A Human Factors Comparison of a Procedural and a Nonprocedural Query Language.*

Charles Welty
David W. Stemple

COINS Technical Report 78-24
December 1978

* This research was funded, in part, by NSF Grant MCS78-07616.

ABSTRACT

An experiment testing the ability of subjects to write queries in two different query languages has been run. The two languages, TABLET and SQL, differ only in their procedurality -- both languages use the relational data model and their Halstead levels are similar. Constructs in the languages that do not affect their procedurality are identical. Subjects were taught using manuals for the two languages that contained identical examples and problems in identical order. The results of the experiment show that subjects write difficult queries significantly better using the procedural language than subjects using the non-procedural language do. The results of the experiment are also used to compare corresponding constructs in the two languages and to recommend improvements for these constructs.

INTRODUCTION

Structured programming concepts have led to the realization that the prime element in computer systems is the human element. Although in terms of cost, speed, reliability and most other "efficiency" measures, the machine is far superior to the human, humans remain a part of the system. Any efficiency in the use of system resources is wasted if a system is not designed to match the needs and abilities of its users. This fact has led to the exploration of new research areas involving the human-oriented aspects of computer systems.

In the field of computer languages, human factors testing can be used to:

1. Test if a language is learnable. Failure of this test may dictate a language's demise.
2. Eliminate minor difficulties in a language. This is used in the planned evolution of the language.

Much of the work in human factors testing has been done on general purpose languages. Language constructs [Gannon and Horning, 1975, Shneiderman, 1974, Sime, Green and Guest, 1973, Weissman, 1974], flowcharting [Shneiderman, Mayer, McKay and Heller, 1975], errors [Youngs, 1976], and debugging [Gould, 1975] have been tested. These studies involve the testing of groups of programmers and the statistical analysis of the results.

Basic research has also been done on non-programmers to study the relationship between problem solving and programming [Miller, 1976 and Miller and Becker, 1974].

Query language users are generally considered to be non-programmers. The querying of a data base is only an occasional part of the user's job. Because of the user's lack of computer experience and intermittent use of the language, it is necessary to determine that a query language is easy to learn, use and remember. This determination may be achieved through human factors testing.

There have been several human factors studies of query languages. Gould and Ascher experimented with IQF [Gould and Ascher, 1975, IBM, 1972]. Their experiment showed the language to be difficult to learn. Zloof's Query by Example [Zloof, 1974] was studied by Thomas and Gould [Thomas and Gould, 1975] and found easy to learn and use. Reisner, et al [Reisner, Boyce and Chamberlin, 1975 and Reisner, 1976] made a study of both SEQUEL [Chamberlin and Boyce, 1974] and SQUARE [Boyce, Chamberlin, King and Hammer, 1975] with the result that SEQUEL was found relatively easy to learn. This experiment helped in the evolution of SEQUEL 2 [Chamberlin, et al, 1970]. Lochovsky [Lochovsky, 1978] studied various data models and their associated data manipulation languages.

All of the query language testing was done without actual systems. The languages were presented in classroom situations and testing was through classroom exams. The reason for this was twofold:

1. The purpose of the testing was to study the language before implementation.

2. Use of an actual system would have introduced the additional factors of terminal availability, system response time, subjects' typing expertise, etc.

Human factors research is readily applicable to query languages for several practical reasons:

1. There are no entrenched query languages which would be either hard or impossible to replace with a language of proven superiority.

2. Non-programming test subjects are in general supply.

3. Query languages are oriented toward individual use which is easily tested.

4. A realistic query problem is quite simple, allowing a large number of them to be done on an exam of reasonable length.

5. Query languages have a small set of data and statement types. There are usually no control structures. Thus, they are easy to learn.

6. Even with their simplicity and restricted problem domain, query languages have the potential to reach a very large user community.

These reasons assume added potency when compared to human factors testing of general purpose languages. General purpose languages have none of these attributes but their study has yielded useful experimental results.

HYPOTHESIS

One of the major issues in database query languages concerns the procedurality of query languages. An experiment has been run testing the learnability of a nonprocedural query language and a procedural query language using human subjects. The experiment tested the basic hypothesis: People more often write difficult queries correctly using a procedural query language than they do using a non-procedural query language. The languages chosen for this experiment must be similar in all respects except in the independent variable -- procedurality -- since the experiment takes the reductionist approach [Shneiderman, 1978].

SQL (formerly SEQUEL 2) [Welty, 1978a, Denny, 1977] and TABLET [Welty, 1978b, Stemple et al, 1978] exhibit the required properties. The similarities between SQL and TABLET are:

1. They both use the same data model, Codd's relational model [Codd, 1970].
2. They are relationally complete [Codd, 1971b].
3. They have similar language levels [Halstead, 1977].
4. Their syntactic differences are a function of their procedurality. Constructs that are independent of procedurality are identical.
5. They both use the same terminal equipment.

These similarities are detailed in [Welty, 1979].

The difference, again, is in the procedurality of the languages. A thorough treatment of procedurality and the procedurality of SQL and TABLET is given by Welty [Welty, 1979]. Generally, a language is procedural if it specifies a step-by-step method for achieving a result. Nonprocedural languages describe the desired result without specifying how it is to be achieved. (The idea is comparable to the difference between constructive and nonconstructive existence proofs in mathematics.) SQL is similar to Codd's relational calculus [Codd, 1971a] and TABLET (The Algebra Based Language for Enquiring of Tables) is based on Codd's relational algebra [Codd, 1971b].

Codd's relational algebra consists of a set of operations defined on relations. An operation on a relation or relations always yields another relation. A relational algebraic query specifies the ordered steps used in generating the result and, thus, is procedural.

A relational calculus query describes the elements of the desired relation. The query is purely descriptive, containing no method for achieving the desired relation. This type of query is nonprocedural.

Further discussion of the procedurality of the algebra and calculus is given by Codd [Codd, 1971a].

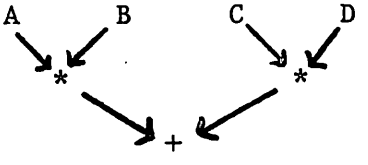
A procedurality metric, P, is suggested by Welty [Welty, 1979] for finding the procedurality of the implementation of any algorithm. The measure is defined as:

$$P = \frac{\text{no. of variable bindings}}{\text{no. of possible orderings of the bindings}} + \frac{\text{no. of operators}}{\text{no. of possible orderings of the operators}}$$

A high P value denotes a high level of procedurality. For example, the FORTRAN assignment statement

$$E = A*B+C*D$$

has one variable binding with only 1 possible order. A dependency diagram of the expression is



Either multiplication may be done first, followed by the other and then the addition. So, for the entire assignment statement

$$P = 1/1 + 3/2 = 2.5.$$

Using this measure, TABLET is significantly more procedural than SQL($p<.001$).

SUBJECTS

The subjects were 72 undergraduate students taking a 1 credit accounting course. Most subjects were business majors. The subjects were divided into two groups -- 35 subjects learned SQL and 37 subjects learned TABLET.

Subjects were also classified as "inexperienced" and "experienced". Inexperienced subjects had no previous experience with computers. Experienced subjects had a course in either BASIC or FORTRAN. A questionnaire showed that the subjects were familiar with the operators >, <, =, etc as well as the set operators union and intersection.

Subjects were motivated to take the course by the credit and a desire to learn about computers.

THE EXPERIMENT

The languages were taught using manuals read outside class. These manuals, one presenting SQL and the other presenting TABLET, contained identical examples and problems presented in the same sequence. Each manual contained 12 lessons. Concepts that were identical in the two languages were presented identically.

There were fourteen class meetings. These meetings were devoted to answering questions on the lessons and to quizzes covering the material presented in the lessons. No lecturing was done in the class meetings. Since no material was presented in class, subjects learned the languages entirely from the manuals.

A final exam was given immediately after the course. The final consisted of 30 English questions (e.g. How many suppliers supply item 19?) and the subjects were required to write the corresponding query in their query language. The final was an open book exam and the students grades were based solely on their final exams.

A retention test was given 3 weeks after the final. This test was of identical format to the final. In fact, although the questions were different on the final and the retention test, the correct response to each question on the final was identical in structure to the corresponding query on the retention test. The retention test was a closed book test. In addition, the subjects were discouraged from studying for the retention test. The retention test did not affect student grades, but it was a required part of the course. Students had to take the test but had no motivation for studying. This delay and lack of studying was used to model the intermittent use of query languages.

GRADING THE QUIZZES, THE FINAL AND THE RETENTION TEST

The grading method for queries in all the tests was similar to Reisner's [Reisner, 1976]. Each solution was classified as one of the following:

correct - the solution was completely correct.

minor language error (ML) - the solution was basically correct but had a small error that would be found by a reasonably good translator.

minor operand error (MO) - the solution has a minor error in its data specification, perhaps a mis-spelled column name.

minor substance error (MS) - the solution yields a result that is not quite correct but its incorrectness is due to the statement of the problem.

correctable (CO) - the solution is wrong but correctable by a good compiler.

major substance error (MS) - the query is syntactically correct but answers a different question than the one specified.

major language error (ML) - a major error in the syntax of the language has been made.

incomplete (IN) - incomplete query.

unattempted (UN) - no sloution was attempted.

The first four categories -- correct, ML, MO, MS -- were called <u>essentially correct responses</u>. The other five categories were classified as incorrect. The correctable solutions used knowledge specific to the database being used and would not be correctable in the general case. If several errors were found in a query, the error is categorized as the lowest one found in the above list.

THE LANGUAGES

The SQL and TABLET query languages are both based on the relational model of data. SQL uses English keywords in a template-like manner for the expression of queries against a database. TABLET specifies the operations that are performed on a relation. Some sample queries follow. These queries all use the COLLEGE database of Figure 1.

Q1. List the names of students from Ohio.

```
SQL:    SELECT NAME
        FROM STUDENT
        WHERE HOMESTATE = 'OHIO'

TABLET: FORM OHIOANS FROM NAME, HOMESTATE OF STUDENT
           KEEP ROWS WHERE HOMESTATE = 'OHIO'
           PRINT NAME
```

This SQL query is called a "simple mapping" and returns a value from the NAME column of a tuple in the STUDENT relation for which the value in the HOMESTATE column is 'OHIO'. The TABLET query first forms a working table named OHIOANS consisting only of the NAME and HOMESTATE columns of the STUDENT table. The KEEP ROWS command specifies that the rows (tuples) of OHIOANS for which the HOMESTATE column contains 'OHIO' are retained. The other rows are eliminated from the table. The values in the NAME column are then printed. Both SQL and TABLET eliminate duplicates from the tuples printed.

Q2. List the average salary of economics faculty members.

```
SQL:    SELECT AVG(SALARY)
        FROM FACULTY
        WHERE DEPT = 'ECONOMICS'

TABLET: FORM ECONSAL FROM SALARY, DEPT OF FACULTY
           KEEP ROWS WHERE DEPT = 'ECONOMICS'
           PRINT AVG(SALARY)
```

data base - COLLEGE

STUDENT

ID	NAME	SEX	HOMESTATE	MAJOR	REPID
1	JOHN JONES	M	MASSACHUSETTS	HISTORY	2
2	JANE DOE	F	OHIO	ECONOMICS	9

TAKING

ID	COURSE	SECTION
1	HIST101	1
1	HIST102	2
1	POLSCI115	1
2	ECON105	3
2	ECON202	1
2	MATH101	1

FACULTY

ID	NAME	SEX	DEPT	COMHEAD	SALARY
312	BILL GRANT	M	ECONOMICS	216	20000
152	JOHN MILTON	M	HISTORY	312	14000
172	ANNE HALL	F	POLSCI	192	19000

DEPARTMENT

DEPT	BUILDING	HEAD
POLSCI	BILLINGS	172
HISTORY	BILLINGS	295
ECONOMICS	KEYNES	312
ENGINEERING	ENGINEERING	207

TEACHING

ID	COURSE	DEPT	SECTION	LIMIT	SIZE
312	ECON105	ECONOMICS	1	35	31
312	MATH101	MATH	2	40	40
152	HIST101	HISTORY	1	28	28
152	HIST102	HISTORY	2	32	19
172	POLSCI115	POLSCI	1	32	30

COURSES

COURSE	DEPT	TITLE	CREDITS
ECON105	ECONOMICS	INTRODUCTION TO ECONOMICS	3
MATH101	MATHEMATICS	COLLEGE ALGEBRA	3
HIST101	HISTORY	AMERICAN HISTORY	3
HIST102	HISTORY	EUROPEAN HISTORY	4

Figure 1
The COLLEGE database

Both languages allow functions in a query. The functions are MAX, MIN, AVG, SUM and COUNT. These functions apply to the given column. Duplicates are not eliminated from the column on which the function operates.

Q3. List the names of students taking ECON105.

```
SQL:  SELECT NAME
      FROM STUDENT
      WHERE ID =
            SELECT ID
            FROM TAKING
            WHERE COURSE = 'ECON105'

TABLET:  FORM ECONSTUDENTS FROM NAME, ID OF STUDENT
         ADD COLUMN COURSE OF TAKING BY ID = ID
         KEEP ROWS WHERE COURSE = 'ECON105'
         PRINT NAME
```

In the SQL query the lower mapping returns a set of ID's to the upper mapping, this is called chaining. In TABLET, the ADD columns statement joins [Codd, 1970] the COURSE column of the TAKING table to the ECONSTUDENTS table using equal ID values from the two tables. This operation results in ECONSTUDENTS containing 3 columns: NAME, ID and COURSE.

Q4. List the ID's of department heads who are also committee heads.

```
SQL:  SELECT HEAD
      FROM DEPARTMENT
    INTERSECT
      SELECT COMHEAD
      FROM FACULTY

TABLET:  FORM HEADID FROM HEAD OF DEPARTMENT
         FORM COMHEADID FROM COMHEAD OF FACULTY
         KEEP ROWS OF COMHEADID WHERE COMHEAD IN HEAD OF HEADID
         PRINT COMHEAD
```

SQL uses the usual set operators -- UNION, INTERSECT and MINUS. TABLET forms two tables and then performs a restriction (KEEP ROWS) of the tuples in one table using the contents of the other table. TABLET uses NOT IN corresponding to SQL's MINUS. The ADD ROWS command is the TABLET analog of UNION.

Q5. List the average salary of faculty members in each department.

```
SQL:    SELECT DEPT, AVG(SALARY)
        FROM FACULTY
        GROUP BY DEPT

TABLET: FORM DEPTAVG FROM DEPT, DALARY OF FACULTY
            GROUP BY DEPT
            PRINT DEPT, AVG(SALARY)
```

Both SQL and TABLET use GROUP BY to denote the partitioning of the relation (table). In SQL, GROUP BY is a clause in the SELECT statement. GROUP BY in TABLET is a command in itself.

Q5. List the name of each student and the names of the courses he is taking (eg. JOHN JONES HIST101).

```
SQL:    SELECT NAME, COURSE
        FROM STUDENT, TAKING
        WHERE STUDENT.ID = TAKING.ID

TABLET: FORM NAMECOURSE FROM NAME, ID OF STUDENT
            ADD COLUMN COURSE OF TAKING BY ID = ID
            PRINT NAME, COURSE
```

The join operation is required by both SQL and TABLET queries. NAME and COURSE come from the STUDENT and TAKING relations, respectively. The ID columns in the SQL WHERE clause are qualified to avoid ambiguity. The TABLET query uses the same format as in Q3.

RESULTS

The mean number of essentially correct solutions as well as mean times to take the final and retention tests, mean difficulty and mean study time for the lessons are found in Table 1. The results are subdivided into various categories, these are:

easy - 10 problems from lessons 1-5 of the manuals covering (in SQL terms) simple mapping, simple mapping with arithmetic operations, simple mapping with built-in functions, and composition (chaining).

hard - 20 problems from lessons 6-12 of the manuals. These cover GROUP BY, set functions, joining and combinations of constructs.

group - 9 problems that require the GROUP BY construct.

join - 3 problems requiring joining in both SQL and TABLET.

chaining - 5 problems that require chaining in SQL (TABLET uses join).

set - 3 problems using UNION, INTERSECT and MINUS (SQL terms).

Statistical analysis

The results of the retention test were analysed using a fully crossed, two way analysis of variance. The two independent variables were the languages (SQL and TABLET) and the experience level of the subjects (inexperienced and experienced). The results are summarized in Table 2.

Using the values of the means from Table 1 we see that experienced subjects required significantly less study time than inexperienced subjects. SQL subjects required significantly less study time than TABLET subjects (Table 2a).

Table 2b shows that inexperienced subjects found the lessons significantly more difficult than experienced subjects did.

The time required to take the retention test was significantly longer for inexperienced subjects than it was for experienced subjects. TABLET subjects took significantly longer than SQL subjects (Table 2c).

	No. of problems	SQL mean (all 35 subjects)	TABLET mean (all 37 subjects)	SQL mean (17 inexperienced subjects)	TABLET mean (20 inexperienced subjects)	SQL mean (18 experienced subjects)	TABLET mean (17 experienced subjects)
final score	30	18.314	18.514	16.824	17.000	19.833	20.294
final time (minutes)		116.71	120.35	123.54	124.10	110.28	115.94
retention score	30	13.600	14.784	12.412	12.800	14.722	17.118
retention time (minutes)		66.457	76.351	69.412	78.050	63.778	74.353
easy final	10	8.4286	8.1081	7.9412	7.5000	8.8889	8.8235
easy retention	10	7.8286	7.4054	7.2353	6.7500	8.3889	8.1765
hard final	20	9.943	10.405	8.8824	9.5000	10.944	11.471
hard retention	20	5.7714	7.3784	5.1765	6.0500	6.3333	8.9412
group final	9	3.2286	4.1892	2.8824	3.4000	3.5556	5.1178
group retention	9	1.0857	2.6486	.8824	1.7000	1.2778	3.7647
join final	3	1.3714	1.8108	1.1765	1.7500	1.5556	1.8824
join retention	3	.5429	1.4865	0.5294	1.4000	0.5556	1.5882
chaining final	5	3.2571	3.3514	2.9412	3.1000	3.5556	3.6417
chaining retention	5	3.1111	3.0000	2.5294	2.3500	3.1111	3.0000
set final	3	2.600	1.7027	2.5294	1.9000	2.6667	1.4706
set retention	3	2.1143	.9459	1.9412	1.0000	2.27778	.8824
average study time (minutes, lesson 1-12)		29.324	36.214	33.378	39.138	25.496	32.775
average difficulty (lessons 1-12; 1-easy, 10-hard)		4.1694	4.1789	4.5041	4.5885	3.8533	3.6971

Table 2

Mean number of essentially correct responses and other results according to subject experience and query category.

	degrees of freedom	F ratio	significance
experience	1,68	4.76	<.05
language	1,68	3.99	<.05
experience × language	1,68	.05	-

a Mean study time of Lessons 1-12.

	degrees of freedom	F ratio	significance
experience	1,68	11.33	<.001
language	1,68	.02	-
experience × language	1,68	.27	-

b Mean difficulty of Lessons 1-12.

	degrees of freedom	F ratio	significance
experience	1,68	1.44	-
language	1,68	6.10	<.05
experience × language	1,68	.06	-

c Time to take retention test.

	degrees of freedom	F ratio	significance
experience	1,68	10.96	<.001
language	1,68	2.03	-
experience × language	1,68	.97	-

d Retention score (essentially correct).

	degrees of freedom	F ratio	significance
experience	1,68	8.91	<.01
language	1,68	.65	-
experience × language	1,68	.10	-

e Retention, easy problems.

	degrees of freedom	F ratio	significance
experience	1,68	8.62	<.005
language	1,68	6.72	<.05
experience × language	1,68	1.53	-

f Retention, hard problems.

	degrees of freedom	F ratio	significance
experience	1,68	10.68	<.005
language	1,68	19.27	<.001
experience × language	1,68	4.92	<.05

g Retention, group problems.

	degrees of freedom	F ratio	significance
experience	1,68	.33	-
language	1,68	26.53	<.001
experience × language	1,68	.19	-

h Retention, joining problems.

	degrees of freedom	F ratio	significance
experience	1,68	4.42	-
language	1,68	.25	<.001
experience × language	1,68	.01	-

i Retention, set problems.

Table 2

Summary of analysis of variance for study time, Lesson difficulty and retention scores. (See Table 1 for means.)

The overall retention score was significantly higher for experienced than for inexperienced subjects. The total scores for SQL and TABLET subjects were not significantly different (Table 2d). The same is true for the easy problems (Table 2e).

The experienced subjects did significantly better than inexperienced subjects on the hard problems. TABLET subjects outperformed SQL subjects on the hard problems (Table 2f). This result supports our basic hypothesis.

Experienced subjects outperformed inexperienced subjects on the group problems (Table 2g). TABLET subjects outperformed SQL subjects. In this case, the interaction term (experience x language) is significant. This means that while the TABLET mean is higher than the SQL mean overall as well as for both the inexperienced and experienced subjects, experienced TABLET subjects did much better than experienced SQL subjects but inexperienced TABLET subjects did only a little better than inexperienced SQL subjects. Therefore, the significance of the TABLET over SQL subjects in the analysis of variance is due primarily to the experienced subjects.

Interestingly, the experience level makes no difference for join problems (Table 2h) or set problems (Table 2i). SQL subjects outperform TABLET subjects in the set problems. TABLET subjects outperform SQL subjects in join problems.

Generally, experienced students outperformed inexperienced students. SQL subjects took less time to learn their language, found it less difficult to learn, finished the retention test faster and did better on set problems than the TABLET subjects did. TABLET subjects outperformed SQL subjects on hard problems, group problems and join problems.

Interpretation of the results

Table 2 shows that the performance of the SQL and TABLET subjects showed significant differences in the following categories:

1. Average study time for lessons 1-12.
2. Time required to take the retention test.
3. Hard problems on the retention test.
4. Join problems on the retention test.
5. Set problems on the retention test.
6. Group problems on the retention test.

TABLET subjects required more study time and more time to take the retention test than SQL subjects did. TABLET has more complex syntax and semantics than SQL has. It takes extra time to learn and write TABLET. TABLET subjects are required to learn how to manipulate tables, SQL subjects are directed mainly by SQL's syntax.

TABLET's complexity yields a reward in the writing of hard queries. The skill acquired in table manipulations is put to use. The skill is analogous to the skill of riding a bicycle, once learned it is easily retained. SQL does not require this sort of skill of its users.

TABLET uses the join construct (ADD ROWS) where SQL uses two constructs, chaining and joining. While SQL subjects used chaining, TABLET subjects acquired experience with joining. On the problems in which both languages used joining, TABLET subjects had an advantage and performed significantly better than SQL subjects.

SQL subjects out performed TABLET subjects on set problems. Set concepts as well as set operators, UNION and INTERSECTION, were familiar to all the subjects due to the new math. SQL uses these familiar concepts. TABLET uses concepts novel to the subjects.

TABLET subjects outperformed SQL subjects on the group problems. This is an interesting result because both languages use the same construct -

GROUP BY column name.

TABLET has many commands that perform operations on relations, but SQL queries are more tuple-oriented. The GROUP BY specifies an operation on a relation, so it is more TABLET-oriented. Also, GROUP BY is an imperative construct, but appears as a subordinate clause within SQL's SELECT. In TABLET the GROUP BY is a separate (imperative) command. Finally, two SQL subjects said that the GROUP BY was in the wrong position in the command, suggesting it occur before the SELECT because the output specified in the SELECT is dependent on the GROUP BY clause. The SELECT clause is analogous to the PRINT in TABLET and GROUP BY does occur before the PRINT in TABLET.

RECOMMENDATIONS FOR LANGUAGE CHANGES

The results of this experiment show that there are problems with the following language elements:

1. Join in SQL.
2. Sets in TABLET.
3. Grouping in SQL.

Recommendations concerning join problems in SQL

The difference between SQL and TABLET for join problems is primarily due to the experience TABLET subjects received in using the ADD COLUMNS command while SQL used chaining.

Lochovsky [Lochovsky, 1978] recommends eliminating the join from the where clause and adding an explicit join specification to SQL. At present, the correct response to the query, "List the names of people who have exceeded their credit limit and the item number of items they have charged," is

```
SELECT NAME, ITEMNO
FROM CHARGEACCTS, CHARGED
WHERE CHARGEACCTS.ACCTNO = CHARGED.ACCTNO
   AND TOTALBILL > LIMIT.
```

The joining is done by the

```
CHARGEACCTS.ACCTNO = CHARGE.ACCTNO
```

in the where clause. A possible substitute query would be

```
SELECT NAME, ITEMNO
FROM CHARGEACCTS, CHARGED
JOINED BY CHARGEACCTS.ACCTNO = CHARGED.ACCTNO
WHERE TOTALBILL > LIMIT.
```

The JOINED BY could be simplified to

```
JOINED BY ACCTNO
```

since this is unambiguous. This syntax makes SQL even less calculus-oriented and more algebraic.

Recommendations concerning set problems in TABLET

TABLET should make use of the set theory background that is so common today. Explicit use of the UNION, INTERSECT and MINUS operators is recommended. For example, the correct response to the query, "List the account numbers of accounts with credit rating of 10 who have charged item 19," was

```
FORM RATING10
  FROM ACCTNO, RATING OF CHARGEACCTS
KEEP ROWS WHERE RATING = 10
FORM ITEM19
  FROM ACCTNO, ITEMNO OF CHARGED
KEEP ROWS WHERE ITEMNO = 19
KEEP ROWS OF RATING10 WHERE ACCTNO
    IN ACCTNO OF ITEM19
PRINT ACCTNO
```

The KEEP ROWS command, above, would be replaced by

```
FORM BOTH
  FROM ACCTNO OF RATING10
  INTERSECT ACCTNO OF ITEM19.
```

The query is still not as succinct as the SQL equivalent but is based on the same concepts.

Recommendations concerning group problems in SQL

It is tempting to use the student comments referred to earlier and put the GROUP BY clause earlier in the SELECT. It should appear before the SELECT itself. Making this change has ramifications that result in a procedural, TABLET-like language.

The simplest change is to make the GROUP BY a subordinate clause. Using the participle, GROUPED or GROUPING has this effect. For example, the correct response to the query, "For each supplier list the supplier name and the average wholesale price of the items he supplies," is

```
SELECT SUPPNAME, AVG(WHOLESALE)
FROM SUPPLIES
GROUP BY SUPPNAME
```

This query becomes

```
SELECT SUPPNAME, AVG(WHOLESALE)
FROM SUPPLIES
GROUPED BY SUPPNAME
```

or

```
SELECT SUPPNAME, AVG(WHOLESALE)
FROM SUPPLIES
GROUPING BY SUPPNAME.
```

The GROUPED BY seems to read better than GROUPING BY, especially if a HAVING clause were to follow.

LANGUAGE USE

The experiment used a population in which about half the subjects were experienced and half the subjects were inexperienced. Given this population mix, TABLET should be used if the application requires writing difficult queries and enough learning time is available. SQL should be used if easy queries are to be written and only a short time is available for learning the language.

Table 1 implies that TABLET is the language of choice for either experienced or inexperienced users writing difficult queries. SQL is the language of choice for either experienced or inexperienced users writing easy queries. If the hard problems are primarily set problems, then SQL is the language of choice.

SUMMARY

A human factors experiment testing the ability of subjects to write queries in two different query languages has been run. One purpose of the experiment was to test the hypothesis: People more often write difficult queries correctly using a procedural query language than they do using a non-procedural query language. Another purpose of the experiment was to compare corresponding constructs in the two languages and recommend improvements for constructs that were found difficult.

The two languages chosen were SQL and TABLET. Both languages use the relational model of data and are relationally complete. They have similar Halstead levels. They also use the same built-in functions. The languages differ in their procedurality -- SQL is non-procedural and TABLET is procedural.

The subjects of the experiment were 72 undergraduates at the University of Massachusetts taking an accounting course. The subjects were divided into two groups -- 35 learning SQL and 37 learning TABLET. Subjects in each group were classified as inexperienced (having no previous computer experience) and experienced (having had a course in BASIC or FORTRAN).

The languages were taught using language manuals that presented the same examples and problems in the same order for each language. There were 14 class meetings in the course, each of which was devoted to answering questions and quizzing the students. At the end of the course a final was given. Three weeks later a retention test was given. Both tests required students to write queries in the language they learned.

The results of the retention test are the main results of the experiment. These query languages were designed as an aid to the casual user, a user who uses the language intermittently. The delay between the end of the course and the retention test models this intermittence. It was also felt that the delay would bring out differences in the retention of the languages.

The results of the retention test supported the hypothesis: Subjects using the procedural language (TABLET) wrote significantly more correct responses to difficult questions than were written by subjects using the non-procedural language (SQL). Comparing the results for corresponding constructs in the language suggested several changes to those constructs:

1. Changing the method of joining tables used in SQL.
2. Using the set operations UNION, INTERSECT and MINUS in TABLET.
3. Rephrasing the GROUP BY clause in SQL.

REFERENCES

BOYCE, R.F.; CHAMBERLIN, D.D.; KING, W.F. and HAMMER, M.M. Specifying queries as relational expressions: The SQUARE data sublanguage. CACM, Nov. 1975. p. 621-628.

CHAMBERLIN, D.D.; ASTRAHAN, M.M.; ESWARAN, K.P.; GRIFFITHS, P.P.; LORIE, R.A.; MEHL, J.W.; REISNER, P. and WADE, B.W. SEQUEL 2: A unified approach to data definition, manipulation, and control. IBM Journal of Research and Development. Vol. 20, No. 6, Nov. 1976. p. 560-575.

CHAMBERLIN, D.D. and BOYCE, R.F. SEQUEL: A structured English query language. Proc. 1974 ACM SIGFIDET Workshop, Ann Arbor, Mich., April, 1974. p. 249-264.

CODD, E.F. A relational model of data for large, shared data banks. CACM, Vol. 13, June 1970. p. 377-397.

CODD, E.F. A database sublanguage founded on the relational calculus. Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access and Control, San Diego, CA., 1971a. p.35-68.

CODD, E.F. Relational completeness of database sublanguages. Courant Computer Science Symposia, Vol. 6: Data Base Systems, Prentice-Hall, NY, 1971b. p. 65-98.

DENNY, G.H. An introduction to SQL, a structured query language. IBM Research, RA 93, San Jose, CA, May, 1977.

GANNON, J.D. and HORNING, J.J. The impact of language design on the production of reliable software. IEEE Trans. on Reliable Software, Vol. 1, No. 2, 1975. p.10-22.

GOULD, J.D. Some Psychological evidence on how people debug computer programs. Int'l. Journal of Man-Machine Studies, 1975, 7. p. 151-182.

GOULD, J.D. and ASCHER, R. Use of an IQF-like query language by non-programmers. RC 5279, IBM Watson Research Center, Yorktown Heights, NY 10598. 1975.

HALSTEAD, M.H. Elements of Software Science. Elsevier North-Holland, Inc. NY, 1977.

IBM Interactive Query Facility User's Guide. GH-1223, 1972.

LOCHOVSKY, F.H. Data base management system user performance. Technical Report CSRG-90, Computer Systems Research Group, Univ. of Toronto, April 1978.

MILLER, L.A. Programming by non-programmers. Int'l. Journal of Man-Machine Studies, 6, 1974. p.273-260.

MILLER, L.A. and BECKER, C.A. Programming in natural english. RC 5137, Watson Research Center, Yorktown Heights, NY 10598, Nov. 15, 1974.

REISNER, P. Use of psychological experimentation an an aid to development of a query language. RJ 1707, IBM Research, San Jose, CA 95193, Jan. 13, 1976.

REISNER, P.; BOYCE, R.F. and CHAMBERLIN, D.D. Human factors evaluation of two data base query languages - SQUARE and SEQUEL. NCC, 1975. p. 447-452.

SHNEIDERMAN, B. Improving the human factors aspect of database interactions. ACM TODS, Vol. 3, No. 4, Dec. 1978. p. 417-439.

SHNEIDERMAN, B.; MAYER, R.; MCKAY, D. and HELLER, P. Experimental investigations of the utility of flowcharts in programming. T.R. No. 36, August, 1975, Computer Science Department, Indiana Univ., Bloomington, Indiana 47401.

SIME, M.E.; GREEN, T.R.G. and GUEST, D.J. Psychological evaluation of two conditional constructions used in computer languages. Int'l. Journal of Man-Machine Studies 5. p. 105-113. 1973.

STEMPLE, D.W.; BECKER, M.; WELTY, C. and MAYFIELD, W. TABLET: The algebra based language for enquiring of tables. T.R. 78-19, Computer and Information Science Dept., Univ. of Mass., Amherst, MA, Nov. 1978.

THOMAS, J.C. and GOULD, J.D. A psychological study of query by example. AFIPS Vol. 44, National Computer Conference, 1975. p. 439-445.

WEISSMAN, L. Psychological complexity of computer programs: an experimental methodology. SIGPLAN Notices, June, 1974. p.25-35.

WELTY, C. SQL manual. Univ. Computing Center, University of Massachusetts, Amherst, MA, Nov. 1978a.

WELTY, C. TABLET manual. Univ. Computing Center, University of Massachusetts, Amherst, MA, Nov. 1978b.

WELTY, C. A study of the effects of procedurality on the learnability of two relational query languages. PHD thesis, Computer and Information Science Department, Univ. of Massachusetts, Amherst, MA, 1979.

YOUNGS, E.A. Human errors in programming. Int'l. Journal of Man-Machine Studies, 1974, 6. p. 361-376.

ZLOOF, M.M. Query by example. Proc. 1975 NCC AFIPS, Vol. 44. p.431-438.