A Unified Approach to the Design and Generation of Complex Database Schemata

J. Peckham
University of Rhode Island

B. MacKellar*
Western Connecticut State University

J. Vorbach
St. John’s University

SORAC (Semantic Objects, Relationships, and Constraints) is a unified approach and environment for the design and automated implementation of active database schema that utilizes object-oriented and semantic techniques. It has been applied to architectural design, real-time, and program stock trading applications, but also represents a general approach that can be used to support other complex applications. This paper reviews and consolidates previously reported pieces of this work, in particular concentrating on the architectural design and real-time applications. Characteristics of these two areas as they relate to active database systems are first reviewed. Then, we describe SORAC and a vision for advanced database systems construction, and describe ongoing research in the SORAC environment. Lessons learned from our experimentation are outlined.

Overview

Database management systems, a mainstay of the business-oriented data processing world are rapidly making inroads into such “advanced” domains as computer-aided design, computer-aided manufacturing, real time systems, and networking. This is driven partially because of the increased capabilities of database management systems, and partially because increased hardware and network capabilities have led to greater expectations for advanced applications. For example, CAD systems for years have used files to store data. Such files often used proprietary formats and were difficult to share among many users. With the growth of networking, companies expect that CAD data should be easily accessible and sharable by multiple users running multiple applications. These are exactly the capabilities that database management systems are designed to support.

With new applications come new demands, however, one of the characteristics of all the application areas mentioned above is increased behavioral complexity. As a result, many companies working in these areas are turning to object-oriented database management systems which support behavior as part of the schema. Active databases which use rules to support the more complex consistency requirements found in these systems are also becoming more popular. Designing a schema for an active database is more difficult because behavior and consistency enforcement, previously part of user applications, now become part of the schema itself. Schema design tools which can support the modeling and verification of behavior and consistency are needed. An example of this type of tool is SORAC, a schema design tool being developed at the University of Rhode Island. This system supports the development of active, object-oriented database schemas. Aspects of this system have been reported in (Doherty et al., 1993; Dong, 1992; Peckham et al., 1996; Qian, 1994; Vora, 1992). In this paper, we bring together the previously reported pieces of this work and in particular show how the same schema design paradigm can be used in two very different

*Partially supported by a CSU-AAUP grant
areas: computer-aided design and real-time systems. We discuss the characteristics of the two application areas, modifications to the SORAC paradigm that had to be made to support each area and the lessons learned from using SORAC in each domain.

**What Is an Active Database?**

Many advanced databases have complex consistency requirements that involve the passage of time, the occurrence of events, or the existence of particular relationships among the data. In the past, the responsibility for maintaining consistency has usually fallen upon the application code that accesses the database. This was acceptable as long as the consistency semantics were relatively simple and access to the data was strictly through well-controlled programs written by database specialists. It works less well as consistency requirements become more complex, the data is more widely accessible, and is used in a greater number of applications.

By adding rule processing capabilities to a database management system, semantics can be expressed declaratively at schema design time and enforced on a uniform basis, no matter which user application is accessing the database. This is the central idea behind active database systems. Examples of active database systems include Postgres, HiPac, Cactis, and Starburst (Jaeger et al., 1995). Rule processing is usually based on the notion of an event-condition-action rule (ECA rule). In this paradigm, an event occurs which triggers the checking of a condition; if the condition evaluates to true, the action is fired. Events can be particular database updates, or the passage of a particular time unit, or the occurrence of a particular action in the application. In SORAC, such rules are called update rules and are associated with relationships that occur between objects in the database. An example might be an update rule associated with the part relationship that can exist between a student object and a course object in a university database. Such a rule might specify that if the student object is deleted (the event) and there are no other students enrolled in the course (the condition), then the course object should be deleted (the action).

**Active CAD Databases**

The active database paradigm can be extremely useful in the domain of computer-aided design (Lockemann et al., 1995; MacKellar et al., 1992). CAD applications are typified by the following characteristics:

- **Structural complexity**
- **Nonstandard relationship types**
- **Multiple representations**

Structural complexity occurs because a typical CAD object is made up of many subobjects. For example, consider a design for a wall containing a door assembly. This door assembly is made up of a frame, the door itself, connecting hinges, and a latching mechanism. The wall contains a hole, which in turn contains the door assembly. Much of the meaning of this object is based on relationships between these component objects. To evaluate whether a door swings into or out of a room (which is often important in code conformance applications), the relationships between the wall and its containing room, the door frame and the wall, and the connection between the door, hinges, and frame must be considered. Behavior is often determined by the kind of relationship as well. If the hole in the wall that contains the door assembly is deleted, the door assembly must be moved; it is physically impossible for it to be contained by the wall at this point. On the other hand, if the connection between a door and its hinges is deleted, no deletions or movement of objects should be required. ECA rules which support such behavior can be associated with each distinct kind of relationship.

Another important characteristic of CAD systems is the need for multiple representations and for active mappings between representations. Large design projects, such as buildings, are normally completed by a number of groups of designers each working on a specialized subsystem of the design. ECA rules can be used to specify propagations between representations and to notify designers working in a particular representation when changes that affect them have occurred. For example, if a structural designer moves a support wall, designers working on the heating systems must be notified. Thus, ECA rules in a CAD system tend to be based on both simple database events such as insertions and deletions of objects and relationships, and on application specific events such as changes in position, or changes in attributes.

**Active Real-Time Databases**

Real-time databases (Ramamritham, 1993) were developed to meet the needs of applications in which temporal constraints are placed upon data and operations to provide up-to-date and timely database functionality. Real-time computing environments require the addition of another dimension to constraint specification and consistency maintenance. That is, real-time databases must be temporally consistent as well as logically consistent. Two types of constraints that are of use in these environments include the following (Prichard et al., 1994):

- **Data Freshness**: Data is typically recorded from incoming sensor information and then quickly accessed. Constraints expressing the freshness of individual data items and the relative freshness of sets of data are important.
- **Operation and Transaction Deadlines**: Operations on the data may have deadlines.
Related Content

Data Clustering for Effective Mapping of Object Models to Relational Models
www.igi-global.com/article/data-clustering-effective-mapping-object/51185?camid=4v1a

A Novel Approach to Distributed Rule Matching and Multiple Firing Based on MapReduce
www.igi-global.com/article/a-novel-approach-to-distributed-rule-matching-and-multiple-firing-based-on-mapreduce/211915?camid=4v1a

Introduction and Trends to Fuzzy Logic and Fuzzy Databases
José Galindo (2008). Handbook of Research on Fuzzy Information Processing in Databases (pp. 1-33).
www.igi-global.com/chapter/introduction-trends-fuzzy-logic-fuzzy/20347?camid=4v1a

Incorporating Spatial Queries into Semantic Sensor Streams on the Internet of Things
www.igi-global.com/article/incorporating-spatial-queries-into-semantic-sensor-streams-on-the-internet-of-things/194998?camid=4v1a