Human factors evaluation of two data base query languages—Square and Sequel

by PHYLLIS REISNER, RAYMOND F. BOYCE and DONALD D. CHAMBERLIN
IBM Research Laboratory
San Jose, California

INTRODUCTION

Boyce et al. have recently described two data base query languages, SQUARE and SEQUEL, which are intended for use in an interactive mode by both programmers and professional non-programmers (e.g., accountants, lawyers, managers). The languages are comparable in the sense that the basic operators, underlying data structures and intended use are the same. They differ primarily in syntactic form, with a few additional differences in some of the specific features. Both of the languages are intended to be easily learned and used by people without specialized computer training.

This paper reports on a human factors experiment intended to evaluate the languages and, where possible in the same experiment, to compare them.

Specifically, the main goal of this experiment was to determine whether the languages could, as expected, be used by the intended populations. We were particularly interested in determining whether non-programmers, of professional caliber, could use the languages without extensive training.

A second goal was to determine whether there was a difference in useability of the two languages.

A third goal, at least as important as the preceding, was to discover, prior to implementation, types of user errors that were commonly made. We felt that a language intended for non-programmers should be “user-friendly,” with extensive use of the machine to catch human error. Common error types, if known, could be handled either by language changes, interactive user aids, or special emphasis in teaching.

GENERAL APPROACH

Arrangements were made with the psychology department of a local university to provide students as experimental subjects and to provide classroom space to run the experiment.

To evaluate and compare the languages, a common teaching curriculum was developed and used to teach four classes to four different subject populations: SEQUEL for programmers, SEQUEL for non-programmers, SQUARE for programmers, and SQUARE for non-programmers. Each class extended over a two week span. Extensive testing in the use of the language taught was carried out both during and after the classes.

SUBJECTS

The subject population consisted of 61 undergraduates and three graduate students from the local university, paid for their participation. They were not preselected in any known way, and thus varied considerably in background. Among the 64 participants in the experiment were students from 29 diverse majors, e.g., accounting, fine arts, recreation, mathematics, nursing, and political science.

We defined a “programmer” as anyone who had taken at least one programming course, and a “non-programmer” as anyone else. Thus there was a considerable spread of experience even within the “programmer” group. Subjects were not required to have any specific mathematics background (other than that required for admission to the university). Consequently many were unfamiliar with concepts and notation used in the languages: set comparisons, logical connectives, and even elementary mathematical notation such as “>.”

The number of subjects completing each class was as follows: SEQUEL programmers, 18; SQUARE programmers, 11; SEQUEL non-programmers, 15; SQUARE non-programmers, 20. Several additional subjects dropped out before completion of the classes for various reasons such as illness.

THE LANGUAGES

The SQUARE and SEQUEL query languages are both based on the relational model of data proposed by E. F. Codd. All information in the data base is assumed to be represented in a series of named tables, or “relations.” The columns of the tables have names, and each row of each table represents information about some entity in the real world such as an employee. Figure 1 shows two example tables which describe the employees and departments of a small company.

SQUARE (Specifying Queries As Relational Expressions) utilizes a mathematical notation to express queries against a data base, while SEQUEL (Structured English Query Language) is based on English keywords. The central concept...
in each language is that of a "mapping," which returns the data-values in some column which are associated with a known data-value in another column, as illustrated by Q1.

Q1. Find the names of employees in Department 50.

SQUARE: EMP ('50')
NAME DEPTNO
SEQUEL: SELECT NAME FROM EMP WHERE DEPTNO = 50

Both SQUARE and SEQUEL allow the result of one mapping to be used as input to another mapping. This process, illustrated by Q2, is called "composition."

Q2. Find the names of those employees who work for a department located in Stockton.

SQUARE: EMP o DEPT ('STOCKTON')
NAME DEPTNO DEPTNO LOC
SEQUEL: SELECT NAME FROM EMP WHERE DEPTNO = 'STOCKTON'

Both SEQUEL and SQUARE allow built-in functions, including AVG, SUM, COUNT, MAX, and MIN, to be applied to any set of numeric data-values (such as the result of a mapping.)

For complex queries, the features of the two languages differ slightly. SQUARE employs a notation called a "free variable" in which the user invents a variable to represent a row, then states a test which defines the rows he wishes to be selected, as in Q3:

Q3. List the department numbers of those departments having average salary greater than 15000.

SQUARE: X ∈ EMP:
AVG(EMP(X)) > 15000
SAL DEPTNO DEPTNO
SEQUEL avoids most uses of variables by means of a feature called GROUP BY, which organizes the rows of a table into groups by matching the values in some column, and then applies a built-in function to the rows in each group. This feature is illustrated by the SEQUEL expression for Q3.

SEQUEL: SELECT DEPTNO FROM EMP GROUP BY DEPTNO WHERE AVG(SAL) > 15000

Both SQUARE and SEQUEL allow use of the set-operators union, intersection and difference. Both languages permit two tables to be joined together by matching values in a common column, as in Codd's "join" operations.

The teaching and testing for this experiment covered the query features of the two languages as presented in the references cited, including assignment of query results to new tables. Insertion, deletion, and update features, although they were defined for the two languages, were not included in the experiment.

TEACHING

The languages were taught in a classroom-type situation with all its attendant problems (late arrivals, Monday fatigue, classes missed for illness, etc.) The expository materials, English examples to be translated into the query language, manuals, and quizzes were the same in all four classes. The classes were highly interactive, with extensive student participation. We decided that the goal of the classes should be to teach the fundamentals of the languages to as many subjects as possible. Therefore, the pace of each class was determined by the slower members.

In general, the programmers were able to learn the languages somewhat faster than the non-programmers; hence the classes for programmers were completed after 12 academic hours of instruction, while the non-programmer classes required 14 academic hours to cover the same material.

Several example data bases were used in teaching to accustom the students to using different data bases (monopoly, school administration, computer dating, car sales).
TESTING

The major tests were: five review quizzes given at intervals during the class, a final exam at the end of the class, and a memory test given one week later. In each, a set of questions was presented in English (e.g., "Find the names of all employees who make more than their managers") and the student was required to write the appropriate query language statement. The English questions on the tests were the same for all four populations. The final exam and memory test contained forty questions each, to be completed in two hours.

The questions on the tests were carefully designed to include all features of the languages. The major emphasis of the testing was on understanding the basic features of the languages, such as mapping, composition, free variables, and set operations. Therefore, 35 of the 40 questions on the final exam were “Basic Feature” questions designed to test each of the basic features individually. The remaining five questions, called “Combination of Features” questions, tested subjects’ ability to combine some of the basic features in ways they had not seen before. The memory test also contained 35 comparable Basic Feature questions, covering the same syntactic features as those on the final exam, plus five filler questions.

We attempted to write English questions for the test that were unambiguous and did not contain features, other than the ones being tested, that might create difficulties. The order of the questions on the final exam and memory test was randomized. Students were instructed that accuracy of writing queries, rather than speed, was the goal on all the exams.

The tests used data bases that had not been seen before by the students (airport administration, department store.) Each student was given a sample data base containing examples of data items in every column, for reference in formulating queries. Students were permitted to use class notes and other teaching materials on the reviews and final exam; however, the memory test was “closed book.”

SCORING

Scoring the responses presented some difficulty. Obviously, a simple ‘correct’ or ‘incorrect’ decision is the simplest choice. Just as clearly, however, errors such as a misspelled word or an omitted quotation sign around a data value represent less serious misunderstanding of the language than failure to use a test on the right side of a free variable statement.

We thus devised the following scoring system:

\[ C = \text{completely correct} \]
\[ D = \text{minor data error} \]
\[ M = \text{minor language error} \]
\[ S = \text{error of substance} \]
\[ F = \text{error of form} \]

Minor language errors (M) were, for example, misspelled column names or omitted quotation marks. Data errors (D) were, for example, using only a surname (Jones) when the sample data base required the full name (John Jones). These data errors were not errors that could reasonably be charged to the query language itself, since they could easily occur with any language. However, a query with such an error would not produce the correct response. We chose therefore to tabulate them separately.

Errors of substance (S) were valid forms that would run, but produce the wrong answer. Errors of form (F) were invalid forms.

In order of judged seriousness, we considered that F errors represented the most severe misunderstanding of the language, then S, M, and D in that order. Each question was given the score of its worst error. Repetitive errors were counted each time they occurred. For example, if subject 5 misspelled “personnel” in both question 10 and question 15, this was scored as two errors.

RESULTS

Overall evaluation and comparison

To obtain a general picture of student performance, we combined the correct responses (C) with those containing minor language errors (M) and minor data errors (D) to obtain the mean percentage of “essentially correct” responses in each population. These results are shown in Table I.

The overall mean score for programmers on the final exam (counting basic feature questions only for both languages) was 77.6 percent; that for non-programmers was 59.8 percent. Analysis of variance showed that the difference in overall mean scores between programmers and non-programmers was significant (p < .01).

<table>
<thead>
<tr>
<th>TABLE I—Mean Percentage of Essentially Correct Responses</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQUARE</td>
</tr>
<tr>
<td>-------</td>
</tr>
<tr>
<td>Non-prog.</td>
</tr>
<tr>
<td>Prog.</td>
</tr>
<tr>
<td>Mean</td>
</tr>
<tr>
<td>a. Final Exam, Basic Functions</td>
</tr>
<tr>
<td>SQUARE</td>
</tr>
<tr>
<td>Non-prog.</td>
</tr>
<tr>
<td>Prog.</td>
</tr>
<tr>
<td>Mean</td>
</tr>
<tr>
<td>b. Final Exam, Combination of Functions</td>
</tr>
<tr>
<td>SQUARE</td>
</tr>
<tr>
<td>Non-prog.</td>
</tr>
<tr>
<td>Prog.</td>
</tr>
<tr>
<td>Mean</td>
</tr>
<tr>
<td>c. Memory Test, Basic Functions</td>
</tr>
</tbody>
</table>

From the collection of the Computer History Museum (www.computerhistory.org)
Analysis revealed that neither the overall difference between SQUARE and SEQUEL nor the interaction effect was statistically significant. However, examination of Table I reveals that non-programmers showed greater facility with SEQUEL than with SQUARE, as expected. The mean scores for the final exam, basic function questions for non-programmers were 54.7 percent for SQUARE and 65.0 percent for SEQUEL; this difference was significant (p<.05).

Table I shows only small differences between the Basic Feature scores and the Combination of Features scores for the various populations. Subjects could apparently use the features they had learned in class to generate the novel forms required on the test about as well as they could use the individual basic features. From this we conclude that the subjects were actually expressing ideas in the new language rather than merely selecting from a set of known syntactic patterns.

In general, scores on the memory test showed no apparent decrease as compared to scores on the final exam, despite the one-week gap and despite the fact that students were not allowed to use notes on the memory test. This shows a high level of retention of the learned languages (and suggests that the final exam itself was probably a learning experience.)

Distribution of scores

Figure 2 shows the cumulative distribution of scores on the final exam (Basic Feature questions) for the four subject populations. The curves show that at least 50 percent of the test queries were expressed "essentially correctly" by the following proportions of the populations:

- 75% of SQUARE non-programmers
- 80% of SEQUEL non-programmers
- 91% of SQUARE programmers
- 94% of SEQUEL programmers

Learning curves

An attempt was made to determine the rate at which subjects learned several of the more important features of the languages, using data from the five reviews and the final exam. Figure 3 gives the learning curve for a simple mapping, showing the mean percentage of simple mapping questions expressed "essentially correctly" on the various tests by each of the four populations. It can be seen that programmers could use simple mapping in either language with over 90 percent accuracy after about two hours' instruction, and non-programmers achieved this level of proficiency after about four hours' instruction.

Learning curves for composition were also obtained but yielded more erratic patterns. On the first review testing use of composition (Review 3) the scores were:

- SQUARE non-programmers: 80%
- SEQUEL non-programmers: 79%
- SQUARE programmers: 91%
- SEQUEL programmers: 89%

Clearly, at this time, subjects understood the composition feature and could select it appropriately from the other features they had learned. After this point, scores on composition questions tended to decline, reaching the following level...
Confusion generated by the later introduction of other features, such as free variables, may account for the declining ability of subjects to use composition.

Scores on questions requiring the use of free variables in SQUARE or GROUP BY in SEQUEL were much lower than those for the simpler features. Comparison of SEQUEL with SQUARE for these features should be undertaken cautiously, since the one-to-one parallelism found between SEQUEL and SQUARE for other features does not always hold between free variables and GROUP BY. On Review 5, immediately after introduction of these features, the scores for free variable and GROUP BY questions were:

<table>
<thead>
<tr>
<th>Category</th>
<th>SQUARE non-programmers</th>
<th>SEQUEL non-programmers</th>
<th>SQUARE programmers</th>
<th>SEQUEL programmers</th>
</tr>
</thead>
<tbody>
<tr>
<td>C (Correct)</td>
<td>42.7%</td>
<td>56.2%</td>
<td>66.0%</td>
<td>68.2%</td>
</tr>
<tr>
<td>D (Minor Data Error)</td>
<td>2.2%</td>
<td>1.9%</td>
<td>2.1%</td>
<td>1.3%</td>
</tr>
<tr>
<td>M (Minor Lang. Error)</td>
<td>9.9%</td>
<td>6.9%</td>
<td>9.6%</td>
<td>7.9%</td>
</tr>
<tr>
<td>Total Essentially Correct</td>
<td>54.8%</td>
<td>65.0%</td>
<td>77.7%</td>
<td>77.4%</td>
</tr>
<tr>
<td>S (Error of Substance)</td>
<td>11.5%</td>
<td>10.9%</td>
<td>10.6%</td>
<td>10.2%</td>
</tr>
<tr>
<td>F (Error of Form)</td>
<td>33.8%</td>
<td>24.2%</td>
<td>11.7%</td>
<td>12.4%</td>
</tr>
<tr>
<td>Total Incorrect</td>
<td>45.3%</td>
<td>35.1%</td>
<td>22.3%</td>
<td>22.6%</td>
</tr>
</tbody>
</table>

By the final exam, the scores for these features had fallen to the following levels:

<table>
<thead>
<tr>
<th>Category</th>
<th>SQUARE non-programmers</th>
<th>SEQUEL non-programmers</th>
<th>SQUARE programmers</th>
<th>SEQUEL programmers</th>
</tr>
</thead>
<tbody>
<tr>
<td>C (Correct)</td>
<td>11%</td>
<td>23%</td>
<td>48%</td>
<td>48%</td>
</tr>
<tr>
<td>D (Minor Data Error)</td>
<td>33%</td>
<td>35%</td>
<td>22%</td>
<td>22%</td>
</tr>
<tr>
<td>M (Minor Lang. Error)</td>
<td>48%</td>
<td>48%</td>
<td>48%</td>
<td>48%</td>
</tr>
</tbody>
</table>

It appears that GROUP BY is somewhat easier to use than free variables, but both features are likely to prove difficult for non-programmers to use correctly. This conclusion is reinforced by comments made by students during the classes.

**Error analysis**

Table II gives the detailed breakdown of responses into five error categories for the four populations on the final exam (Basic Feature questions). It should be noted that the M and D scores do not represent the total occurrence of minor language and data errors in the experiment, because each response is placed in the category of the worst error it contains.

We observe that F-type errors, denoting inability to write syntactically correct language forms, occurred more often among non-programmers than among programmers. Also, among non-programmers, F-type errors occurred more often with SQUARE than with SEQUEL.

One of the goals of this experiment was to identify common types of user errors for which special aids could be built into the system. Study of M-type and D-type errors has yielded some preliminary results in this area.

A common tendency among subjects was to insert words from the English sentence into the SEQUEL or SQUARE query in place of the correct table name, column name, or data value. Thus, subjects wrote "NAMES" rather than "NAME," "SOLD" rather than "SALES," "PERSONNEL" rather than "EMPLOYEE," etc. These "intrusion errors" were made in both languages, by both programmers and non-programmers, in spite of the fact that subjects had sample data bases to refer to during the tests. A "user-friendly" system might be able to correct some of these minor errors by such techniques as word-stem matching or alternative-form dictionaries.

Misspellings of keywords, table names, and column-names were also common, despite the availability of the sample data base. This suggests that a spelling corrector would be another important part of a "user-friendly" system.

**DISCUSSION OF METHOD**

An early decision was made that the SQUARE and SEQUEL classes for this experiment would be taught in a conventional classroom situation with a human teacher rather than by a system of programmed or computer-assisted instruction. This decision was made because classroom teaching is relatively quick to implement and known to be effective, and because it provides opportunity for on-the-spot feedback between the teacher and the student. On the other hand, a programmed instruction or CAI curriculum would have been more reproducible. Furthermore, an individualized curriculum allowing each student to learn at his own rate and to take an examination when ready would have provided a more accurate measure of learning times for individual students.

All testing for the experiment was done on paper in the classroom. If an actual interactive data base management system had been available for testing, it might have been able to provide feedback to prevent subjects from making the same error repeatedly. On the other hand, such a test would confuse difficulty in use of the language with difficulty in learning to use the terminal system.

A limitation of the experiment was that the effectiveness...
of the teaching techniques and the learnability of the languages tend to be confounded. Since the results of the experiment were favorable, we conclude both that the languages are reasonably easy to learn and that the curriculum was reasonably effective; however, it is difficult to separate the two factors. To some extent, the goals of the experiment were conflicting (e.g., we wished to study user errors, but if the teaching were perfectly effective, errors would be eliminated).

NEED FOR FURTHER WORK

The field of language evaluation is in its infancy and provides fertile opportunities for further research. The present experiment focused on the learnability of languages, but other language aspects remain to be measured; for example, the ease with which one subject can understand a query written by someone else, or the ease with which a query can be modified.

In our experiment, subjects were unsystematically assigned to classes with the expectation that individual differences among subjects would "balance out" and allow the different classes to be compared (e.g., SQUARE class vs. SEQUEL class for non-programmers). This aspect of the experiment could be made more rigorous if an "aptitude measure" were developed which would enable matching the aptitudes of students in two populations before the experiment is begun.

SUMMARY

A series of experiments was conducted to evaluate the learnability of two data base query languages, SQUARE and SEQUEL, using university students as subjects. The students were divided into two groups according to whether they had had programming experience. Experiments showed that both populations were able to use either language with reasonable proficiency after 12 to 14 academic hours of instruction (lowest class mean was 55 percent for SQUARE non-programmers; mean score for programmers was 77 percent in both languages). Programmers learned both languages more quickly and more completely than did non-programmers, and the non-programmers showed greater proficiency with SEQUEL than with SQUARE. Test scores showed that both programmers and non-programmers were able to combine the basic language features in ways they had not been explicitly taught. Also, it was shown that, one week after the end of instruction, subjects still retained nearly all their proficiency and were able to use the languages successfully on a closed-book examination.

The individual features of the languages varied considerably in learnability. The basic language feature, a simple mapping, was learned in each language with near-perfect accuracy by programmers after two hours and by non-programmers after four hours. However, considerable difficulty was experienced in learning and retaining the more complex "free variable" and "GROUP BY" features, especially by non-programmers.

A study of errors made by subjects suggests that a real data base system should be prepared to correct minor syntactic errors and to search for poorly-specified data values by some technique such as stem-matching or a synonym dictionary.

REFERENCES


BIBLIOGRAPHY