**EXPERIMENT 12**

# DATA MODELING USING ERWIN

**Objective**

## Familiarize you with Erwin Data modeling tool.

## Introduction

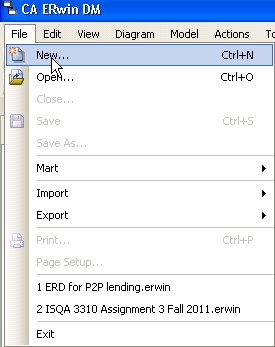
ERwin is a popular data modeling tool. The product supports a variety of aspects of database design, including data modeling, forward engineering (the creation of a database schema and physical database on the basis of a data model), and reverse engineering (the creation of a data model on the basis of an existing database) for a wide variety of relational DBMS. This brief tutorial steps you through the process of creating a data model using Er*win*. Creation of a basic data model (Conceptual data model)

## Creation of a basic data model

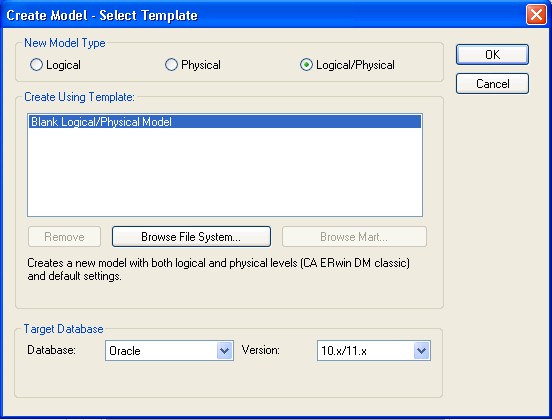
The Entities involved in this model include:  Employee, Department.

From the File Menu choose to create a new model: File->New

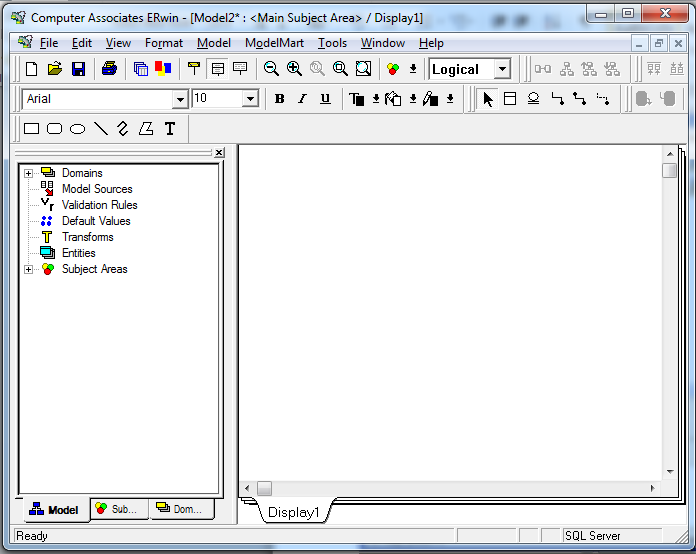
As shown in Figure 1 below

  
**Figure 1: Create a new model**

The next dialog box, shown in Figure 2, will ask you to choose the template to be used to create the new model. Choose Logical/Physical as the new model type.  This choice will allow us to switch back and forth easily between a logical model (ER Diagram) and a physical model (database schema).

  
**Figure 2: Selecting a model type**

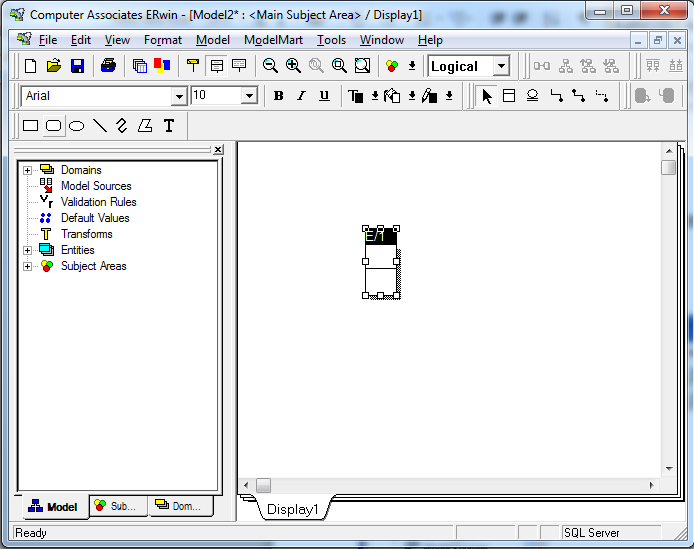
ERwin will now display the main window from which most of your ER diagram development will be done, as shown in Figure 3.

  
**Figure 3: The ERwin Workplace**

The ERwin workplace consists of two main parts.  On the left is the Model Navigator, which displays a hierarchy of items of importance, such as entities, domains, and subject areas.   On the right is the Display Window, which will show the ER diagram itself.  As you create objects, they will appear in the display window (if they are visual in nature, like entities), and appear in the hierarchy within the Model Navigator.

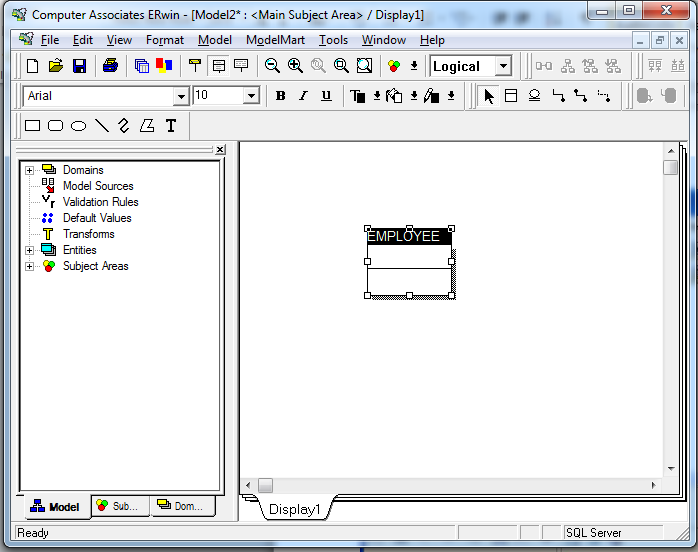
## Creating an Entity

To create a new entity, click on the entity icon (http://www.isqa.unomaha.edu/wolcott/tutorials/erwin/erwin_files/04_EntityIcon.jpg) on the toolbar, or right-click on the word *Entity* in the Model Navigator.  If you click on the entity icon, you then should click on the Display Window where you would like the entity to appear, as shown in Figure 4.

  
**Figure 4: A new entity**

Notice that the default name for the entity is E/x, where x is some number (1 in this case).  Click on the *Tab* key several times and notice what happens.  Pressing the *tab* key cause the focus to cycle between the three main parts of the Entity:   the name of the entity, the primary key attribute(s), and the non-primary key attribute(s).   In general, to modify one of these three parts of the entity, you will press the *Tab* key to cycle to the appropriate part of the entity, then type to add or modify that part of the entity.

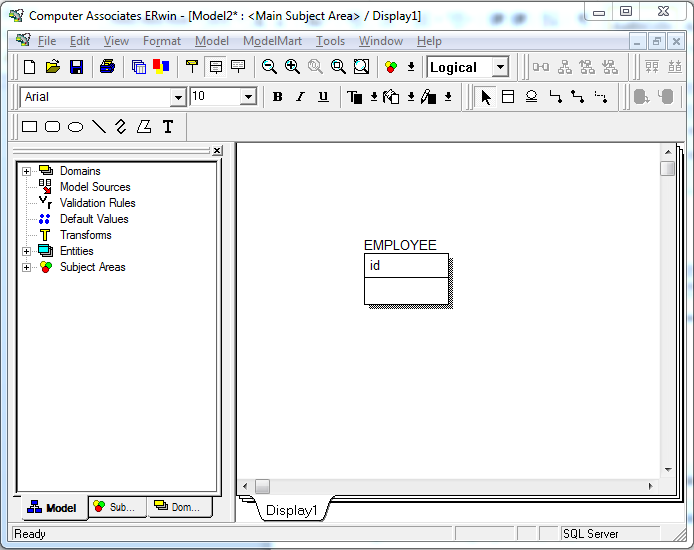
Right now, press the *Tab* key until the entity name is highlighted.  Then type EMPLOYEE, as shown in Figure 5.

  
**Figure 5: Changing the name of the entity**

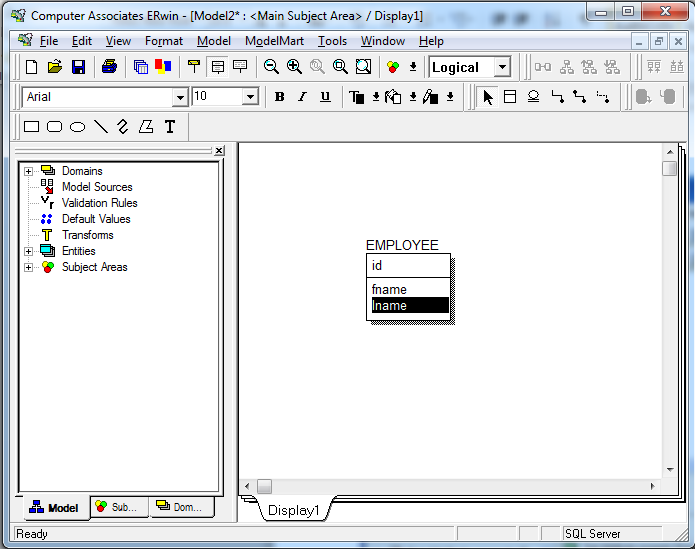
At this point, you may wish to save and name your diagram to avoid loss should the system or application crash.

**Adding primary key columns**

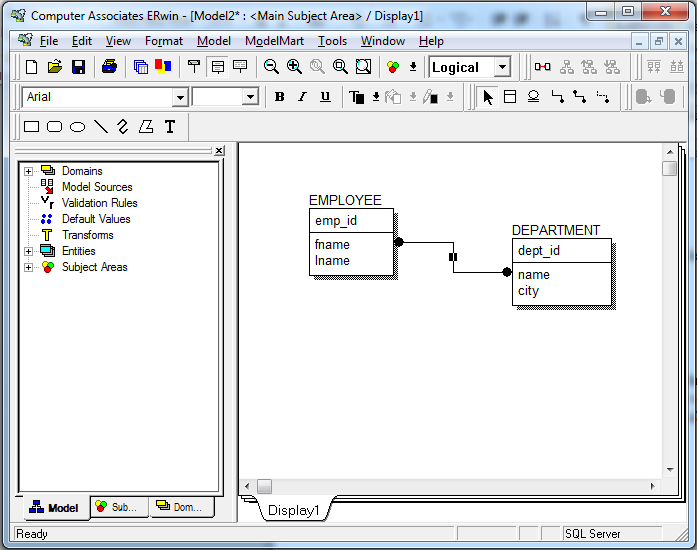
Once you have changed the name to EMPLOYEE, press the *Tab* key again to move the focus to the next part of the Entity, adding a primary key attribute.  Then type the name of the primary key attribute, **id**, as shown in Figure 6.

  
**Figure 6: Adding a primary key attribute**

Press the *Tab* key one more time to bring the focus below the horizontal line in the Entity, where you will add in a number of non-primary key attributes.   Type fname, as shown in Figure 7.   When you have typed fname, press the *Enter* key (not *Tab*).   Notice what happens.  The cursor is now positioned for you to add another attribute in this same portion of the Entity, the non-primary key attribute portion.  Type another attribute lname.

  
**Figure 7: Adding non-primary key attributes**

Make another entity, DEPARTMET with primary key dept\_id and attributes name and city shown in the Figure 8 below.



**Figure 8: Employee and Department Entities**

## Creating Relationships

ER*win* supports the creation of relationships with three basic kinds of connectivity:   one-to-one, one-to-many, and many-to-many.  Within the one-to-many category, ER*win* allows us to distinguish between *identifying*and *non-identifying* one-to-many relationships.

**Q:** [What's the difference between identifying and non-identifying relationships?](http://stackoverflow.com/questions/762937/whats-the-difference-between-identifying-and-non-identifying-relationships)

* An **identifying relationship** is when the existence of a row in a child table depends on a row in a parent table. This may be confusing because it's common practice these days to create a pseudokey for a child table, but *not* make the foreign key to the parent part of the child's primary key. Formally, the "right" way to do this is to make the foreign key part of the child's primary key. But the logical relationship is that the child cannot exist without the parent.

**Example**: A Person has one or more phone numbers. If they had just one phone number, we could simply store it in a column of Person. Since we want to support multiple phone numbers, we make a second table PhoneNumbers, whose primary key includes the person\_id referencing the Person table.

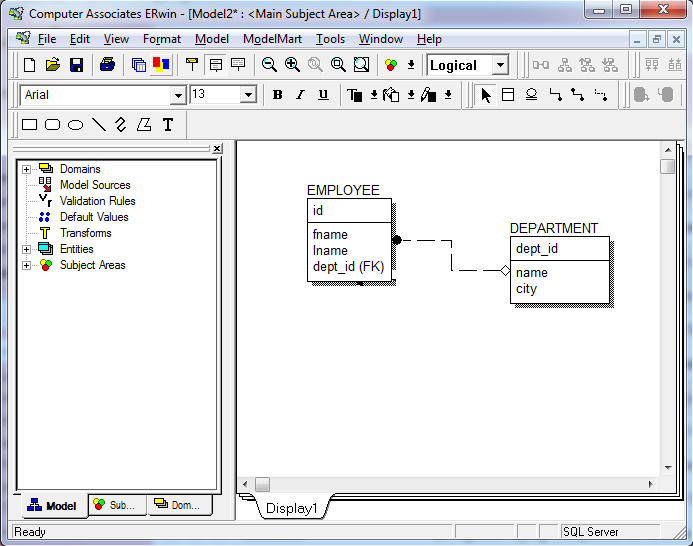
We may think of the phone number(s) as belonging to a person, even though they are modeled as attributes of a separate table. This is a strong clue that this is an identifying relationship (even if we don't literally include person\_id in the primary key of PhoneNumbers).

* A **non-identifying relationship** is when the primary key attributes of the parent *must not* become primary key attributes of the child. A good example of this is a lookup table, such as a foreign key onPerson.state referencing the primary key of States.state. Person is a child table with respect to States. But a row in Person is not identified by its state attribute. I.e. state is not part of the primary key of Person.

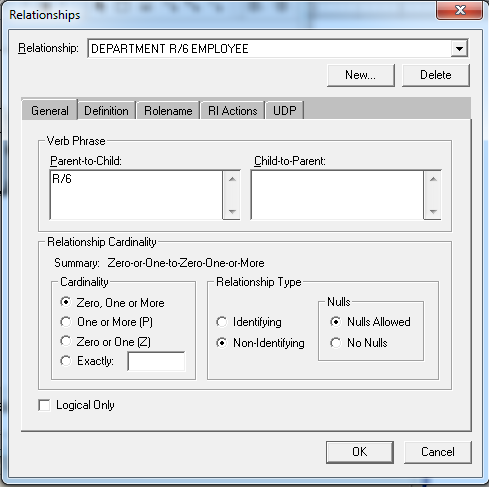
A non-identifying relationship can be **optional** or **mandatory**, which means the foreign key column allows NULL or disallows NULL, respectively.

### One-to-many Relationships

An *non-identifying* relationship is created by clicking first on the non-identifying relationship icon ().  To create a *non-identifying* relationship, click first on this icon, then click on the parent entity (on the one side of the relationship) and then click on the child entity (on the many side of the relationship).   In this case, you will click first on the *non-identifying* relationship icon, then on DEPARTMENT, then on EMPLOYEE.  The results are shown in Figure 9 below.

  
**Figure 9: A non-identifying relationship**

Double-click on the relationship itself to bring up a dialog box in which we can further refine the relationship definition.   As shown in Figure 10 below.

  
**Figure 10: Relationships definition**

In the Relationship Cardinality portion of this window, we can determine how many child entity occurrences may be associated with each parent entity occurrence.

You can further inquire about the functionality of Erwin By going though the guide provided in the Lab folder.

***Forward Engineering***

When you have a physical data model, ERwin automatically generates the schema for the target server as you build the model. *Forward Engineering* is the process that Erwin uses to transfer the schema from the data model to the target server. Graphically shown in the Figure 11 below.

When you forward engineer a data model, you can choose to generate a script file, which you can use to update the database using a database administration tool or you can forward engineer by directly connecting to the database catalog. Either way, ERwin does the majority of work for you by generating the script for your target server.

## 

**Figure 11: Model**

Before you forward engineer, you can view the *schema,* which is a text-based representation of the database objects that will be created in the database from the script. ERwin uses the data definition language (DDL) for the target database to write the script. Each time you add an object or property to your data model, Erwin automatically updates the script file to reflect the change to the data model.

**To Preview the Script File**

1. From the File menu, choose Open and locate *the .er1 file*, which you saved in a previous exercise.
2. When your *.er1* opens, using the Model Type Indicator, switch to the Physical model. Then from the Tools menu, choose Forward Engineer/Schema Generation.
3. When the Schema Generation dialog appears, click the Preview button at the bottom of the dialog. The Schema Generation Preview dialog appears as shown in the figure 12 below.



**Figure 12: Script File**

1. When you are finished previewing the schema, click the Close button to return to the Schema Generation dialog.

**To Generate The Script File**

1. In the Oracle Schema Generation dialog, click the Report button at the bottom of the dialog shown in the figure 13 below.



**Figure 13**

1. In the Save As dialog, in the File Name box, type Mymodel**.sql** and click Save.

As shown in figure 14.



**Figure 14**

***Reverse Engineering***

When you have an existing database from which you want to create a data model, ERwin lets you reverse engineer the database schema, which is converted by Erwin into a graphical representation of the database structures. A graphical model of the process is shown below in the Figure 15. The process of reverse engineering saves considerable time because ERwin does the work of creating the data model directly from the database.

****

**Figure 15: Reverse Engineering**

**Using The Reverse Engineer Wizard**

1. From the Tools menu, choose Reverse Engineer shown in the Figure 16.



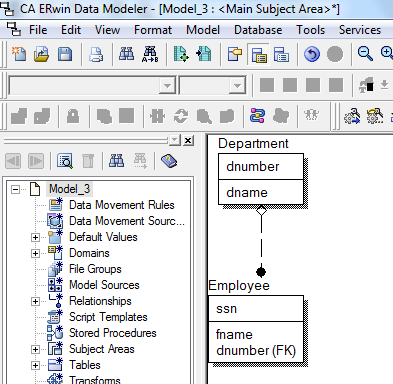
**Figure 16**

1. In the dialog, select Physical as the New Model Type, select Blank Physical Model as the template, and SQL Server 2000 as the target database.
2. Then click Next. The Reverse Engineer Set Options dialog box appears as shown in Figure 17:



**Figure 17**

1. In Reverse Engineer From, select Script File and then click Browse to locate the revEng.sqlfile lying in your lab folder. For the purpose of this exercise, accept the default options in the remaining areas of the dialog and then click Next.
2. You will see a small dialog with text that describes the database structures that ERwin is reverse engineering. When the process ends, a new data model appears in the Diagram Window as shown in Figure 18.



**Figure 18**

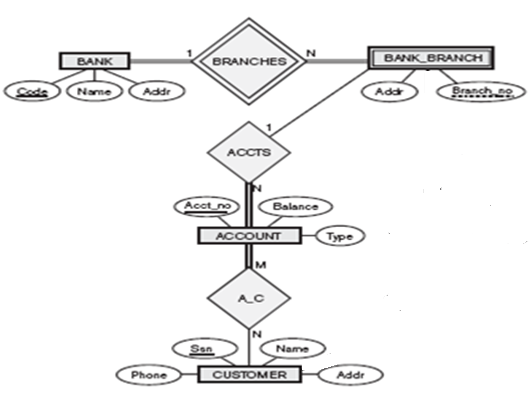
1. From the File menu, choose *Save* and save this model as *My ERwin Model.ER1*. When asked if you want to replace the existing file, click Yes.

# References

* <http://www.isqa.unomaha.edu/wolcott/tutorials/erwin/erwin.html>
* http://stackoverflow.com/questions/762937/whats-the-difference-between-identifying-and-non-identifying-relationships

**In Lab Task:**

1. Using Erwin make the following ER diagram as shown in Figure 19: Please give a screenshot of your data model in the space provided



**Figure 19**

1. Forward Engineer the data model you created for the inLab and save the SQL file that gets generated.
2. Create a small sql file of your choice and then reverse Engineer it. Paste your sql file as well a screenshot of the data model that gets generated in the lab report.