ERDraw: An XML-based ER-diagram Drawing and Translation Tool

Shuyun Xu
Wayne State University
Detroit, MI 48202
sxu@cs.wayne.edu

Yu Li
Micro Research Ins. Inc
Chicago, IL 60532
yuli@visviva.com

Shiyong Lu
Wayne State University
Detroit, MI 48202
shiyong@cs.wayne.edu

Abstract
The Entity-Relationship (ER) model is one of the most popular methodologies for designing relational databases. Several commercial products have been developed to support drawing ER-diagrams in a graphical fashion. Their architectures and implementation details, however, are not available in public. Inspired by these products and recent developments in XML technology and semantic drawing framework Drawlet, we have developed an educational prototype ERDraw that supports drawing ER-diagrams visually and translating them to relational database schemas automatically. In this paper, we describe the architecture of ERDraw and its implementation details to illustrate how such a tool can be developed.

Keywords: ER model, XML, database design.

1. Introduction
Entity-Relationship (ER) model is one of the most popular methodologies for designing relational databases. It contains three essential components, entity sets, relationship sets and integrity constraints. Entity sets model the objects that are involved in an enterprise such as students, professors, and courses in a university. Relationship sets model the associations among these objects. For example, students take courses, and professors advise students. Integrity constraints specify the application-dependent correctness constraints over entities and relationships. For example, a professor must teach at least one course each semester. An important advantage of the ER methodology is that, a designer can focus on conceptual modeling of an enterprise, making decisions on what entity sets, relationship sets and constraints to use. Once an ER-diagram is created, one can convert it to a relational database schema based on the set of standard translation rules that are available in most database textbooks [10, 11].

The ER model was introduced as a tool for data modeling by P. Chen in 1976 [1]. Together with its different variants [3, 4], it has been used successfully in data modeling and database design in more than two decades. Based on the notion of ER modeling which was generalized to conceptual modeling afterwards, an international conference was initiated in 1979 and has been held annually [5] thereafter. Several commercial products have been developed to support drawing ER-diagrams in a graphical fashion. They include Embarcadero Tech’s ERStudio [6], Microsoft’s Visio [7], and Dia [8] that are developed by A. Larsson, et al.

Inspired by these products and with the support from recent developments in XML technology [12] and semantic drawing framework Drawlet [9], we have developed an educational prototype ERDraw that supports drawing ER-diagrams visually and translating them to relational database schema automatically. In this paper, we describe the architecture of ERDraw and its implementation details to illustrate how such a tool can be developed. ERDraw support the following set of attractive features in one framework seamlessly:

- **Representing ER-diagrams in XML.** As addressed by P. Chen, the ER model has a close relationship with XML and the web [2]. To support this trend, we have defined ERML, a language based on XML to convert an ER-diagram into ERML format and vice versa. In this paper, we describe the DTD of ERML, hoping that it will be used as a public XML exchange format for ER-diagrams between different ER tools.

- **Drawing ER-diagrams semantically.** ERDraw recognizes ER-diagram components such as entity sets and relationship sets semantically. For example, when one drags and drops an entity set, line connections between this entity set and other components will be maintained automatically. This greatly
facilitates the designer to change the layout of an ER-diagram.

- **Validity verification.** ERDraw supports the verification of the validity of an ER-diagram and ensures that only well-formed ER-diagrams can be exported in ERML files or translated to relational database schemas.

- **Automatic translation to relational models.** After an ER-diagram is created with ERDraw, one can simply translate the ER-diagram to a relational database schema (i.e., a set of CREATE TABLE statements) automatically. In particular, integrity constraints such as total participation constraints and key constraints will be implemented in the resulting schema automatically.

- **Platform independence.** ERDraw is implemented in Java, therefore it can be executed in all environments where a Java virtual machine is available.

2. **System architecture**

The overall system architecture of ERDraw is illustrated in Figure 1. Basically, it consists of the following five major modules.

- **Graphical User Interface.** It provides a user-friendly graphical user interface to support the interaction between ER-diagram designers and ERDraw. A snapshot of this interface is shown in Figure 3.

- **ER-diagram Objects.** It consists of a set of raw graphical objects such as rectangles, diamonds and ellipses. The whole diagram, however, is not necessarily a valid ER-diagram. This allows one to store an incomplete ER-diagram for future continuing work.

- **ER Semantic Object Model.** It is the key internal data structure that represents the complete semantic information of an ER-diagram. An ER Semantic Object Model is created either from ER-diagram Objects or from an ERML file.

- **ERML Object Model.** It is the intermediate data structure that maps an ER Semantic Object Model to an ERML file. It is used to generate an ERML file, or is constructed from an ERML file.

- **Relational Object Model.** It is the intermediate data structure that maps an ER Semantic Object Model to a relational database schema.

3. **Implementation**

To support interoperability between different ER-diagram tools, we have defined ERML, a language based on XML to convert an ER-diagram into an ERML file and vice versa. The structure of ERML documents is described by the XML Document Type Descriptor (DTD) shown in Figure 2.

To translate an ER diagram to a relational database schema, ERDraw follows the following rules:

a) If there is no constraint between entity set $A$ and its participating relationship set $R$, then implement $A$ as a separate table.

b) If entity set $A$ has a key constraint but no total participation constraint over relationship set $R$, then implement $A$ and $R$ as a combined table. $A$’ primary key
becomes the primary key of the combined table.

c) If entity set \( A \) has both a key constraint and a total participation constraint over relationship set \( R \), the implementation is similar to b). In addition, attributes in \( R \) should be declared as NOT NULL.

d) If entity set \( A \) has a total participation constraint but no key constraint over relationship set \( R \), an SQL assertion will be generated to enforce the constraint.

e) Finally, all relationship sets, unless they are processed by rule b) or c) will be implemented as a separate table, with the union of all roles as the primary key, and each role as a foreign key referencing the corresponding participating entity set.

4. An example

To illustrate the translation process, consider the ER-diagram for a hospital management system as shown in Figure 3. Due to space limit, we show only part of the generated database schema to illustrate each of the above cases.

Doctor will be implemented as a separate table based on rule a).

CREATE TABLE Doctor (  
  licenseno VARCHAR(50),  
  telephone VARCHAR(50),  
  specialty VARCHAR(50),
  name VARCHAR(50),  
  location VARCHAR(50),  
  primary_key(licenseno))

Patient and Cover are combined into one table according to rule b).

CREATE TABLE Patient (  
  SSN VARCHAR(50),  
  name VARCHAR(50),  
  birthday VARCHAR(50),  
  gender VARCHAR(50),  
  Policy_id VARCHAR(50),  
  PRIMARY KEY ( SSN ),  
  FOREIGN KEY (Policy_id ) REFERENCES Policy ( id )
)

Policy and Belong are combined into one table according to rule c). Notice the NOT NULL constraint for attribute InsComp_name.

CREATE TABLE Policy (  
  id VARCHAR(50),  
  to VARCHAR(50),  
  from VARCHAR(50),  
  InsComp_name VARCHAR(50) NOT NULL,  
  PRIMARY KEY ( id ),  
  FOREIGN KEY (InsComp_name ) REFERENCES InsComp ( name )
)

The following assertion implements the total participation between Policy and Cover based on rule d).

CREATE ASSERTION AtLeastOnePatient  
CHECK NOT EXISTS(  
  SELECT *  
  FROM Policy  
  WHERE NOT EXISTS(  
    SELECT *  
    FROM Patient  
    WHERE Patient.Policy_id=Policy.id  
  )
)

Based on rule e), Accept is implemented as a separate table.

CREATE TABLE Accept (  
  InsComp_name VARCHAR(50),  
  Doctor_licenseno VARCHAR(50),  
  PRIMARY KEY ( InsComp_name,  
  Doctor_licenseno ),  
  FOREIGN KEY (InsComp_name ) REFERENCES InsComp ( name ),  
  FOREIGN KEY (Doctor_licenseno) REFERENCES Doctor ( licenseno )
)
5. Conclusion

We have developed an XML-based ER-diagram and translation tool for educational purposes. This tool incorporates OO technology, XML and the relational database theory. ERML facilitates the interoperability of various ER tools. The semantic drawing tool and verification process guarantee the semantic correctness of ER-diagrams. The automatic translation from ER-diagrams to relational data schemas is practically useful.

We are currently extending ERDraw to support other features such as ISA relationship sets. Other future work includes the design of a language to specify functional dependencies [10, 11] and explore the possibility of automatic database normalization based on these dependencies.

References


