Using Visual ER Query Systems in Real World Applications

Peter Rosengren

Swedish Institute for Systems Development
Electrum 212
S - 164 40 Kista
Sweden
Telephone: #46-8-752 16 00
Fax: #46-8-752 68 00
email: peterros@sisu.se

Abstract. This paper presents experiences from using a visual ER-based query system within large organisations. The system, Hybris, has been used by end-users for retrieving information from relational databases. Hybris exploits ER-schemes, graphical querying and an integrated dictionary browser as important facilities in the user interface.

1. Introduction

Today, organisations need to make more efficient use of existing information resources. Large investments have been made in building databases and the information stored in databases represents a valuable asset. Unfortunately so far it has been difficult for end-users to access the information they need. Several reasons can be given for that - the structure of a large complex database is hard to understand and interpret, and query languages are often difficult to learn and use.

The objectives of the Hybris project have been to investigate the requirements for an end-user tool to retrieve information from large, complex corporate databases and to define and implement such a tool. The solution taken in the Hybris project was to use an ER-based approach for the graphical user interface.

The Hybris software was put into operation 1990 and have been tested and evaluated by approximately 100 users at Swedish Telecom working with a production management database and at Sweden Post with 15 users working with a market and sales support database. Also several prototype installations have been made at other sites.

The purpose of this paper is to report about our experiences from designing and evaluating a visual ER query system. The paper is organised as follows. The next chapter surveys related research on visual ER query systems. Chapter 3 gives an overview of Hybris and explains its functionality. In chapter 4 we present results of

three different evaluations of Hybris. Conclusions are given in chapter 5 which also points out some important issues for future research.

2. Related Work

Research on ER-based query interfaces started in the early eighties. Two of the first research systems to use an ER-schema as a main component in the user interface were GUIDE [23] and gql/ER [24]. Another pioneering approach for database retrieval was presented by Fogg who combined ER-models with a browsing interface [8]. The list of early contributions can be made quite long, some examples are [4, 7, 16, 18].

More recent research system make use of other data models such as the universal relation model [13] or extensions of the ER-model with specialisation, generalisation and aggregation. Examples of visual query systems exploiting extended ER-models are Candid [20], Pasta-3 [14, 15], and Super [1]. Czedjo et al also describe a query interface based on extended ER-models [6]. Visual querying and browsing interfaces for object-oriented data models have also been proposed [17, 22]. The reader is further referred to surveys of contemporary research [2, 6, 14, 21].

Although there have been many solutions suggested, as Batini et al point out, surprisingly few reports about user satisfaction have been produced [2].

3. Lessons learned from Hybris

In this section we give an overview of Hybris and its functionality. We will also discuss various design decisions and explain how they relate to previous research system. It should be noted that it is beyond the scope of this paper to present all details about Hybris, instead we want to point out some issues that are important to consider when implementing a visual ER query system in real world applications.

The basic architecture for a visual ER query system is, as a result of extensive research, quite well understood by now. Basically, such a system requires three components - *schema browser*, user environment for *query formulation* and *database query translator* [4]. The way these three components have been implemented differs between various systems.

3.1. Large Schema Navigation

We have learned that a large ER-schema requires good mechanisms for structuring the overall schema into different views. How to deal with views have been described by several authors. Wong and Kuo suggest a filtering approach based on *relevance ranking* [23]. However, it is not clear how to do such an objective relevance ranking since entities may be of varying importance depending on users and their tasks. Catarci and Santucci suggest two solutions - predefined *personalised schemes* that can be fetched from a library and *top-down browsing* to allow a user to view the schema at different levels of detail [4].

Our approach have been to design schema views together with users, where each view reflects concepts important for a well-defined task. The figure below shows the Task view at Swedish Telecom. In that view all entity classes relevant when querying about tasks are shown.

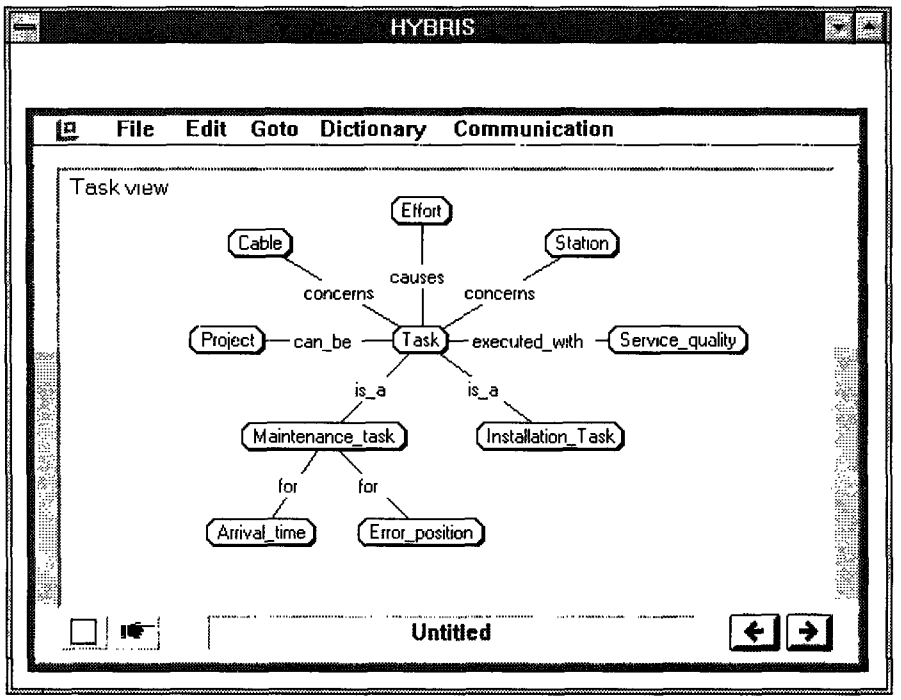

Fig. 1. A task-oriented view designed together with users.

To further simplify navigation within large schemas Hybris allows the user to type in a search string. Hybris then locates entities and attributes that match the search string and displays an appropriate view containing the entity searched for. Similar features exists in the Pasta-3 system [15] and Graqula [21].

3.2. Integrated Dictionary Browser

Our experience from evaluating Hybris is that schema browsing facilities are necessary but not enough to support a user's understanding of the database contents. The databases we have been working with so far have contained between 40 and 80 entity types which have had an average of 8-10 attributes each. In many cases there has also been multiple relationships between entity types.

To simply display the ER-schema and let the user browse through it is not enough to convey the semantics of data to the user. We therefore have designed another browsing facility - a dictionary browser where users can browse through meta data.

Figure 2 and 3 show how a user can browse through the dictionary in a hypertext fashion. The user does so by clicking on attribute names, entity names or relationship names.

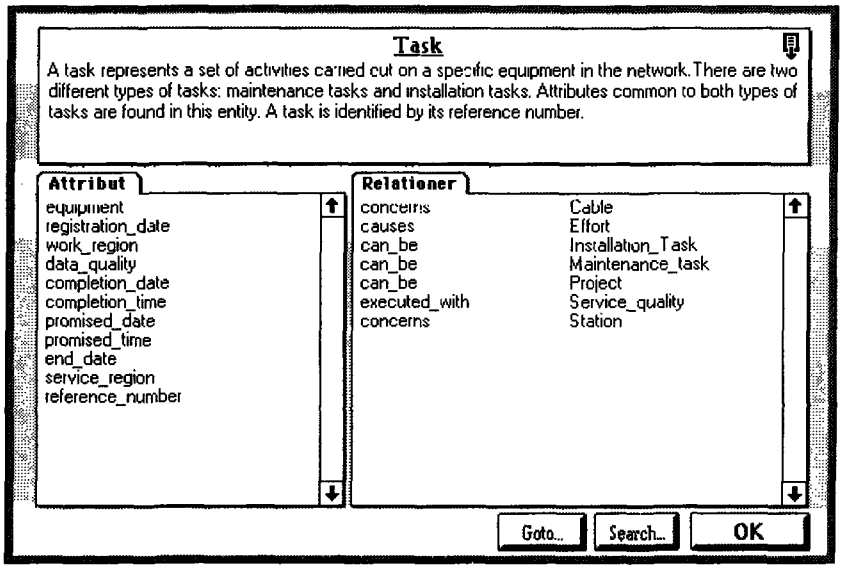

Fig. 2. The Integrated Dictionary Browser

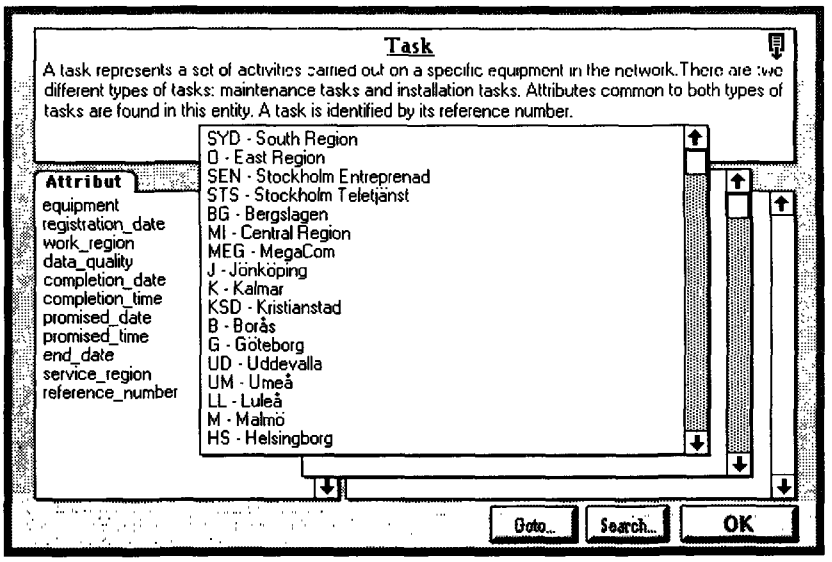

Fig. 3. The Integrated Dictionary Browser (cont.). The user has clicked on an attribute and then on the value domain for that attribute. Explanations and descriptions are shown to the user.

The dictionary browser can be accessed at anytime, during schema browsing or query formulation. A user can look at entity explanations, attribute definitions, relationship constraints etc.

The dictionary browser also solves another problem apparent in several research systems - *cognitive overload.* In order to represent as many semantic aspects as possible many systems end up with a visually too complex layout [3, 6, 20]. Our approach has been to visualise a plain ER-schema to the user and instead put extra semantic information in the dictionary browser.

3.3. Simplified Query Formulation

Most visual ER query systems imply that the user works in the following steps - the schema is browsed in order to understand the database contents, a subschema is selected for further investigations, restrictions are added for attributes belonging to relevant entities. Finally the query is submitted to the database.

Hybris differs from most research systems in that we do not separate schema browsing and query formulation. We wanted to give the user a *mental model* [9] of the schema as being the actual database. Queries are formulated by directly pointing at entities and restricting attributes in the schema. Immediate feedback is given in the ER schema. See figure 4.

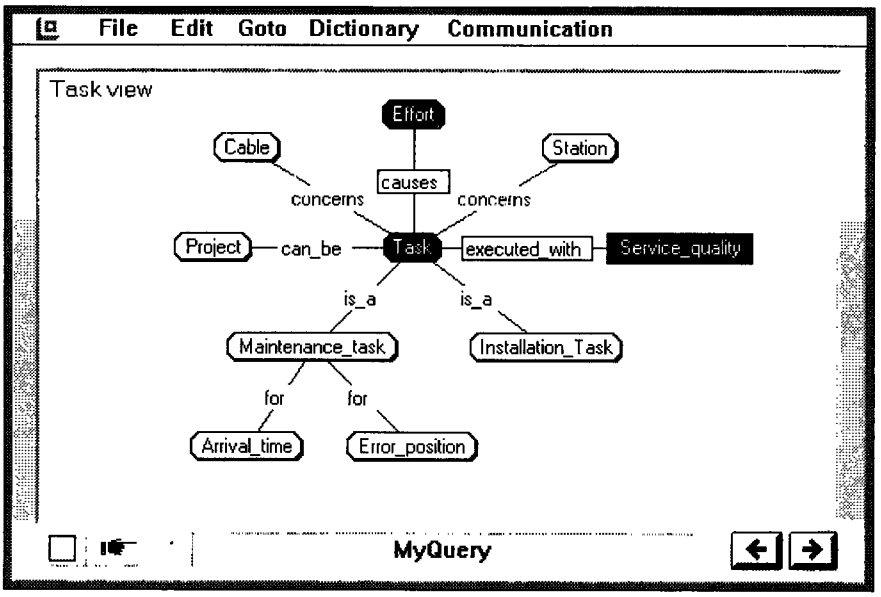

Fig. 4. Direct query feedback is given in the schema. When an entity is displayed as a rectangle it shows the user that restrictions have been added to some attribute of the entity.

This approach showed both advantages and disadvantages. The advantage were that it gave the users a straightforward and easy-to-learn model of the system. The disadvantage was that it limited the expressive power of the query language. An example is a query where an entity type plays different roles. This would require duplication of the entity directly in the schema which proved to be a difficult usability problem to solve, because it would produce clutter and confusion [21]. If we would have used two separate windows, one for schema browsing and another for query formulation this particular problem could have been solved. But then, on the other hand, the user interaction would not have been as straightforward since it would have required an extra step for the user to move items from the schema window to the query window.

One problem in real world applications is the use of cryptically numerical codes to represent attribute values. This seriously effects the ease-of-use when formulating queries. Users might very well know what they are looking for, e.g. sales in the region "Stockholm North", but if they do not know the code value for "Stockholm North" they cannot get the information.

In Hybris, we solved the problem in the same way as in GUIDE [23], by supporting value domains at the user interface level, i.e. when querying the user can select values from a list of explaining text strings, see figure 5. Hybris then automatically translates it into the correct database code.

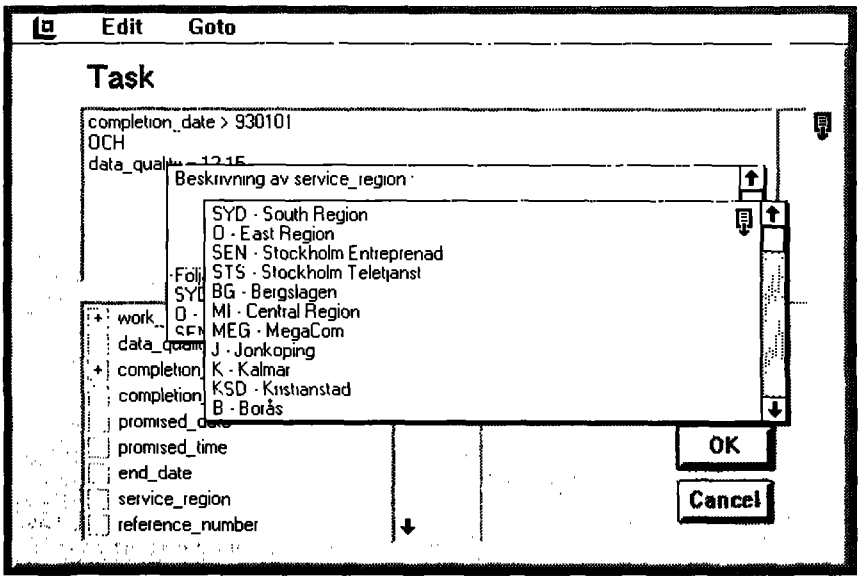

Fig. 5. The Integrated Dictionary Browser is always available. This figure shows a user that consults it for code value explanation during query formulation.

Hybris also make extensive use of value domains and value rules to ensure that a user does not enter an illegal value for the database search.

3.4. Query Translation

So far we have described how the schema browsing facilities have been implemented in Hybris and how it was extended with an integrated dictionary browser. We have also explained how query formulation works. The final component in a visual ER query system is the database query translator.

To be able to translate the visual query into SQL, Hybris uses an internal dictionary. In the dictionary the underlying database schema, the ER schema and the mapping between the ER schema and the database schema are stored. The SQL translator reads a textual description of the visual query and uses dictionary information to construct the appropriate SQL-query.

As is pointed out by Hohenstein transformations of conceptual ER-models into database schemas has been quite well investigated, but less attention has been paid to transformation of query languages [11]. If visual ER query systems are to be used in real world applications the queries formulated in the ER-tool have to be translated into SQL queries since this is the standard query language for databases.

Few published papers deal with this issue. Czedjo et al mention that they translate graphical ER-queries into SQL [6]. Hohenstein has proposed an algorithm for automatic translation of an ER-query language into SQL [11]. In both cases it is assumed that for each entity set there exists one relation in the database.

We make the same assumption but our experience has shown that this restriction needs to be relaxed. While the one-to-one assumption might be true for a newly designed database it is not true after one or two years in operation. The database structure changes due to optimisation issues but also because of new user requirements. The changes cause a denormalised database structure, for instance the entity set "Customer" might be divided into three underlying database tables - "Northern Region Customer", "Southern Region Customer" and "Other Customer".

3.5. Implementation

Two versions of Hybris have been implemented. The first one runs on Macintosh and has been built with HyperCard. The second version runs under Windows and is implemented with the tool PLUS. The query translator is written in C with the tools LEX and YACC. The database communication is implemented differently depending on specific applications.

4. Evaluations

Hybris has been evaluated in three different studies. In this chapter we will give a summary of the different studies. The first study was a usability study at Swedish Telecom that was conducted during the first three months of use [19]. The application

was a production management database which was accessed to measure the quality of service offered by Swedish Telecom to their customers and to plan for future work and investments.

Prior to Hybris the users accessed the database either through predefined standard reports or through SQL for unpredictable, ad-hoc queries. However, the use of SQL at this time was limited. A study had shown Swedish Telecom that only 20 percent of the users who had been taught SQL in a 3 day course, were actually using SQL three months after the course.

Most users had very little experiences of using a graphical user interface.

The result of the Hybris evaluation was promising:

- Inexperienced users could quickly and easily get access to information needed in different situations.
- Users felt that they had access to "new" information because the overview provided by the ER-schema made them aware of new relations among data that they didn't know existed.
- Many errors in the database was discovered by users once they had the opportunity to query about their own data.
- According to users they felt a tool like Hybris would lead to increased efficiency within the organisation. They gave two reasons for that; 1) the quality of information increased since it was possible to verify the accuracy of data at the finest level of transactions, 2) they could focus their attention on their task instead of on how to handle a database query language.
- The ER-schema gave a visual view of the database which made the database less abstract and difficult to comprehend.

Sahlin concludes that Hybris gives powerful support for retrieving information and that it is especially well-suited for infrequent and novice database users [19].

The second study was a laboratory study by Karlgren and Wideroth [12]. The subjects where students at the Department of Psychology at University of Stockholm and the application used was a hospital database.

Karlgren and Wideroth showed that Hybris worked well for queries with simple logic, but that the subjects had problems when a query involved complex logical formulas, especially when the connectives AND and OR have to mixed in nested statements. The same was also noticed from use at Swedish Telecom.

This is an inherently difficult usability problem to solve since logic is difficult to master. However, there are certain things that can be done to ease the burden on the user.

The following is an example that user had many problems with. Consider the query "Efforts during April for Regions 15, 17 and 25" where the restrictions should be formulated:

```
date BETWEEN 910401, 910430
AND (region = 15 OR region = 17 OR region = 25)
```

Most users forgot the parenthesis which of course lead to an unexpected answer:

```
date BETWEEN 910401, 910430
AND region = 15 OR region = 17 OR region = 25
```

This usability problem was solved by changing the Hybris interface and allowing several values as input for a constraint:

```
date BETWEEN 910401, 910430
AND region = 15, 17, 25
```

Another improvement is suggested by Kuntz and Melchert who use indentation to show the user the different levels of nesting in a complex restriction [14].

Another problem discovered during the second study was that users were confused when they had to link entities together that were not located in the same view. This taught us the importance of having a good design methodology for ER-based query systems. The different views shown in the system have to be designed together with users and with respect to the different tasks a user may have.

A third study was done by Hogedahl and Zakizadeh for a market and sales support system at Sweden Post. Their study showed that Hybris gave the users an increased understanding of the database contents [10]. The tool also made it easier to formulate spontaneous ad-hod queries and made the users less dependent on specialist support functions, such as controllers.

5. Conclusions and further work

The main contribution of this paper is to present experiences of using a visual ER query system within large organisations. As was mentioned in chapter 2 very little has been reported on real world applications of visual ER query system.

We have also presented the Hybris system and the design decisions taken when implementing the system. Especially the integrated dictionary browser is a novel approach to supporting users understanding of the database contents.

From our experiences with Hybris we are confident that visual ER query systems are a very promising approach for increasing usability of large corporate databases. Moreover, it is an approach that has shown to be applicable in real world environments. This should encourage further research in this area.

Some important issues are:

- Improved techniques for schema visualisation. Catarci et al have started work in this area by experimenting with *iconic* interfaces [5].
- Methodology for designing visual query applications.
- Security issues, such as whether access rights should be controlled at the database level or at the ER level.
- Tools for maintaining the query system once the database structure starts to change.

We also need to study a user's overall retrieval environment. So far research have considered ER-based retrieval from structured databases. However, as part of their work, users also need to retrieve information from other sources such as a text retrieval system, a picture archive or a hypertext system.

We should also adopt an organisational perspective on visual ER-query systems not only viewing them from a single-user perspective but study which support large organisations with many users with different levels of expertise and tasks need.

6. Acknowledgements

I acknowledge the invaluable work done by my colleagues in implementing Hybris. First of all I would like to thank Stefan Paulsson without whom Hybris would never had existed. Many thanks to Jesper Lundh who worked with the Macintosh version and Peeter Kool with the Windows version. Special thanks to Ulf Wingstedt, SISU, for his encouraging support and stimulating discussions on ER subjects. The original idea for Hybris came from Björn Nilsson, SISU. Special thanks to Bertil Andersson, Swedish Telecom, for his never ending support for this work.

Finally, many thanks to all Hybris users whose enthusiasm have been the main source of inspiration for this work.

This work has been sponsored by Swedish Telecom and Sweden Post.

7. References

1. A. Audinno, Y. Dennebouy, Y. Dupont, E. Fontana, S. Spaccapietra, Z. Tari. "Super: A comprehensive approach to DBMS Visual User Interfaces", Ecole Polytechnique Fédérale, Lausanne, Switzerland.
2. C. Batini, T. Catarci, M. F. Costabile, S. Levialdi, "Visual Strategies for Querying Databases", IEEE Workshop on Visual Languages, 1991, pp 183-189.
3. D. Bryce, R. Hull, "SNAP, a Graphics-Based Schema Manager", Proceedings of the 2:th IEEE International Conference on Data Engineering, pp 151-164, 1986.

4. T. Catarci, G. Santucci, "Query by Diagram: A Graphic Query System", Proceedings of the 7:th International Conference of Entity Relationship Approach, pp 157-174, 1988.

5. T. Catarci, A. Massari, G. Santucci, "Iconic and Diagrammatic Interfaces: An Integrated Approach", IEEE Workshop on Visual Languages, 1991, pp 199-204.

6. B. Czejdo, R. Elmasri, M. Rusinkiewicz, D. Embley, "A Graphical Data Manipulation Language for an Extended Entity-Relationship Model", IEEE Computer, March 1990

7. R. Elmasri, J. Larson, "A Graphical Query Facility for ER Databases", Proceedings of the 4:th International Conference of Entity Relationship Approach, 1985.

8. D. Fogg, "Lessons from a "Living in a Database" Graphical Query Interface", Proceedings ACM Sigmod, 1984.

9. D. Genter, L. Stevens, "Mental Models", Hillsdale, New Jersey, Lawrence Erlbaum Associates.

10. H. Hogedahl, H. Zakizadeh, "PimWin - en fallstudie vid Posten", TRIAD-rapport U3, 1993, SISU, Electrum 212, 164 40 Kista, Sweden.

11. U. Hohenstein, "Automatic Transformation of an Entity-Relationship Query Language into SQL", Proceedings of the 8:th International International Conference of Entity Relationship Approach, 1989.

12. K. Karlgren, M. Wideroth, "En utvärdering av Hybris", SISU Technical Report 12.

13. H. J. Kim, H. F. Korth, A. Silberschatz, "PICASSO: A Graphical Query Language", Software - Practice and Experience, March 1988.

14. M. Kuntz, R. Melchert, "Pasta-3´s Graphical Query Language: Direct Manipulation, Cooperative Queries, Full Expressive Power", Proceedings of the 15:th International Conference on Very Large Databases, 1989.

15. M. Kuntz, R. Melchert, "Ergonomic Schema Design and Browsing with more Semantics in the Pasta-3 Interface for E-R DBMSs", Proceedings of the 8:th International Conference of Entity Relationship Approach, 1989.

16. J. Larson, J. B Wallick, "An Interface for Novice and Infrequent Database Management System Users", AFIPS Conference Proceedings, National Computer Conference, 1984.

17. M. Leong, S. Sam, D. Narasimhalu, "Towards a Visual Language for an Object-Oriented Multi-Media Database System", Visual Database Systems, T. L. Kunii (editor), Elsevier Science Publishers B. V. (North-Holland), IFIP 1989.

18. T. R Rogers, R. G. G Cattell, "Entity-Relationship Database User Interfaces", Proceedings of the 6:th International Conference of Entity Relationship Approach, 1987.

19. C. Sahlin, "Erfarenheter från användning av Hybris. - Ett multimedia hjälpmedel för navigering i Televerkets PULS databas", Försvarets Forskningsanstalt, Linköping, Sweden, report 90-5991/S. In Swedish.

20. M. Schneider, C. Trepied, "A Graphical Query Language Based on an Extended E-R Model", Proceedings of the 8:th International Conference of Entity Relationship Approach, 1989.

21. G. H. Sockut, L. M. Burns, A. Malhotra, K.-Y. Whang, "Graqula: A graphical query language for entity-relationship or relational databases", pp 171-202, Data&Knowledge Engineering, vol 11, no 2, october 1993,

22. F. Staes, L. Tarantino, A. Times, "A Graphical Query Language for Object-Oriented Databases", IEEE Workshop on Visual Languages, 1991, pp 199-204.

23. H. Wong, I. Kuo, "GUIDE: Graphical User Interface for Database Exploration", Proceedings of the 8:th International Conference on Very Large Databases, 1982.

24. Z. Q Zhang, O. Mendelzon, "A graphical query language for entity-relationship databases", Entity-Relationship Approach to Software Engineering, pp 441-448, 1983.