Data Clustering for Effective Mapping of Object Models to Relational Models

Narasimha Bolloju
City University of Hong Kong

Kranti Toraskar
City University of Hong Kong

Today the object-oriented model is increasingly used during the analysis and design stages of information systems development, while relational database management systems (RDBMS) are still the most popular implementation tools. Consequently, in practice it is becoming increasingly common to map the object model to an appropriate relational model. This mapping often results in excessively fragmented tables, and denormalization is a commonly used approach for improving the system performance in such cases. However, denormalization affects the flexibility, integrity and data accessibility of implementation, while reducing correspondence between the implementation and the original object model. Based on a particular type of physical data organization, called data clustering, this paper presents an approach to avoid or minimize the need for denormalization. We first examine the use of denormalization and discuss the associated problems in the context of mapping object models to relational models. Next, we present the concept of data clustering and its effect on the performance and storage requirements. Finally, we describe and illustrate how data clustering can be employed to avoid denormalization and to achieve a greater degree of correspondence between an object model and its relational implementation. We also discuss the various trade-offs involved in the use of data clustering.

Currently, an increasing number of information systems (IS) applications are being developed using object-oriented concepts and techniques, while the relational data model and relational database management systems (RDBMS) are still the most popular implementation vehicles (Blaha et al., 1994). Object-oriented modeling is especially attractive during the early stages of system development due to its unique power to capture the structural semantics of information systems (Markowitz and Makowsky, 1990). As a result, in practice it is becoming increasingly necessary to map the object model to a relational model during the implementation phase of the IS development cycle.

In general, the mapping between object model and relational model is regarded as being many-to-many (Premerlani and Blaha, 1994). Thus, in practice it is always possible to identify more than one, in fact several, relational models that correspond to a given object model. In such a situation, the selection of a particular relational model entails making a trade-off between two conflicting goals: performance maximization and conformity to the object model. On the one hand, the relational model should allow the maximum level of operational performance to be achieved. On the other hand, the relational model should conform to the object model as much as possible. Object conformity is important for a smooth development and maintenance of the IS application, but, at the same time, it also means a lower performance level, as explained next.

In the case of totally normalized implementations, poor system performance is mainly caused by the fragmentation of real world objects. In general, the joining of the fragments of interest is expensive, and it affects performance because of the
additional computations and disk accesses required. These adverse effects will be particularly visible in transactions that use complex objects. Westland (1992), referring to anecdotal evidence, notes that normalization-induced fragmentation which causes significant inefficiencies is related to the cost of join operations. Thus, in the case of normalized implementations, direct mapping essentially guarantees object conformity, but at the cost of lower performance.

In order to solve this problem, we propose making physical database changes because they do not affect the logical schema; therefore, the logical database independence of the application is preserved. This can be better understood with reference to the external, conceptual, and internal schemas specified in the ANSI/SPARC 3-schema architecture (Tsichritzis and Klug, 1978). In terms of these schemas, logical database independence refers to “the capacity to change the conceptual schema without having to change external schemas or application programs” (Elmasri and Navathe, 1994, p. 28). Physical database changes affect only the mapping between internal schema and conceptual schema, and therefore have no effect on conceptual and external schemas which determine the logical data independence. Date (1990, p. 40) mentions the strategy of restricting any database design changes below the conceptual level as a way of preserving logical data independence.

Some of the common approaches to changing the physical data organization include creation of additional indexes and redistribution of storage of tables on different disk-drives. In practice, when such changes in physical data organization fail to provide the required performance levels, database professionals resort to what is known as denormalization to meet those performance requirements (Elmasri and Navathe, 1994, p. 469). Introducing redundant or duplicate attributes, combining two or more tables, and introducing repeating attributes are some common examples of denormalization techniques (Edwards, 1990; Elmasri and Navathe, 1994). Invariably, this approach causes a change in the logical database design, and can, therefore, significantly affect the application programs that manipulate data pertaining to the changed structures. Consequently, denormalization should be considered as a desirable option only if the normalized relations (i.e., at least in the third normal form) are unable to deliver the required performance, and only after considering all possible physical database organizations. In this paper, we consider the physical organization of data clustering to avoid denormalization.

Several problems associated with denormalization provide the motivation for our approach. Although denormalization improves performance, it compromises the flexibility, integrity and accessibility of data (Edwards, 1990). For example, the flexibility of making certain database design changes is compromised when repeating attributes are used in a table and one wants to change the size of that repeating attribute. Similarly, combining two or more tables results in integrity problems when the resultant table is not in at least the third normal form. Integrity problems can also surface with the introduction of redundant or duplicate attributes since the application programs need to handle the likely update anomalies. Finally, repeating an attribute in a table complicates database accessibility for any queries involving that particular attribute, for example, queries involving a join operation. We illustrate these problems later in the next section using an example case.

In view of the above discussion, approaches that do not affect the logical database independence and the closeness to the object model, and at the same time improve the overall performance will be of interest to database designers and administrators for the following reasons. First, in the context of this paper, such approaches avoid the problems associated with denormalization. Second, and more generally, such approaches will also minimize the complications observed by Premerlani and Blaha (1994) in the context of reverse engineering of relational databases. According to them, these complications arise out of the wide range of styles and unusual implementation constructs used for performance improvement. By minimizing the need for unusual styles and constructs, approaches such as ours will have better opportunities for future migration to an object-oriented database management system (OODBMS) or to an RDBMS enhanced with object-oriented features.

A number of researchers and practitioners (e.g., Johnson and Fotouhi, 1995; Edwards, 1990; Gorla and Quinn, 1991) have advocated data clustering as an effective approach for performance improvements in relational databases. In this paper we present an application of data clustering from a somewhat different perspective, namely the perspective of mapping an object model to relational model. We argue that the data clustering is an effective approach to avoid or minimize the need for denormalization that is often carried out in response to the excessive fragmentation in relational models. The role of denormalization in mapping an object model to a relational model is examined in the next section. The third section presents the concept of data clustering and its effect on performance and storage requirements. In the fourth section we illustrate how data clustering can be used to avoid denormalization and still bring in desired performance improvements, in general leading to a more effective implementation of the object model. We also discuss the various trade-offs involved in the use of data clustering.

**Mapping an Object Model to Relational Model with Denormalization**

As discussed above, denormalization is the process of deliberately violating normalization principles of the relational data model in order to improve the database performance when processing certain important queries or transactions. Introducing redundant or duplicate attributes, combining two or more tables and introducing repeating attributes are