Triggers, Rules and Constraints in Databases

Juan M. Ale
Universidad de Buenos Aires, Argentina

Mauricio Minuto Espil
Universidad Católica Argentina, Argentina

INTRODUCTION

Databases are essentially large repositories of data. Since the mid-1980s up to the mid-1990s, considerable effort has been paid to incorporate reactive behavior to the data management facilities available. Reactive behavior is characterized by variants of the event–condition–action model. Applications areas include checking for integrity constraints, system alerts, materialized view maintenance (especially useful in data warehousing), replication of data for audit purposes, data sampling, workflow processing, implementation of business rules, scheduling, and many others. Practically all products offered today in the database marketplace support complex reactive behavior on the client side. Nevertheless, the reactive behavior supported by those products on the server side is poor. Recently, the topic has regained attention because of the inherent reactive nature demanded in Web applications and the necessity of migrating many of the functionalities of browsers to active Web servers (Bonifati, Braga, Campi, & Ceri, 2002).

BACKGROUND

Several applications that support reactive behavior in the electronic commerce arena appeared recently, as is the case in the following: the Active Views system, described in Abiteboul et al. (1999); the event–condition–action (ECA) rule language for XML repositories, described in Bailey, Poulovassilis, and Wood (2002); and the set of classes for remote notification within a Web service environment, described in Bonifati, Ceri, and Paraboschi (2001).

Supporting reactive behavior implies that a database management system must be viewed from a production rule system perspective (Baralis, Ceri, & Paraboschi, 1996). An active database system must support the definition of production rules. Production rules are well known nowadays, in database terminology, under the name of active rules or, simply, triggers.

Active rules and integrity constraints are related topics (Ceri, Cochrane, & Widom, 2000). Database engines do not bring a full support of declarative integrity constraints in their kernels. When a complex constraint must be enforced on data, and the constraint cannot be declared, it must be emulated by means of triggers.

From a user's point of view, reactivity is a concept related to object state evolution over time. Dynamic constraints, constraints making assertions on the evolution of object states, may be needed to control changes in the states of data objects (Sistla & Wolfson, 1995b).

ACTIVITY WITHIN DATABASES

Usually, a database system performs its actions in response to requests from users in a passive way. In some cases, it is desirable that actions be taken with no human intervention, that is, automatic response to certain events.

Traditionally, the latter behavior has been obtained by embedding it into user applications; that is, the application software recognizes some events triggered by an user and performs some actions in response.

Because of the complexity in supporting reactive behavior, it would be desirable that the active functionality be provided by the database system. A database with a capability of reacting to external or internal stimuli is called an active database. An active database system can be thought of as coupling a database management system with a rule-based programming environment (Paton & Diaz, 1999). Among the applications that use active database systems nowadays, we can mention inventory control systems, online reservation systems, and portfolio management systems, just to name a few.

Knowledge Model

A central issue in the knowledge model of active databases is the concept of active rule.

An active rule is defined throughout three dimensions: event, condition, and action. In this case, this is termed an ECA rule.

An event is something that happens at a point in time. The source of the event may be transactional (an abort, commit, or begin transaction), operational (insert, delete, or update operations), temporal (clock signaling), or external (generated by the environment). An event can be classified according to its complexity as primitive or composite.

A condition, i.e., a predicate evaluated on data states, is the second component of an active rule. Moreover, because the state of data may change, before and after the occurrence of an event, the condition should be able to refer to previous and new states.

An action consists in a sequence of operations. There are several options for possible actions: the update of the contents or structure of the database, the call of an external procedure, the abort of the current transaction, or the notification about some unexpected situation.

Execution Model

The execution model for a set of active rules determines how the rules are managed at execution time. This model is strongly dependent on the particular implementation. However, it is possible to describe it in general using a set of common activities or phases: signaling, triggering, evaluating, scheduling, and executing.

How these phases are synchronized depends on the so-called coupling modes of ECA rules. The relationship among the aforementioned activities of the rules involves the concepts shown in Table 1.

Termination and Confluence

The behavior of active rules is hard to understand and control (Baralis, Ceri, & Widom, 1993). Rule interaction is one of the most important aspects related to rule set behavior. Two important properties related to this problem are observed: termination and confluence. It is said that a rule set is guaranteed to terminate if, for any database state and initial event, rule processing cannot continue forever. A rule set is confluent if, for any database state and initial event, the final database state after rule processing is independent of the order in which activated rules are executed.

Basic methods that perform termination analysis of a rule set have been discovered. However, because of the undecidability of the problem in general (Bailey, Dong, & Ramamohanarao, 1998), we cannot always decide whether a rule firing process is guaranteed to finish.

According to the time when those methods are applied, they can be classified as static, if the rule set is analyzed at compile time, or dynamic, if the rule set behavior is analyzed at run time. Deciding whether the condition of one rule is affected by the action of other rules, and when two rule actions commute, is known as the propagation problem. Propagation has been thoroughly studied (see Baralis & Widom, 2000). Propagation is the crux of static methods to determine confluence and termination (Widom & Ceri, 1996).

In the commercial systems side, an approach consists of imposing syntactic limitations, in order to guarantee termination or confluence at run time, although in some cases, counters are used to prevent infinite execution.

Table 1. Concepts

• Activation time
• Transition granularity
• Net effect policy
• Cycle policy
• Consumption modes
• Rule execution ordering

ACTIVE RULES AND DECLARATIVE CONSTRAINTS

Declarative constraints are user definitions specifying restrictions that the database states must satisfy. In a SQL-1999 (Standard Query Language-1999) compliant system, four classes of declarative constraints are supported: check predicate constraints, referential constraints, assertions, and view check options. Check predicate constraints aim at validating conditions against the actual state of *one* table in the database, and they include *primary key* and *unique* definitions, *not null* column definition, and *explicit check* clauses that validate general predicates on the values of some of the columns of the table. Referential constraints aim at guaranteeing that a many-to-one relationship holds on the actual state of two

Table 2. Why declarative constraints and triggers must be distinguished

• Declarative constraints should be processed after all changes are effectively applied.
• Inconsistent states would lead to unpredictable behavior when firing a trigger.
• Processing constraints and triggers together should be confluent.