An Experimental Study of Object-Oriented Query Language and Relational Query Language for Novice Users

Chun-Zhi Wu, Hock-Chuan Chan, Hock-Hai Teo, and Kwok-Kee Wei
National University of Singapore

This study compares object-oriented query language (OQL) and relational query language (SQL) for novice users. The comparison focuses on the effects of different query languages on user performance for query writing task and query reading task in terms of time, confidence and accuracy. To accomplish this objective, a laboratory experiment was conducted. The results indicate that OQL subjects performed significantly better than SQL subjects for query writing in terms of time and accuracy and for query reading in terms of time, confidence and accuracy.

Object-oriented database management systems (OODBMS) have been suggested as the next generation of database technology (Committee for Advanced DBMS Functions, 1990; Hsiao, 1992; Kim, 1990b). Since the 1980’s, both industrial and research laboratories have developed products and prototypes of OODBMS such as EXODUS (Carey et al., 1988), Gemstone (Butterworth et al., 1991), IRIS (Fishman et al., 1988), Orion (Kim, 1989; Kim, 1990a), O2 (Lecluse, 1988; Bancilhon et al., 1989), Ontos (Ontologic, 1989), and POSTGRES (Stonebraker and Kemnitz, 1991). OODBMS are developed to meet the requirements of new applications such as CAD/CAM, CASE, AI, image and voice processing and other complex tasks that exceed the capabilities of conventional (relational) DBMS (Banerjee et al., 1987; Cattell, 1991). An important question is whether these OODBMS are useful for commercial data processing. It remains an open question. Like other DBMS, OODBMS also need to provide a comprehensive set of capabilities. Query language which caters especially for end users to retrieve information is perhaps one of the most important and necessary facilities that should be provided.

Being an essential tool to the users, object-oriented query language (OQL) is an integral part of OODBMS. To gain wide acceptance by current database users, OODBMS have to rise to the challenge to support a more powerful and easy-to-use OQL. In the commercial world, relational DBMS have now become the de facto standard for data processing application (Hughes, 1991). Indeed, the popularity of relational DBMS mainly stems from the strength of SQL as a universal query language, which saves the programmers from writing tedious navigational paths and algorithms for efficient retrieval. It is easy to use and widely accepted by the users.

The earlier generations of OODBMS (Copeland and Maier, 1984) did not provide any special support for queries. However, later OODBMS such as EXODUS, O2, Orion and POSTGRES have all begun to provide SQL-like OQL. SQL-like OQL have not yet been much explored since they are relatively new. In addition, there is still no standard OQL for OODBMS to-date. Despite the lack of standards, there do exist several prototype OQL which share common functionalities and similar expressions. Orion’s OQL is one such example which we will use as the basis for our study.

To-date, there is a dearth of literature that can provide evidence to substantiate the effectiveness of OODBMS and OQL in terms of user performance. Several human factors researchers have focused on studies which compare data modelling and query language capabilities on different data models. Bock and Ryan (1989) compares success in data modelling on the extended ER model and OO data model. There are few empirical studies which compare the effectiveness of query languages of OO data model and other data models. This paper attempts to explore this gap. Specifically, we would like to determine if OQL are better than relational SQL in terms of user performance which is measured by time, confidence, and accuracy.
Literature Review

Review of OQL and SQL

Carey et al. (1988), Bancilhon et al. (1989), Kim (1989), Alashqur et al. (1989), and Bertino et al. (1992) provide a comprehensive comparison between OQL and SQL in theory. The differences between OQL and SQL lie in the different expressiveness of the abstractions in the two data models and in the capability of the query language for capturing and enriching these abstractions with operators. Below, we show some main differences that affect the usability from the viewpoint of users. The analytical comparison favors OQL. Hence, we would expect OQL to be better than SQL. This is the basis for our hypotheses stated in the section, Research Hypotheses.

Abstractions

Table 1 illustrates the difference of OO and relational data models. A relational model uses classification as the only type of abstraction to model the real world. It simply presents the real world as a group of flat structure relations. Associations in the real world are represented by embedded foreign keys. The object-oriented data model is much richer since it adopts three types of abstractions: classification, generalization and aggregation abstractions. The classification abstraction is used for defining one concept as a class of real world objects characterized by common properties (Batini et al., 1991). An aggregation defines a new class from a set of (other) classes that represent its component parts. A subset relationship between the elements of two (or more) classes are defined through the generalization abstraction. Based on these abstractions, OQL can capture these semantic relationships more directly and therefore enhance the functionalities of OQL with additional operators such as path expression, class restriction operator and quantifiers (all, one) placed before the multi-valued attributes.

Path Expression/Navigation

A path expression is a sequence of nested attributes linking objects to objects. Due to the navigational access of path expression, OQL is considered as both associative (value-based) and navigational (link-based). However, SQL is associative and declarative. Stonebraker (1990) argued that navigational access is a major step backward in OQL because SQL made the navigational details transparent to the users. On the other hand, SQL demands logical navigation through joins. Users would not find it difficult to understand the links in path expression for three reasons: (1) There is graphic representations (links) in the OO data model; (2) Navigational access in OQL is on the conceptual level (attribute-domain) as opposed to the physical level of the CODASYL and Hierarchical models; and (3) It obviates the use of explicit joins.

Generalization

Besides aggregation, generalization abstraction simplifies the information retrieval about the relationships among the groupings. For instance, students are specialized persons. In relational data model, we use two relations Student and Person to represent them. To avoid data redundancy, we record specific attributes of a student as well as the identification number (a person) for the relation Student. Whenever one wants to know the basic information (as being a person) about a student, one has to do a join on the common key (identification number) between the two relations.

In contrast, the OO data model allows one to directly retrieve information in the specialized class without any join operation and without referring to the generalized class. In addition, OQL allows users to specify a subset of classes in the generalization class hierarchy so that users can easily restrict their queries to a certain range.

Quantifier

SQL has the existential quantifier EXISTS, The universal quantifier is expressed by the negated form of EXISTS (NOT EXISTS). Therefore, whenever one needs to do this kind of retrieval, the query has to be paraphrased (Date, 1986). It is not a natural way of thinking. OQL avoid such a problem by using the universal quantifier ALL and existential quantifier ONE directly before the attribute since the attribute in OO data model can be set-valued.

Artificial Key

It is unnecessary for OO data model to maintain the artificial primary key for each class. The identity of each object is automatically maintained by the system and users do not have to be aware of it. In contrast, each relation in relational data model must have a key. In some cases, it seems awkward to maintain various identification number attributes in relations. For instance, when one refers to persons, one normally uses attributes such as name, age and sex, and seldom uses identification numbers. OO data model alleviates this problem and results in users’ better understanding of the data model which helps query formulation.

It seems evident that OQL is more powerful and natural than SQL. However, the analyses above are purely theoretical. Empirical evidence is necessary. As the notion of