5

Diagrammatic Vs Textual Query Languages: A Comparative Experiment

T. Catarci and G. Santucci
Dipartimento di Informatica e Sistemistica
Università degli Studi di Roma "La Sapienza"
Via Salaria, 113 - 00198 Roma, Italy
(catarci/santucci)@infokit.dis.uniroma1.it

Abstract

The significance of usability as one of the most important system characteristics is widely recognized in all the application fields, including databases. This also implies the involvement of the user in the activities of design and testing of any interface. However, very few studies aiming at validating the usability of a system have been proposed in the database field, in contrast to many other computer science areas. In this paper we present an experiment comparing, from the point of view of the ease-of-use, a novel visual query language, namely QBD*, against a well-known traditional textual language such as SQL. The results of the experiment confirmed the superiority of the QBD* approach, which is based on a conceptual data model, closer to the user view of the reality than the relational model, a visual representation of such a model, more attractive and graspable than a textual list of table names, and direct manipulation commands, having a syntax much more easier than the SQL one.

1 INTRODUCTION

Several classes of users, both professionals and naives, need to be able to access databases for extracting meaningful information. The traditional interaction style with databases relies on using textual query languages, such as SQL (Date 1987). This kind of language is hard to use by non-programmers. Thus, alternative query modalities have been proposed in the literature, often based on using visual representations and direct manipulation interaction mechanisms (see (Batini et al. 1991) for a survey of visual query languages, and (Shneiderman 1983) for the definition of direct manipulation). All the proposed approaches emphasize in principle the role of the final user in the design life-cycle of the query systems. Also, the significance of usability as one of the most important system characteristics is widely recognized ((Nielsen 1993, Bevan Macleod 1993), see also a recent panel on user interfaces at VLDB'94 (Spaccapietra 1994)). However, very few studies aiming at testing and validating the usability of a system have been proposed in the database field.

(Reisner 1988) is an interesting paper recalling the majority of the experiments done on the ease of use of traditional query languages, mainly SQL and QBE. Other earlier surveys on experiments are (Shneiderman 1978, Shneiderman 1980, Thomas 1977), and a recent one is reported in (Ahlberg Williamson Shneiderman 1993). A common goal of such experiments is to provide a quantitative estimate of the ease of use. The general approach in order to do this is:
1. define precisely what one is to measure;

S. Spaccapietra et al. (eds.), *Visual Database Systems 3*

2. develop a task for users to perform;
3. measure relevant parameters of user performance (i.e. error rate, time to complete a task, etc.).

Unfortunately, measuring ease of use is not as precise as measuring the quantity of a substance in a physics experiment, it involves cognitive activities and is determined by a large number of variables. To capture at least some aspects of ease of use, experimenters have developed a number of different tasks (Reisner 1988). The most common are query writing tasks, in which subjects are given questions in natural language and asked to write the corresponding query language statements. The tasks can be used in different kind of tests to capture different facets of ease of use. Two different classes of experiments have been reported in the literature: one aiming at evaluating the ease of use of a single language, the other aiming at establishing which is the easiest to use among two or more languages.

Our approach falls in the second category. We performed a series of experiments in order to compare, based on their usability, QBD*, a diagrammatic query language developed in our Department (Angelaccio Catarci Santucci 1990a; Angelaccio Catarci Santucci 1990b) and SQL. Analogously to previous experiments, also in our case the task was query writing, and the actual usage experiment was preceded by a teaching period, not part of the variables observed in the experiment. We tried to make as many of the conditions of the experimentation of the two languages the same as we reasonably could (i.e., teaching method, exam questions, exam procedure, method of scoring). We also examined some background information about the various users in order to divide them into equivalence classes and to assign to each language the same set of classes. This is commonly done in experiments to determine whether factors other than the one being measured might influence the experiment results.

It is worth noting that, as far as we know, this is the first rigorous study of comparison between a visual query language and a well-known traditional language, such as SQL. The results of the experiment confirmed our belief (see (Catarci et al. 1994)) that a visual approach is particularly suited for medium and naive users, and for certain types of queries.

The paper is organized as follows. Section 2 provides a short description of the QBD* system (we assume the reader familiar with SQL). Section 3 introduces the evaluation techniques that have been adopted in the experiment. Section 4 describes the experimental framework and reports the numerical results. In Section 5 a discussion of these results is presented. Finally, conclusions are drawn in Section 6.

2 QBD*

The QBD* (Query By Diagram) system is a visual query system based on a diagrammatic representation of the Entity Relationship schema (Chen 1976) describing the underlying (relational) database. It balances a high expressive power with a noticeable facility of use. Concerning the expressive power, it has been proved to be relationally complete; moreover it includes a significant class of recursive queries (transitive closure) and handles an extension of the relational algebra set-oriented operators that we call *generalized set-oriented operators* (Santucci Sottile 1993). Through those operators it is possible to compute set-oriented operations on any pair of relational tables sharing the same identifier. The ease of use has been reached through a fully graphical environment. In that environment the user interacts with the system mainly with a mouse-like device, using the keyboard only when necessary. To increase the friendliness of the system a set of *helping facilities* has been devised. Among them, the most important ones are top down browsing and schema transformation. The former allows for browsing a library of top-down refinements documenting an ER schema at different levels of abstraction. While browsing through the top-down schemata, the user can easily locate the fundamental concepts and links. Top-down browsing may be also used to locate the subschema of interest. The schema transformation allows for converting a schema into a new version closer

to the query. As an example it is possible to collapse a path in the schema giving rise to a new concept or to replace an ISA relationship with the child entity.

Nevertheless, the helping facilities of QBD* have not been posed at the user disposal in the experiments. In order to equate it with SQL, the user interacted with the query module only.

The general structure of the QBD* query is based on the location of a distinguishing concept, called *main concept* (entity or relationship), that can be seen as the entry point of one or more subqueries; these subqueries express possible navigations from the main concept to other concepts in the schema. The attributes belonging to each subquery are determined by the following strategy: the attributes of the main concept are automatically considered (unless explicitly deleted by the user), while the other ones are shown in the result only if requested by the user. The presence of a main concept associates a type with each subquery: as an example, if the main entity is the entity person, the result of the query will be a set of persons (possibly enriched with attributes coming from other concepts), no matter what kind of subsequent operations the user performs to specify it.

The subqueries can be combined together by applying several set-oriented operators on two or more subqueries derived from the same entity. The structure of the involved subqueries being, in general, different, it is impossible to perform simple set-oriented operations, and a more general approach is needed (see (Santucci Sottile 1993) for the specific solution adopted within QBD*).

Once the main concept has been selected, two different types of primitives are available for navigating in the schema. The first one allows the user to follow existing paths on the schema (see Figure 1); the other one (see Figure 2), is used for comparing two concepts by building a new path (relationship) on the schema. Comparison conditions, as well as generic conditions on the attributes, are expressed by means of a simple window mechanism including a suitable set of icons. Roughly speaking, we can say that a path on the schema corresponds to an ordered sequence of natural joins[1] among the pairs <entity, relationship> constituting the path, followed by a final selection and projection. The explicit presence of relationships releases the user from the need of looking for concepts like "foreign keys" and the system can perform in the right way the join operations. On the other hand, comparing two concepts via a new relationship corresponds to a theta-join between the two involved entities, and it is up to the user the specification of the theta-condition.

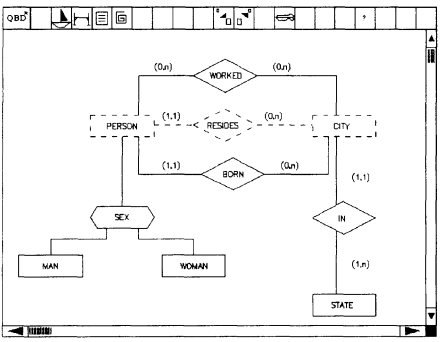
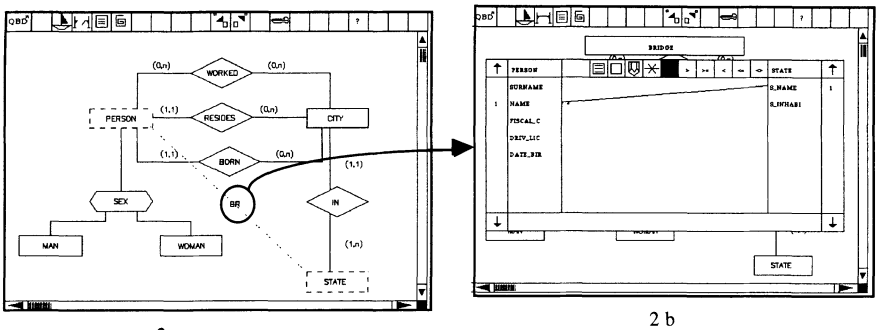


Figure 1 Schema navigation through existing paths.

Queries involving transitive closure are expressed by a visual approach similar to the one in Figure 2b, the difference being the pre-selection of a particular icon, which signals the

[1] We mean here the natural join and theta-join operators as defined in relational algebra (Codd 1972).

beginning of a recursive session. However because of standard SQL does not support recursion, this kind of queries has not been included in the experiments.



2 a 2 b

Figure 2 Schema navigation through the building of a new relationship (2a) on the basis of attribute comparisons (2b).

3 EVALUATION TECHNIQUE

Recently, a lot of emphasis has been placed on evaluation techniques, since there are several reasons to evaluate software. People involved in developing software products are interested in evaluations to assist them in making design decisions and to determine whether or not the products achieve the quality measure that must be met. Individuals, who buy the software because they want to use it, need to evaluate the software before purchasing it to check whether it answers key question about usability measures. Even if there is some similarity in the various evaluations, since they are all looking to answer whether the system adequately meets the needs of the user, the above interests are not served by a single evaluation technique. On the contrary, a variety of techniques have been proposed.

The first requirements in order to design a system evaluation are to assess the evaluation purpose, what is being evaluated, who the evaluation is for and, finally, what one hopes to gain. In our case the purpose of the evaluation is to compare two different query systems (a visual one and a traditional one) in order to understand which one is the most easy to use by different user classes. In other words, we are trying to compare the *usability* of the two systems.

Several different definitions of usability exist (Miller 1971; Shackel Richardson 1991; ISO 1991 ISO 1993; Badre 1993; Maissel et al. 1993). In particular, in MUSiC (Maissel et al. 1993) a very comprehensive definition of usability is given as "the extent to which a product can be used with efficiency, effectiveness and satisfaction by specific users to achieve specific goals in specific environments". From this point of view, the usability is the quality of interaction between the user and the overall system. It can be evaluated by measuring three factors: 1) effectiveness, i.e. the extent to which the intended goals of the system can be achieved; 2) efficiency, i.e. the time, the money, the mental effort spent to achieve these goals; 3) satisfaction, i.e. how much the users feel themselves comfortable using the system. It is worth noting that the usability depends on the overall system, i.e. the context, which consists of the types of users, the characteristics of the tasks, the equipment (hardware, software and materials) and the physical and organizational (e.g. the working practices) environment.

The elements that make up the quality of interaction need in general to be measured. Since the quality of interaction depends on the context of use, there is no general rule for how measures can be combined. Usually we will have to provide at least one measure for each element of the quality of interaction. In our experiments we aimed at measuring the effectiveness and the efficiency of the query systems. The effectiveness has been evaluated by relating the goals or subgoals of using the system to the accuracy and completeness with which these goals can be achieved. In our case the main goal is to extract information from the database by asking queries and the accuracy in achieving such a goal has been measured in terms of the user correctness rate when writing such queries.

Measures of efficiency relate the level of effectiveness achieved at the expense of some resources, like mental and physical effort, time, financial cost, etc. In principle, we can consider both the user and the organization point of view, while for our experiment purposes we have measured only the former in terms of time spent carrying out the task. In the latter, we could measure the economic efficiency as the effectiveness in relation to the total cost, which is the labor costs of the user's time, the cost of both resources and equipment used, and the cost of any training required by the user.

Moreover, other kinds of measures can be evaluated. Measures of satisfaction describe the comfort and acceptability of the overall system used by people. Some measures are related to the need to quantify learnability and flexibility, which can be assessed by measuring effectiveness, efficiency and satisfaction across a range of contexts. The learnability of a product may be measured by comparing the usability of a product handled by one user, keeping in mind his/her knowledge and experience. The flexibility of a product handled by users to carry out different tasks can be assessed by measuring usability in different contexts.

The method we adopted in order to measure the effectiveness and efficiency of the systems was the so-called *observational evaluation*, one of the most well known methods for evaluating usability (Macleod 1992; Bevan Macleod 1993). Such a method involves real users that are observed when interacting with the system and offers a broad evaluation of usability. We may either apply the observational evaluation by direct observation or record the interaction between users and system. The recording (done by video camera) is more valuable, since it allows to store a lot of information, for examples the critical points during the interaction (when the user has to consult the manual, when and where s/he is blocked, etc.), the time a user spends to perform a task, the mistakes a user makes, and so on. However, it is also very expensive, not for setting up the equipment, but for the time required to analyze the recorded data. For this reason we preferred to rely on a very careful direct observation.

4 THE EXPERIMENT FRAMEWORK

The goal of this section is to describe the overall experiment framework, i.e., the involved users, the queries we chose, the training the users were given, and the experiment results.

4.1 The Sample Users

The subjects involved in the experiment were undergraduate students, secretaries and professionals for a total of 102 people. They were classified in three categories:

1) Naive users, i.e., persons having little or no knowledge about computer science, for an amount of 56 people;
2) Intermediate users, i.e., persons having a general knowledge about computer science but naive about databases, for an amount of 30 people;
3) Expert users, i.e., persons having a precise knowledge about databases, for an amount of 16 people.

The subjects were assigned to one of two groups: one involved in QBD* tests, the other one in the SQL tests. Each group was composed by 28 naive users, 15 intermediates, and 8 experts. We tried to make the groups as homogeneous as we could in terms of relevant subjects' characteristics, such as educational level, background preparation, employment, sex, and age.

It is worth noting that most subjects were naive users, being the QBD* environment mainly thought for non experts. Nevertheless, in order to get results also on intermediate and expert users, we analyzed the performance of each category separately, as explained in Section 4.4.

4.2 The Query Classes

The users were expected to answer queries of increasing complexity. At this aim we defined a metric for evaluating the complexity of the queries. Note that here we are not dealing with the expressive power of the queries: with the word complexity we mean the mental effort required to the user to compute the query.

After a discussion involving several experts in query languages, we assigned an initial weight to each relational operator and we used such weights to compute the complexity of a set of queries, slightly adjusting the initial weights on the basis of the total score we got for the measured queries. In this phase, looking at the SQL query structure, we decided to add additional weights taking into account the increase of complexity for the presence of subqueries. The weight is proportional to the subquery depth. In Figure 3 the correspondence among relational operators and weights is shown.

Operator	Weight
Identity/Projection in a (Sub)query	$n^{1.5}$ (n=depth level)
Renaming	3
Selection	$n^{1.2}$ (n=number of conditions)
Complement	1.5
Self join	3.5
Natural join	1.5
Theta join	1.5

Figure 3 Adopted weights in measuring query complexity.

In Figure 4 we show two examples of the application of the above metrics.

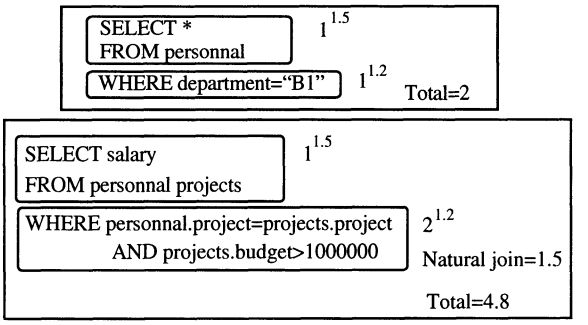


Figure 4 Evaluation of two sample queries.

The strategy we adopted for evaluating the query complexity is based on the expertise of the people asked for designating the weights rather than on a formal approach. Nevertheless, the results we got are quite reasonable and do not affect, directly, the comparison between the two environments. In fact, we used the introduced metric to determine three sets of six queries, of low, medium, and high complexity and the QBD* and SQL sets of users were asked to answer the same sets of queries. Moreover the experiment gave us a significant feedback on the goodness of our metric: the more the query weight increased, the more the users took a large time and made a high number of mistakes in computing them (see Section 4.4). In general, we conjecture that there is a significant relationship between the independent variables, user skill and query classes, and the visual query language used.

More precisely, the naive users (SQL+QBD*) were exposed only to the low complexity set of queries, the intermediate users were exposed to the low and medium complexity sets, the expert users were exposed to all the queries.

Each set of queries was composed by six queries. In particular, the low complexity queries consisted of selections on single relational tables (single entities in QBD*), with selection conditions as follows: simple positive (queries D1, D2, D3), simple negative (D5), multiple in conjunction (D6), multiple in disjunction (D4). The medium complexity queries comprised join operations (path navigations in QBD*) with different join and selection conditions. Queries D1 and D3 were simple equijoins involving two relational tables, D2 was a join with a simple positive selection condition on the join result. Queries D4 and D5 were similar to D2, but with a negative condition. Finally, D6 was the only query of this group containing a join on three relational tables. High complexity queries comprehended both set operations and complex joins. In particular, D1 was a join among several tables; D2, D3, D4, D5 involved set operations with growing complexity conditions; D6 contained a self-join.

4.3 The Learning Phase

Our goal was to compare the effectiveness and the efficiency of the two approaches to query formulation, i.e., a textual declarative language against a graphical query language, discarding each aspect involving the learning time of the two environments. Following this guideline we set up two exhaustive courses about the two environments, giving to the users all the time they were needing to study them.

The SQL course was about the relational model and the SQL language, the QBD* course was about the Entity-Relationship model and the usage of the QBD* environment. In this way the users were totally acquainted with the query language, the underlying model, and, in the QBD* case, also with the environment implementation. In this way, during the final test, they were free of concentrating exclusively on the query formulation.

From a technical point of view, courses were set on a PC based environment and arranged in a self-learning fashion. Each user was provided with a floppy containing her/his course and was left free to study it. Several check points insured us of the correct learning of the subjects.

4.4 The Experiment Results

For each group of users, naives, intermediates, and experts, we measured the number of errors plus the time spent in formulating the query. Thus, the independent variables in the experiment were the type of query language, the users' skill, and the class of queries; the dependent variables were the time in seconds to complete a query and the accuracy of query completion.

In this section we report the Figures summarizing the whole experiment, a more detailed discussion about its results is in the next section. For each group of users we report the results obtained for the corresponding classes of queries (low complexity for naive users, low and medium for intermediate users, low, medium and high for the experts). Figure 5 shows the results outcoming from the SQL user groups while Figure 6 is concerned with the results of the

QBD* groups, were the number of correctly completed queries is expressed as a percentage (out of 100) and the answering time is the average time.

SQL		Low complexity		Medium complexity		High complexity	
	Query #	Right answer	Time (sec)	Right answer	Time (sec)	Right answer	Time (sec)
Naive	D1	99	74	-	-	-	-
Users	D2	96	67	-	-	-	-
	D3	78	88	-	-	-	-
	D4	98	100	-	-	-	-
	D5	93	120	-	-	-	-
	D6	84	145	-	-	-	-
Intermediate	D1	97	50	93	70	-	-
Users	D2	87	40	57	110	-	-
	D3	71	39	68	170	-	-
	D4	95	52	93	130	-	-
	D5	91	70	91	107	-	-
	D6	82	63	70	226	-	-
Expert	D1	100	36	100	102	100	270
Users	D2	100	42	100	84	64	165
	D3	87	36	73	108	100	108
	D4	100	42	100	70	26	183
	D5	100	48	100	126	94	162
	D6	96	57	93	180	92	126

Figure 5 SQL users' results.

5 DISCUSSION

The hypothesis that a graphical query language would perform better than SQL was confirmed by the experiment results.

In Figure 7 we show the comparison of SQL and QBD* in terms of average time (left side) and correct answers (right side) of the low complexity queries per user group. Referring to SQL it is interesting to note that the naive users performed better than the intermediate ones. This is probably due to the fact that the intermediate users were mainly acquainted with some types of procedural languages, then it was difficult for them to learn a declarative language such as SQL. Moreover, they felt themselves confident in solving so easy questions, as a consequence they did not put enough attention in doing it. On the other hand, the naive users were completely unaware of programming techniques, so their mind was free of influences in learning a certain kind of language. Furthermore, even expert users did errors in simple questions. Such errors were mainly due to the need of remembering table names and using a precise syntax. This confirms our believes that even experts can profit from a more flexible query language.

QBD*		Low complexity		Medium complexity		High complexity	
	Query #	Right answer	Time (sec)	Right answer	Time (sec)	Right answer	Time (sec)
Naive	D1	100	26	-	-	-	-
Users	D2	100	40	-	-	-	-
	D3	100	51	-	-	-	-
	D4	83	82	-	-	-	-
	D5	86	80	-	-	-	-
	D6	95	100	-	-	-	-
Intermediate	D1	100	18	100	49	-	-
Users	D2	100	28	100	50	-	-
	D3	100	47	100	49	-	-
	D4	100	52	100	54	-	-
	D5	100	52	100	90	-	-
	D6	100	61	87	122	-	-
Expert	D1	100	17	100	50	100	125
Users	D2	100	27	100	48	100	62
	D3	100	45	100	50	87	90
	D4	100	48	100	52	50	143
	D5	100	51	100	91	87	126
	D6	100	57	87	128	76	136

Figure 6 QBD* users' results

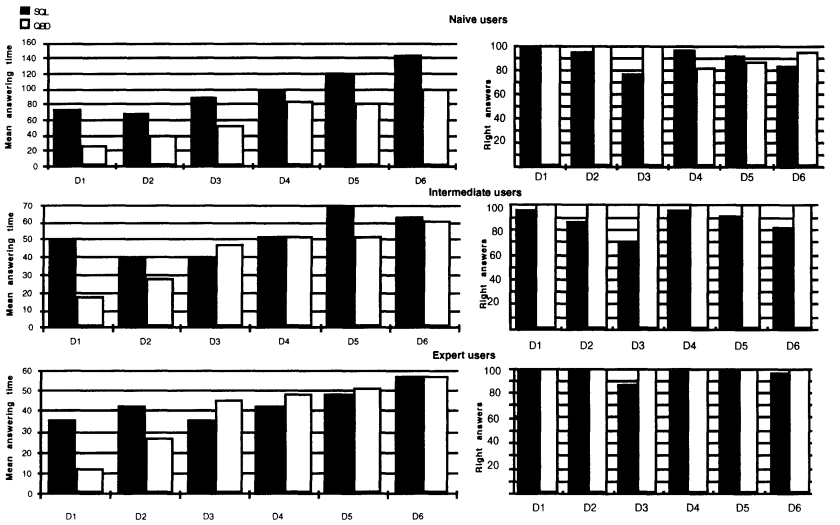


Figure 7 Mean answering times and percentage of correctly completed queries for low complexity queries

Referring to QBD*, we noticed that the fourth query, involving the use of the "OR" connective between the selection conditions, gave the most of the problems to the users. This is certainly due to the fact that the system makes use of some default assumptions that can be modified by the user. In this case the default consisted in considering all the selection conditions in conjunction. This means that if the user is interested in using the "AND" connective s/he does not need to specify it, while he should do it for the "OR", but s/he probably tends to forget this different treatment of the two connectives. We noticed that, in general, the use of the default does not help the user, on the contrary, it introduces more error possibilities.

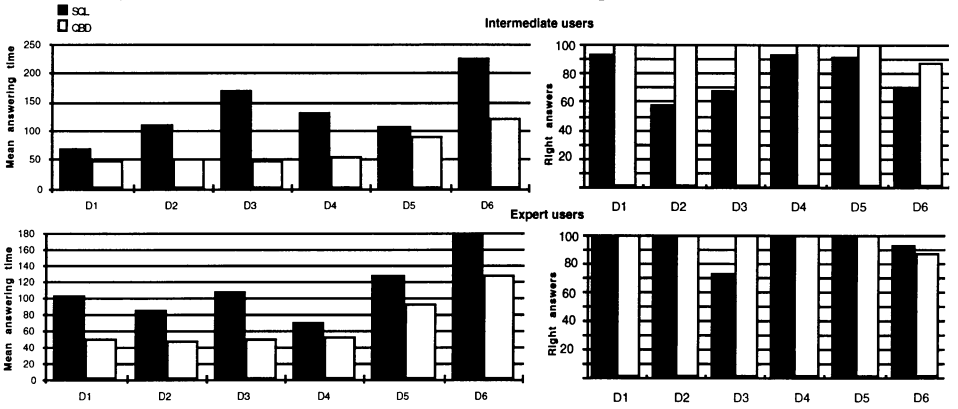


Figure 8 Mean answering times and percentage of correctly completed queries for medium complexity queries

In Figure 8 we show the comparison of SQL and QBD* for medium complexity queries. As for the SQL users, this set of queries gave the expected results, in the sense that the more experts were the users, the more they performed well.

Referring to QBD* we noticed that in the last question (D6) both intermediate and expert users forgot to include the attributes not belonging to the main entity in the final result. We recall (see Section 2) that the system automatically includes the main concept attributes in the final result, while the attributes of the other concepts selected along the path have to be explicitly added. Again, we note that having some default rules in the system behavior confuses the users. On the other hand, the users never did "conceptual" errors, such as a wrong choice of the main concept or of the path to be followed.

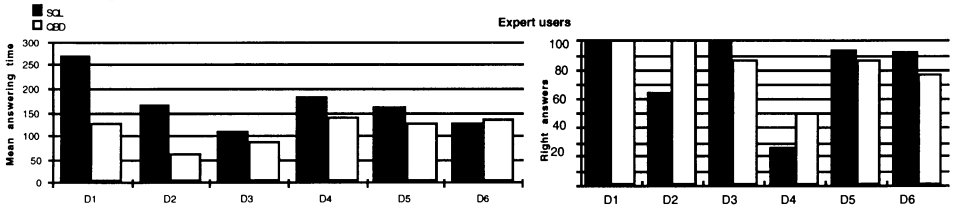


Figure 9 Mean answering times and percentage of correctly completed queries for high complexity queries.

In Figure 9 we show the comparison of SQL and QBD* for high complexity queries. SQL users, even if experts, did not gain great results for this kind of queries. In particular, they

found very difficult the use of the self-join in query D6. Also, the time results were particularly bad compared with the performance of the expert users in simple and intermediate queries.

QBD* users had a main problem in understanding the correct way of composing subqueries through the set operators. It is worth noting that for this last set of queries the differences in performance between the two query languages were significantly reduced, meaning that the main advantage in using a direct manipulation, graphical query language is in composing simple and intermediate queries. When the complexity of the query increases there is a conceptual comprehension gap that needs to be overcome by the user.

Finally, we compared the overall performances of the users by grouping the results on the three query classes.

		SYSTEM	
		SQL	QBD*
USERS	NAIVE	87%	100%
	INTERMEDIATE	83%	99%
	EXPERT	90%	93%

Figure 10 Overall average of correctly completed queries.

Figure 10 shows the overall average of correctly completed queries for SQL and QBD* . An Analysis of Variance test (ANOVA) yielded no significant difference for expert users on accuracy between SQL and QBD* at $F(1,14) = 1.83$, $P > 0.05$. There was however a significant difference on the subject's accuracy performance between the two systems at $F(1,28) = 40.7$, $P < 0.05$ (intermediate users) and $F(1,54) = 32.4$, $P < 0.05$ (naive users).

		SYSTEM	
		SQL	QBD*
USERS	NAIVE	99	63
	INTERMEDIATE	93	56
	EXPERT	108	74

Figure 11 Overall average of completation time

Figure 11 shows the overall average of time completation scores for SQL and QBD*. An ANOVA test yielded no significant difference for naive users on completation time between SQL and QBD* at $F(1,54) = 3.6$, $P < 0.05$. There was however a significant difference on the subject's performance between the two systems at $F(1,28) = 29.5$, $P < 0.05$ (intermediate users) and $F(1,14) = 8.2$, $P < 0.05$ (expert users).

In summary, we can say that QBD* was superior to SQL in many respects:

- it offers a complete view of the database at a higher abstraction level, which has been particularly appreciated by the users;
- the syntax is extremely simple; the user does not need to remember table or attribute names, but just to choose them navigating on the schema;
- the time needed to directly manipulate objects on the screen is less than the time used for writing SQL statements;
- the error rate is reduced, in particular for simple queries (but we understand from the experiments that we have to improve the default mechanisms);
- the users find QBD* more attractive, so they are not bored by working with it.

6 CONCLUSIONS AND FUTURE WORK

This paper describes an experiment aiming at estimating the advantages (or disadvantages) of a visual approach to database querying against a traditional one. In order to do this we compared QBD*, a visual query system based on the direct manipulation of Entity-Relationship diagrams, against the well-known SQL language. The results of the experiment were largely in favor of QBD*. However, we have to note that QBD* not only adopts direct manipulation mechanisms and a diagrammatic representation of the database, but also an external data model, the Entity-Relationship model, conceptually more expressive than the relational model. This choice was done on purpose, because we believe that the model of the database information presented to the user should be close to his/her mental model of the reality of interest, and far from the database model. So, we think that user interfaces to database systems can gain from the adoption of effective external models and metaphors, expressive visual representations, friendly interaction mechanisms. We recall that the data model and its visual representation are two distinct aspects of the interface and different mappings can be established among them (see (Batini et al. 1993) and (Haber Ioannidis Livny 1994)), but it is surely a good choice to try to improve both of them.

Our future work along these lines will be the comparison of two visual query systems based on the same conceptual model, but proposing different visual representations, namely iconic and diagrammatic, and interaction mechanisms. We already have two system prototypes and are presently setting up the experiment environment.

ACKNOWLEDGEMENTS

We are deeply grateful to Carlo Zerbo and Emanuele Mioni, who assisted us in designing and realizing the experiments. Also, we wish to thank Albert Badre for his helpful comments on a previous version of this paper.

7 REFERENCES

Ahlberg, C. Williamson, C., and Shneiderman, B. (1993)- Dynamic Queries for Information Exploration: an Implementation and Evaluation - In: *Sparks of Innovation in Human-Computer Interaction*, B. Shneiderman Ed., Ablex Publ., Norwood, NJ.

Angelaccio, M., Catarci, T., and Santucci, G. (1990) - QBD*: A Fully Visual Query System - *Journal on Visual Languages and Computing*, Vol. 1, No. 2, pp.255-273.

Angelaccio, M., Catarci, T., and Santucci, G. (1990) - QBD*: A Graphical Query Language with Recursion - *IEEE Transactions on Software Engineering*, Vol.16, No 10, pp.1150-1163.

Badre, A.N. (1993) - Methodological Issues for Interface Design: a User-Centered Approach - Technical Report DI/DS - 93/01 of Dipartimento di Scienze dell'Informazione, University of Rome "La Sapienza".

Batini, C., Catarci, T., Costabile, M.F., and Levialdi, S. (1991)- Visual Query Systems - Technical Report N.04.91 of Dipartimento di Informatica e Sistemistica, University of Rome "La Sapienza".

Batini, C., Catarci, T., Costabile, M.F., and Levialdi, S. (1993) - On Visual Representations for Database Query Systems - Proc. of the Second International Conference "Interface to Real & Virtual Worlds", Montpellier, France.

Bevan, N. and Macleod, M. (1993) - Usability Assessment and Measurement - In: "*The management of Software Quality*", M. Kelly Ed., Ashgate Technical/Gower Press.

Catarci, T. and Santucci, G. (1994) - Query By Diagram: A Graphical Environment For Querying Databases - , Proc. of ACM SIGMOD Conference on Management of Data.
Catarci, T., Chang, S.K., Costabile, M.F., Levialdi, S., and S.Santucci, G.(1994) - A Visual Interface for Multiparadigmatic Access to Databases - *IEEE Transactions on Knowledge and Data Engineering,* to appear.
Chen, P.P.(1976) - The Entity-Relationship Model toward a Unified View of Data - *ACM Transactions on Data Base Systems*, Vol.1, N.1.
Codd, E.F.(1972) - Relational completeness of database sub-languages - In *Data Base Systems*, R.Rustin, Ed., Prentice Hall, Englewood Cliffs, 65-98.
Date, C.J.(1987) - *An Introduction to Database Systems* - Vol.I, Addison-Wesley Publishing Company.
Haber, M.E., Ioannidis, Y.E., and Livny, M. (1994) - Foundations of Visual Metaphors for Schema Display. *Journal of Intelligent Information Systems*, Special Issue on "Advances in Visual Information Management Systems", *Vol. 3*, N. 3, pp. 1-38.
ISO (1991)- ISO 9126: Software product evaluation - Quality characteristics and guidelines for their use.
ISO (1993) - ISO DIS 9241-11: Guidelines for specifying and measuring usability.
Macleod, M. (1992) - An Introduction to Usability Evaluation, NPL Report DITC 102/92.
Maissel, J., Macleod, M., Dillon, A., Thomas, C., Rengger, R., Maguire, M., Sweeney, M. and Corcoran, R. (1993) - *Context guidelines handbook* - Version 2.1, National Phisical Laboratory, DITC, Teddington, UK.
Miller, R.B. (1971) - Human Ease of Use Criteria and their Tradeoffs - IBM Report TR 00.2185, 12 April. Poughkeepsie, NY: IBM Corporation.
Nielsen J. (1993) *Usability Engineering*. Academic Press, San Diego, CA, 1993.
Reisner, P. (1988)- Query Languages - In *Handbook of Human-Computer Interaction*, M. Helander, Ed. Elsevier Science Publ., 257-280.
Santucci, G. and Sottile, P. A. (1993) - Query By Diagram: a Visual Environment for Querying Databases - *Software Practice and Experience*, Vol. 23, No. 3.
Shackel, B. and Richardson, D.J. (1991)- *Human Factors for Informatics Usability* - Cambridge University Press.
Shneiderman, B. (1978) - Improving the Human Factors Aspect of Data Base Interactions - *ACM Transactions on Database Systems*, 3, 4, 417-439.
Shneiderman, B. (1980)- *Software Psychology* - Winthrop, Cambridge, Mass.
Shneiderman, B. (1983) - Direct Manipulation: A Step beyond Programming Languages - *IEEE Computer*, 16, pp. 57-69.
Spaccapietra, S. (1994) User Interfaces; Who Cares?, Proc. of the 20th International Conference on Very Large Data Bases, Santiago, Chile.
Thomas, J.C. (1977) - Psychological Issues in Data Base Management - Proc. of the 3rd International Conference on Very Large Data Bases, Tokyo, Japan.

8 APPENDIX: NATURAL LANGUAGE QUERIES

In this appendix we list the natural language queries used in the experiment together with their english translation (since the experiment subjects were italian speakers, the original queries were expressed in italian).

LOW COMPLEXITY QUERIES

D1) Elenca tutti i dati delle divisioni con un budget inferiore a 200.000.000.
List all the information on the departments having a budget less than 200.000.000 liras.

D2) Elenca i dati di tutti i reparti della divisione "SECONDA".
List all the information on the second division.

D3) Elenca nome e cognome dei dirigenti.
List names and surnames of the company managers.

D4) Elenca i dati del personale appartente al reparto Al o al reparto A2.
List all the information on employees belonging to either the A1 or the A2 Department.
D5) Elenca il personale che non sia dirigente.
List all the employees who are not managers.
D6) Elenca i nomi dei dipendenti del reparto 'Al' e sono dirigenti.
List all the employees belonging to the A1 Department who are managers.

MEDIUM COMPLEXITY QUERIES

D1) Per ogni reparto, elenca nome e cognome di tutti i dipendenti.
For each department, list names and surnames of the employees working in it.
D2) Elenca nome ed indirizzo dei dipendenti del reparto 'Bl'.
List the name and address of the employees belonging to the B1 department.
D3) Elenca divisione e settore di ogni reparto.
For each department, list its area and division.
D4) Elenca nome e cognome dei dipendenti di reparti non appartenenti alla divisione 'SECONDA'.
List names and surnames of employees working in departments which do not belong to the second division.
D5) Elenca i dati dei dipendenti dei reparti della divisione 'PRIMA', che non siano dirigenti.
List all the information on the employees of departments of the first division, who are not managers.
D6) Elenca il nome, cognome e reparto dei dipendenti di tutte le divisioni del settone 'MARKETING'.
List name, surname, and department of all the employees belonging to the MARKETING area.

HIGH COMPLEXITY QUERIES

D1) Elenca le province di residenza del personale dei reparti con budget annuo superiore a 100.000.000.
List the states where employees working in departments which have a budget greater than 100.000.000 liras live.
D2) Elenca nome e cognome degli impiegati che risiedono a Milano o a Roma e possiedono almeno un appartamento.
List names and surnames of employees living in Rome or Milan and owning at least one apartment.
D3) Elenca tutte le citta' dove non sono situati appartamenti del Signor ROSSI.
List all the cities where Mr. ROSSI does not own apartments.
D4) Elenca tutte le divisioni che non hanno ne' manager donne, ne' manager con meno di 40 anni.
List all the divisions whose managers are both non female and older than 40.
D5) Elenca tutti gli impiegati che risiedono a Roma e hanno un appartamento anche a Milano e tutti gli impiegati che risiedono a Milano.
List all the employees who either live in Rome and own an apartment in Milan or live in Milan.
D6) Elenca i nomi di tutti i dirigenti con retribuzione inferiore a quella di Smith
List all the managers whose salary is less than the Smith's salary.

9 BIOGRAPHIES

Tiziana Catarci

Tiziana Catarci was born in Roma, November 6, 1961.
She was graduated in Electrical Engineering from the University of Rome "La Sapienza," Rome, Italy, on 1987. She received the PhD degree in Computer Science from the University of Roma "La Sapienza", 1992.

Since 1987 she teaches courses in Computer Science at Italian universities. From 1987 to 1990 she was research assistant at the University of Roma "La Sapienza", where she is now an assistant professor.

Her main research interest is in visual formalisms for database representation and querying. Moreover, she worked in semantic database modeling, cooperative information systems, statistical databases, methodologies for database design. Dr. Catarci has published over sixty papers, and written or edited four books.

She is member of the Association for Computing Machinery (ACM), and the Institute of Electrical and Electronics Engineers (IEEE).

Giuseppe Santucci

Giuseppe Santucci was born in Rome, Italy, on February 8, 1958. He was graduated in Electrical Engineering from the University of Rome "La Sapienza," Rome, Italy, on 1987. He is currently Assistant Professor in Computer Science at Department of Computer Science of the same University. His main research activity concerns both theoretical and practical aspects of visual query language and user interfaces. Other fields of research are data dictionaries and graphical environment. He is the co-author of several scientific papers on these topics published on international journals and conference proceedings.