AN EXPLORATORY STUDY OF AD HOC QUERY LANGUAGES TO DATABASES

by

John E. Bell and Lawrence A. Rowe

Memorandum No. UCB/ERL M91/112

12 December 1991

AN EXPLORATORY STUDY OF AD HOC QUERY LANGUAGES TO DATABASES

by

John E. Bell and Lawrence A. Rowe

Memorandum No. UCB/ERL M91/112

12 December 1991

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

AN EXPLORATORY STUDY OF AD HOC QUERY LANGUAGES TO DATABASES

by

John E. Bell and Lawrence A. Rowe

Memorandum No. UCB/ERL M91/112

12 December 1991

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

This paper will be presented at the IEEE 8th Int'l Conf. on Data Engineering, Feb. 3-7, 1992, Phoenix, AZ

An Exploratory Study of Ad Hoc Query Languages to Databases

John E. Bell
Instructional Technology Program
University of California
Berkeley, CA 94720

Lawrence A. Rowe
Computer Science Division–EECS
University of California
Berkeley, CA 94720

Abstract

This paper describes an exploratory study performed to compare three different interface styles for ad hoc query to a database. Subjects with wide ranging computer experience performed queries of varying difficulty using either an artificial, a graphical, or a natural language interface. All three interfaces were commercial products. The study revealed strengths and weaknesses of each interface and showed that interaction with the natural language interface was qualitatively different than interaction with either the graphical or artificial language systems.

1. Introduction

Increasing computer power and decreasing costs are causing radical changes in user interfaces. Graphical and natural language interfaces are now practical alternatives to alphanumeric interfaces. This paper describes an an exploratory study that compared three different interface styles for database query: 1) artificial language, 2) graphical language, and 3) natural language.

Other researchers have compared different interfaces to query languages with varying success. Some have compared various artificial languages (e.g., [1–4]), others included natural language systems (e.g., [5–7]), and others have studied various other query language interface issues (e.g., [8–17]).

This paper describes an exploratory study comparing full function query languages with fifty-five subjects working on actual interfaces in a controlled environment. Subjects ranged in computer experience from none at all to experienced database users with programming experience. The tasks included simple queries (e.g., single table) to complex queries (e.g., multiple table and aggregate counting). Subject performance was analyzed across interfaces and across task types giving a rich and broad picture of the usability of these interfaces.

Research supported in part by a grant from BP America and by the National Science Foundation under grant MIP 87-15557.

This study was an exploratory study because of the lack of research regarding how these three interface styles compare using actual systems. Concepts from case studies were applied to gain an understanding of *how* these different interfaces compare as a prerequisite to finding the result of such a comparison. A case study, like ethnographic research, allows us to rely on a variety of means of observation, to discover what questions should be asked in a comparison of such interfaces, and to recognize the complexity of comparing the use of these interfaces by actual users [18]. The need to emphasize qualitative aspects of research has been recognized as well by other researchers in computer science (e.g., [19] and [20]).

This research stands out from the previous research because: 1) it is the first to include a graphical interface for database query, 2) it was performed in a relatively controlled environment using actual products, and 3) it emphasizes the qualitative data to help us understand the nature of the comparison of these systems.

The remainder of this paper describes the study and the results. Section 2 describes the sample database and the three interfaces. Section 3 describes the details of the experiment, including its design, the subjects involved, and the treatment. Section 4 presents the quantitative results and section 5 presents the qualitative results. Section 6 describes a small experts experiment and those results compared to the results presented in section 4. Finally, section 7 contains our conclusions.

2. The experimental database and the interfaces

The experimental database included information about students, teachers, classes and activities for a high school. Figure 1 shows an entity-relationship diagram for the database. Entities are represented by boxes, and relationships are represented by diamonds.

The high school database was chosen because it was familiar to all subjects and it is easy to understand. Also, it has sufficient entities and relationships to allow a variety of simple and complex questions to be asked.

A fundamental problem encountered when comparing different interfaces for the same task is to find a fair method to present the task to the experimental subject. Several different approaches have been used by other researchers (e.g., [5], [6], and [9]). We chose a pictorial representation of the tasks because it causes minimal bias toward any of the interfaces, it can represent all of the desired tasks, and it was readily understandable by subjects. Note that the representation was not intended to be a complete query language.

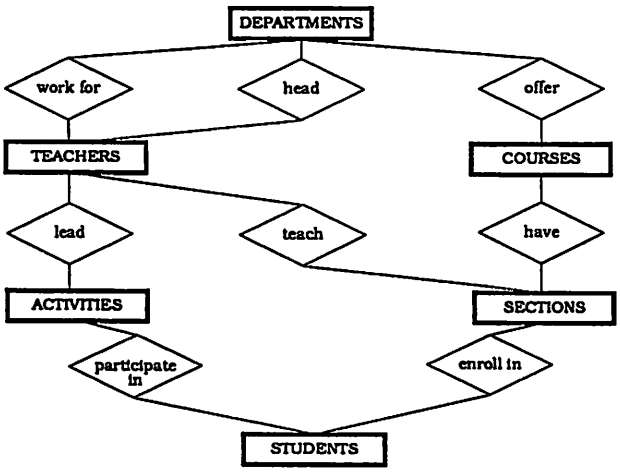

Figure 1: ER diagram of high school database.

Figure 2 shows an example. The task represented in the figure is to find the first and last names and salary of teacher number 101.[1] The icon represents the entity, in this case a teacher, and the captions represent the attributes. Attributes with values are restrictions and attributes without values specify the information to be retrieved. Subjects could refer to a sheet showing these icons and their meaning during the experiment. During the experiment, subjects appeared to have little trouble understanding the pictures, though in a few instances they misread a picture (e.g., not noting that "101" was specified by the teacher number).

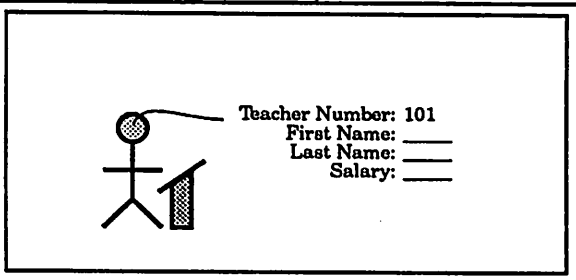

Figure 2: An example of pictorial task presentation.

Figure 3 shows a task description for a two table query. The task is to list all courses taught by each teacher. Notice that the join is not specifically stated but is implied because it uses attributes from different entities (i.e., teachers and section). The remainder of this section shows how this query can be entered into the three interfaces.

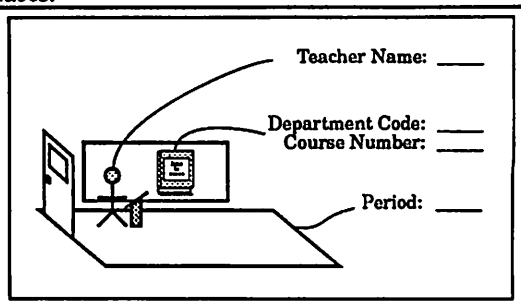

Figure 3: Task: List all courses taught by each teacher.

2.1. SQL

The first interface was an artificial language (AL) interface called *isql* that used SQL [21]. SQL uses English keywords (e.g., *select*, *from*, and *where*) to make it more readable and easier to remember. SQL is claimed to be user friendly because it is a non-procedural language. An SQL query that solves the sample task is:

```
select teachers.tfname, teachers.tlname,
  sections.dcode, sections.cnumber, sections.speriod
from teachers, sections
where teachers.tnumber = sections.tnumber
```

2.2. Simplify

The second interface was a graphical language (GL) called Simplify developed by Sun Microsystems [22]. Earlier versions of Simplify were developed at Xerox PARC [23]. Simplify is essentially a graphical interface to an SQL processor. A query constructed in Simplify is translated into an SQL query which is then run. Consequently, Simplify and SQL users specify the same commands but in different ways.

Figure 4 shows a Simplify window that contains the query for the sample task. The query is created by the following steps:

- Click on entity buttons for the tables in the query (i.e., teachers and sections). The entity boxes in the qualification window are displayed when the entity is selected.
- Click on the attributes in the entity boxes that are to be included in the query. Selected attributes are indicated by check marks (✔) and are automatically entered into the output format window.
- Click on tnumber in teachers, select "Create a join" in a pop-up menu, and click on tnumber in sections. This specifies the join between teachers and sections. A line joining these two attributes is displayed.

[1] The actual representation used in the study was slightly different but the difference did not affect understanding by the subjects. For more details, see [Bell, 1990 #53].

The query is then executed by choosing "Execute query" in a pop-up menu. The query result is displayed in another window.

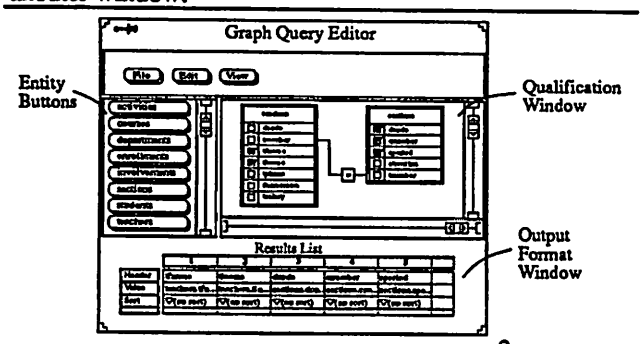

Figure 4: A sample Simplify screen.[2]

2.3. DataTalker

The third interface was a natural language interface (NL), called DataTalker, developed by Natural Language, Inc. [24]. It allows users to specify a query by entering plain English on a keyboard.

One solution to the sample task is the following:

```
Show the courses taught by each teacher
```

It is important to realize that this is only one solution among many that the user can enter to request this information. For example, another way to ask for this information is:

```
List the teachers and their courses
```

One objective of a natural language interface is to allow a variety of questions to access the same data.

All interfaces used the Ingres DBMS to store data. The SQL interface used in the experiment was the *isql* full-screen editor supplied with Ingres 5.0. The Simplify interface used was version 1.0 (beta) released in May 1989. The DataTalker interface used was version 3.0 released in March 1989. The experiment was performed on a Sun 3 computer. The same database was used for all subjects.

3. The experiment

The experimental design is shown in figure 5. Subjects were assigned to one of four groups (the vertical axis) based on prior experience and were randomly assigned to one of three treatments (the horizontal axis). Each subject worked through two phases: a learning phase and a performance phase. The learning phase was composed of task levels that covered seven types of queries as shown in figure 6. The performance phase included queries to test how well the user learned to use the interface to which they were assigned during the experiment.

[2]The entity boxes in figure 4 are shown side-by-side to enhance readability.

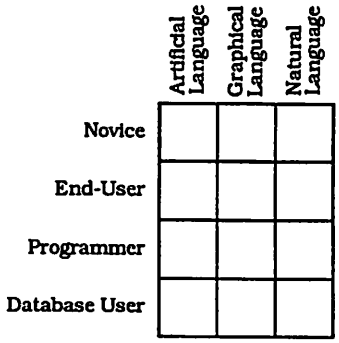

Figure 5: Experimental design.

Level	Description
L1	One table, no restrictions
L2	One table, one restriction
L3	One table, two restrictions
L4	One table, one or two restrictions, sorting
L5	One table, no restrictions, aggregation counting
L6	Two table join
L7	Three table join

Figure 6: Learning phase task level definitions.

Group	Definition
Novice	< 10 hours of computer experience ever
End User	experience with applications, no programming experience
Programmer	programming, no database experience
Database User	knowledgeable in SQL or QUEL

Figure 7: Subject group definitions.

Most of the 55 subjects were volunteers drawn from students at the University of California at Berkeley. The subjects were classified according to prior computer experience as shown in figure 7.

There were fifteen subjects in each user group except for database users. Since the database users knew SQL, none of them were assigned to the AL condition. Consequently, only ten database users were included in the study.

Subjects were randomly assigned to one of the three treatments: AL, GL, or NL. Everything was identical about the three treatments except for the interface used and the content of the help materials provided during the learning phase. Each subject spent about two hours working with the assigned interface.

Each subject was given a brief introduction to the experiment, the high school database, and the pictorial task presentation method. The only information they were given regarding the database was the set of icons used in query pictures along with a brief explanation of each icon. The ER diagram in figure 1 was not made available to the subjects.

3.1. Learning phase

Subjects then worked through the learning phase based on the following procedure. Subjects were given a task beginning at level 1 which they were to perform. If they could not solve the task, help was provided in the form of written advice. Help was broken down into the four levels shown in figure 8. After completing the task, another task of similar difficulty was given. These tasks were given until a task could be solved without help at which point subjects were advanced to the next level. A minimum of two tasks per level was required even if subjects could complete the first task without help.

Level	Description
Hint	A two sentence description of the essence of the solution.
Example	A query used to solve a similar task.
Explained Example	The steps followed to generate the example query.
Answer	A query that would solve the current task.

Figure 8: Help level descriptions.

3.2. Performance phase

After completing all levels, subjects moved to the performance phase of the experiment. The performance phase was the same as above except that no help was given. Subjects were allowed to work until they were successful or until they were making no progress toward the solution. The performance phase tasks are shown in figure 9.

Tasks 6 and 7 were included to see if subjects were able to extend their knowledge of the interfaces to situations that had not been explicitly taught.

4. User performance results

The quantitative results of the performance phase of the study are briefly presented in this section. Two metrics were examined: success and average time to task completion. Success is defined as completing a given task in a level. This metric assessed how well subjects could perform a variety of tasks after the training they received in the learning phase. The second metric was the average time required to complete each task. The average is considered only when tasks were successfully completed by at least half of the subjects. This metric assessed how much work was required of subjects to complete tasks in the performance phase.

Perf. Task	Description	Learning Level
1	One table, no restrictions	L1
2	One table, two restrictions	L2 & L3
3	One table, aggregate counting	L5
4	Three table join	L7
5	Two table join, one restriction, sorting	L4 & L6
6	Non-existence	none
7	Self-referential join	none

Figure 9: Performance phase task level definitions.

Note that because this study was an exploratory study these data are not statistically significant. We wanted to understand the qualities of the comparison between these interfaces before we attempted to achieve statistically significant quantitative results. The qualitative results, as presented in Section 5, are thus the more important and informative results of this study. The quantitative results are presented here for completeness. Throughout this section, we present our interpretation of the quantitative results based on our qualitative observations. These interpretations, however, are based on observations of individual cases and thus should not be considered as conclusions based on statistically significant numbers.

4.1. Trained tasks

The first metric examined for the trained tasks was the success rate. The number of tasks completed successfully out of all of the trained tasks is shown by user group and by interface in figure 10. Five tasks (i.e., levels 1 thru 5) were given to each of the five subjects in each user group, so the maximum possible value in each cell is 25. This table shows that the skills and experience of programmers and database users had a strong positive effect on their performance. This advantage was most pronounced on SQL and Simplify. This table also shows that the knowledge that end users had, as compared to novices, was most important when learning DataTalker. Finally, this table shows the overall strength of Simplify for trained tasks.

Though certain patterns are very clear from this data (e.g., end users on SQL and Simplify did no better than novices on those systems, while end users on DataTalker did much better than novices on that system), a Chi-

Square analysis does not yield any significant differences overall. Future research which includes more subjects is needed in order to determine if there are significant differences in this table.

User Groups	Interfaces: SQL	Simplify	DataTalker
Novices	11	11	7
End Users	10	11	12
Programmers	21	24	16
DB Users	•	24	20

Figure 10: Number of successful tasks by user group for trained tasks.

Figure 11 shows the success metric in greater detail. It contains a breakdown of figure 10 by task levels. The vertical axis shows the number of successful tasks, and the horizontal axis shows the task level.

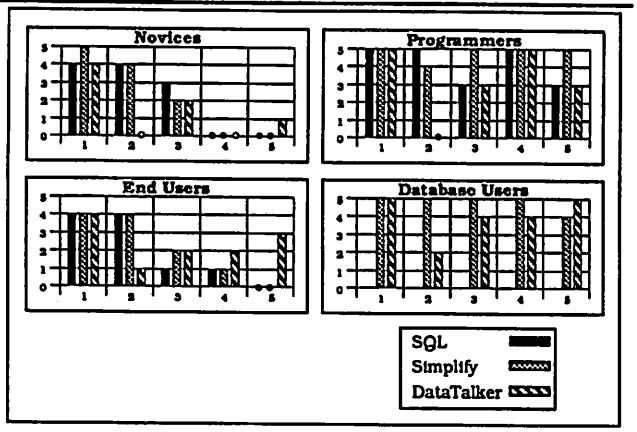

Figure 11: Number of successful tasks by user group and task level for trained tasks

The second level caused problems for all subjects using DataTalker. This task involved multiple restrictions which in and of themselves may be hard for natural language. The problems may also have arisen, however, due to difficulty related to the specific task involved. This pattern is explained in more detail in Section 5.1.

Novices on all systems performed roughly the same. This result is surprising in that novices using DataTalker were no more successful on these tasks overall than novices using either SQL or Simplify in spite of the expected benefits of natural language interfaces.

End users did approximately the same on all systems except for the fifth level. This level involved a two table join which had a more complex interaction than was found in the three table join of level 4. Because users of DataTalker did not have to deal with this complexity, 3 of the 5 subjects using DataTalker were able to complete this task. In contrast, this complexity resulted in no SQL or Simplify users being successful.

Overall, the success rates on these tasks were very similar across these interfaces with the exception of level 2 for all subjects, and of level 5 for end users.

The second metric examined was the average time needed to complete each task, including the time used by subjects to read the pictorial representation of the task. The results are shown in figure 12.

End users took about the same amount of time on all systems. Note, however, that while novices were just as fast as end users on Simplify and DataTalker, end users on SQL were much faster than novices on SQL.

Database users took about the same amount of time on Simplify and DataTalker. However, database users on DataTalker were faster than programmers on DataTalker. This result is surprising because the benefits of natural language interfaces are expected to be greater for the less experienced, and yet here, the extra knowledge and skill of database users made the interface apparently more powerful.

We collected and analyzed this data on the time to complete a query because we thought one or more interfaces might require less time for a user to express a query. We hypothesized that the time would be less if the time to plan a solution or to execute the plan was shorter. This experiment did not reveal many large time differences so further experimentation and analysis is required to evaluate our hypothesis.

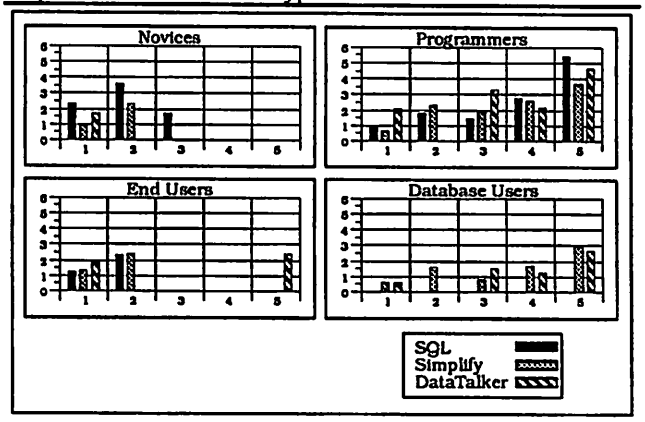

Figure 12: Average time (in minutes) for trained tasks.

4.2. Untrained tasks

The first untrained task was a non-existence query. None of the subjects using SQL or Simplify was successful on this task. DataTalker subjects, however, did extremely well. All five of the programmers and database users and three of the five novices as well as end users were successful. We were not surprised by the Simplify result because the system is incapable of performing this task. However, we were surprised by the SQL result. This query was a clear winner for the natural language interface.

The second untrained task was a self-referential join query. Only one subject, a database user using Simplify, was able to perform this task. Hence, Simplify is capable of performing self-referential joins, but most subjects were unable to discover how to do it. Furthermore, no SQL subjects were successful because self-referential joins in SQL are hard to figure out if you've never done one before. No DataTalker subjects were able to perform the self-referential join. This result does not mean that DataTalker cannot solve the task but that no one was able to discover how to do it. We were somewhat surprised by these results since we thought more subjects would get this query. This result suggests that subjects did not really understand the self-join concept and/or the way joins are specified in the three interfaces. Further study of this issue is needed.

The timing data on the untrained tasks did not reveal any obvious patterns.

5. Qualitative results

In an exploratory study, qualitative data is a rich source of information which helps the researcher understand more about the object of study. Subject comments both during and after the experiment, common mistakes and usage patterns, and experimenter observations all provided insight concerning what was easy and hard about each of these interfaces for the different types of tasks in this study. This section contains a discussion of the qualitative results as they relate to the following topics: 1) single and multiple restriction tasks, 2) join tasks, 3) counting tasks, 4) unknowingly getting the wrong answer, 5) compounded uncertainty, and 6) the natural language connection.

5.1. Single and multiple restrictions

The transition from single to multiple restrictions results in dramatically different performance on each of the three interfaces. In Simplify, multiple restrictions are specified by repeatedly forming single restrictions. Consequently, subjects did as well with multiple restrictions as with single restrictions. SQL was a little harder since subjects had to include the 'and' operator. DataTalker was much harder on multiple restrictions than on single restrictions partly because of the many possible ways that multiple restrictions can be stated in English. These statements also were more complex and led to greater user difficulty.

5.2. Joins

Both SQL and Simplify required users to understand the concept of forming joins by setting the join terms to be equal. This process was complex, largely due to figuring out which join terms should be used for each task. Simplify was slightly easier to do on two table joins because the graphical representation was simpler than the SQL syntax, allowing subjects to focus more on the question of which terms to include than on how to express the join.

Specifying joins was an area in which DataTalker was clearly superior because users did not have to specify joins at all. All potential joins in the database had to be described to DataTalker in the connection process. When users asked a question that involved attributes in separate tables, DataTalker automatically included the join terms between the tables. Subjects who used DataTalker were unaware that joins ever took place. In fact, some subjects thought that some of the two table join tasks had been given before since they were so much like the earlier tasks in other respects.

This power of DataTalker is similar to the universal relation interface as described by Maier, et al. [25]. SQL also provides a similar though more limited functionality with views which was not included in this experiment.

5.3. Counting

Counting was a very simple concept for subjects to understand, but it met with varying difficulty in each of the interfaces. Programmers had an especially hard time learning counting in SQL. They could not see the logic in the new commands. Consequently, they refused to believe it had been designed as it appeared. Once they learned how it worked, however, they were able to perform counting tasks relatively quickly. In contrast, end users using SQL did not have this difficulty with counting. Subjects had little trouble in Simplify. A construct that was similar to SQL was used, but it was easier to use in Simplify. In DataTalker, most subjects had almost no trouble at all. An obvious way of stating these tasks in English worked very reliably.

5.4. Unknowingly getting wrong answer

We expected to find that subjects using DataTalker would unknowingly get the wrong answer more often than when using SQL or Simplify. However, this behavior happened with about the same frequency on all systems. In general, the wrong answer occurred unnoticed only at the fringes of a subject's knowledge of the database and the interface they were using.

5.5. Compounded uncertainty

Each part of a query that was uncertain in the user's mind dramatically increased the difficulty of completing

the query. Several combinations had to be tried before finding an appropriate way to write each part of the query. This compounded uncertainty was most apparent in DataTalker because of the tremendous variation possible. Compounded uncertainty was the least common in Simplify because of the immediate feedback.

5.6. Natural language connection

Ogden has noted that the quality of the connection must be considered when evaluating natural language interfaces [26]. It is unreasonable to expect that actual users will be able to develop as consistently good connections as the authors of a natural language system. We spent several weeks working on the knowledge base. Then, a pilot study was run with several users to debug the experimental design and the connection. Problem areas were discussed with NLI and the changes they suggested were made. Nonetheless, not all problems were fixed. More research is needed in order to understand what impact the connection has on user performance.

6. Experts experiment

A brief experiment with two experts on each interface was also performed to serve as a benchmark to compare with the performance of subjects in the performance phase. This experiment was performed in exactly the same manner as the other experiment. On all systems, experts did better than the best non-experts (programmers and database users) but not by a wide margin. The experts were often faster and more successful though not always.

As was expected, in SQL experts did much better than non-experts on the untrained tasks, and Simplify experts did not do much better than non-experts on any task. SQL requires greater knowledge of complex features while most of Simplify's features are visible in its graphical interface. DataTalker experts did no better overall than non-experts. This result was surprising but is reasonable because every connection in DataTalker is a new interface which may not draw on more general expertise.

7. Conclusions

The following conclusions can be drawn from this experiment.

First, no interface was best. It comes as no surprise that none of these interfaces outperforms the others in all cases. Each does better under certain conditions. Furthermore, each interface can be improved.

Second, interaction with DataTalker is different than with SQL and Simplify. The performance of subjects who used DataTalker was very different than the performance of subjects who used SQL or Simplify. This result shows that an implicit goal of natural language interfaces, that is, to change the style of human-computer interaction, has been achieved in DataTalker. The difficulty of tasks in DataTalker is largely independent of the difficulty in more formal languages.

This result also shows the significance of the performance of subjects using DataTalker. Using a radically different style of interface, DataTalker's overall performance is comparable to the performance of more established interfaces.

Third, user experience affects performance differently on each interface. No one doubts the fact that in general the more experience users have the better they will do. What is surprising is the way experience affects performance on each system. In particular, performance on DataTalker was greatly affected by computer and database experience. It is often assumed that natural language interfaces will be best for those with less computer expertise, but it appears that such expertise is still an important asset to users of today's natural language interfaces.

Another surprise is that Simplify was strongest for programmers. The general expectation is that graphical interfaces are better for those with less experience. In this study, however, the complexity of the graphical interface prevented that from happening.

As our conclusions state, these results apply specifically to the interfaces tested. However, to the extent that other query interfaces are like these interfaces, similar patterns can be expected to occur [27]. For example, for other graphical languages that follow a model similar to Simplify (e.g., VQL from Sybase [28]), and for other natural language interfaces that have a similar coverage of English and allow a similar dialog as in DataTalker, strengths and weaknesses like those seen here can be expected to occur. Actual experimentation is needed, however, to explore the similarity of other interfaces to these systems, and consequently, the applicability of these results.

Many other topics for research in the comparison of interfaces for ad hoc database query also remain. A focused study on each of the user groups, especially end-users, needs to be performed in order to determine a specific and reliable comparison of the three interface styles. Research should also be performed which focuses more on the qualitative aspects such as evaluating user strategy when working with natural language interfaces or developing a model of user knowledge and behavior when working with query interfaces in general. Research should also consider other interfaces such as NLMenu which is a menu-based interface to a natural language system [29]. Finally, similar research should be performed as

enhancements are made to these interfaces in order to determine how user performance is affected.

References

1. Reisner, P., R.F. Boyce, and D.D. Chamberlin. *Human factors evaluation of two database query languages – Square and Sequel.* in *National Computer Conference.* 1975. Anaheim, CA: AFIPS.
2. Greenblatt, D. and J. Waxman, *A study of three database query languages,* in *Databases: Improving Usability and Responsiveness,* B. Shneiderman, Editor. 1978, Academic Press, Inc.: New York. p. 77-97.
3. Boyle, J.M., K.F. Bury, and R.J. Evey. *Two studies evaluating learning and use of QBE and SQL.* in *Proceedings of the Human Factors Society 27th Annual Meeting.* 1983. Santa Monica, CA: Human Factors Society.
4. Zloof, M.M. *Query by Example.* in *National Computer Conference.* 1975. Anaheim, CA: AFIPS.
5. Shneiderman, B., *Improving the human factors aspect of data base interactions.* ACM TODS, 1978. 3(4): p. 417-439.
6. Small, D.W. and L.J. Weldon, *An experimental comparison of natural and structured query languages.* Human Factors, 1983. 25: p. 253-263.
7. Jarke, M., *et al., A field evaluation of natural language for data retrieval.* IEEE TSE, 1985. SE-11(1): p. 97-114.
8. Thomas, J.C. and J.C. Gould. *A psychological study of query by example.* in *National Computer Conference.* 1975. Anaheim, CA: AFIPS.
9. Ogden, W.C. and S.R. Brooks, *Query Languages for the Casual User: Exploring the Middle Ground between Formal and Natural Languages.* CHI'83, 1983. : p. 161-165.
10. Fink, P.K., A.H. Sigmon, and A.W. Biermann, *Computer control via limited natural language.* IEEE Transactions on Systems, Man, and Cybernetics, 1985. 15: p. 54-68.
11. Ogden, W.C. and A. Sorknes. *What do users say to their natural language interface?* in *Proceedings of Interact '87 - 2nd IFIP conference on Human-Computer Interaction.* 1987. Amsterdam: Elsevier Science.
12. Krause, J., *Natural language access to information systems. An evaluation study of its acceptance by end users.* Information Systems, 1980. 5: p. 297-319.
13. Damerau, F.J., *Operating statistics for the transformational question answering system.* American Journal of Computational Linguistics, 1981. 7: p. 30-42.
14. Welty, C. and D.W. Stemple, *Human Factors Comparison of a Procedural and a Nonprocedural Query Language.* ACM TODS, 1981. 6: p. 626-649.
15. Lochovsky, G.H., *Data base management system user performance.* 1978, University of Toronto, Canada:
16. Brosey, M. and B. Shneiderman, *Two experimental comparisons of relational and hierarchical database models.* International Journal of Man-Machine Studies, 1978. 10: p. 625-637.
17. Bierman, A.W., B.W. Ballard, and A.H. Sigmon, *An experimental study of natural language programming.* International Journal of Man-Machine Studies, 1983. 18: p. 71-87.
18. Wolcott, H.F., *Ethnographic research in education,* in *Complementary methods for research in education,* R.M. Jaeger, Editor. 1988, American Educational Research Association: Washington, DC.
19. Moher, T.G., *Estimating the distribution of software complexity within a program.* CHI '85 Proceedings, 1985. : p. 61-64.
20. Soloway, E., K. Ehrlich, and J.B. Black. *Beyond Numbers: Don't Ask 'How Many'...Ask 'Why'.* in *Proceedings of the Conference on Human Factors in Computing Systems.* 1983. Boston, MA:
21. *Using INGRES Through Form and Menus.* 1989, Relational Technology, Inc.:
22. *SunSimplify™ 2.0 Reference Manual.* 1989, Sun Microsystems, Inc.:
23. Cattell, R.G.G., *An Entity-based Database User Interface.* ACM SIGMOD, 1980. : p. 144-150.
24. *Natural Language™ Database Retrieval System User Manual.* 1988, Natural Language Incorporated:
25. Maier, D., *et al. Toward logical data independence: A relational query language without relations.* in *ACM SIGMOD Conference.* 1982.
26. Ogden, W.C., *Using Natural Language Interfaces,* in *Handbook of Human–Computer Interaction,* M. Helander, Editor. 1988, Elsevier Science Publishers: p. 281-299.
27. Cornfield, J. and J.W. Tukey, *Average values of mean squares in factorials.* Annals of Mathematical Statistics, 1956. 27: p. 907-949.
28. *VQL.* 1989, Sybase, Inc.:
29. Tennant, H.R., K.M. Ross, and C.W. Thompson. *Usable natural language interfaces through menu based natural language understanding.* in *CHI 1983 Conference on Human Factors in Computer Systems.* 1983. Boston, MA: North-Holland.