6 The window with the tree view that displays the navigational structure of the NDI. The window can also be used for editing.

The navigational structure is shown in a tree view (Fig. 6) that may be edited with the functions the window provides. A meta information window (Fig. 7) gathers the information associated with a node or query. The information can be entered in several languages at the same time.

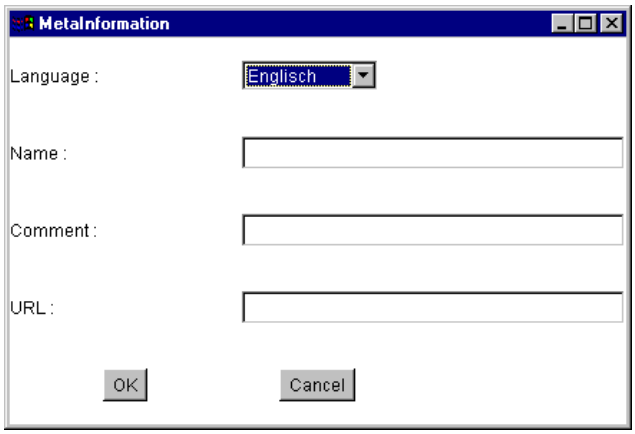

Fig. 7 The window that gathers the meta information for a node or statement. The choice in the top switches the language.

Simple queries can be designed with a query design window (Fig. 8).

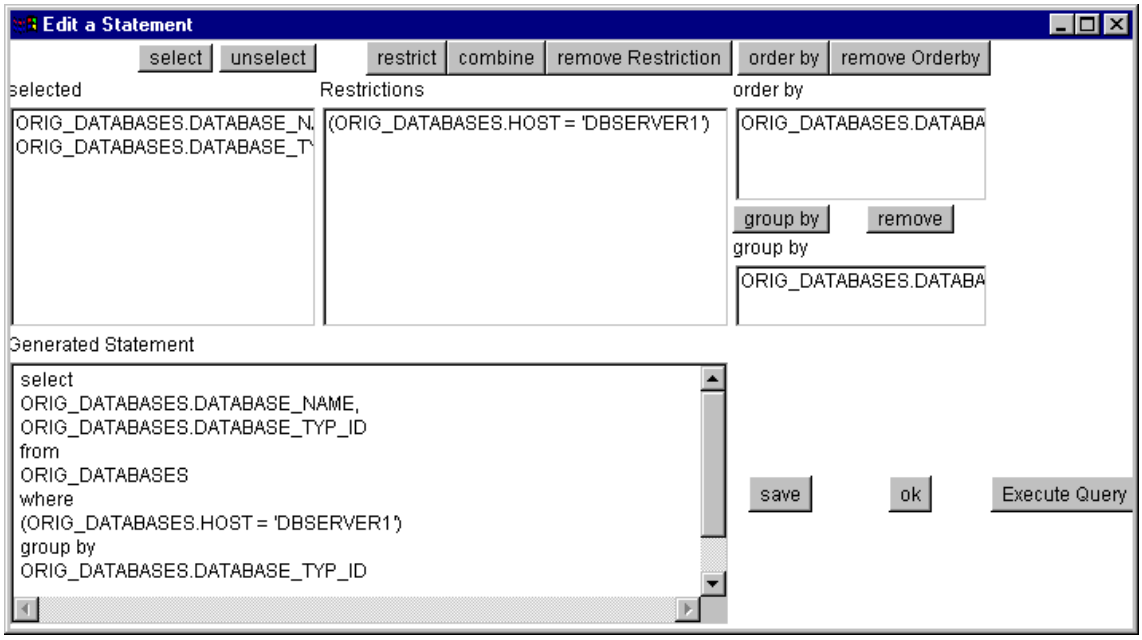

Fig. 8 The window used to generate simple select statements.

Each query may be attached to several nodes. This enables different navigational structures leading to the same data. So users with different preferences concerning navigation can be satisfied. While the first tool is implemented as an application, the second one will run as an Applet as well. This allows database administrators to create their queries for the databases maintained by themselves via Internet.

3. Possible Employment of the System

The main aim of the NDI is to increase the availability of existing databases. The system provides one general user interface for databases of different vendors. It does not bother users with the need for knowledge of each databases existence, content and handling. Users just select what kind of information they need by following the predefined paths built by experienced people (administrators). The system will lead them to the available database queries. So Users do not even need to know anymore, which database answers their queries. They just receive a table containing a result set. If they need to gather further information, they may read the comment or the additional help page belonging to the query they executed. Because of its multilingual implementation and the meta information it provides, the system could be employed for the documentation of databases, too.
The most important use of the NDI is probably the intranet of a multinational company or an international organization. Even public databases, e.g. in environmental protection (Siemonsen 1998) could possibly use the system. Usually such companies and institutions own several “older” databases with very valuable data. Often it is unreasonable to migrate such systems to more up to date database engines, since there are still some applications running, that require the existing database engines. Up to now, very often one has to use special, database-dependent, difficult to use proprietary tools, which are very often platform-dependent, in order to extract information from these databases.

4. Conclusion and Outlook

In this paper we described the concept and prototype implementation of a Java based and therefore platform- independent database interface on the World Wide Web. This interface combines navigation and database queries. Both are enriched with multilingual metadata. Of course the system will not be satisfying for people being used to access databases directly with SQL in a proprietary environment. But it works very well for a great percentage of common users that have to execute nearly the same set of queries on several databases every day. Furthermore it supports users with their search for information, of which they do not know where it is stored.

Database administrators can generate statements satisfying users needs and provide them by using the interface.

While still being a prototype, the system can be improved in several ways. In order to create a more general way to run the server component, we intend to implement the server service described above as a servlet. Then each webserver capable of running Java servlets could be extended with the Navigation and Database Interface (NDI). At the moment only database querying is provided. If we decide to manage sessions and implement user registration, it is also feasible to allow data manipulation. In the near future we plan to extend the system in regard of URL managing. Then not only database queries can be attached to nodes, but arbitrary URLs pointing to content related data sources. This will allow a navigation in an even more open information space. In contrast to a search engine the system would deliver URLs that have been reviewed and sorted by experienced people.

Literature

[Benn 1998]
Benn, W. Gringer, I. : *Zugriff auf Datenbanken über das World Wide Web*, Informatik Spektrum 21:1-8(1998), Springer Verlag 1998

[Codd 1990]
Codd, E.F. : *The Relational Model for Database Management: Version 2. Reading*, MA: Addison-Wessley 1990

[Date 1997]
Date, C.J., Darwen,H.: *A Guide to SQL Standard. 4. Auflage. Reading*, MA: Addison-Wessley 1997

[Reese 1997]
Reese, G.: *Database Programming with JDBC and Java*, O'Reilly 1997

[Siemonsen 1998]
Siemonsen, K. Heuer, A. Engel, T. and Meinel C.
Ein Web-basiertes Navigationssystem für allgemeine, heterogene Datenbanksysteme, Proceedings of 1. Workshop *Hypermedia im Umweltschutz*, Ulm (1998) ,p. 83-86

[Sridharan 1997]
Sridharan, P. : *Advanced Java networking*, Prentice Hall 1997, p.67-75