E-R Modeler: A Database Modeling Toolkit for Eclipse

Song Zhou, Chuanxi Xu, Hui Wu, Jing Zhang, Yuehua Lin, Juanqin Wang, Jeff Gray, Barrett Bryant
Department of Computer and Information Sciences
University of Alabama at Birmingham
Birmingham, AL 35294 USA
{zhous, xuc, wuh, zhangj, liny, wangjq, gray, bryant}@cis.uab.edu

ABSTRACT

Eclipse is a Java integrated development environment (IDE) and tool integration platform that offers numerous extension points for customization through a plug-in architecture. This paper describes the design of an Eclipse plug-in called E-R Modeler. As a database design toolkit, the E-R Modeler provides an E-R (Entity-Relationship) diagram development environment that supports XML and DDL (Database Definition Language) generation tools. It also provides database connection and schema creation capabilities. As an Eclipse plug-In, it offers multiple extension points for other applications to build upon.

Categories and Subject Descriptors

D.2.6 [**Software Engineering**]: Programming Environments – *Integrated Environments.*

Keywords

Eclipse, Plug-In, E-R diagram, Data Modeling, Design Patterns.

1. INTRODUCTION

Eclipse is a development platform for constructing customized integrated development environments (IDEs) that can be used to create applications as diverse as web sites, embedded Java programs, C++ programs, and Enterprise JavaBeans (EJBs) [6]. The core of Eclipse is composed of: a Java development environment, a tool integration platform, and an open source community. Eclipse provides a Plug-In Development Environment (PDE) that extends the Java Development Tools (JDT). From the plug-in viewpoint, Eclipse is a collection of "places-to-plug-things-into" (extension points) and "things-to-plug-in" (extensions) [2]. An Eclipse plug-in has the ability to integrate with other plug-ins to extend functionality.

Following the Open-Closed principle [4] (i.e., "Open for extension, but closed for modification"), Eclipse is an extremely extensible platform and can be extended in different levels and aspects. Menus can be extended with new menu items, perspectives can be extended with new views, and a workbench can be extended with new perspectives. The basic building blocks of Eclipse are plug-ins, which can be as simple as an online help file or as complex as a full-fledged Java development environment. Starting from a primitive bootstrap plug-in, all the other functionality and features are implemented as layered plug-ins. One plug-in can extend the functionality of another plug-in by implementing the interface defined by the extension point of the other. Developers can provide new functionality to Eclipse by extending several existing extension points, and, at the same time, provide further development opportunities for others by publicizing new extension points

From a user's point of view, Eclipse provides a consistent GUI among different features. The menus, toolbars, preference, online help of different plug-ins are integrated together. Eclipse also provides a standard framework for implementing many high-level GUI components, like viewers, wizards, and dialogs. These facilities relieve a developer from rebuilding low-level infrastructure.

The E-R Modeler is implemented as an Eclipse plug-in that provides an E-R (Entity-Relationship) diagram [1] development environment with XML and DDL (Data Definition Language) generation tools. The plug-in also provides database connection and schema creation capabilities from within Eclipse. The primary motivation for constructing the plug-in is to offer tool support for an undergraduate database course. During the Spring 2004 semester, the tool will be introduced into the undergraduate database course at UAB on a trial basis. It will be used as a pedagogical aid toward teaching students the primary concepts of database schema development with a user-friendly modeling tool.

The first version of E-R Modeler contains a powerful E-R diagramming tool, a navigator and property view for visualization, and provides manipulation of the E-R models. Additional features have been implemented such as capabilities for DDL generation, database connection, and schema generation (please see Figure 1). In using the tool, there are several views that are provided of the modeled schema. The outline view is used to display the hierarchy of the entities, relationships, and attributes. The property view is used to display and set some basic properties (e.g., name, location and multiplicity) of each entity set and relationship. A pop-up property dialog is used to insert/delete attributes of an entity set. All the views and E-R model editor are always synchronized; i.e., changes made in one part are reflected in other parts immediately.

The paper is organized as follows. The next section introduces the functional modules that are contained in E-R Modeler. The detailed implementation challenges are described in Section 3. A conclusion contains a description of related and future work.

ACMSE '04, April 2-3, 2004, Huntsville, Alabama, USA.

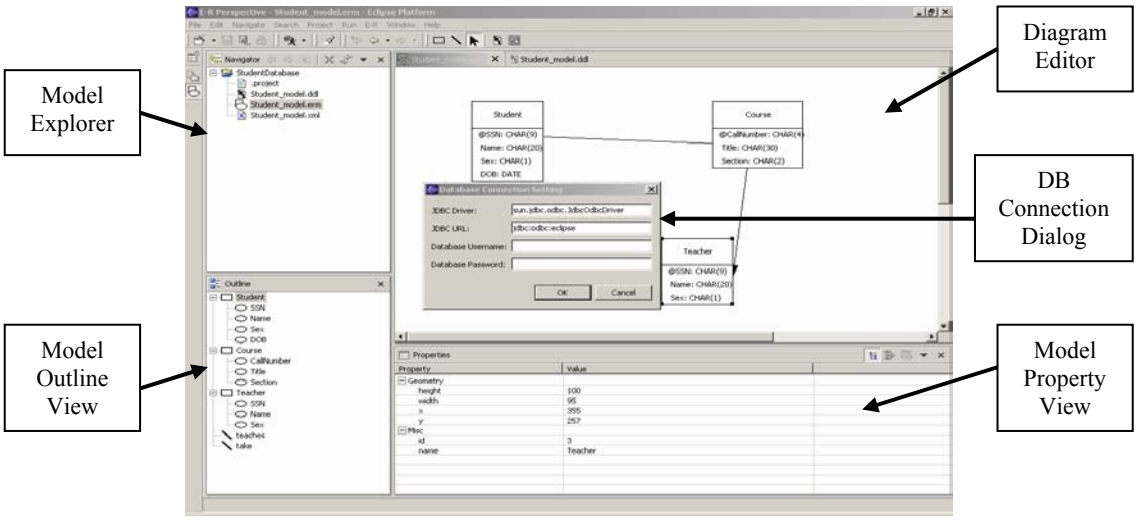

Figure 1 E-R Modeler GUI View

2. SYSTEM ARTHITECTURE

E-R Modeler not only helps users design a logical data model that captures application requirements, but also supports the design of the physical data model for the target server. This enables users to forward engineer the physical data model and automatically generate physical database structures to the system catalog. In the next generation, E-R Modeler will support reverse engineering of existing databases and provide both a physical and logical data model so that users can maintain an existing database, or migrate from the current target server to a different one (see Figure 2).

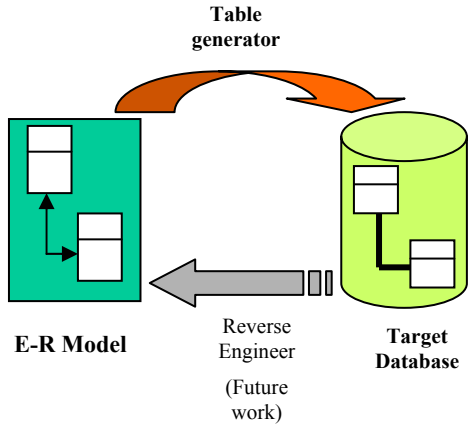

Figure 2 E-R Modeler Functionality

2.1 Primary function modules

Figure 3 illustrates the E-R Modeler architecture. There are two independent plug-ins: the E-R Model Editor plug-in containing a diagram editor to create and edit E-R models, and the Generator plug-in for DDL generation, database connection and table creation.

2.1.1 Diagram Editor

The Diagram Editor is the kernel of the entire project. It supports the construction of three Entity-Relationship objects: entity, relationship, and attribute. The look and feel of an entity resembles that of an object in a UML diagram, being composed of attributes in its body. A relationship between entities has the typical relationship representation in the E-R diagram.

The entity set is displayed as a rectangular box. It has two display strategies: one only shows the name of an entity set, while the other shows both the names and the attributes of an entity set. When using the latter display strategy, the rectangle representing the entity set can automatically adjust its size after inserting/removing the attributes. The entity set box can be moved arbitrarily, followed immediately by the moving of its associated connection points. When one entity set is deleted, all the associated relationships are cascade deleted simultaneously.

A relationship is created by defining two connection points, each of which must be associated with one entity set. A connection point can only be defined by clicking along the border of an entity set and can also be moved along the border. Dangling relationships are always forbidden. A relationship can be deleted independently. In addition, the multiplicity of a relationship is multiple-to-multiple by default, but can be changed to one-to-multiple and one-to-one through the property view, along which the shape of the relationship is changed to reflect the multiplicity.

A list is maintained for all of the ER objects. Each entity object has two categories of properties: geometry and relational. An entity has position, size, and decoration points as its geometry properties while ID, name, and attributes are relational. A relationship has end points as its geometry property while connected entities are relational.

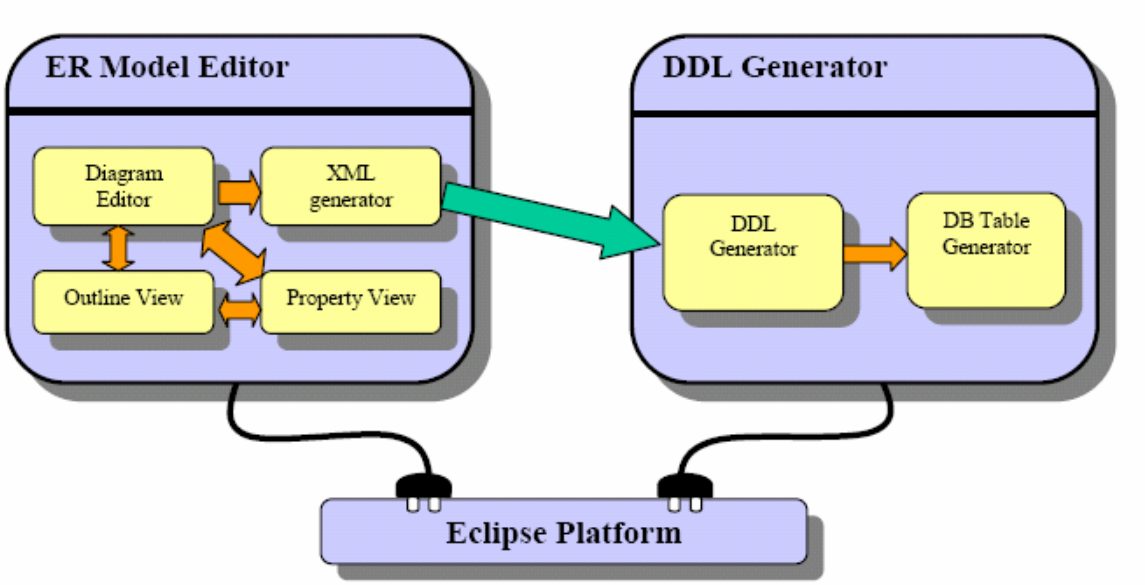

Figure 3. E-R Modeler Architecture

2.1.2 Outline View

The outline view is used to display the hierarchy of the entities, relationships, and attributes. The only widget in the view is a `TreeViewer`, the input to which is the object list from the diagram editor. The `ContentProvider` and `LabelProvider` to the `TreeViewer` could be customized to reflect the personalities of E-R Modeler.

2.1.3 Property View/Dialog

The property view is used to display some basic information of each entity and relationship. In addition, users can change the name of the selected entity set and multiplicity of any relationship. The property view is always synchronized with the graphical editor and other views. Changes made in one of them are reflected in all the others immediately.

The property dialog is used to modify the attributes of an entity set, which can designate not only the name and type of an attribute but also the primary key of an entity set. The data types available are the typical data types in JDBC.

2.1.4 State Persistence

By default, E-R Modeler serializes the E-R objects on the Save command. During the serialization process, both the relational and geometry information is saved. Consequently, the persisted E-R model can be reloaded completely from the saved file. All information but the geometry properties of the E-R objects can also be exported to a XML file, which can be used to generate DDLs or utilized by other plug-ins providing additional processing of the database model.

2.1.5 DDL Generator

During DDL generation, E-R Modeler loads the saved XML file as an XML DOM tree. The DOM tree is traversed to generate DDL based on the underlying database. Considering JDBC support for different database systems, the JDBC data type was chosen as the default DDL data type. The generated DDL includes primary key definition and foreign key definition besides table definition. An SQL editor is embedded within E-R Modeler so that the user can preview and edit the generated DDL file.

2.1.6 Database Table Generator

To generate database tables, E-R Modeler can connect to a particular DBMS (e.g., Oracle, Access, SQL Server, SQL Plus) through a JDBC driver, which can map general JDBC data types used by the generated DDL to specific data types of the corresponding database. After connecting to the database, E-R Modeler passes the translated DDL commands to the database server. The server then executes the DDL commands to generate tables and other database artifacts. The DBMS and database location is specified in the form of a JDBC driver and URL, which are supplied by the user through a pop-up dialog.

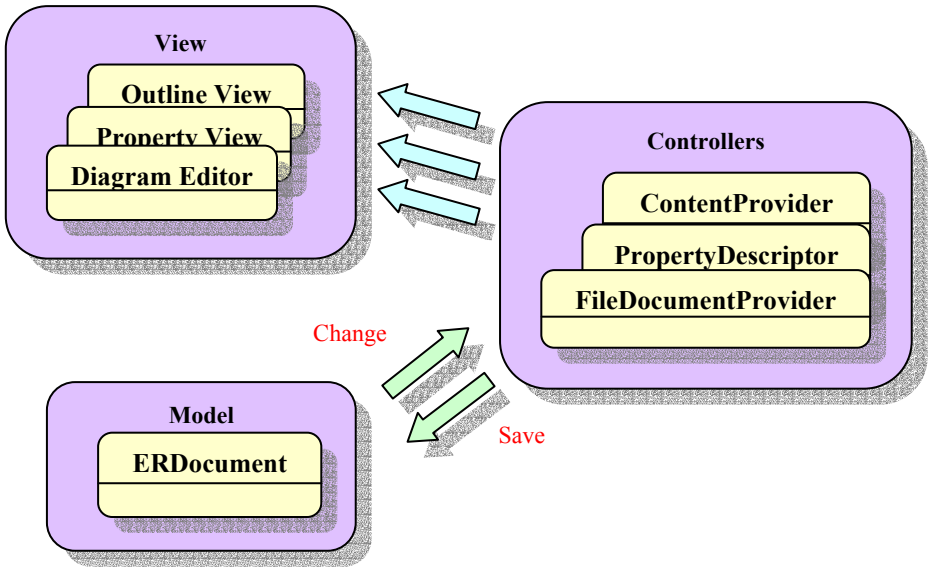

Figure 4. Model-View-Controller (MVC) Architecture

2.2 Extension points used

This section describes numerous extensions that are used in E-R Modeler.

2.2.1 newWizards – new project and file types

The newWizards extension point is extended to support the creation of a new E-R model project and a new E-R model file.

2.2.2 perspectives – framework

The E-R perspective is activated automatically when an E-R model project is created or opened. The E-R perspective is composed of the following parts: an E-R model editor, a navigator view, an outline view, and a property-sheet view.

2.2.3 editors – the diagram, XML and SQL editors

The extension point "org.eclipse.editors" is extended for building the diagram editor, the XML editor and the SQL editor. The diagram editor is associated with file type ".erm" to provide a drawing canvas for the E-R objects. The actions invoked by the menu and toolbar are defined in the contributor class. The XML editor is shipped with Eclipse through PDE wizards. The parsing of the XML file is implemented by defining text rules, scanners and tokens. The SQL editor is used to view the generated DDLs. Data types and keyword definitions are compatible with those defined by JDBC.

2.2.4 actionSets – toolbar actions

Action sets are associated with the E-R model editor appearing automatically when the editor is activated. The action set is used when creating a model, selecting an E-R element, generating DDLs, and uploading the database.

2.2.5 propertyPages – property view

The propertyPages extension point is extended to manage the attributes of an entity. Users can add, delete or modify the attributes in the property page. Each attribute is defined by its name, type, and optionally length, and scale.

2.2.6 preferencePages – XML editor and coloring

The extension point org.eclipse.ui.preferencePages is used to provide a preference page for changing the color of the characters in the XML editor.

2.2.7 toc – online help

The extension point org.eclipse.help.toc is used to create the online help of the E-R Model editor plug-in.

3. IMPLEMENTATION CHALLENGES

In developing this E-R modeling plug-in, there were numerous challenges encountered while building the graphical editor. In particular, the synchronization of the textual and graphical views, as well as the persistence of the run-time canvas information, proved challenging. These issues are further described in this section.

3.1 Constructing a visual editor from scratch

The E-R Modeler has several special requirements (e.g., rich graphical elements to represent diverse database objects, customizability of the graphical elements such as changing position and size). Several existing plug-ins failed to meet the requirements of four core tools within E-R Modeler:

3.1.1 Creation tool

The creation tool is responsible for the initiation of new E-R elements into a model. Numerous design patterns [3] were applied, such as the Builder pattern to abstract the creation process for different E-R elements. For example, creating an entity needs only one graphical point, while creating a relationship needs two points. Strategies were also used in order to define different creation processes for the same element. For instance, users can use either single-click or drag-and-drop operations to create a new entity. All of the creation tools are themselves generated by an Abstract Factory, which is also a Singleton.

3.1.2 Selection tool

The selection tool is responsible for the selection of different E-R elements. In each E-R element class, there is a Strategy method to determine whether the element is selected given the current cursor position. For an entity, it needs to consider whether the cursor falls inside the rectangle denoting the entity. But, selecting associations can be complex. Several vector computations are needed to compute the distance between a point and a line segment, which is much harder than computing the distance between a point and a line.

3.1.3 Drawing tool

The drawing tool is responsible for rendering the different E-R elements. The modeler has implemented two strategies to accommodate the variations in displaying an entity. One only shows a fixed sized rectangle with the name enclosed within the entity. Another strategy displays all the attributes of an entity. The display of a relationship is relatively easy. The task is to determine how an arrow is to be drawn along the line. For arrowed associations, vector computations (*scaling, rotation and shift*) are also needed to compute the three points that define a triangle. Based on our algorithm, both the size and the angle of the arrow are configurable. In addition to highlighting the selection of an E-R element, a Decorator is used to decorate the selected E-R element with additional highlight points. All of the drawing tools are created by a Factory Method of the relevant E-R element.

3.1.4 Drag-and-Drop tool

The drag-and-drop tool is responsible for handling drag-and-drop actions happening inside the editor. Different elements have different drag-and-drop behaviors. There is another Factory Method in each E-R element, which is responsible for returning the current active drag-and-drop tool for the element. For an entity, a drag-and-drop tool can be used to generate a *moving* action. For a relationship, a drag-and-drop tool can be used for *resizing* and *reconnecting*.

3.2 Synchronizing between editor and views

Synchronization is implemented using the MVC (Model-View-Controller) architecture (please see Figure 4). The model can be altered in the graphical editor and the outline view. All the changes are sent to the model, which, after updating itself, will broadcast the change event to the editor and all the views. To implement this Observer pattern, the E-R document (i.e., the object that stores all the data) is a `FileDocumentProvider` of the Diagram Editor, a `ContentProvider` of the Outline View and a `PropertyDescriptor` of the Property View.

Additionally, all the views and editors must receive the selection change event so that they can activate necessary actions. For example, the property view must always display the property of the selected E-R element. Selections can be made in both the editor and the outline view. To exchange the selection change event, the editor and Outline view will be a `SelectionListener` of each other, while the property view has to listen to all of the selection events that are generated.

3.3 Persisting the run-time canvas

It is necessary to be able to override the object read/write methods during serialization and restore the run-time information separately. Object serialization is used to save the state of E-R objects (i.e. all ER objects - entity, attribute, relationship are implementations of `java.io.Serializable`). Special care was taken for the SWT member fields that are not serializable. The state can be restored when reloading the saved file. All the geometry information is kept intact during the restoration process.

In addition, the logical information can be saved in XML format, which can be further processed by a DDL generator or other third parties interested in further development. To make the plug-in work on both JDK 1.3 and JDK 1.4, a thread context Classloader [5] is used so that the XML parser factory classes defined in the Java extension package can instantiate the Xerces XML parser implementation shipped with Eclipse.

4. CONCLUSION

The current version of E-R Modeler contains a powerful E-R diagramming tool, a navigator and property view for visualization, and provides manipulation of the E-R models. Additional features have been implemented such as capabilities for schema generation and database connection. E-R Modeler not only helps users design a logical data model that captures business requirements, but it also supports the design of the physical data model for the target server. This enables users to forward engineer the physical data model and automatically generate physical database structures to the system catalog.

4.1 Related work

Several database-related design tools already exist, including commercial tools such as AllFusion ERwin Data Modeler (http://www3.ca.com/Solutions/Product.asp?ID=260). However, these commercial tools are generally stand-alone applications that need to be deployed and configured separately from existing programming IDEs like Eclipse. Meanwhile, existing Eclipse plug-ins such as EasySQL and JFaceDbc only provide limited functionality for database design. E-R Modeler improves the modeling capability of Eclipse and provides database access ability from within Eclipse.

4.2 Future work

Although there is a core base of functionality within the tool, there are many additional capabilities that need to be added before the tool is mature enough to be adopted as a key component of a database course. There are numerous extensions to E-R Modeler that we are currently developing. Each extension supports the maturation of the tool such that its applicability as a teaching tool is enhanced. The following list represents our current and future work on E-R Modeler:

- Implement support for reverse engineering: We propose to extend E-R Modeler to provide support for reverse engineering of existing databases into physical and logical data models. To accomplish this task, the database meta-data will be queried and used to construct the model. This would permit maintenance of an existing database that was not previously modeled. In addition, E-R Modeler will automate model and database synchronization by providing

a capability to compare the model with the actual database schema from the server. Differences between the model and actual schema will be displayed and analyzed. The long-term vision for reverse engineering is the capability to visually evolve a database schema by dragging and dropping database instances and automating migration to a new schema.

- Automatic layout support: The layout manager for the visualization of modeling entities needs to be improved such that aesthetic appearance is preserved as model entities are moved around the canvas.
- Support for additional file types: E-R modeler will be extended such that it can export and import different data formats, such as XML Metadata Interchange (XMI).
- Enrich the supported ER object types: Currently, E-R Modeler only supports limited types of database objects. Additional support for many other diverse Entity-Relationship object types will be provided (e.g., weak entity and subclass types).
- Add extension-points for other plug-ins to build on: E-R Modeler will provide several additional extension points such that other Eclipse applications can build on top of E-R Modeler. The possible extension points may be the diagram editor drawing tools, and access to the underlying model data structure to support extensions to the DDL generator. The internal structure of the E-R model will be exposed through an interface such that other generator tools can be created for various database vendor implementations.

5. AVAILABILITY

The complete source code and other artifacts related to the project are available at:

http://www.cis.uab.edu/info/Eclipse/ERModeler/

6. REFERENCES

[1] Chen, P. P., "The Entity-Relationship Model: Toward a Unified View of Data", *ACM Transactions on Database Systems*, March 1976, pp. 1-36.

[2] Gamma, E., and Kent, B., *Contributing to Eclipse: Principles, Patterns, and Plug-Ins,* Addison-Wesley, 2004.

[3] Gamma, E., Helm, R.., Johnson R., and Vlissides, J. H., *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley Press, 1995.

[4] Meyer, B., *Object Oriented Software Construction*, 2nd ed., Prentice Hall, 1997.

[5] Roubtsov, V., "Find a way out of the ClassLoader maze," *Javaworld*, June 2003.

[6] Shavor, S., D'Anjou, J., Fairbrother, S., Kehn, D., Kellerman J. and McCarthy, P., *The Java Developer's Guide to Eclipse*, Addison-Wesley Press, 2003.