Assuring Database Integrity

David Stemple
University of Massachusetts

Eric Simon
INRIA, France

Subhasish Mazumdar
University of Massachusetts

Matthias Jarke
University of Passau, West Germany

Database integrity has many facets. Both consistency, the property of a database modeling a possible state of the world, and correctness, the exact correspondence of the data to the state of the world being modeled, are important aspects of integrity. Integrity also includes guaranteeing that meaningful views and changes result from interleaved access to the database occurring in typical transaction processing, a property known as atomicity of transactions. Assuring database integrity requires several mechanisms, including language capabilities, runtime processing, and operational controls. Here we concentrate on the maintenance of consistency, both the means of stating database consistency properties and techniques of efficiently assuring that a database can only change from one consistent state to another. We present a theorem-based method of simplifying integrity constraints at the heart of most techniques for achieving efficient integrity assurance. We argue that these techniques are currently practical enough to be included in database system products in the near future, but that languages used to specify constraints and transactions will have to be of a high level of abstraction, reasonably formal, and uniform in order for this to happen.

The problem of assuring that a database contains the right data can be broken down into several problems. The first is to create a structure or design for the database that allows representation of all the states of the world that can legitimately occur. The second is to allow only data representing “actual” facts to be entered in the database. Since the database system very seldom has direct access to the real world, such as physical sensors, comparison of data to “actuality” is not normally feasible. For this reason, operational controls outside the computerized system are ultimately needed to solve the problem of factual data. However, if the data can be tested to assure that it produces at least a consistent state of the database, i.e., a state that models a possible state of the world, then many database instances that could not possibly be factual can be avoided. Thus consistency, while not totally capturing correctness, is an important precondition for correctness, and is achievable within the computerized system, unlike full correctness.

In this paper we will concentrate on the problem of assuring database consistency. We will present the various aspects of the problem of expressing integrity constraints, which will be the linguistic means of specifying the consistency of a database. We will not deal with either the problem of building controls for assuring the factual nature of a database or the issues of concurrency control, the means of effecting meaningful changes and views of a database subjected to interleaved access and update.

We will first outline the varieties of integrity and relate the consistency issues with those of “good” database design, such as those incorporated in relational
normalization theory. We find it useful to separate the consistency problem into two important subcases, static and dynamic. Transaction atomicity will be discussed briefly to explain why it is a requirement for maintaining consistency of a database, but is unrelated to the details of a particular database's integrity constraints. Means of specifying integrity constraints and transactions are discussed in detail. We then survey the ways in which the efficiency of maintaining integrity constraints can be improved by using theorems that relate integrity predicates with update primitives. The main problem is to attain efficiency in processing transactions while assuring that the integrity of the database is maintained.

Varieties of Integrity

Database integrity addresses a variety of concerns, all subsumed under the desire that the data in a database be correct, that is, represent the proper state of the modeled world. However, consistency, the property of a database modeling a possible state of the world, is also an important concept since, as explained above, it is a precondition for correctness and has the possibility of system-maintenance, an option that does not exist for full correctness. Integrity also includes guaranteeing that meaningful views and changes result from interleaved access to the database occurring in typical transaction processing, a property known as atomicity of transactions. In this section we outline the varieties of integrity that are included under the notion of consistency, and discuss the issue of atomicity only briefly.

State Predicates and Static Integrity. Many of the properties that define the consistency of a database can be represented by predicates on the database state (Eswaran & Chamberlain, 1975), i.e., statements containing variables which, when the variables are assigned values from the database state, can be evaluated as true or false. If all of a set of such predicates defining the integrity of a database are evaluated as true using the values from a database state, then the state is consistent. Such predicates define what we will call state-based or static integrity.

In order to provide a concrete basis for talking about consistency, we will use an example database. The entity-relationship diagram in Figure 1 will provide the basis for our examples. The world modeled by the diagram consists of employees, projects and skills. Employees possess skills and projects require skills. Projects depend on each other, which means that a dependent project cannot start until the project(s) on which it depends is (are) completed. Employees are managed by employees and are assigned for a certain percentage of their time to projects.

Examples of state-based integrity constraints in a database representing employees, skills and projects include:

1. An employee number is less than 200,000.
2. An employee who is a manager has a salary greater than $40,000.
3. Each employee has a unique employee number.
4. All employee numbers in the database are employee numbers for employees described in the database.
5. The set of skills possessed by the employees working on a project contains the skills required by the project.
6. The starting date of a project must not be earlier than the end date of any project on which it depends.
7. The way in which projects depend upon each other cannot be cyclic.
8. The salary budget of a project will be the weighted sum of the assigned employees' salaries, where the weighting is the percentage of time the employee is assigned to the project.

The variables in these constraints are employee(s), skills and project(s). It should be clear that if we understand the meaning of the data in the database, we could evaluate these predicates to determine whether the database was consistent or not.

The amount of consistency that can be expressed by static integrity constraints clearly depends on the amount of facts that are represented in the database. This is most obvious when the way in which the state of the
Related Content

Information Privacy: Understanding How Firms Behave Online
Gerald C. Kane, Kathy Stewart Schwaig and Veda C. Storey (2011). Theoretical and Practical Advances in Information Systems Development: Emerging Trends and Approaches (pp. 81-100).
www.igi-global.com/chapter/information-privacy-understanding-firms-behave/52953?camid=4v1a

An Enterprise Model Repository: Architecture and System
www.igi-global.com/article/enterprise-model-repository/3246?camid=4v1a

A Framework for Analyzing Mobile Transaction Models
www.igi-global.com/article/framework-analyzing-mobile-transaction-models/3267?camid=4v1a

E-Government Databases
www.igi-global.com/chapter/government-databases/11147?camid=4v1a