Available online at www.sciencedirect.com
ScienceDirect
Data & Knowledge Engineering 63 (2007) 568–589
DATA & KNOWLEDGE ENGINEERING
www.elsevier.com/locate/datak

Extracting generalization hierarchies from relational databases: A reverse engineering approach

Nadira Lammari [a], Isabelle Comyn-Wattiau [b,*], Jacky Akoka [c]

[a] CEDRIC-CNAM, 292 Rue Saint Martin, 75141 Paris Cedex 03, France
[b] CEDRIC-CNAM and ESSEC, 292 Rue Saint Martin, 75141 Paris Cedex 03, France
[c] CEDRIC-CNAM and INT, 292 Rue Saint Martin, 75141 Paris Cedex 03, France

Received 28 August 2004; received in revised form 26 April 2006; accepted 11 April 2007
Available online 21 April 2007

Abstract

Relational Data Base Management Systems (RDBMS) are currently the most popular database management systems. The relational model is a simple and powerful model for representing real world applications. However, it lacks the expressiveness of conceptual models. Unlike the latter, the relational model does not offer the generalization abstraction. Therefore, it does not allow the designer to represent directly a large variety of integrity constraints. Moreover, inclusion dependencies formalizing inter-relational constraints cannot directly be represented in the relational model, due to the fact that its basic construct, the relation, is the unique structure. Finally, relational databases do not enable a natural way to represent inheritances. In this paper we describe a reverse engineering method which particularly deals with the elicitation of inheritance links embedded in a relational database, combining heuristic and algorithmic approaches. We provide rules for detecting intersection constraints and inclusion dependencies. Heuristics are proposed for understanding null value semantics. Finally, we present decision rules for detecting existence dependencies and translating them into IS-A hierarchies among entities. An example is used to illustrate our approach.

Keywords: Reverse engineering; IS-A inheritance links; Relational databases; Extended entity-relationship (EER) model; Generalization hierarchies

1. Introduction

The relational data model is known to be a semantically poor model. As an illustration, the concept of generalization/specialization (G/S) cannot be directly represented. The G/S concept has been introduced by Smith and Smith [27]. It has been incorporated in the Entity-Relationship (ER) model by Scheuermann et al. [24]. Basically, entity-type E may be considered as a generalization of the entities $E_1, E_2, \ldots, E_n$ if every instance

* Corresponding author. Tel.: +33 1 34 43 30 76; fax: +33 1 34 43 30 01.
E-mail addresses: lammari@cnam.fr (N. Lammari), wattiau@cnam.fr (I. Comyn-Wattiau), akoka@cnam.fr (J. Akoka).

0169-023X/$ - see front matter
doi:10.1016/j.datak.2007.04.002

of E is also an instance of at least one E_i. In other words, $E_1, E_2, \ldots, E_n$ are specializations of E. Each specialization may usually be distinguished by the value of one or several attributes. Specialized entities inherit the properties of more general entities. The basic idea of inheritance is that new entities may be defined in a database schema as specializations of previously defined entities. The main advantage is to facilitate evolutions of the database schema. It also allows the designer to integrate different user views of the database. It provides the designer with a semantically sound mechanism for representing IS-A hierarchies whatever the number of specializations.

As an illustration, let us consider a database which maintains details of various types of persons (students, clerks, professors, etc.) in a university. We choose to define a PERSON as a generalization of STUDENT. In the Extended Entity-Relationship (EER) model, inheritance represents the IS-A relationship: "every student is a person". It implies that every attribute relevant for PERSON instances is inherently applicable to STUDENT instances. The relational model provides no mechanism for incorporating such semantic information. The resulting relational schema may be defined as follows:

PERSON (SSN, Name, Age)
STUDENT (SSN, Name, Age, Department, Degree)

The relation PERSON contains all the instances of persons except students in order to avoid redundancy. A view representing all the instances of the generalization can be specified. Another possible implementation consists of one relation PERSON defined by all the attributes of the hierarchy. In this case, null values are introduced and existence dependencies must be specified. A third usual solution consists in implementing the following relations:

PERSON (SSN, Name, Age)
STUDENT (SSN, Department, Degree)
with a referential integrity constraint between SSN of student and SSN of Person.

More generally, relational databases lack concepts that enable a natural representation of inheritance. As a consequence, designers have to resort to different strategies such as representing only specializations, introducing null values, defining inclusion dependencies and views.

Database reverse engineering is a set of techniques allowing the designer to extract semantics from existing databases in order to facilitate their maintenance and to enable their migration. This paper deals with relational database reverse engineering, especially with generalization hierarchies reverse engineering. Discovering these concealed semantic links is a crucial task. The latter requires the elicitation of the underlying concepts of existence and inclusion dependencies. Many algorithms have been proposed to perform a reverse engineering process of relational databases. While they share a number of features in common, only seven algorithms adopt a specific approach for eliciting generalization/specialization hierarchies [6,8,12,15,20,23,29].

The aim of this paper is to focus on generalization hierarchies reverse engineering. This process is integrated into a relational database reverse engineering global approach called MeRCI (a French acronym for Méthode de Rétro-Conception Intelligente) [10]. One contribution of this paper is to formalize the detection of generalization hierarchies using three common information sources explored by the reverse engineer when available: the database structure expressed as Data Definition Language (DDL) specifications, the database dynamics through Data Manipulation Language (DML) queries and programs and, finally, the actual data. Another main contribution of this paper is related to the tight combination of algorithmic and heuristic rules. Our approach structures the rules described in past approaches for generalization hierarchies reverse engineering and enriches them with new rules, especially based on null values, homonyms and synonyms, existence dependencies and intersection constraints.