Enhancing the ER Model with Integrity Methods

Mira Balaban and Peretz Shoval
Ben-Gurion University of the Negev, Israel

Entity Relationship (ER) schemas include cardinality constraints, that restrict the dependencies among entities within a relationship type. The cardinality constraints have direct impact on application transactions, since insertions or deletions of entities or relationships might affect related entities. Application transactions can be strengthened to preserve the consistency of a database with respect to the cardinality constraints in a schema. Yet, once an ER schema is translated into a logical database schema, the direct correlation between the cardinality constraints and application transaction is lost, since the components of the ER schema might be decomposed among those of the logical database schema. We suggest to extend the Enhanced ER (EER) data model with integrity methods that take the cardinality constraints into account. The integrity methods can be fully defined by the cardinality constraints, using a small number of primitive update methods, and are automatically created for a given EER diagram. A translation of an EER schema into a logical database schema can create integrity routines by translating the primitive update methods alone. These integrity routines may be implemented as database procedures, if a relational DBMS is utilized, or as class methods, if an object-oriented DBMS is utilized.

Chen (1976) introduced the Entity Relationship (ER) data model as a means for describing in a diagrammatic form, entities and relationships among entities in the subject domain. The ER model enjoys widespread popularity as a tool for conceptual database design, and received many extensions and variations, which are generally termed the Enhanced ER (EER) model. An EER schema can be translated into logical database schemas, usually relational, and implemented with some specific DBMS, using its specific DDL (data definition language). Application programs that manipulate the database access the DBMS via its DML (data manipulation language), either directly or through a host programming language.

EER can be used not only to design a conceptual schema that will later on be translated into a logical schema, but also as a platform for database integration, i.e. to create a meta-schema for a multi database environment, in which there are heterogeneous databases, utilizing different data models. Cooperation or federation of such databases is possible if a common meta-schema is created. EER can be the high-level model used for that purpose. Similarly, the EER model is used in database reverse-engineering: the data model of a legacy-system is first reverse-engineered to an EER schema, and later on translated and implemented in a new DBMS. Yet, the EER model deals only with the static (structural) aspects of the data model (namely, entities, relationships and attributes) but not with behavioral aspects (namely procedures to manipulate the data that is defined by the schema, and to preserve the integrity of data). These aspects are taken care of at the implementation level, either by the DBMS (for example, when a relational DBMS performs referential integrity checks), or by the application programs.

An EER schema supports the specification of cardinality constraints, that restrict the dependencies among entities within a relationship type (see, for example, Lenzerini & Santucci, 1983; Lenzerini & Nobili, 1990; Ferg, 1991; Thalheim, 1992; Thalheim, 1998). For example, the cardinality constraints can specify that a department must have at least five workers and at most eighty. The cardinality constraints have direct impact on maintenance transactions of the target system, since insertions or deletions of entities or relationships might affect related entities. This impact can be captured by operations that a transaction must trigger in order to preserve the cardinality constraints. Yet, once an EER schema is translated into a logical database schema, the direct correlation between the
The Enhanced-Entity-Relationship (EER) Data Model

EER is a data model for describing entities, their properties, and inter-relationships. A set of entities that share a common structure is captured as an entity type. Regular properties of entities are captured as their attributes. The attributes are associated with the entity types, and can be either simple or composite. In most entity types, entities are identified by an attribute (or attributes), called a key (or keys). In some entity types, entities are identified by their inter-relationships to other entities. Such entity types are termed weak.

Interactions among entities are modeled by relationships. A relationship type relates several entity types; it denotes a set of relationships among their entities. The number of entity types related by a relationship type is its arity (• 2). The role of an entity type within a relationship type is specified by its role-name. Role-names are mandatory in case that an entity type plays several roles within a single relationship type. Two-ary relationship types are called binary. A binary relationship type that relates an entity type to itself is called unary. Specialization and generalization inter-relationships among entity types are singled out as special kinds of relationships.

The cardinality constraints are set on relationship types, and characterize numerical dependencies among entities within the relationship types. Existing EER models support a variety of cardinality constraints and maintain an OO schema, within an OO-DBMS or any other OO programming environment, the following strategy is recommended: first, design an EER schema, then map it to an equivalent OO schema. Finally, augment the OO schema with the necessary behavioral constructs (e.g., methods and messages).

A recent effort to standardize all aspects of object modeling is accomplished within the Unified Modeling Language (UML) initiative (Fowler, 1997; Booch et al., 1999). UML is a collection of visual languages that covers the static data, process and behavioral perspectives in modeling. The data modeling aspect in UML is handled by class diagrams and object diagrams, and also provides an Object Constraint Language. The static part of UML is richer than the conventional object schemas in object-oriented databases and in programming languages. The EER model is an integral part of UML — actually, UML includes all constructs of EER schema, except for weak entity types. Indeed, we use the EER model as a representative for the static part of object modeling, in general. Our work can be viewed in the wider aspect of extending UML with structure sensitive methods. The extension of the EER data model with methods blurs the difference between OO schemas to EER schemas.

In the following section the EER data model is introduced, and subsequent sections describe the suggested enhancement with structure methods.