Chapter I

Transformation-Based Database Engineering

Jean-Luc Hainaut, University of Namur, Belgium

Abstract

In this chapter, we develop a transformational framework in which many database engineering processes can be modeled in a precise way, and in which properties such as semantics preservation and propagation can be studied rigorously. Indeed, the transformational paradigm is particularly suited to database schema manipulation and translation, that are the basis of such processes as schema normalization and optimization, model translation, reverse engineering, database integration and federation or database migration. The presentation first develops a theoretical framework based on a rich, wide spectrum specification model. Then, it describes how more complex transformations can be built through predicate-based filtering and composition. Finally, it analyzes two major engineering activities, namely database design and reverse engineering, modeled as goal-oriented schema transformations.
Motivation and Introduction

Modeling software design as the systematic transformation of formal specifications into efficient programs, and building CASE tools that support it, has long been considered one of the ultimate goals of software engineering. For instance, Balzer (1981) and Fikas (1985) consider that the process of developing a program [can be] formalized as a set of correctness-preserving transformations [...] aimed to compilable and efficient program production. In this context, according to Partsch (1983),

“a transformation is a relation between two program schemes \( P \) and \( P' \) (a program scheme is the [parameterized] representation of a class of related programs; a program of this class is obtained by instantiating the scheme parameters). It is said to be correct if a certain semantic relation holds between \( P \) and \( P' \).”

These definitions still hold for database schemas, which are special kinds of abstract program schemes. The concept of transformation is particularly attractive in this realm, though it has not often been made explicit (for instance, as a user tool) in current CASE tools. A (schema) transformation is most generally considered to be an operator by which a data structure \( S_1 \) (possibly empty) is replaced by another structure \( S_2 \) (possibly empty) which may have some sort of equivalence with \( S_1 \). Some transformations change the information contents of the source schema, particularly in schema building (adding an entity type or an attribute) and in schema evolution (removing a constraint or extending a relationship type). Others preserve it and will be called semantics-preserving or reversible. Among them, we will find those which just change the nature of a schema object, such as transforming an entity type into a relationship type or extracting a set of attributes as an independent entity type.


The goal of this chapter is to develop and illustrate a general framework for database transformations in which all the processes mentioned above can be