Chapter XXXVII
Improving Constraints Checking in Distributed Databases with Complete, Sufficient, and Support Tests

Ali Amer Alwan
Universiti Putra Malaysia, Malaysia

Hamidah Ibrahim
Universiti Putra Malaysia, Malaysia

Nur Izura Udzir
Universiti Putra Malaysia, Malaysia

INTRODUCTION

A **database state** is said to be **consistent** if and only if it satisfies the set of integrity constraints. A **database state** may change into a new state when it is updated either by a single **update operation** (insert, delete, or modify) or by a sequence of updates (**transaction**). If a constraint is false in the new state, the new state is inconsistent, therefore the enforcement mechanism can either perform compensatory actions to produce a new consistent state, or restore the initial state by **undo**ing the update operation. The steps generate integrity tests that are **queries** composed from the integrity constraints and the update operations, and run these **queries** against the **database**, which check whether all the integrity constraints of the database are satisfied; are referred to as *integrity checking* (Alwan, Ibrahim, & Udzir, 2007; Ibrahim, 2006; Ibrahim, Gray & Fiddian, 2001), is the main focus of this chapter.

The growing complexity of modern database applications in addition to the need for support of multiple users has further increased the need for a powerful integrity subsystem to be incorporated into these systems. Therefore, a complete integrity subsystem is considered to be an important

part of any modern DBMS. The crucial problem in designing this subsystem is the difficulty of devising an efficient algorithm for enforcing database integrity against updates (Ibrahim, Gray & Fiddian, 2001). Thus, it is not surprising that much attention has been paid to the maintenance of integrity in centralized databases. A naïve approach is to perform the update and then check whether the integrity constraints are satisfied in the new database state. This method, termed *brute force checking*, is very expensive, impractical, and can lead to prohibitive processing costs. Enforcement is costly because the evaluation of integrity constraints requires accessing large amounts of data, which are not involved in the database update transition. Hence, improvements to this approach have been reported in many research papers (Henschen, McCune & Naqvi, 1984; Martinenghi, 2005; McCune & Henschen, 1989; Nicolas, 1982; Qian, 1990; Simon & Valduriez, 1986). The problem of devising an efficient enforcement is more crucial in a distributed environment.

The brute force strategy of checking constraints is worse in the distributed context since the checking would typically require data transfer as well as computation leading to complex algorithms to determine the most efficient approach. Allowing an update to execute with the intension of aborting it at commit time in the event of constraints violation is also inefficient given that rollback and recovery must occur at all sites which participated in the update. Moreover, devising an efficient algorithm for enforcing database integrity against update is extremely difficult to implement and can lead to prohibitive processing costs in a distributed environment (Grefen, 1993; Ibrahim, Gray & Fiddian, 2001). A comprehensive survey on the issues of constraint checking in centralized, distributed, and parallel databases are provided in (Feras, 2005; Ibrahim, 2006).

Works in the area of constraint checking for distributed databases concentrate on improving the performance of the checking mechanism by executing the complete and sufficient tests when necessary. None of the work has to look at the potential of support test in enhancing the performance of the checking mechanism. Also, the previous works claimed that the sufficient test is cheaper than the complete test and its initial integrity constraint. These depend solely on the assumption that the update operation is submitted at the site where the **relations** to be updated is located, which is not necessary the case. Thus, the aim of this chapter is to analyze the performance of the checking process when various types of integrity tests are considered without concentrating on a certain type of test as suggested by previous works. The most suitable test will be selected from the various alternative tests in determining the consistency of the distributed databases. Here, suitable means the test that minimizes the amount of data transferred across the network and the number of sites involved during the process of checking the constraints.

RELATED WORK

For distributed databases, a number of researchers have looked at the problem of semantic integrity checking. Although many research works have been conducted concerning the issues of integrity constraint checking and maintaining in distributed databases but these works are limited due to the fact that (i) they cater limited type of constraints, (ii) their approaches did not use the available information during the process of checking, and (iii) their approaches are able to generate limited type of integrity tests. This is briefly shown in Table 1, where column labeled 1, 2, 3, 4, 5, 6, and 7 represent the work by Simon and Valduriez (1986), Qian (1989), Mazumdar (1993), Gupta (1994), Ibrahim, Gray and Fiddian (2001), Ibrahim (2002), and Madiraju and Sunderraman (2004), respectively.

The work presented in Simon and Valduriez (1986) constructed a simplification method for integrity constraints expressed in terms of as-