Chapter XXI
Event–Condition–Action (ECA) Rules for Maintaining the Integrity Constraints of Mobile Databases

Hamidah Ibrahim
Universiti Putra Malaysia, Malaysia

Zarina Dzolkhifli
Universiti Putra Malaysia, Malaysia

ABSTRACT

An important problem for a database system is to guarantee database consistency. Difficulty arises in devising an efficient algorithm for checking database integrity against updates and for maintaining the consistency of the database, which indisputably desires great attention. However, it is intricate to accomplish particularly in a distributed and mobile environment due to the nature and characteristics of these environments. This chapter presents a model for maintaining the consistency of mobile databases. A rule-based mechanism based on the Event-Condition-Action (ECA) rules is proposed wherein the process of constraint maintenance will trigger the appropriate rule depending on the location (site) where the update is executed. In mobile databases, violation of integrity constraints can occur (detect) at three different levels (locations), namely: mobile host, local base station and remote base station. We believe that the proposed mechanism will reduce the amount of data accessed in addition to the amount of data transferred across the network during the process of maintaining the consistency of the mobile databases. Besides, it is without a doubt that it will speed up the checking process.

INTRODUCTION

A **database state** is said to be consistent if the database satisfies a set of statements, called *semantic integrity constraints* (or simply *constraints*). Integrity constraints specify those configurations of the data that are considered semantically correct. Any update operation (insert, delete or modify) or **transaction** (sequence of updates) that occurs must not result in a state that violates these constraints. Thus, a fundamental issue concerning integrity constraints is *constraint checking*, that is the process of ensuring that the integrity constraints are satisfied by the **database** after it has been updated. Checking the consistency of a database state will generally involve the execution of *integrity tests* (**query** that returns the value true or false) on the database which verify whether the database is satisfying its constraints or not. Nevertheless, another important issue is *maintaining* the integrity of the database. This process, which involves triggering the appropriate actions to secure the integrity of data, has always been neglected especially in distributed and mobile environment.

In the literature, three approaches for guaranteeing database consistency have been reported. In the first approach, the responsibility for ensuring the consistency of the database when a transaction occurs is part of the transaction design process. The transaction designers are responsible for ensuring that transactions are *safe*: that is, when executed, the transactions are guaranteed to bring the database from one **consistent** state to another. Consequently, transactions can get very complex and a *transaction design tool* is usually incorporated into the system to assist designers to construct safe transactions. In the second approach, transactions have integrity tests embedded in them to perform the necessary integrity checking. The modified transactions can then be executed by standard transaction facilities. This approach is based on the *query modification*[1] and *transaction modification* strategies, where an arbitrary query or transaction that may violate the integrity of a database is modified, such that the execution of the modified query or transaction is sure to leave the database in a consistent state. In the third approach, integrity tests are general rather than transaction-specific and thus no knowledge of the internal structure of a transaction is required. Typically, this approach requires rule mechanisms to implement integrity constraint enforcement (Grefen, 1990; Ibrahim, 2002; Ibrahim, Gray & Fiddian, 2001).

Checking and maintaining database integrity in a mobile environment require different mechanisms from those used in centralized and distributed databases due to the following facts: (i) Characteristics of the mobile environment which include high communication latency, intermittent wireless connectivity, limited battery life and changing client location (Elmasri & Navathe, 2007). (ii) Data is unevenly distributed among the base stations and mobile units (Elmasri & Navathe, 2007). (iii) Integrity constraints may spread over several base stations in the system. A large amount of data may therefore need to be transferred around the network in order to determine the truth of such statements. (iv) Frequent updates can lead to frequent executions of expensive violation testing operations. (v) Allowing an update to execute with the intention of aborting it at commit time in the event of constraint violation is also inefficient since rollback and recovery must occur at all stations which participated in the update, which can be a very costly operation in mobile systems.

Figure 1 presents an overview of the mobile database architecture. It consists of mobile support stations (MSS), or base stations (BS), and mobile units (MU), or hosts (MH). The base station is a stationary component in the model and is responsible for a small geographic area called a cell. They are connected to each other through fixed networks. The mobile host is the mobile component of the model and may move from one cell to another. These mobile hosts communicate