Extracting an Entity Relationship Schema from a Relational Database through Reverse Engineering

Martin Andersson

Database Laboratory, Department of Computer Science
Swiss Federal Institute of Technology
CH-1015 Lausanne, Switzerland
E-mail: Martin.Andersson@di.epfl.ch

Abstract. This paper presents a method for extracting a conceptual schema from a relational database. The method is based on an analysis of data manipulation statements in the code of an application using a relational DBMS. Attributes representing references between tables in the relational schema, and possible keys are determined by an analysis of join conditions in queries and view definitions. Knowledge about which attributes link tables is used to investigate the database extension in a selective manner. When the keys cannot be unambiguously determined, possible solutions are generated by the system under guidance of the user. The approach makes it possible to efficiently construct a conceptual schema from only rudimentary information.

1 Introduction

The current rapid progress in the telecommunications domain will allow geographically distributed computers to interact more closely than today. However, this evolution is slowed down by existing information systems (IS) based on old-fashioned technology that does not allow them to be part of a distributed computing environment. These systems are referred to as legacy systems [2] and are characterized by e.g. old-fashioned architecture, lack of documentation and non-uniformity resulting from numerous extensions. These properties lead to inflexible systems and high maintenance costs. Reverse engineering can help by extracting a conceptual, implementation independent specification that will provide a basis for future evolution of the IS. We propose a method for extracting *ERC+* [13] specifications from a relational database using only rudimentary information that can be expected to be found in a legacy system.

Most previous methods for translation from the relational to a conceptual model assume that functional and inclusion dependencies are given beforehand, e.g. [1], [12], [6], [9], [11], [7], [15], [10]. A few approaches do not, e.g. [14] where several different sources are used to obtain information on keys and foreign keys, and in [3] where a method is presented for extracting functional dependencies from the database extension. In [5], possible inclusion dependencies are deduced from information on keys and foreign keys and are verified against the database extension.

The main interest of this work is reengineering of older systems, where the only information provided by the DBMS is table names, field names, indices and possibly view definitions. We try to deduce information on functional dependencies, keys and inclusion dependencies. The idea in this paper is to look for this information in data manipulation statements that can be extracted from the application code. In [14], the use of queries in the reverse engineering process is briefly mentioned. Our approach emphasizes the use of data manipulation statements in queries and views.

The rest of this paper is organized as follows. Section 2 explains the notation and gives a description of the *ERC+* model. Section 3 presents how information is extracted from the database application and represented in a *connection diagram*. Section 4 gives an overview of the rules applied to the information in the diagram to generate an *ERC+* schema. Finally, section 5 summarizes and gives directions of future work.

2 The *ERC+* Data Model

ERC+ extends the ER model, as defined in [4], with multivalued and complex objects, and multi-instantiation. Figure 1 shows an *ERC+* schema with an entity type *Faculty* that is a subtype of *Person*. *Faculty* is involved in the relationship type *Offer*, with cardinality 1:n, meaning that each faculty gives one or several courses. *Offer* is a ternary relationship type that associates exactly one instance of *Faculty*, *Department*, and *Course*. *Person* has a complex attribute *child* with cardinality 0:n, with two sub-attributes *name* and *age* both with cardinality 1:1. *Course* participates in *Offer* with cardinality 0:1.

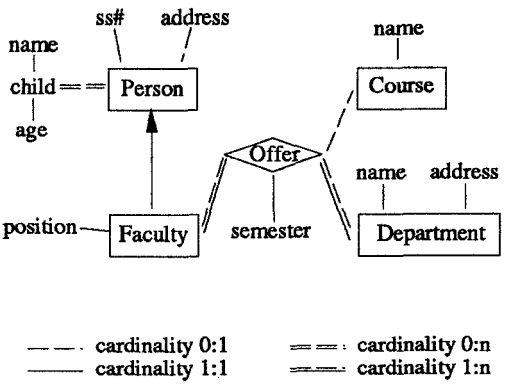

Fig. 1. *ERC+* schema

Multi-instantiation in *ERC+* comes in two flavours: *maybe* and *isa*. *maybe* is used to express that object instances can be part of the extension of several object types. *isa* means that the extension of one object type is included in the