



# ENHANCING AN END USER DEVELOPMENT IN DATABASE DESIGN USING ENTITY RELATIONSHIP DIAGRAM MAPPER

<sup>1</sup>MOHAMMED A. OTAIR, <sup>2</sup>AHMAD M. ODAT

<sup>1</sup>Assoc. Prof., Faculty of Computer Sciences and Informatics, Amman Arab University, Amman-Jordan

<sup>2</sup>Assoc. Prof., Faculty of Science and Information Technology, Irbid National University, Irbid-Jordan

E-mail: <sup>1</sup>otair@aau.edu.jo, <sup>2</sup>aodat@yahoo.com

## ABSTRACT

Today, Database systems plays a large part of our everyday lives –especially with the importance of the growing concept of database and storing information and keeping the information in a way secured in the database. Entity Relationship Diagram (ERD) is one of the popular and important concepts in database modeling. The design of the correct database (or good design) depends on the correct ERD that presents the requirements of the required system to the customer. In this paper, an Entity Relationship Diagram Mapper (ERDM) is built to provide developers of database as an End User Development (EUD) tool that enables them to draw any Entity Relationship Diagram then the system automatically map or translate these diagrams into their corresponding relations (or tables). The constructed database by the system has a good design or normalized relations. This system will be useful when the developer or user does not know many things about the mapping process of an Entity Relationship Diagram into Database schemas (its logical design). In order to test the developed system, sample of 100 ERDs collected from different sources were mapped using the system. The experiments showed that the accuracy of the system was 100%, without even any single error.

**KEYWORDS:** *Database Design, Normalization, Entity Relationship Diagram, End User Development, Database Schema.*

## 1. INTRODUCTION

Since last three decades, end-user development (EUD) and outsourcing are the most growing and common techniques that used as information resources for organizations. End users are collaborating in the tasks achieved normally by developers actively and many of these tasks are delivered to them [3]. In most researches, End User Development (EUD) and End User Computing (EUC) are considered as synonymous concepts, because the roles of the users and developers are being too closed. According to [5], End User Development (EUD) can be defined as a set application tools that assist software systems users as amateurish developers without requiring them to understand complicated of programming languages. The impact of end user development on organizations is ongoing. So, the researchers and students in the domains of management should take in their considerations to study of how EUD and EUC influence the trends of their careers [7].

Spreadsheets as software tools are mostly used by the developers or end users, despite many tools are available such as marketing and accounting applications [3]. Unfortunately, the difficulty task of database design according to the casual database developers or users prevents them to be active player in the end user development field [2, 4, 6, 9]. However, more experienced end users seek to gain benefits of sophisticated database products to improve their databases, but many of them do not capture the skills of data modeling. Hsiang et al. in their research [3] developed EUD based on 5C technique (Count, Comply, Compare, Consolidate, and Convert), which is an easy normalization algorithm for bottom-up database design.

The aim of this paper is to develop a computer aid system called Entity Relationship Diagram Mapper, which helps users who are working in the field of the database design and they cannot or make mistakes in the converting of the ERDs into good logical design or tables stored in the database. So, the system allows the user (as a Database designer or even as an end-user) to draw the ERDs



then the system will automatically convert them into tables stored in a database.

ERDM is a full package, started by drawing the conceptual design and ended by design a relational DB schema as code. This solution model generates an SQL statements, especially DDL which related to create statement, like create database and create tables, with preserving the normalization. Other tools isolated between conceptual design and logical design. To convert manually the conceptual design into logical design, you should to have a good knowledge about normalization otherwise DB design will be very poor.

ERDM is simple and easy to use, all users can use this tool without needing to be professional in normalization, ERDM also save time and effort, once you finished correctly drawing ERD the software will automatically generate a suitable running code.

As a researchers, we know very well the importance of previous works (studies) paragraph, where this paragraph is considered one of the important basic ingredients of scientific paper, this paragraph absent because of the lack of scientific papers talking about this topic. Researchers have made a great effort to find the previous Studies, but all attempts failed.

The objectives of ERDM is to facilitate design a relational database schema for database designers and other users. ERDM also will help the DB designers to obtain correct SQL statement 100% within a short time and a little efforts.

## 2. THE DATABASE DESIGN STEPS

Designing a database for any system is achieved via a series of development steps as follow:

### 2.1 Requirement Analysis

By user interview a user view can be defined where the system can have more than user view. By determining the user views of the major database users, all the requirements for new system will be considered. These views assist in construct of sophisticated database system assisting requirements to be divided into tractable subsystems. The requirements of the users of the new database system can be analyzed by the gathered information during the interviews. The gathered information could include a metadata and how it can be used.

### 2.2 Drawing The ERD Diagram (Conceptual Design)

After the requirements are collected, analyzed, and determined, then the system should be modelled using one of the most common used data models which called an Entity Relationship Diagram (ERD). It contains three main constructs: Attribute (six types: single or multi-valued, simple or composite, stored or derived), Entity (two types: strong or weak), and Relationship (two types: Identifying or relationship type). An ERD is a standard data model to represent relational databases via to design modes Conceptual and Logical. In the conceptual design, the real world can be represented by focusing on collecting the system requirements and convert them into diagram which represents the whole system. Whereas, the logical design mode tries to transform or map a conceptual ER diagram into a good designed (normalized) relations or tables. The transforming or mapping process is done using very well seven rules (will be discussed in sub-section 2.3).

### 2.3 Converting An ERD Into Logical Database

A relational database (tables) can be created using the following mapping rules on the ERD [1]:

#### 2.3.1 The Strong Entity

Every strong entity will be represented by a new table in the database. The primary key of this table is the indicated key in the ERD. The simple/single/stored attributes of the entity will form the columns (fields) of the table. At the other hand, composite attributes by taking the leaves (atomic) parts of the composite attributes. Multi-valued attribute is mapped into separate table with composite primary key consists of the multi-valued attribute(s) and the primary key from the original table.

#### 2.3.2 The Weak Entity

As the strong entities, each weak entity is mapped to new table. The attributes of the weak entity treated exactly as the strong except that the primary key in the weak table is always composite primary key which relates the weak entity to its owner via the partial key from the weak entity and the primary key from the owner entity.

#### 2.3.3 The Binary M:N Relationship

Many to many relationship maps to a new table with composite primary key from the two primary keys of the two owner entities that are being participated in the M:N relationship. The attributes

of the M:N relationship will be added to the new table of the relationship.

### 2.3.4 The Binary 1:N Relationship

This relationship is not represented as a new table. However, a copy of the primary key attribute(s) at the one side entity is placed into the table at the many side entity to be as a foreign key.

### 2.3.5 The Binary 1:1 Relationship

A copy of the primary key attribute(s) at the one side entity is placed into the table at the other one side entity to be as a foreign key (or vice versa). However, it is better to choose the primary key from the entity side that has partial participation with the relationship.

### 2.3.6 The Recursive Relationship

Create a shadow entity and convert the recursive relationship into a binary relationship. After that, the rules of mapping binary relationships can be applied. At the end, apply the binary mapping rules, and then one of the redundant tables (the table with lesser attributes) will be removed.

### 2.3.7 The N-ARY Relationship

Create a new table to represent the n-ary relationship. The primary key of the created table will be composite of the primary keys of all entities that participate in the relationship.

## 2.4 Creating The Database

After applying the converting rules on the diagrams, then the database using several DBMSs can be created by the system automatically. The converting rules are listed in the next section in details.

## 3. THE DESIGN OF ENTITY RELATIONSHIP DIAGRAM MAPPER

The system developed in this paper is mainly consists of three main parts or it works based on three phases:

- A. Create new ERD: that involves specifying (drawing) the entities and its attributes then make the appropriate relations between these entities, or this phase may be open an existing ERD that was saved before.
- B. Convert the ERD into Tables: the system then automatically converts this ERD into the equivalent logical design of the tables using the well knows ERD mapping rules.
- C. Create the database: involves creating the suitable database and save the logical designs of the tables. In other words, it

creates the corresponding physical design of that mapped logical design.

The interfaces of an ERD Mapper is developed using Adobe Flash Builder and Flex Builder. An Adobe Flex Builder is an open source application environment which assists the developers to facilely build their applications as a SWF file using an open source Flex framework and a scripting language called ActionScript for browser. Adobe Flash Builder offers built-in code editors for MXML (Macromedia eXtensible Markup Language) and ActionScript for modifying MXML applications [adobe.com/products/flex.html].

The ERD Mapper is divided into three packages as follow: blocks, converter, and db. The required classes are distributed along the three packages to achieve the objectives if the developed system. The main package is the blocks which consists of the following classes: Attributable, Attribute, AttributeDomain, CompositeAttribute, Element, Entity, ERDiagram, Relation, and RelationConnection. The second important of the system is converter with two classes: QueryGenerator and TableGenerator. The db package consists of DBColumn and DBTable classes. The following figure show the class diagram of the system.

*Figure 1 Class Diagram Of ERD Mapper  
(In Appendix A-2)*

The fragment code in Appendix B-1 represents the main important code of the system which generates the create tables commands based on the drawn EDR by the system.

## 4. THE IMPLEMENTATION OF ENTITY RELATIONSHIP DIAGRAM MAPPER

The system consists of three main interfaces which are as the following:

- 4.1 The main form: which provide the user an option to navigate and use all the components of the system. Figure 2 shows the start-up window of the developed system.

*Figure 2 Start-Up Page Of ERD Mapper  
(In Appendix A-3)*

The above numbers into the circles around the above figure refer to the main functions of the system:

- 1- Open the existing diagram.



- 2- Save the diagram for the first time or save any changes on it.
- 3- Create the database schemas by mapping the diagram in the panel based on the mapping seven rules.
- 4- Reload or refresh any changes done on the diagram.
- 5- Add a new entity to the diagram.
- 6- Add a new relationship to the diagram.
- 7- Edit the selected entity by add/delete attributes(s)
- 8- Delete the selected entity.
- 9- Edit the selected relationship.
- 10- Delete the selected relationship.

**4.2** Adding new entity: that allows the user to add new entity and specify the attributes of that entity. When the user click on the button referenced by number 1 in figure 2, then the create entity form will be opened as in figure 3. A user can type name the entity and then the attributes one by one by typing its name, type and domain

*Figure 3 Add New Entity  
(In Appendix A-4)*

After the below *add* button will be clicked, then the entity with its attributes will be created and it is automatically drawn on the drawing panel as shown the following figure.

**Figure 4** Append the New Entity to the Drawing Panel (in Appendix A-5)

The user can edit the name and attributes of an existing entity by doing the following functions:

- 1- Click *edit selected* entity button.
- 2- Open the edit entity window.
- 3- The user can make the following:
  - Change the name of the entity.
  - Remove an existing attribute.
  - Change the type and domain of an existing attribute.
  - Add new attributes.
- 4- Click the *save* button to *submit* changes.
- 5- Automatically make the equivalent changes for the determined entity on the drawing panel

In addition to create and edit entities, the user can delete an existing entity with its attributes and it will be automatically deleted from the drawing panel.

**4.3** Adding new relation: that allows the user to create a relationship between the chosen

entities (it could be unary, binary, ternary or more). By this form shown in figure 5, the user can do the following tasks:

- 1- Determine the relationship name and if it is identifying (relates weak with strong entities) or relationship type (relates strong with strong entities).
- 2- Select the entities that will be connected to this relationship, then the cardinalities of each entity with its participation (total/partial participation) can be determined.
- 3- Add the attributes one by one to the relationship (if the relationship has attribute(s)).

*Figure 5 Add New Relationship (In Appendix A-6)*

4- When the user clicks on *new* relation button, then the new relation and its attributes will automatically be drawn on the drawing panel as shown in the following figure.

*Figure 6 Append The Relationship To The Panel  
(In Appendix A-7)*

The user can make an editing on existing relations (edit the relationship name, entities and attributes) by doing the following tasks:

- 1- Select the relationship needed to be edited.
- 2- Click the edit selected relationship button.
- 3- The edit relation form will open, and then the user can make the following tasks:
  - Change the relationship name.
  - Remove an existing entity.
  - Remove an existing attribute.
  - Change the type and domain of an existing attribute.
  - Change the cardinality of an existing entity.
  - Add new attributes.
- 4- Click the *save* button to submit changes.
- 5- Automatically make the equivalent changes on the drawing panel.

As the editing, the user can delete an existing relationship and its attributes. Then, automatically delete the selected relation from the drawing panel.

**4.4** Create a database by mapping the drawn diagram. When the user clicks on the create database button, the form with drop-down list is shown as in the following figure.

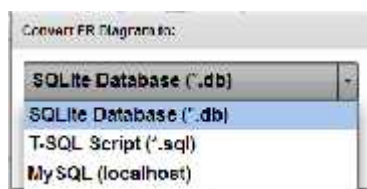


Figure 7 Converted Db Options

The system offers three options for the converted database: SQLite, T-SQL Script, and MySQL. SQLite is a relational database management system contained in a C programming library [8], Transact-SQL (T-SQL) is an expansion to SQL owned by Sybase and Microsoft to generate data scripts in an executable and readable T-SQL format, and MySQL is a well-known open source SQL relational database management system used for the web. So, when the user select the type of the database and clicks on the convert button (as in figure 8), then the diagram in the panel will be automatically converted to its corresponding database schema.



Figure 8 Convert ERD Into DB Schema

## 5. CONCLUSIONS

This paper presents a system called ERD Mapper which plays a role as a very effective tool for End User Development. The system developed in this paper achieves all the required steps for database design which involves drawing Entity Relational Diagrams, converting the ERDs to logical design, create the database and store the resulted tables in the database. So, it can support expert developers or even naïve developers without having to know much about the database design. The ERD Mapper can be improved to encompass an Enhanced ERD (E-ERD or E2RD). Finally, the developed system could be embedded into any database management system to enrich the use of these systems.

## REFERENCES:

[1] A. Silberschatz, H. Korth, and S. Sudarshan, Database System Concepts, 6th ed., McGraw-Hill, 2011.

- [2] C. Govindarajulu, "End users: Who are They", Communications of the ACM, vol. 46, no. 9, pp. 152-159., 2003.
- [3] H. Kung, H. Tung, and A. Gardiner, "Improving End-User Database Development Quality: A 5C Data Modeling Method", Issues in Information Systems, vol. IX, No. 2, pp. 305-312, 2008.
- [4] M. Taylor, E. Moynihan, & A. Wood-Harper, "End-user computing and information systems methodologies". Information Systems Journal, Vol. 8, No. 1, pp. 85-96, 1998.
- [5] N. Sharma and N. Deswal, "Web Engineering: End User Development (EUD) Of Web Applications", International Journal of Innovative Research in Technology, vol. 1, No. 5, pp. 725-730, 2014.
- [6] R. Nelson, & P. Todd, "Strategies for managing EUC on the Web", Journal of End User Computing. Vo. 11, No. 1, pp. 24-31, 1999.
- [7] S. Barker, "End User Computing and End User Development: Exploring Definitions for the 21st Century", Managing Worldwide Operations & Communications with Information Technology, pp. 249-252, 2007.
- [8] T. Dongare, A. Babar, & M. Nivangune, "Android Application for Ticket Reservation with GPS as Ticket Validation", International Journal of Emerging Research in Management & Technology, Vol. 3, No. 3, pp. 13-141, 2014.
- [9] T. Ouellette, "Giving users the keys to their Web content", Computerworld, pp. 66-67, 1999.



Appendix A-1 Class Diagram of ERD Mapper

ERDiagram
-entities -relations
+ERDiagram() +addEntity() : <unspecified> +getEntityByName() +entityExists() +addRelation() +getRelationByName() +removeRelation() +removeEntity()

Relation
-connections -identifying
+Relation() +addConnection() +removeConnection() +printConnections() +connectedToEntity() +connectionExists() +connectionCount()

DBTable
-pk -name -columns
+DBTable() +Name() +ColumnsCount() +PK() +addPK() +addColumn() +removeColumn() +getColumn()

Attribute
-type -domain -key -partial -multiValue -derived
+Attribute() +checkType() +isKey() +Domain() +Type() +isPartialKey() +isMultiValue() +isDerived() +isComposite()

AttributeTable
-attributes
+AttributeTable() +getCompositeAttributes() +removeAttributeByName() +removeAttribute() +addAttribute()

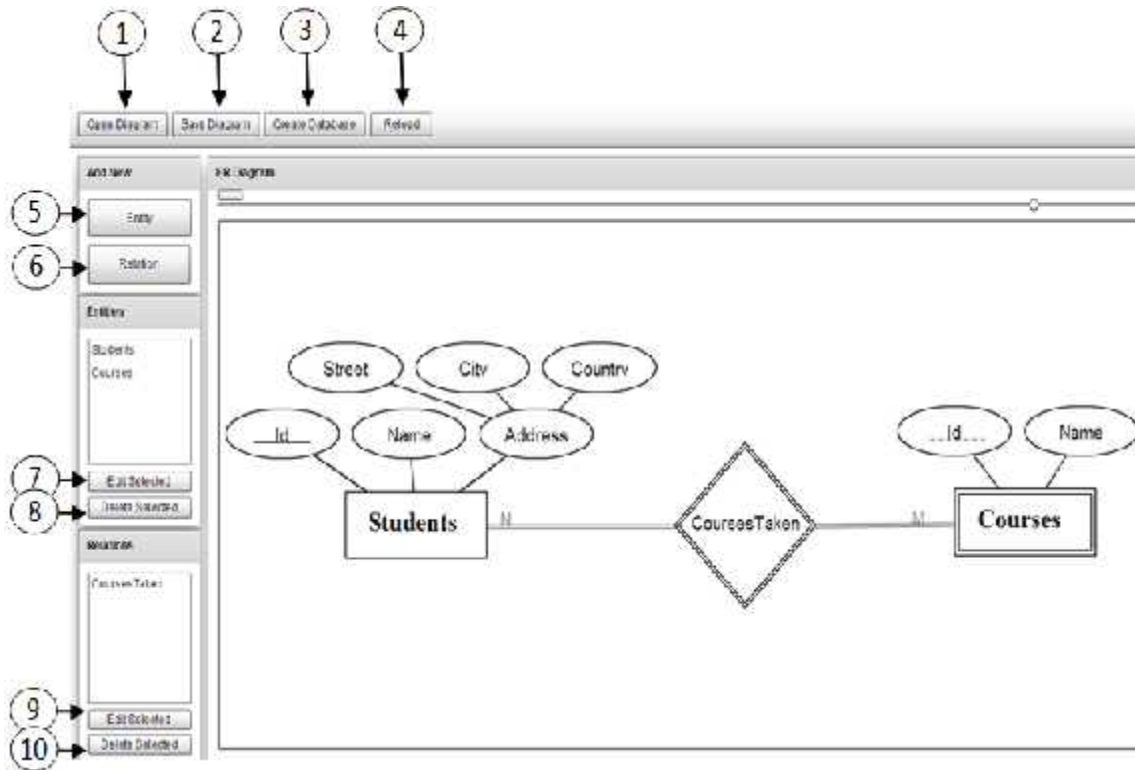
DBColumn
-name -domain -fk -fkTable -fkColumn -notNull
+DBColumn() +Name() +Domain() +NotNull() +FK() +FKTable() +FKColumn() +setFk()

CompositeAttribute
-attributes
+CompositeAttribute() +removeAttributeByName() +removeAttribute() +addAttribute() +getAttributeAt()

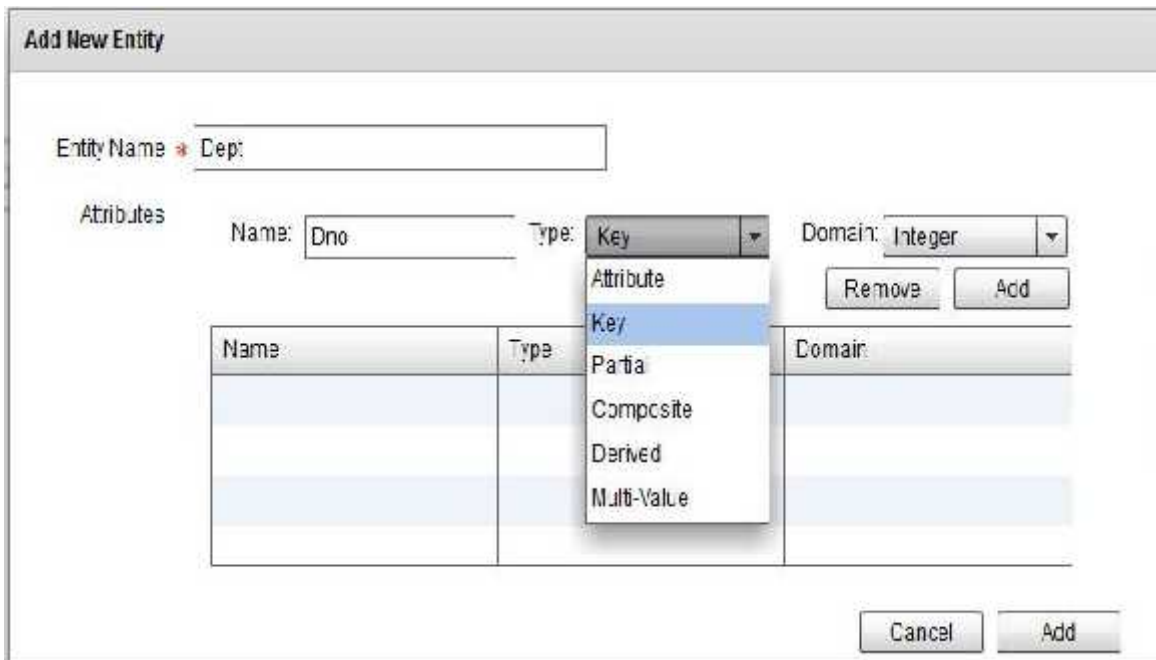
Entity
+Entity() +isWeak()

RelationConnection
-entity -cardinality -totalParticipation
+RelationConnection()

Appendix A-2 Start-up Page of ERD Mapper

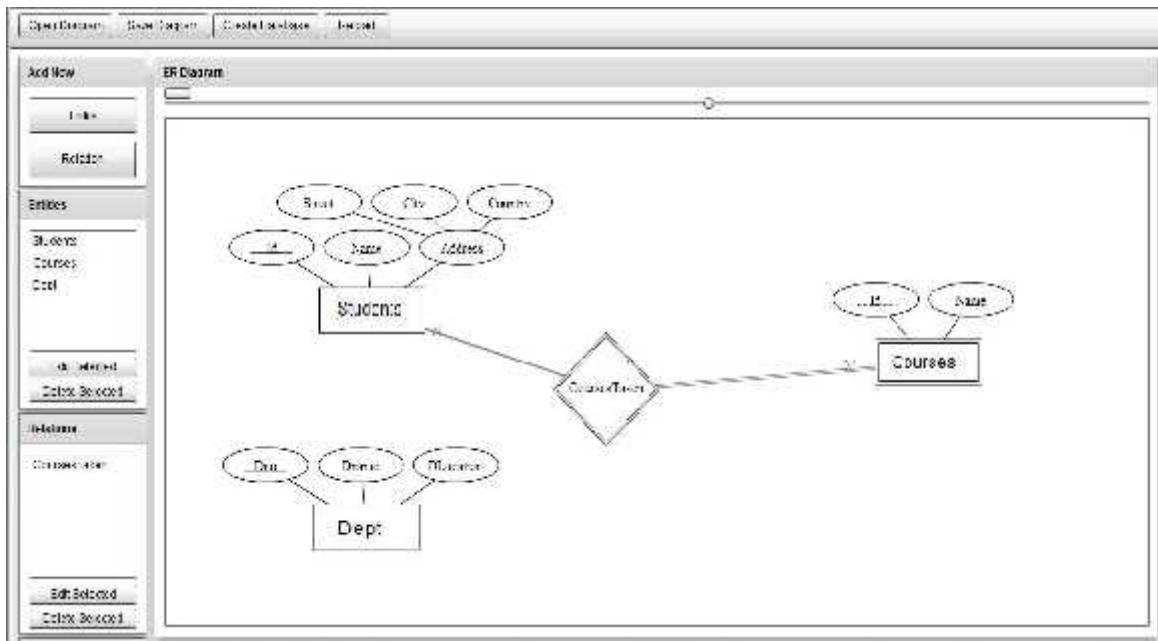


Appendix A-3 Add New entity



The 'Add New Entity' dialog box is shown. The 'Entity Name' field contains 'Dep:'. The 'Attributes' section has a table with columns 'Name' and 'Type'. One attribute is listed: 'Dno' with type 'Key' and domain 'Integer'. A dropdown menu is open over the 'Type' field, showing options: Attribute, Key, Partia, Composite, Derived, Multi-Value. There are 'Remove' and 'Add' buttons next to the attribute table, and 'Cancel' and 'Add' buttons at the bottom of the dialog.

### Appendix A-4 Append the New Entity to the Drawing Panel







Appendix A-5 Add New Relationship

**Add New Relation**

Relation Name:

Identifying:

Entities: Entity:  Cardinality:   Total

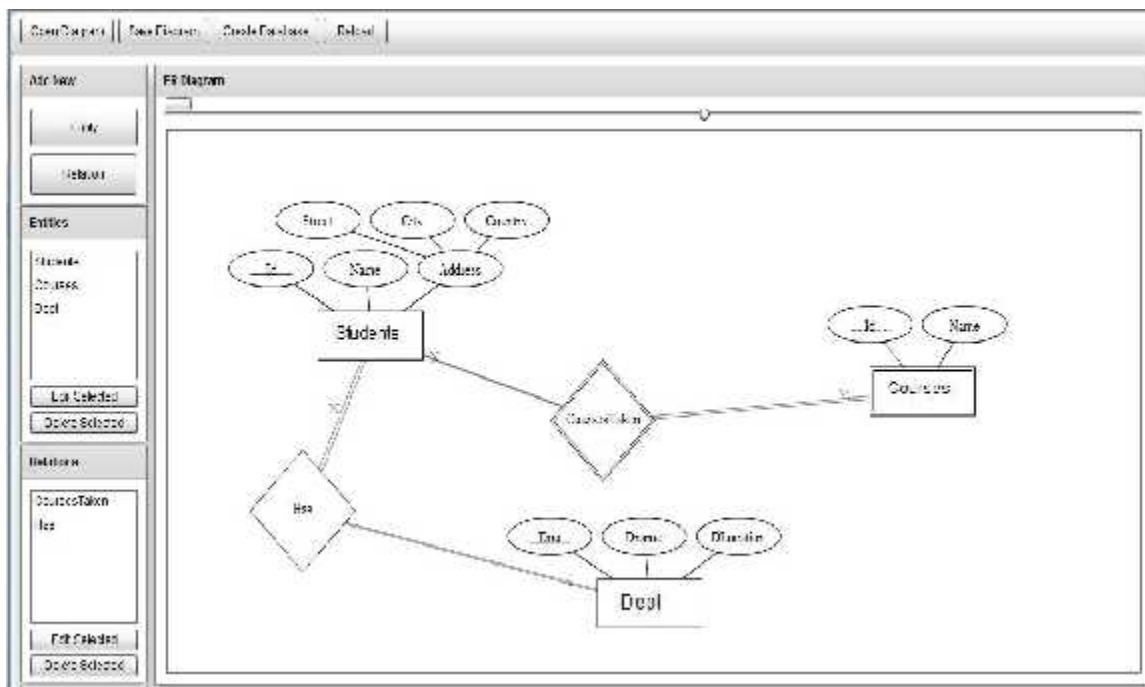
You need to connect at least two entities

Entity	Cardinality	Total

Attributes: Name:  Type:  Domain:

Name	Type	Domain

## Appendix A-6 Append the Relationship to the Panel



## Appendix B-1 Query Converter Class

```

package org.e2rd.converter
{
    import org.e2rd.blocks.AttributeDomain;
    import org.e2rd.db.DBColumn;
    import org.e2rd.db.DBTable;
    public class QueryGenerator
    {
        public static function getQuery(tables:Vector.<DBTable>) :
            Vector.<String>
        {
            var q:Vector.<String> = new Vector.<String>();
            tables = sortTables(tables);
            for(var i:int = 0; i < tables.length; i++)
                q.push(getTableQuery(tables[i]));
            return q;
        }
        private static function sortTables(tables:Vector.<DBTable>)
            :
            Vector.<DBTable>
        { return tables; }

        private static function getTableQuery(t:DBTable) : String
        {
            var q:String = "CREATE TABLE " + t.Name + " \n(\n";
            var i:int;
            //get columns
            for(i = 0; i < t.ColumnsCount; i++)
    
```



```

        q += getColumnQuery(t.getColumn(i));
        //get fk
        for(i = 0; i < t.ColumnsCount; i++)
            if(t.getColumn(i).FK)
                q+= getForeignKey(t.getColumn(i))
        //get pk
        var pkList:String = t.PK[0].Name;
        for(i = 1; i < t.PK.length; i++)
            pkList += "," + t.PK[i].Name;
        q += "CONSTRAINT pk_" + t.Name +
            " PRIMARY KEY (" + pkList +
    ")\n" (
        return q;
    }

    private static function getColumnQuery(c:DBCColumn) : String
    {
        return c.Name + " " + mapType(c.Domain) +
            (c.NotNull?" NOT NULL,\n":",\n(");
    }
    private static function getForeignKey(fk:DBCColumn) : String
    {
        return "FOREIGN KEY (" + fk.Name + ") REFERENCES " +
            fk.FkTable.Name + "(" + fk.FkColumn.Name +
    ")\n";
    }
    private static function mapType(d:String):String
    {
        switch(d)
        {
            case AttributeDomain.ANY:
                return "TEXT";
            case AttributeDomain.STRING:
                return "TEXT";
            case AttributeDomain.DATETIME:
                return "INTEGER";
            case AttributeDomain.INTEGER:
                return "INTEGER";
            case AttributeDomain.FLOAT:
                return "REAL";
        }
    }
    return "Text";
}}}
```