5-1995

The effect of a database feedback system on user performance

H. CHAN

K. WEI

Keng SIAU
Singapore Management University, klsiau@smu.edu.sg

BEHAVIOUR & INFORMATION TECHNOLOGY, 1995, VOL. 14, NO. 3, 152–162

The effect of a database feedback system on user performance

HOCK CHUAN CHAN and KWOK KEE WEI

Department of Information Systems and Computer Science, National University of Singapore, Lower Kent Ridge Road, Singapore 0511, Republic of Singapore

KENG LENG SIAU

Faculty of Commerce and Business Administration, University of British Columbia, 2053 Main Mall, Vancouver, B.C., Canada V6T 1Z2

Abstract. There are two main approaches to improving the effectiveness of database interfaces. One is to raise the level of abstraction for the content of the user–database interaction. The relational model belonging to the logical level has replaced the hierarchical and network models that belong to the lower physical level. It is likely that the relational model will eventually be replaced by models belonging to the even higher conceptual level, such as entity relationship models and object-oriented models. The second approach is to enhance the actual interaction process. This can be done by providing better feedback to the user. Feedback can be in the form of more comprehensible error messages, and the provision of a natural language interpretation of user's query. Such a feedback system was developed, and its effectiveness tested in an experiment. The results showed that the feedback system enhanced user performance greatly. Specifically, users who used the feedback system were 12.9% more accurate than those without the feedback system. They were also 41.2% more confident of their answers, and they took 29.0% less time than those without the feedback system.

1. Introduction

A good user–database interface (UDI) allows fast and accurate query formulation (Chamberlain 1980). This area is expected to become more important as more end-users gain access to sophisticated database systems. A major stumbling block for end-users, in accessing relational databases, is the need to understand and manipulate unfamiliar data objects, such as relations, fields, and joins (Poonen 1979, Templeton and Burger 1986, Junet 1987).

This difficulty can be avoided by providing an interface of a higher abstraction level, i.e., a conceptual level interface, where the user can work directly with entities and relationships instead of relations and joins (Jarvenpaa and Machesky 1986, Batra *et al.* 1990, Chan *et al.* 1991). The resulting UDI is much better, but it is still far from perfect.

It is proposed that the UDI include an active feedback system to further improve user performance. The implemented system can serve as a training aid. Another contribution is towards quality management, as the ultimate objective of the research is to increase the quality of the output from the database users. More specifically, the feedback system under development will be able to take over part of the management function of providing feedback to the users. This will be highly desirable as the experiment by Ang *et al.* (1991) showed that people are reluctant to seek feedback from another person, and that they will rather seek feedback from the computer, especially if the feedback is also computer generated.

The system is described in more details in section 2. Section 3 describes an experiment to test empirically the effect of the system on user performance, as measured by query accuracy, user confidence and time taken to write queries. The results are reported and discussed in section 4, and section 5 offers concluding remarks.

2. Database feedback system

2.1. *Objectives*

The feedback studies by Campbell (1988) and Earley *et al.* (1990) indicated that the nature and content of the feedback must fit the task, if the feedback is to be valuable.

0144-929X/95 $10·00

Furthermore, good feedback should give specific information on how to modify the query.

Existing database systems give very inadequate feedback. The user only sees the syntax errors and the retrieved data. A query language interface defines a formal language, in which all retrieval requests must be expressed. Even if the user manage to formulate a syntactically correct query expression, there is no guarantee that it is logically correct. In a study of Query-By-Example (Thomas and Gould 1975), it was found that 27% of the queries analyzed were syntactically correct, but they did not correspond to the questions the users thought they had asked. Thus, although query language interfaces have the advantages of generality (the ability to express arbitrary requests) and non-ambiguity (each statement has clear semantics), using query languages requires considerable proficiency (Motro 1990).

When a query is executed, only the data set is returned. It is difficult to check for query accuracy. Worse still, the answer might be correct even if the query is incorrect. The study by Katzeff (1989) confirmed that users had great difficulty in verifying the values retrieved by the system. In that study, the subjects had all the data in the computer and they could hand-check the results shown by the computer. This required not only detailed knowledge of the actual data but also the data manipulations such as joins. Hand-checking is possible only for small sets of data. Even then, this can be very tedious.

A better feedback is to state the query semantics in English. This was in fact done for the standard relational query language SQL in research systems (Amano and Kambayashi 1991, Luk and Kloster 1986). There are however two main problems with the SQL to English translation. One problem is that customization is necessary for each database. Another problem is that SQL is hard to use (Welty 1985). After knowing that the query is wrong, the user may not be able to do the corrections.

A system is proposed to overcome these two problems. First, the system is domain independent; it can work with any ER database without customization. Second, it is based on an ER language, KQL, that has been empirically shown to be much easier than SQL (Chan *et al.* 1991). In addition, it actively offers solutions to correct syntax as well as possible semantic errors.

The database feedback system has the following specific objectives:

- to provide English translation of KQL queries, even when there are syntax errors;
- to highlight syntax errors, and to suggest solutions;
- to highlight possible semantic errors, and to suggest solutions.

2.2. *Syntax and semantic errors*

The following is a list of some of the syntax errors handled by the system:

- some spelling errors;
- misplaced or omitted commas;
- missing *a* or *an* in the instance clause;
  e.g. *e is employee*
  suggested solution:
  syntax error: < *e is employee* > is incorrect
  correct syntax: < *e is a/an employee* >
- wrong comparison operands;
  e.g. *e1 work dept* = *e2 work dept*
  suggested solution:
  syntax error: < *e1 work dept* > = < *e2 work dept* >
  wrong usage of compare condition
  correct syntax: > *select....*
  where e1 work-related dept;
  e2 work-related dept >
- wrong usage of clauses, conditions.

The system detects quite a number of semantic errors. The following are some of the errors handled by the system:

- redeclaration of instance variable error;
- usage of unknown entity or relationship type;
- undeclared instance variable name;
- inheritance of wrong attributes in an is–a relationship;
- disjoint entities or relationships

An example of the last case is when a query has two entity instances but no condition connecting these, e.g.,

```
select employee.name
where dept = 'research'
```

In such cases, error is suspected and the user will be informed.

2.3. *English language translator*

A parse tree is built from a KQL query and a set of translation rules are then applied to the tree for translation to English. The translation of a query is the translation of all the parts of the query, preserving the order of the parts. This makes the algorithm simple and enables the user to find the correspondence between the feedback and his query. This can be important for the user in identifying and correcting mistakes. The translation process uses knowledge from the entity relationship model. No additional knowledge and hence no customization is required. Examples of KQL and the English translations are given in Appendix A for the test

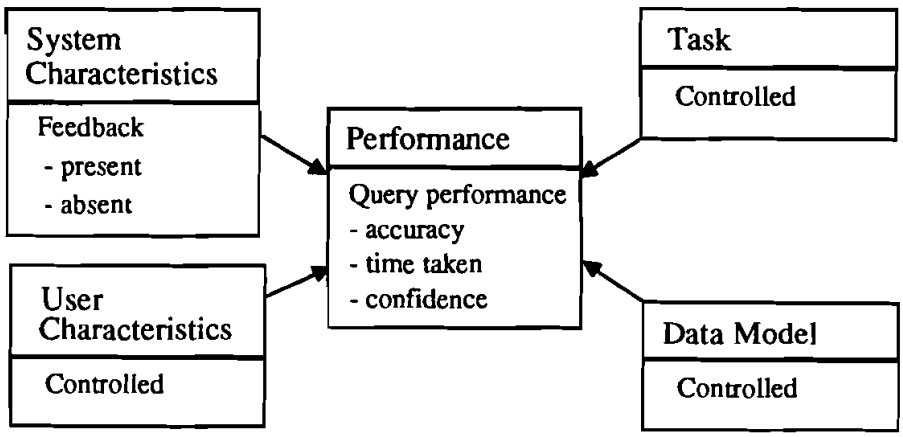


Figure 1. A research model.

questions. The syntax for KQL, and the algorithm to translate KQL to English can be found in Chan *et al.* 1993.

2.4. *Implementation*

The database feedback system is implemented on an AT&T 3B4000/15 computer operating under UNIX/System V, release 3.1.5. The underlying relational database system used is INGRES, release 5.0 from Relational Technology Inc.

A simple interactive terminal interface, similar to INGRES terminal monitor interface is implemented for KQL user interface. The user can enter a query and execute it. The UNIX vi editor can also be invoked to make changes to the existing query. The interface also allows the user to read or write files containing queries or execute operating system level commands from within the interface. Additional features are included to allow the user to display the translated SQL commands as they are executed or without executing them.

3. Empirical testing

3.1. *Research model and hypotheses*

The research model for this study is shown in figure 1. This model is adapted from Reisner (1981). Surveys of empirical studies on query languages show that there are four important factors that have effects on user performance:

1. the data model characteristics;
2. the task characteristics;
3. the user characteristics (human factors);
4. the system characteristics.

In this experiment study, we controlled three of the four factors by fixing the model to be the entity relationship model, fixing the task to writing retrieval queries for ten wide ranging questions, and by randomizing the subjects to control user characteristics. System characteristic is varied by having two interfaces, one with feedback and one without. In all other ways, the interfaces are the same. The objective of this controlled laboratory experiment is to evaluate the effect of a database feedback system during the training session and during the test session. Based on the premise that feedback is beneficial to users, the following hypotheses are made

H1: Subjects experiencing feedback during training (group WN and group WW, in table 1) will perform better than subjects not experiencing feedback during training (group NN and group NW).

H2: Subjects experiencing feedback during testing (group NW and group WW) will perform better than subjects not experiencing feedback during testing (group NN and group WN).

Better performance means higher accuracy, higher confidence and faster timing in writing queries.

3.2. *Experiment design*

The two-by-two factorial experimental design is given in table 1, together with the number of subjects successfully completing the experiment in each treatment condition. There are four groups with two separate treatments—either the presence or absence of a feedback system during training and/or during testing.

In this study, the independent variables manipulated are the presence/absence of feedback during training, and the presence/absence of feedback during testing. The dependent variables are accuracy, confidence, and time. These are measures commonly used in experiments on database retrieval. Accuracy measures the correctness of the queries formulated by the subject during the test. Confidence measures a subject's confidence in his queries formulated

Table 1 Experimental design.

Treatment		Test: Without feedback	Test: With feedback	Total
Training	Without feedback	Group NN 15 subjects	Group NW 14 subjects	29 subjects
	With feedback	Group WN 15 subjects	Group WW 16 subjects	31 subjects
Total		30 subjects	30 subjects	

during the test. Time measures how long a subject take to formulate his queries during the test.

3.3. *Subjects*

Sixty-four subjects were selected randomly from first-year computer science undergraduate students. They were assigned randomly to the various treatment conditions. Sixty subjects completed the research experiment successfully. Student subjects were employed for the following reasons: they were available in sufficient numbers to provide adequate statistical power for statistical tests; and they were relatively homogeneous in terms of knowledge, intelligence, and organizational experience, thus reducing the presence of extraneous variable differences.

On average, the students were 20 years old. To motivate the subjects, they were informed that course credit would be awarded based on their accuracy in constructing the queries, and also based on the correlation between the accuracy of their queries and the self-reported confidence level. This encouraged them to report their confidence honestly rather than to show excessive confidence. The subjects are representative of users who are intelligent, have some computer experience, and little database training. Users matching these characteristics are likely to be at the executive or managerial level.

3.4. *Experimental procedure*

The experiment proceeds in two steps, a training followed by a test.

3.4.1. *Training:* At the beginning of the training, the trainer (administrator) gave an introduction and explanation of the study to the participants. All subjects were then given a copy of the training booklet. Two training booklets were used during the experiment: training booklet A (a booklet with no examples of feedback messages from the feedback system) for the two groups that do not experience the feedback system during training (group NN and group NW) and a training booklet B (a booklet with examples of feedback messages from the feedback system for the two groups that experience the feedback system during training (group WN and group WW). The two training booklets gave a brief overview of ER data model and the query language KQL. To maintain consistency, the same database domain (supplier and parts) and the same example queries were used in both booklets.

The same trainer provided separate training for the four groups. All examples in the booklets were discussed. Subjects practised answering a question after each example. All questions from the subjects were answered. The training for all the groups lasted one hour and all the subjects were given a break of ten minutes after the training.

The purpose of the training is to let the subjects have hands-on experience with the database management system.

Table 2. Accuracy means and standard deviation for each group.

	Treatment	Test: Without feedback	Test: With feedback	Total
Training	Without feedback	42.67 (1.16)	45.11 (0.68)	43.84 (1.56)
	With feedback	46 .77 (0.68)	48.19 (0.96)	47.50 (1.10)
	Total	44.71 (2.28)	46.75 (1.77)	

Table 3. Analysis of variance for accuracy.

Source of variation	Degrees of freedom	Sum of squares	F ratio	Prop > F
With/without feedback during training	1	192.91	238.51	0.0000*
With/without feedback during testing	1	55.79	68.97	0.0000*
Interaction	1	3.89	4.81	0.0325*
Error	56	45.29		
Total	59	304.23		

The training allowed the subjects to be acquainted with the mechanics of the interface so that, for the test, they would not have to spend time figuring out how to enter the query, how to report the confidence level, or how to get to the next question. The measured time for the real test will then be the real query formulation time.

3.4.2. *Test:* Finally, the subjects were asked to run the test program. They had to answer ten questions based on a new database domain (employees and departments). All the four groups were given a diagram, on paper, of the ER model. The diagram, the set of ten questions and the sample answers and feedback are given in Appendix A. Sample sessions of how the subjects changed their queries after feedback are given in Appendix B.

The subjects had to enter their name at the beginning of the program. The same test program was used for all the four groups. This meant that all groups answered the same set of questions in exactly the same order. Every subject had to finish each question before proceeding to the next one. The program displayed the questions one by one. Answers were entered directly into the computer via a simple text editor. Subjects could refer to the training material and use paper and pencil to help formulate the answers.

The computer automatically timed the interval between the display of the question (or system messages) and the time when the student entered \g (or \eng) signifying that the query had been constructed. Immediately after each answer, the subjects were asked for their confidence in their answer. The value ranged from 0 (zero confidence) to 5 (absolute confidence). The query, the confidence level and the time in seconds for each query were recorded by the program.

The test questions cover quite a comprehensive range. The first two are very simple. Others can be quite complicated, e.g., those that need two instances of the same entity type and their relationship specifications, and those that need counting, ALL/NO qualifications. While some of these may appear easy, they are really difficult queries for existing relational languages like SQL, needing up to two levels of nesting.

3.5. *Measurement of dependent variables*

3.5.1. *Accuracy:* There were ten questions in the test. Thus, each subject was required to write queries for each of the ten questions. A subject could enter as many queries as he wished for each question. The accuracy of each query was determined separately by two markers. The separate marking provided an estimate of the reliability. The accuracy was an overall assessment of the correctness of the answer. Both semantic and syntax accuracies were considered.

The accuracy of each query of each question could get a maximum of 5 marks (absolute accuracy) and a minimum of 0 mark (zero accuracy). Only the last query of each question was taken into consideration during statistical analysis of accuracy. The marks for the ten questions were totalled to give the accuracy of the ten queries formulated by each subject. Thus, the accuracy of the queries formulated by each subject is measured on a scale ranging from 0 to 50.

3.5.2. *Confidence:* As mentioned earlier, each subject could

Table 4. Analysis of variance for accuracy (no feedback during training).

Source of variation	Degrees of freedom	Sum of squares	F ratio	Prob > F
With/without feedback during testing	1	43.13	46.72	0.0000*
Error	27	24.92		
Total	28	68.05		

Table 5. Analysis of variance for accuracy (feedback during training).

Source of variation	Degrees of freedom	Sum of squares	F ratio	Prob > F
With/without feedback during testing	1	15.63	22.25	0.0001*
Error	29	20.37		
Total	30	36.00		

enter as many queries as he wished for each of the ten questions in the test. Immediately after entering each query, the subject was asked for his confidence in his query. The value entered by the subject ranged from 0 (zero confidence) to 5 (absolute confidence) and was recorded by the program. Only the last query of each question is taken into consideration during statistical analysis of confidence. Each subject's confidence of the last query in each question for the ten questions were totalled to give the subject's confidence in the ten queries formulated. Thus, each subject's confidence in his or her queries is measured on a scale ranging from 0 to 50.

3.5.3. *Time:* As mentioned earlier, each subject could enter as many queries as he wished for each question in the test. The program automatically timed the interval between the display of the question (or system messages) and the time when the subject entered \g (or \eng) signifying that the query had been constructed. The time in seconds was recorded by the program. All the queries of each question are taken into consideration during statistical analysis of time. The time taken to answer each query in each question for the ten questions were totalled to give the subject's time taken to complete the test. Time taken is measured in seconds.

4. Results and implications

4.1. *Statistical methods*

The analysis of variance (ANOVA) test was used to detect significant main and interaction effects. Tukey–Kramer's test was also used to confirm the results. A five percent level of significance was used for all the statistical tests. JMP (version 2), a software for statistical visualization on the Apple Macintosh, was used to perform the statistical tests. At the beginning of the test, the experiment data were subjected to the following to ensure that the data satisfied the requirements of ANOVA test:

- *Homogeneity of variances.* This requirement was tested using Hartley's test and all the data satisfied this test.
- *Normality of error terms:* This requirement was tested using Shapiro–Wilk's test. Data which failed the test were subject to a suitable transformation (Weisberg 1985) that allowed this requirement to be met.
- *Independent samples:* This requirement has been satisfied by randomly assigning subjects using random generator to groups under the various treatment conditions.

4.2. *Statistical results*

For each dependent variable, the results of statistical analysis are presented in two tables. The first table presents the mean and standard deviation for the dependent variable under each treatment condition. The second table presents the details of the ANOVA for the dependent variable.

4.2.1. *Accuracy:* Table 2 summarizes the measurement of accuracy for each treatment condition. There was a high correlation between the separate assessment of accuracy by the two markers. The first set of marks was used for the statistical tests. Table 3 reports the details of the ANOVA test on accuracy.

Feedback during training had a significant main effect on accuracy. Feedback during testing also has a significant main effect on accuracy. There is an interaction effect on accuracy. Results of Tukey–Kramer's test on accuracy using the effect of feedback during training confirm that queries from subjects who did not experience feedback during training are significantly less accurate than queries from subjects who

Table 6. Confidence means and standard deviation for each group.

Treatment		Test: Without feedback	Test: With feedback	Total
Training	Without feedback	32.27 (2.34)	37.93 (1.21)	35.00 (3.42)
	With feedback	40.53 (1.41)	45.56 (1.46)	43.13 (2.92)
Total		36.40 (4.61)	42.00 (4.09)	

Table 7. Analysis of variance for confidence

Source of variation	Degrees of freedom	Sum of squares	F ratio	Prob > F
With/without feedback during training	1	946.00	340.61	0.0000*
With/without feedback during testing	1	427.67	153.98	0.0000*
Interaction	1	1.50	0.54	0.47
Error	56	155.53		
Total	59	1573.60		

experienced feedback during training. Similarly, results from Tukey–Kramer's test on accuracy using feedback during testing confirm that queries from subjects who did not experience feedback during testing are significantly less accurate than queries from subjects who experienced feedback during testing.

There is an interaction effect on accuracy. An in-depth analysis of the interaction is required because the interpretation of an interaction take precedence over the interpretation of a significant main effect (Keppel 1982). The method of analysis of simple effects (Keppel 1982) was used to examine the interaction and the results are listed in tables 4 and 5. Feedback during testing was assumed to be the more important dimension.

Table 4 indicates that when feedback is absent during training, the presence of feedback during testing makes a significant difference. Table 5 indicates that even when feedback is present during training, the presence of feedback during testing makes a significant difference.

4.2.2. *Confidence:* Table 6 summarizes the measurement of confidence for each treatment condition. Table 7 reports the details of the ANOVA test on confidence. Feedback during training has a significant main effect on confidence. Feedback during testing also has a significant main effect on confidence. There is no interaction effect on confidence. Results of Tukey–Kramer's test on confidence based on feedback during training confirm that subjects who did not experience feedback during training were significantly less confident than subjects who experienced feedback during training. Similarly, using the effect of feedback during testing, the Tukey–Kramer's results confirm that subjects with feedback during testing were significantly more confident than those without.

4.2.3. *Time:* Table 8 summarizes the measurement of time for each treatment condition. Table 9 reports the details of the ANOVA test on time. Data for time were transformed by a square root function (Weisberg 1985) to meet the requirements of the ANOVA test (Neter *et al.* 1985). Feedback during training has a significant main effect on time. Feedback during testing has no significant main effect on time. There is no interaction effect on time. These results were confirmed by Tukey–Kramer's tests.

The hypotheses supported by the statistical results are summarized below.

- Subjects with feedback during training perform better than subjects without feedback during training. This is true for the measures of confidence and time.
- Subjects with feedback during testing perform better than subjects without feedback during testing. This is true only for the measure of confidence.
- In addition, significant interaction effect was found for the measure of accuracy. When feedback is absent during training, the presence of feedback during testing makes a significant improvement. Even when feedback is present during training, the presence of feedback during testing also makes a significant improvement.

4.3. *Discussion*

Figure 2 shows the dependent measures relative to those for the group without feedback during training and testing, i.e., this group's scores are set at 100%, and the other groups' scores are divided by this group's scores. Subjects who had

Table 8. Time means and standard deviation for each group.

Treatment		Test: Without feedback	Test: With feedback	Total
Training	Without feedback	2905.20 (1620.92)	2682.07 (1125.00)	2797.48 (1383.54)
	With feedback	2189.67 (749.94)	2062.12 (949.78)	2123.84 (847.17)
Total		2547.43 (1293.18)	2351.43 (1064.38)	

Table 9. Analysis of variance for time.

Source of variation	Degrees of freedom	Sum of squares	*F* ratio	Prob > *F*
With/without feedback during training	1	567.14	4.12	0.047*
With/without feedback during testing	1	39.82	0.29	0.593
Interaction	1	2.68	0.02	0.89
Error	56	7,724.75		
Total	59	8,345.75		

feedback both during training and testing produced the best scores while subjects who did not have any feedback produced the worst scores. Compared to group NN, group WW had 12.9% higher accuracy, 41.2% higher confidence, and took 29.0% less time. A reason for the moderate percentage improvement in accuracy is that the starting point, group NN's score, is already a high 85.3% of the maximum score. Group WW's score is a very high 96.4% of the maximum. Group WW's score for confidence is also a high 91.1% of the maximum.

As discussed in Katzeff (1989), formulating a correct expected reply and applying an efficient strategy for evaluating the computer reply are two crucial steps in the query writing process. Users are more successful in problem-solving tasks within a particular system if they are able to form a mental model of that system. A critical role played by the mental model is to prepare the users for novel situations. Through the mental model the user is able to form expectations about the system consequences produced by his actions. In the present study, subjects who experienced feedback had apparently formed mental models which fulfilled this role. Error and help messages provided by the feedback system assisted subjects in resolving possible difficulties. In addition, feedback of English translations of queries allowed subjects to double-check their queries.

4.4. *Enhancement*

Although the feedback has improved user performance, more can be done to enhance the feedback quality. For example, it can provide suggestions to more syntax errors. Presently the system has only a very limited number of suggestions on how to correct syntax errors, even though all syntax errors can be located.

The common mistakes observed in the experiment can be used as a guide to offer more suggestions. A possible approach is to generate a set of templates based on the observed errors. In the presence of error, the system will determine the type of error made and select a suitable template. The system then replaces the variable names in the template by the appropriate words in the query and in the particular data model. Consider the following example from the test:

Show the names of employees who work in the research department.

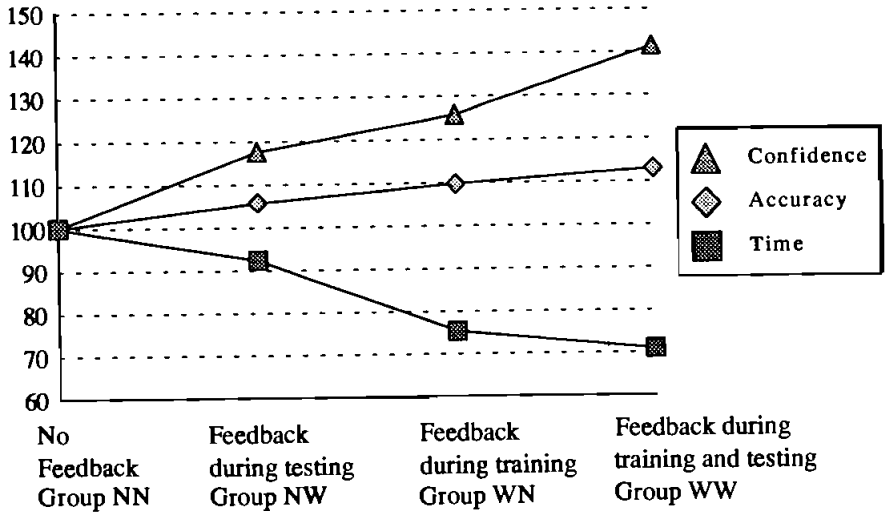


Figure 2. Percentage of scores relative to group NN's scores.

A common mistake as observed in the experiment is:

```
select employee.name
where employee.dept = 'research'
```

The feedback message is:

Instance variable employee does not have attribute dept

To cater for this common mistake, a template could be provided:

where < variable_name1 > < relationship-name > < variable_name2 > , < variable_name2.attribute > = ' '

In this case, whenever similar type of error is detected, the query could be matched to the template, variable names substituted accordingly, and the following result suggested to the user:

```
where employee work dept
dept.name = 'research'
```

5. Conclusion

We have presented the need for an active feedback system to improve user's performance during database feedback. Such a system was developed to provide feedback for user queries based on the entity relationship model. The system was empirically tested with 60 subjects in a two-by-two factorial experiment.

The results confirm the hypotheses that feedback will help to improve user performance in writing database queries, in particular for the measures of accuracy and confidence. For the measure of time, though subjects with feedback took less time than subjects without feedback, statistical significance is found only for the presence of feedback during training, and not for feedback for testing.

This study has concentrated on database retrieval. Another important type of user–database interaction is database update. For the higher level models, such as entity–relationship model, there are many inherent constraints such that a simple looking update may have many propagation updates. For example, deleting an entity instance should result in the deletion of its relationship instances. For systems that will allow user-defined constraints, there can be even more propagation updates. This is one area that feedback on the semantics of an update command *and* the propagation updates will be important for users, especially for users who do not know all the connections in the database model.

References

AMANO, H. and KAMBAYASHI, Y. L. 1991, Translation of SQL queries containing nested predicates into pseudonatural language, *The Second International Symposium on Database Systems for Advanced Applications*, 116–125.

ANG, S., STRAUB, D. W., CUMMINGS, L. L., EARLEY, P. C. and CURTIS, L. 1991, Effects of information technology on feedback seeking, in J. I. DeGross, I. Benbasat, G. DeSantis and C. M. Beath (eds), *Proceedings of the Twelfth International Conference on Information Systems*, 165–175.

BATRA D., HOFFER, J. A. and BOSTROM, R. P. 1990, Comparing representations with relational and EER models, *Communications of the ACM*, **33**, 126–139.

CAMPBELL, D. J. 1988, Task complexity and strategy development: a review and conceptual analysis, *Academy of Management Review*, **13**, 40–52.

CHAMBERLAIN, D. D. 1980, A summary of user experience with the SQL data sublanguage, *Proceedings of the International Conference on Data Bases*, 181–203.

CHAN H. C, WEI K. K. and SIAU K. L. 1991, Conceptual level versus logical level user–database interaction, in J. I. DeGross, I. Benbasat, G. DeSantis, and C. M. Beath (eds), *Proceedings of the Twelfth International Conference on Information Systems*, 29–40.

Chan, H. C., Wei. K. K. and Siau, K. L. 1993, A rule-based system for query feedback, in J. F. Nunamaker, Jr. and R. H. Sprague, Jr. (eds) *Proceedings of the 26th Annual Hawaii International Conference on System Sciences, Vol. III, Decision Support Systems and Knowledge-Based Systems* (IEEE Computer Society Press, New York), 53–61.

EARLEY, P. C., NORTHCRAFT, G. B., LEE, C. and LITUCHY, T. R. 1990, The impact of process and outcome feedback on the relation of goal setting and task performance, *Academy of Management Journal*, **33**, 87–105.

JARVENPAA. S. L. and MACHESKY, J. J. 1986, End user learning behavior in data analysis and data modeling tools, *International Conference on Information Systems*, 152–167.

JUNET, M. 1987, Design and implementation of an extended entity–relationship data base management system (ECRINS/86), in S. Spaccapietra (ed.) *Entity–Relationship Approach* (Elsevier, Amsterdam), 305–322.

KATZEFF, C. 1989, Strategies for testing hypotheses in database query writing, in F. Klix. N. A. Streitz, Y. Waern and H. Wandke (eds), *Man–Computer Interaction Research* (Elsevier, Amsterdam), 125–147.

KEPPEL, G. 1982, *Design and Analysis: A Researcher's Handbook* (Prentice Hall, New Jersey).

LUK, W. S. and KLOSTER, S. 1986, ELFS: English language from SQL, *ACM Transactions on Database Systems*, **11**, 447–472.

MOTRO, A. 1990, Flex: a tolerant and co-operative user interface to databases, *IEEE Transactions on Knowledge and Data Engineering*, **2**, 231–246.

POONEN, G. 1979, CLEAR: A Conceptual Language for Entities and Relationships, in W. W. Chu and P. P. Chen (eds) *Centralized and Distributed Data Base Systems* (IEEE Computer Society, New York), 194–215.

TEMPLETON, M. and J. BURGER 1986, Considerations for the development of natural language interfaces to database management systems, in L. Bolc and M. Jarke (eds), *Co-operative Interfaces to Information Systems* (Springer-Verlag, Berlin), 67–99.

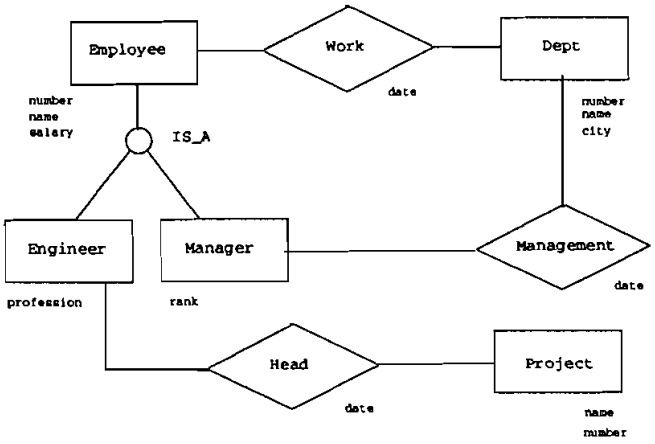
THOMAS, J. C. and GOULD, J. D. 1975, A psychological study of query by example, *Proceedings of the National Computer Conference* (AFIPS Press, New York), pp. 439–445.

WELTY, C. 1985, Correcting user errors in SQL, *International Journal of Man–Machine Studies*, **22**, 463–477.

Appendix A: Test materials

Entity–relationship model

The following diagram shows the database in the test.



Test questions, sample answers, and English feedback

Each of the ten test questions are listed below. The sample answers are given, followed immediately by the English feedback in a different font.

1. Show the names and numbers of all employees.
   select employee.name, employee.number
   Find an employee
   Show employee's name, employee's number

2. Show the department name and city.
   select dept.name, city
   Find a dept
   Show dept's name, dept's number

3. Show the engineers' numbers, names and professions.
   select engineer.number, name, profession
   Find an engineer
   Show engineer's number, engineer's name, engineer's profession

4. Show the names of employees who head any projects.
   select employee.name
   where employee head project
   Find an employee
   find a project
   find a head
   show employee's name
   such that employee is related to project
   through the relationship instance head.

5. Show the names of employees who work in the research department.
   select employee.name
   where employee work dept, dept.name = 'research'
   Find an employee
   find a dept
   find a work
   show employee's name
   such that employee is related to dept
   through the relationship instance work
   and dept's name = 'research'

6. Show the names of employees who work in the same department as Jack.
   e is an employee. e1 is an employee
   select e1.name
   where e1 work-related dept,
   e work-related dept,
   e.name = 'jack'
   find an employee e
   find an employee e1
   find a dept
   show e1's name
   such that e1 is related to dept through the relationship work
   and e is related to dept through the relationship work
   and e's name = 'jack'

7. Show the names of the employees with higher salary than Jack.
   e is an employee,
   e1 is an employee
   select e1.name
   where e1.salary > e.salary
   e.name = 'jack'
   Find an employee e,
   find an employee e1.
   Show e1's name
   such that e1's salary > e's salary and e's name = 'jack'.

8. List the names of managers who manages more than one department.
   select manager.name
   where manager manage-related > 1 dept
   Find a manager,
   Show manager's name
   such that manager is related to > 1 instances of dept
   through the relationship manage.

9. List the names of engineers who do not head any projects.
   select engineer.name
   where engineer head-related no project
   Find an engineer
   Show engineer's name
   such that engineer is NOT related to any instance of project
   through the relationship head

10. List the names of engineers who head all projects.
    select engineer.name
    where engineer head-related all project
    Find an engineer
    how engineer's name
    such that engineer is related to ALL instances of project
    through the relationship head.

Appendix B. Sample sessions

Two examples are given below of how a subject modified his queries based on the computer feedback. The first example shows a simple question where the subject only need to modify once. The second example shows a more complicated question, where the subject had to modify four times to get the correct query.

Question 1: Show the name of employees who work in the research department

Query version 1: select employee.name
where dept.name = 'research'
Feedback: find an employee, find a dept
show employee's name
such that dept's name = 'research'
note: employee and dept are not connected
Query version 2: select employee.name
where employee work-related dept,
dept.name = 'research'
Feedback: find an employee, find a dept
show employee's name
such that employee is related to dept through the relationship work and dept's name = 'research'

The subject was happy with version 2, which was correct.

Question 2: Show the name of employees who work in the same department as Jack.

Query version 1: A is employee, B is employee
select A.name
where A dept-related B, B.name = 'jack' >
Feedback: find an employee B, find an employee A.
show A's name
such that B's name = 'jack'
syntax error: < A is employee > is incorrect!
Correct syntax is < A is a/an employee > .
syntax error: < B is employee > is incorrect!
Correct syntax is < B is a/an employee > .
ERROR: Bad combination condition: dept is not of relationship type
Query version 2: A is an employee, B is an employee
select A.name
where A work-related B, B.name = 'jack'
Feedback: find an employee B, find an employee A.
show A's name.
such that B's name = 'jack'
ERROR: Bad combination condition: Both entity instance variables A and B are of the same type employee.
Query version 3: A is an employee, B is an employee
select A.name
where A dept-related B
Feedback: find an employee B, find an employee A.
show A's name
ERROR: Bad combination condition: dept is not of relationship type
Query version 4: A is a employee, B is a employee
select A.name
where A work-related dept = B work-related dept,
B.name = 'jack'
Feedback: find a dept, find an employee B, find an employee A
show A's name
such that A is related to dept through the relationship work and B's name = 'jack'
syntax error: < A work-related dept > = < B work-related dept >
Wrong usage of compare condition!!
correct answer: < select .... where A work-related dept, B work-related dept
Query version 5: A is an employee, B is an employee
select A.name
where A work-related dept, B work-related dept,
B.name = 'jack'
Feedback: find a dept, find an employee B, find an employee A
show A's name
such that A is related to dept through the relationship work
and B is related to dept through the relationship work
and B's name = 'jack'.

The query is finally correct.