Chapter 8
A Model for Ranking and Selecting Integrity Tests in a Distributed Database

Ali A. Alwan
Universiti Putra Malaysia, Malaysia

Hamidah Ibrahim
Universiti Putra Malaysia, Malaysia

Nur Izura Udzir
Universiti Putra Malaysia, Malaysia

ABSTRACT
Checking the consistency of a database state generally involves the execution of integrity tests on the database, which verify whether the database is satisfying its constraints or not. This paper presents the various types of integrity tests as reported in previous works and discusses how these tests can significantly improve the performance of the constraint checking mechanisms without limiting to a certain type of test. Having these test alternatives and selecting the most suitable test is an issue that needs to be tackled. In this regard, the authors propose a model to rank and select the suitable test to be evaluated given several alternative tests. The model uses the amount of data transferred across the network, the number of sites involved, and the amount of data accessed as the parameters in deciding the suitable test. Several analyses have been performed to evaluate the proposed model, and results show that the model achieves a higher percentage of local processing as compared to the previous selected strategies.

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, the enforcement mechanism can either perform compensatory actions to produce a new consistent state, or restore the initial state by undoing the
A Model for Ranking and Selecting Integrity Tests in a Distributed Database

update operation. The steps, generate integrity tests, which 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 (Ali, Hamidah, & Nur Izura, 2009; Ibrahim, Gray, & Fiddian, 2001; Ibrahim, 2006) is the main focus of this paper.

The growing complexity of modern database applications plus the need to support 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 (Martinenghi, 2005; McCune & Henschen, 1989; Nam, 1998; Nicolas, 1982; Qian, 1989; Simon & Valduriez, 1989). 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 intention of aborting it at commit time in the event of constraints violation is also inefficient since 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 is provided in (Feras, 2006; 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 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. They 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 paper is to analyze the performance of the checking process when various types of integrity tests are considered without concentrating on certain type of test as suggested by previous works. The most suitable test is 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, the amount of data accessed, and the number of sites involved during the process of checking the constraints.

In this paper we provide the previous works related to this research, the basic definitions, notations, and examples which are used in the rest of the paper. In addition, discussion on the importance of considering different types of integrity tests is given. Next we introduce our proposed model to rank and to select integrity tests. Finally, the results of the proposed model are extensively discussed.