An Analysis of Semantic Overload in Database Access Systems Using Multi-table Query Formulation

Sangkyu Rho
Seoul National University
Salvatore T. March
University of Minnesota

Theory suggests that semantic overload in database representations and query languages may be a significant contributing factor to difficulties in end-user access to corporate databases. With respect to join queries, two aspects of SQL-based relational database management systems are overloaded: (1) foreign key columns are overloaded to represent both attributes and relationships and (2) the WHERE clause in SQL is overloaded to include both selection criteria and join criteria. We experimentally test the semantic overload theory by comparing end user performance using data access systems (DASs) that address these aspects individually and in combination. Eliminating the overload in foreign key columns by adding an explicit relationship construct to the standard tabular relational database representation significantly improved end user performance. Eliminating the overload in the SQL WHERE clause by adding an explicit join construct, similar to that included in SQL2, reduced certain types of errors but did not significantly improve performance. We conclude that for organizations seeking to improve end user performance in multi-table database queries it is more effective simply to use an explicit relationship representation in documenting the database structure than to invest in a database query language, such as SQL2, that has an explicit join syntax.

End user computing (EUC) is a vital part of organizational computing systems (Niederman et al., 1991; Caudle et al., 1991; Clark, 1992). EUC allows IS departments to off-load some of their work to end users, especially in the area of management reporting and ad-hoc information retrieval (Clark, 1992; Lehman and Wetherbe, 1989). Furthermore, it allows end users, who are facing ever-changing business environments, to meet their information needs in a timely manner. Relational database management systems and their high level query languages (QLs) such as SQL and QBE have played an important role in enabling EUC (Lehman and Wetherbe, 1989; Podhorn and Pikner, 1991).

With the emergence of client/server technology and distributed database systems, end users can have access to many organizational databases (Dewire, 1993, Madnick, 1991). In such an environment, it is crucial to have a database access system (DAS) that enables the best possible performance in obtaining data from the database. We use the term DAS to emphasize the fact that effectively accessing a database involves two formal systems: a database representation describing the structure of the database and a query language through which retrieval and manipulation operations on the data are expressed.

Standard relational DASs use tabular database representations and query languages like SQL. Although relatively easy to use, they may not be easy enough for casual end users (Leitheiser and March, 1996; Podhorn and Pikner, 1991). Users may not adopt a DAS in their work because learning and using it are too difficult and time-consuming (Gerlach and Kuo, 1991; Davis, et. al., 1992). Worse yet, users may unknowingly write incorrect queries due to lack of understanding of the database structure or the query language. Inaccurate information derived from such queries may lead to inappropriate business decisions (Davis, 1989; Podhorn and Pikner, 1991). The use of a DAS could actually be counterproductive.
When user information requirements can be specified in advance, views or even pre-compiled queries can be created to simplify the querying task. When information requirements are truly ad hoc, the difficulties of effectively using a DAS must be addressed.

Prior studies of the usability of DASs have compared one DAS with another (e.g., Reisner et al., 1975; Yen and Scamell, 1993, Suh and Jenkins, 1992). They have focused on evaluating particular DAS implementations. Although it is important to know whether one DAS is better than another, it is more important to understand why one DAS is better than another, for what kind of people, and under what conditions (Reisner, 1988). That is, we must theorize and justify theories about the underlying constructs that make a DAS effective (March and Smith, 1995). These provide principles by which to improve the usability of DASs. The purpose of this paper is to test one such theory, semantic overload.

Norman (1986) and Weber and Zhang (1991) present theory suggesting that semantic overload may explain the difficulties observed when end users attempt to formulate multi-table join queries. Semantic overload occurs when a single construct has multiple purposes. In tabular relational database representations, foreign key columns are overloaded to represent both attributes (descriptors of entities in the real world) and relationships (associations between entities). In SQL (ANSI, 1986), the most commonly used database query language, the WHERE clause is overloaded to specify both selection and join conditions (QBE, another popular QL, has the same type of a semantic overload).

In our laboratory study, we eliminate the semantic overload in foreign key columns by explicitly representing relationships. We eliminate the overload in the SQL WHERE clause for multi-table join queries by adding an explicit JOINED BY clause. Theory suggests that these should individually, and in combination, improve end user performance. SQL2 (Date and Darwen, 1993), the current language standard, has a similar language construct for the join operation (however, the FROM clause is overloaded to specify both the tables in the join and the join conditions).

Our study is not directed at specific language comparisons per se (e.g., SQL versus SQL2), but addresses the general issue of explicit relationship representations and explicit join syntax. We focus on underlying constructs to test the theory of semantic overload. The principles learned from this study should apply to any DAS.

We study the multi-table join operation because it is one of the most basic data retrieval operations in relational databases, yet it is difficult to learn and to apply properly (Chan et al., 1991, 1993; Welty and Stemple, 1981). Furthermore, it is a prototypical operation to enable a test of the semantic overload theory. Other relational operations such as select, project, union, subtract, and intersect could also be studied.

Our findings suggest that an explicit join construct is not as effective as explicit relationship representations. Eliminating the overload in foreign key columns by adding an explicit relationship construct significantly improved end user performance. Eliminating the overload in the SQL WHERE clause by adding an explicit join construct reduced certain types of errors but did not significantly improve performance. Thus we conclude that for organizations seeking to improve end-user performance in multi-table database queries it is more effective simply to use an explicit relationship representation in documenting the database structure than to invest in a database query language, such as SQL2, that has an explicit join syntax. User organizations that expect to obtain significant improvements in query performance by acquiring a DBMS with an explicit join construct (such as Access 2 (Microsoft, 1994)) to replace one without an explicit join construct (such as FoxPro (Microsoft, 1993)) will be disappointed.

The remainder of this paper is organized as follows. In the next section we describe prior studies on the usability of DASs. In the following section we present our research framework. We then describe the experiment and present our findings and their implications. In the final section, we summarize our results and present directions for future research.

## Significant Prior Research

Several studies have addressed the effects of database representations and query languages on end user performance in formulating queries. Lochovsky and Tsichritzis (1977) compared three traditional database representations: hierarchical, network, and relational using, respectively, the IMS language DL/1, the DBTG COBOL DML, and Codd’s ALPHA as query languages. For less experienced users the relational representation and ALPHA resulted in significantly better performance than any others. They conclude that a relational representation and language result in improved query performance. They could not, however, differentiate the effects of the database representation from those of the language, nor did they explain why performance was improved.

Jih et al. (1989) compared end user performance in writing SQL queries using a tabular relational representation versus an entity-relationship representation. There were no significant differences in the semantic accuracy of the SQL queries. Subjects using the tabular representation made fewer syntactic errors but took more time to complete the queries. The authors conclude that neither representation is clearly superior. This result could be attributed to a “ceiling effect.” The database and queries used in these experiments were extremely simple. The most complex queries required joining two tables from a three table database. Furthermore, SQL is based on the relational model. Users of the entity relationship representation needed to transform that representation into a
Related Content

Evaluation of MDE Tools from a Metamodeling Perspective
[www.igi-global.com/article/evaluation-mde-tools-metamodeling-perspective/3393?camid=4v1a](www.igi-global.com/article/evaluation-mde-tools-metamodeling-perspective/3393?camid=4v1a)

Integrating Mobile-Based Systems with Healthcare Databases
[www.igi-global.com/chapter/integrating-mobile-based-systems-healthcare/7928?camid=4v1a](www.igi-global.com/chapter/integrating-mobile-based-systems-healthcare/7928?camid=4v1a)

Indexing Textual Information
[www.igi-global.com/chapter/indexing-textual-information/7911?camid=4v1a](www.igi-global.com/chapter/indexing-textual-information/7911?camid=4v1a)

Framework for a Geographic Districting DSS using an Intelligent Object-oriented Model
[www.igi-global.com/article/framework-geographic-districting-dss-using/51180?camid=4v1a](www.igi-global.com/article/framework-geographic-districting-dss-using/51180?camid=4v1a)