Transforming Extended Entity-Relationship Model into Object-Relational Database using Triple Graph Grammars

Nguyen Thi Nhung, Sanggeun Song, and Sangjun Lee

Abstract -- The entity-relationship and extended entity-relationship have been the most popular paradigms for conceptual database design. However, with the appearance of a new generation Object-relational database and UML, conceptual database modeling using the EER approach has not attracted much interest, although the EER model is still powerful when working with object-oriented databases. Therefore in this thesis we proposed a transformation method for ORDB based on the EER model. The transformation rules are defined using triple graph grammars, and represented in the MOFLON TGG editor. To verify the proposed method, we analyze the completeness and correctness of the model transformation. Following triple graph grammar rules, our proposed method can be applied in automatic ORDB development frameworks.

Index Terms -- Database, Triple graph grammar, ORDB, EER model

I. INTRODUCTION

Introduced by Chen in 1976, the Entity-Relationship(ER) model[1] has become the most important paradigm for conceptual design in relational database development. Thanks to its simplicity and clarity structuring concepts, users can easily model real-world objects and their relationship naturally. Once an ER diagram is obtained after the conceptual design phase, database designers have to transform it into the logical data model of a target system. Various logical models of target systems have been transform from the ER model, include relational models, network models, hierarchical models and object-oriented database models[2].

With the increasing complexity of database applications and objects to modeling, basic concepts of ER modeling are no longer adequate. An improved ER model was introduced, called Extended Entity-Relationship(ER) model[3]. It extends the basic ER model, includes additional concepts such as generalizations/specializations and inheritances. However, when transforming these complex concepts into a Relational database(RDB), database designers face some obstacles because of the fixed data type system supported by RDB.

A new wave of DB generations appeared to overcome RDB’s problems: object-based database, which included Object-Relational database(ORDB)[4]. ORDB technology extends the RDB by providing a richer data type system and object oriented features, such as objects, classes, and inheritance. This DB technology is well established based on SQL:1999[5] and SQL:2003[6] standards, and supported in many major DB products.

In this paper, we propose a method for transforming EER model into ORDB model to those people who have been already familiar with EER modeling techniques but want to adopt ORDB technology.

II. RELATED WORK

A. ER and Extended ER model

The Entity-Relationship model[1] became one of the most popular method in database design and related fields. In database design, it is commonly used in the conceptual design phase to describe the structure of the database independent of the underlying system. The original ER model represents information in terms of entities, their attributes, and associations among entity occurrences called relationships. Entity sets are the principal things or objects about which information was collected in the real world, such as a person, place, or thing. Attributes are used to describe the entities by giving them descriptive properties such as a name, age, and address. A Relationship represents associations among entities.

B. Object-Relational Databases

The limitations of a relational database, such as a restricted set of built-in data types that accommodate only numbers and strings, flat structures, and un-supported object-oriented features has caused some deficiencies when working with complex objects and applications. Therefore the requirement for a new database model emerged. To overcome the limitation of RDB, an object-oriented database was developed to extend object-oriented programming techniques into the field of persistent data management[2]. However, OODB are not widely used because of the difficulties of tying DB design too closely to application design. Maintaining and evolving an OODB-based system is an arduous undertaking.
C. Triple Graph Grammars

Triple Graph Grammars (TGG) were introduced by Schürr in 1994 as a technique for model transformation[7]. This is a useful technique for defining a transformation in a declarative way and executing the transformation in both directions. TGG also allow us to define the relationship between two types of models and to transform a model of one type into another type. The correspondence between two models is computed incrementally to maintain the consistency between them, and to record and synchronize all changes.

As shown in Fig. 1, the left object depicts the source model, the right one demonstrates the transformed model or target, and the middle object represents the correspondence rules to mapping those models.

III. TGG TRANSFORMATION FOR ORDB DEVELOPMENT

A. EER Model

As we mentioned above, to apply TGG rules we need a triple graph schema or metamodel for each model integration scenario. In our method for ORDB development, the abstract model or PIM is a conceptual model, represented by EER and developed according to a requirement analysis process. The logical design of a DB is developed using obtained conceptual data. Based on a set of mapping rules, we obtain the ORDB schema at the PSM level that will be detailed in following sections.

Fig. 2 shows the metamodel of EER, which is expressed by notations in the MOFLON tool[8]. The metamodel states that an EER model consists of Entity, Relationship and Attribute.

Fig. 1. EER Model to ORDB Triple Graph Grammers Mapping

Fig. 2. EER Metamodel

Entity and Relationship can both contain Attribute. The Entity objects may be weak entity, that is, they need keys from other entities to establish their uniqueness. An entity can be a super type of another entity set and can also be a subtype of another one in the generalization hierarchy (generalize association). The Attribute objects have a type of primitive data or other entities. There are two types of Attribute: identifiers and descriptors. Both identifiers and descriptors may consist of either a single attribute or some composite of Attributes. Some Attributes may be multivalued or derived from other attribute.

Entities associate with others through Relationship object. The connectivity property of Relationship expresses the number of each entity participating in the Relationship, which may have value of one-to-one, one-to-many or many-to-many. We use one-to-many connectivity to represent for many-to-one connectivity, with the replacement of the two entities in both sides. The degree of Relationship may be unary (recursive binary), binary, ternary or more(n-ary). Relationship among relationships is represented by Aggregation(aggregate association). The relationship among entities is treated as a higher-level of entities that can involve into relationship with other entities[8].

B. ORDB Model

Fig. 3 presents the given metamodel for the ORDB resulting from the transformation. In this paper we focus on the data type characteristics of the SQL:2003[6] OR metamodel; that is, the artifacts corresponding to the relational metamodel are not included. Fig. 3 shows three different kinds of Data Types: Constructed, Predefined and User Defined Types (UDTs). The Constructed types can be Composite Types and Reference Types. The Composite types can be Collection Types (Array and Multisets) and Row Types. While in collections, all the elements have the same type (thus the corresponding data type is defined for the collection concept). In rows, fields may have different types (thus they have a connection with Attribute).

The UDTs can be Structured types and Distinct types (the latter are defined over a predefined data type). Structured types are composed of one or more Attributes and optional Methods. Each Attribute has one data type. Direct inheritance among Structured types or Reference types may occur.
C. Mapping EER to ORDB

After defining a metamodel for EER and ORDB graphs and mapping links between these metamodels, we have to define detailed TGG rules for each link. The specification of each rule describes how elements in a metamodel evolve into other ones.

1) Transformation of Entities

**Strong Entities** : For each strong entity in the EER model (with isWeak attribute is false), a structured type or a row type is added to the target model with the same name as the entity name. From the added structured type or row type, we can create a typed table or row table in the ORDB schema. Each of the entity’s attributes has a corresponding table attribute of the appropriate data type. Mandatory elements are established with NOT NULL constraints. The attribute with the “isPrimary” value is true and can either be added to the attribute of a created type to identify the object uniquely, or use a system-generated identifier.

**Weak Entities** : Weak entities are also represented using structured types or row types; the data type name is the same as the entity’s name, and it has a set of columns that correspond to the entity’s attributes in much the same way that a strong entity’s created type does. The “has a” weak entity relationship in the source model is replaced by “references” relationship in the target model. This means that the columns defining the primary key of the related strong entity are added to the columns defining the weak entity’s attributes, and a reference relationship between the two objects is created.

2) Transformation of Attributes

**Multivalued Attributes** : There are different ways to represent collection types in ORDB, depend on whether the information we wish to record requires ordering or duplicating or not. The Array provides a semantic ordering of the elements, where as a Multiset or Nested table contains elements that can be duplicated and do not require ordering. The entity is mapped to a Structured or Row type, and its multivalued attribute is mapped to the Collection type.

**Composed Attributes** : The Composed attributes which are included in the entity, are mapped to the Structured or Row Type, but we will not create an associated Typed table because the new Structured type will be included as an attribute of the Structured type that maps the entity in which the composed attribute was found. Also all the component properties of the Composed attribute are mapped to properties of the created Structured type (or Row type).

**Derived Attributes** : The value for this type of attribute can be derived from the values of other related attributes or entities. To map this kind of attribute to ORDB, we just add a method or trigger in a Structured type that maps the entity in which the attribute is included. The value of derived attribute is not stored but is computed when required.

3) Transformation of Relationship

There are various degrees of relationships, such as unary (recursive binary), binary, ternary, and n-ary relationship. For each kind of relationship degree, there are three kinds of connectivity between entities : one-to-one, one-to-many(or many-to-one), and many-to-many.

For one-to-one and one-to-many relationship, a Structured type is created with embedded foreign key of the parent entity. This Structured type is created for the entity in the “many”(child) side in a one-to-many relationship, and for one of the two entities in a one-to-one relationship.

For a many-to-many relationship, a new Structured object is created from the relationship, and a Typed table is derived based on this structured object, containing the references to all the Constructed objects in the relationship. This new Structured object may also contains attributes of the original relationship.

IV. CONCLUSION

In this paper, we have completely defined the transformation from the EER model to the object-relational schema objects related to the SQL:2003 standard. The main contribution of this thesis is the formalization, by means of definitions and mapping rules in triple graph grammars, of the steps involved in the design of object-relational databases.

The formalization of the model transformation can be used to detect errors and inconsistencies in the early stages of developing and increasing the quality of the built model, as well as the subsequent code generated from them. By using triple graph grammars, it is possible to develop a framework to allow the automatic generation of an OR schema design. To develop an automatic ORDB transformation tool from EER, we need to define more detailed rule, as well as consider other aspects of ORDB besides the mapping of data type.

REFERENCES