An Agent-Based Approach to Correctness in Databases

Herbert Stoyan, Stefan Mandl, Sebastian Schmidt, Mario Vogel

Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
{hstoyan, stefan.mandl, sebastian.schmidt, mario.vogel}@informatik.uni-erlangen.de

Abstract. When defining the schema of a relational database, integrity constraints are included to describe simple syntactic constraints of correctness that can easily be tested in a centralized way when tuples are inserted, deleted or updated. Complex dependencies may exist between different tuples of a relation. The description of them can be difficult with current formalisms. An example for such an inconsistency is the problem of duplicates, the existence of different tuples describing the same real world entity. Duplicates can occur when one makes typographic or other errors while transferring the representation of a real-world entity to the database. In this paper, we describe a new method to detect dependencies of that kind using continuously active agents that check consistency of a database and propose steps to improve the content of the database.

1 Introduction

Consistency plays an important role in the database literature [2], e.g., it is discussed how to ensure that transactions realize the transition from a consistent database state to another. If a transaction fails, obeying the principle of ACID at least guarantees a roll-back to the last known consistent state. There is consensus to centrally organize control of consistency in the DBMS.

Integrity constraints are classified into static and dynamic constraints. The latter restrict permitted updates (the dead cannot become alive again), the former restrict sets of tuples by constraining singular tuples by relations between some attributes or by claiming relationships between the database relations.

The question is then how to achieve on the basis of integrity constraints the goals of consistency (free of contradictions), data uniqueness, completeness and semantic correctness—under central control. One may move it to the database administrator (or schema designer): He has to formulate the right integrity constrains. At present, typical integrity constraints give restrictions of value sets or enumerate permitted values. A schema designer often can not formulate the right constraints at design time, because usually the complex ones evolve when the database fills.

T. Härder and W. Lehner (Eds.): Data Management (Wedekind Festschrift), LNCS 3551, pp. 37-47, 2005.

Using this, input will be denied if it violates these constraints. This might be a good principle for a database with qualified input personal.

2 The Problem, Preliminaries

Let us define the content of a tuple to be a statement about the existence of an entity with some properties or a statement about relationships between some entities. If so, a database is a set of representations of such statements. The schema of a database specifies the formal structure of the representation of those statements of existence and of relationship. The statements, i.e. the tuples, are constrained by the definition of the database schema. They must be of a certain structure to be acceptable by a relation. Entities of the real world—similar to the statements—cannot be stored, but only represented. Those representations are made up of numbers, enumerations, strings and references. Computers can operate on various ways on numbers and have a contentful comparison facility.

Strings are less flexible: The only comparisons are equality or difference, an operation is the concatenation—both are weakly related to the representation function of the strings. For references the comparison of identity is sufficient. References relate to other statements of existence which can be regarded of being representations of the entity of which the existence is stated.

Generally, databases should contain only (representations of) statements of existence and relationship that are true, although it might be possible to store only false statements for special applications. At a meta-level, false statements are also true: the statement x has been made, but is false. If a database contains only representations of true statements, we call it "absolute correctness".

Because databases usually contain thousands or even millions of such representations (records) striving for absolute correctness is very hard.

Errors have two causes: The representation of a statement relates to a false or a pointless statement. If the original statement was pointless or false already or if the representation failed is unimportant. For our scenario of a community collecting a data set, the latter could be interesting: If several members of the community want to add the same statement the result may be a set of representations: several correct, some false, and some pointless.

Dates of events are typical content of databases. If only exact data are stored, there could be a global consistency check for a date.

Pointless statements like the "X was born 30th February" could be detected this way. But if date approximations or incomplete dates matter the only alternative may be a string representation. If historical dates are to be represented, aberrations from the current Gregorian calendar might obscure the data correctness. So, date descriptions of the "6.Trecember" or "190a|1908" (as an alternative of years) may occur. To exclude such strings from being stored in the date field is much more difficult. The integrity constraints have to enable the definition of a date syntax grammar, which is beyond the current possibilities.