Human Factors Studies of Database Query Languages: SQL as a Metric

Charles Welty
University of Southern Maine

Over the years, SQL has been used repeatedly in human factors experiments measuring the ease of use of database query languages. These experiments have studied a wide range of issues from basic language comparisons, through design issues, to attempts to understand the human cognitive processes involved in querying a database. This paper summarizes the issues and the experiments that have involved SQL.

Introduction

Since its appearance in 1974, SQL has become the premier database language. SQL (Chamberlin & Boyce, 1974; Chamberlin et al., 1976; Date, 1989; Date & White, 1989) has also gained early and continued acceptance as the language against which others are measured in human factors experiments.

In general, human factors studies of software systems measure how well a system can be learned and used. Query languages have been widely studied for a variety of reasons. First, they are used by individuals as opposed to, for example, programming languages which are usually used in group projects. Individual use is easier to test than group use. Query languages also require minimum training compared to many other systems. Query language users do not need special backgrounds and, thus, possible experimental subjects are in general supply.

Of course, the motivation for query language studies is much more than the comparative ease of doing the studies. A major reason for the studies is to test the claims of the language designers as to the ease of use of a particular language. The designers and implementers of database systems have a very different background from the proposed user population. The designer’s intuition as to how easy the system is to use is not sufficient. Human factors studies replace intuition by testing whether the query language achieves appropriate levels of usability. Thus, a human factors experiment may result in changes to the language, insights into how the language is best used, or the language’s demise.

Another primary reason for studying query languages has been the perception of who the users are and how the language will be used. The user of a query language is commonly considered to be a “casual user”. Casual use of a query language implies that query generation is not a primary aspect of the user’s job. Use of the system is ancillary to, but in support of, the user’s primary function. The casual user population consists of executives and clerical staff who use the language intermittently on the job. If the language does not match the needs and abilities of the users, it will not be accepted. Other methods will be found to support their need for information. Thus, query languages must be easily learned and used by these target populations.

In addition, it is possible to abstract from a particular language syntax to more basic concepts. These concepts may deal with the basic design of the language, how people perceive and use the language, or how the language is learned. Many researchers are interested in how these and other languages match the way human beings acquire and use knowledge, that is how well the languages match our cognitive processes.

This paper surveys a variety of experiments that
have addressed a wide range of human factors issues related to database query languages. Other papers that summarize previous experiments and discuss general issues are Ogden, 1988; Reisner, 1977, 1988; Shneiderman, 1978; Thomas, 1977.

Human Factors Experiments

Human factors experiments are based on the established field of controlled psychological experimentation. Every attempt is made to isolate the phenomenon being studied from outside influences. Usually this means finding a homogeneous group of subjects and randomly dividing them into groups that will then receive different treatments. The treatments differ only in the elements to be studied. These are the independent variables. In the case of query language experiments, the treatments consist of learning and using different query languages. Various measurements are made to determine the learnability and ease of use of each language. These measurements are the dependent variables. Then the measurements are statistically analyzed to determine if the results have statistical significance or if the differences could be due to random variation.

There are many experimental designs based on the above methodology. Most of the studies we will pursue investigate two different query languages, one of which is SQL. Thus, there are two groups of subjects. One group learns SQL and the other learns the other language. They are then tested and the test results statistically analyzed. This is called a between group design. A problem with this method is that one group may have more ability to learn the query language, even though they are randomly selected. This problem is compensated for in several of the experiments by having each subject learn both languages. Since the order in which the languages are learned may well have an effect on the outcome, subjects in each group learn the languages in a different order. This is called a counterbalanced within subjects design. It is within subject because each subject learns each language. The counterbalancing is due to the two orderings.

It is not always easy to control for outside influences. For example, the time of day at which each group meets may influence the experiment. If the subjects are college students and one group meets when an honors class is in session, no honors students may be in that group. If the experiment is on-line and uses a time-sharing system, the system may have poor response time for one group compared to the other. Obviously, the experimenter must attempt to eliminate these influences. The simplest way is to select subjects who are available at both times and randomly assign them to groups.

If the experiment is well conceived and executed the results can be replicated and generalized. Replication of the experiment, based on documentation from the original experiment, should yield the same result. Generalization means that the result can be applied to a population other than the individuals in the original experiment.

Many of the early experiments were paper-and-pencil studies. The systems were as yet unimplemented and were simulated by subjects writing queries and experimenters grading them. This method eliminated differences in typing ability and fear of technology from the experiment. But they did not simulate the real system, especially if the languages differed mainly in the types of terminal interactions required. Later experiments used working systems. In the following two sections each experiment will be noted as either a paper-and-pencil study or on-line.

There are a variety of performance measures that have been interesting to experimenters. These include: number of correct queries generated; time to learn the language; time to write queries; level of confidence that a query is correct; and ability to write queries after not using the language for some period of time. The following sections will present a number of experiments using a variety of measures.

SQL versus Formal Languages

SQL versus SQUARE. Reisner (1977) ran the first experiment comparing SQL to SQUARE. SQUARE (Specifying Queries As Relational Expressions) used a notation based on relational algebra (Codd, 1971) to specify queries. The experiment used four groups, two groups used SQL and two used SQUARE. For each language, one group consisted of programmers and the other nonprogrammers. The numbers of subjects in each group were: 18 SQL programmers, 15 SQL nonprogrammers, 11 SQUARE programmers, and 20 SQUARE nonprogrammers. This was a between groups, paper-and-pencil study.

The subjects required 10-12 hours of instruction in their language. After the learning phase (which contained some quizzes) two tests were given. Both consisted of English questions to be translated into the appropriate language. One test was given directly after the learning phase. The other test was given after a week in which the language was not used. The purpose of the latter test was to check language retention, an important aspect of languages designed for casual use.

The results showed that programmers could use either language with minimal (certainly not statistically significant) differences. Programmers significantly outperformed nonprogrammers overall (programmers - 78% correct, nonprogrammers - 60% correct). The most
Related Content

A Distributed System for The Management of Fine-grained Provenance
[www.igi-global.com/article/a-distributed-system-for-the-management-of-fine-grained-provenance/142071?camid=4v1a](www.igi-global.com/article/a-distributed-system-for-the-management-of-fine-grained-provenance/142071?camid=4v1a)

Shape Matching for Foliage Database Retrieval
Haibin Ling and David W. Jacobs (2009). *Semantic Mining Technologies for Multimedia Databases* (pp. 100-129).
[www.igi-global.com/chapter/shape-matching-foliage-database-retrieval/28830?camid=4v1a](www.igi-global.com/chapter/shape-matching-foliage-database-retrieval/28830?camid=4v1a)

A Tool for Fuzzy Reasoning and Querying
[www.igi-global.com/chapter/tool-fuzzy-reasoning-querying/20361?camid=4v1a](www.igi-global.com/chapter/tool-fuzzy-reasoning-querying/20361?camid=4v1a)

Modeling of Business Rules for Active Database Application Specification
Youssef Amghar, Madjid Meziane and Andre Flory (2002). *Advanced Topics in Database Research, Volume 1* (pp. 135-156).
[www.igi-global.com/chapter/modeling-business-rules-active-database/4326?camid=4v1a](www.igi-global.com/chapter/modeling-business-rules-active-database/4326?camid=4v1a)