Chapter VI

Mapping FuzzyEER Model Concepts to Relations

This chapter shows the transformation of the FuzzyEER model to a logical design by using relational databases. The FuzzyEER-to-Relational mapping algorithm is based on the “classical” EER-to-Relational mapping algorithm, published in Elmasri and Navathe (2000) and summarized in the first section of this chapter, but other versions are very similar (De Miguel, Piattini, & Marcos, 1999; Silerschatz, Korth, & Sudarshan, 2002). The FuzzyEER-to-Relational mapping algorithm includes additional rules for mapping fuzzy concepts.

The following sections translate the FuzzyEER concepts, that is, the definitions in Chapter IV, to the FIRST-2 schema, which was exposed in Chapter V. Thus, this chapter relates Chapter IV with Chapter V, obtaining a fuzzy relational database. In addition, we need a comprehensive fuzzy database language with statements for data definition, query, and update. This language is FSQL (Fuzzy SQL), and we describe it in Chapter VII.

It should be noted that some definitions in Chapter IV define fuzzy degrees to the model (see the “Zvieli and Chen Approach” section in Chapter III and the “Fuzzy Degree to the Model” section in Chapter IV). Of course, these degrees are not mapped to the relational database. As such, Definitions 4.7, 4.8, and 4.12 are not treated in this chapter.
EER-to-Relational Mapping Algorithm

Basically an EER schema may be mapped into the corresponding relational database schema in 10 steps. The first seven steps are related to ER models, and the last steps are related to the superclass/subclass relationships in EER models.

**STEP 1**: For each regular (strong) entity type E, create a relation R that includes all the simple attributes or the simple component attributes of composite attributes of E. One or some attributes must be the primary key of E.

**STEP 2**: For each weak entity type W with owner entity type E, create a relation R, including the attributes as in Step 1. Furthermore, R must include a foreign key to the relation of E. The primary key of R is the combination of the primary key of the owner (relation of E) and the partial key of W, if any. It is common to choose the propagate option for the referential triggered action, that is, the SQL clauses `ON UPDATE CASCADE` and `ON DELETE CASCADE` in the foreign key, because a weak entity has an existence dependency on its owner entity.

**STEP 3**: For each binary 1:1 relationship type, choose one of the participating entities and include the primary key of the other one as foreign key of one of the entities’ relations. It is better to choose an entity type with total participation in the relationship. Include all the simple or composite attributes in the relation of the chosen entity. An alternative mapping of a 1:1 relationship type is possible by merging the two entity types and the relationships into a single relation. This is better when both participations are total.

**STEP 4**: For each regular binary 1:N relationship, identify the relation S that represents the participating entity type at the N-side of the relationship type. Include as foreign key in the S relation the primary key of the other relation. Include all the simple or composite attributes in the relation of S.

**STEP 5**: For each binary M:N relationship type, create a new relation R. Include as foreign key in R the primary keys of the relations that represent the participating entity types; their combination will form the primary key of R. Include in R all the simple or composite attributes in the relationship. The