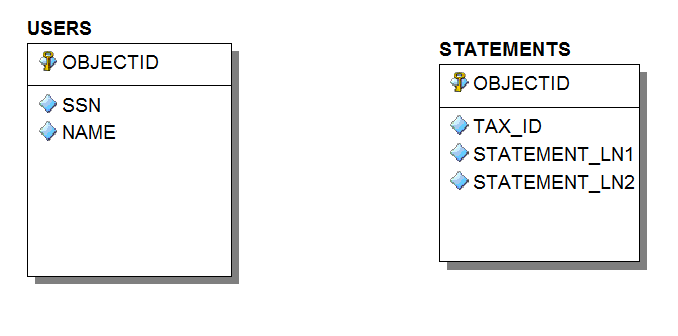
# Adding FK Constraint in ER/Studio

## Introduction

One would think it would be an easy thing to do to add a Foreign Key constraint in a tool designed to model entities and relationships for database systems. If one were starting their model from scratch, this would be an easier process. The ER/Studio tool lets you drag and drop a line from parent to child to define the FK relationship. Unfortunately, many times the tool is used to reverse-engineer a model from an existing database which can complicate matters.

## The Problem

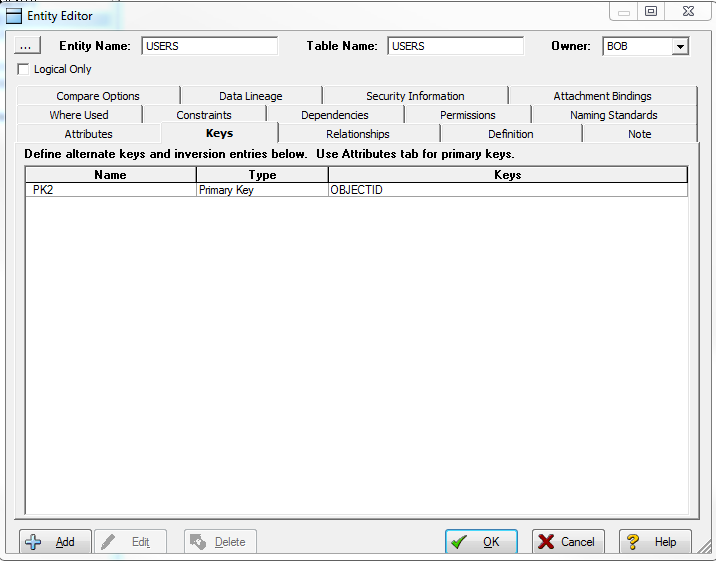
The ER/Studio tool assumes that you want to define the FK relationship between primary keys of the two tables in the relationship. To illustrate, look at the following tables in ER/Studio:



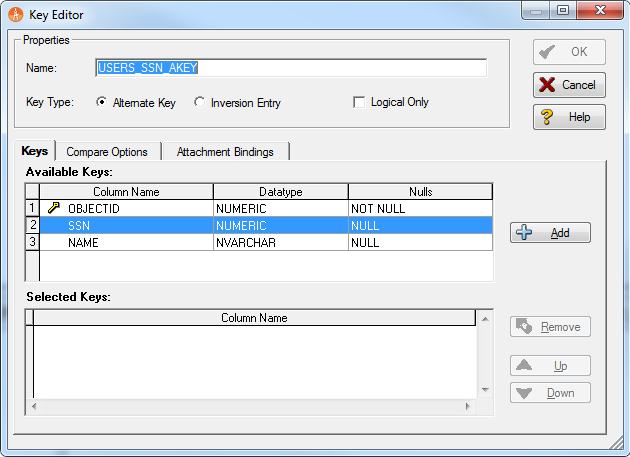
In this example, the USERS table is the parent and the STATEMENTS table is the child. If I drag and drop the relationship from parent to child, ER/Studio will assume that the relationship is defined on the PK column. In my case, the OBJECTID column in both tables is populated with an Oracle sequence and is a synthetic primary key. While synthetic keys are bad design practice and natural keys should be used, it is too late to change the model as it has been rolled into production. In my relationship between the two tables above, the TAX\_ID column is a child of the SSN column in the USERS table. ER/Studio has two issues when trying to define this relationship. One, neither column is the Primary Key of its table. Two, the column names are different. This document shows you how to define a FK constraint between these tables when neither column is a primary key and the column names differ.

## The Solution

In Oracle, you are not required to reference a Primary Key constraint in your Foreign Key definition. However, the column(s) in the parent table must be unique, so you should have defined a unique constraint for the parent table. So the first thing to do is to make sure our parent table, USERS, will allow the SSN column to be usable for a FK relationship. In ER/Studio parlance, we need to make this column an Alternate Key. First, double click on the USERS table to bring up the Entity Editor. Then click on the Keys tab.



Click the Add button.

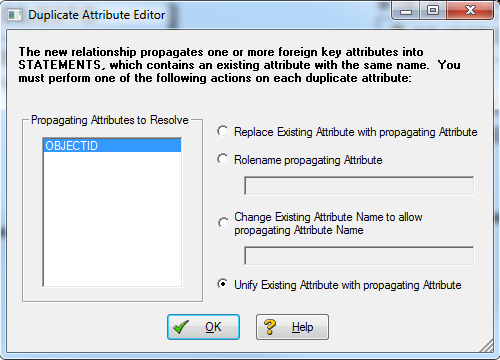


Enter a name for the new key. Make sure the Alternate Key radio button is selected. Select the SSN column under Available Keys, click the Add button, then click OK. The entity editor should now show this alternate key.

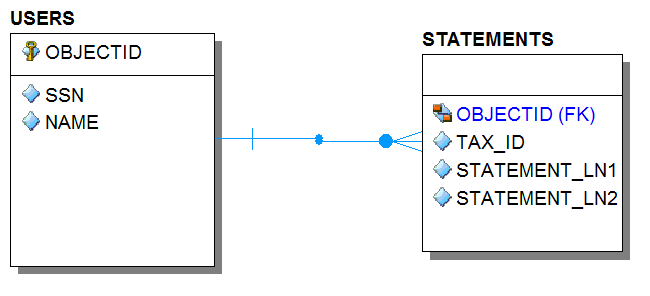


Click OK to close the Entity Editor box.

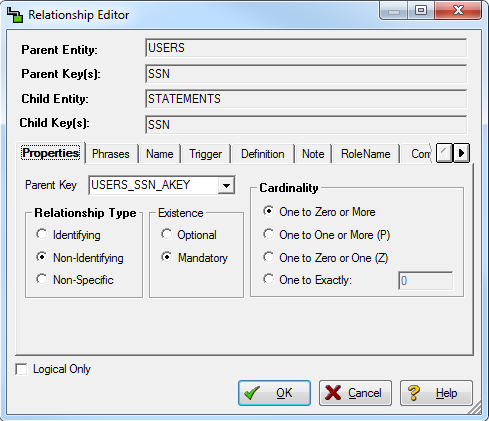
Now let’s add that relationship, click the Non-Identifying, Mandatory relationship tool button. Then click on the Users (parent) table and then click on the Statements (child) table. ER/Studio will detect a duplicate attribute. Select the last radio button and then press OK.



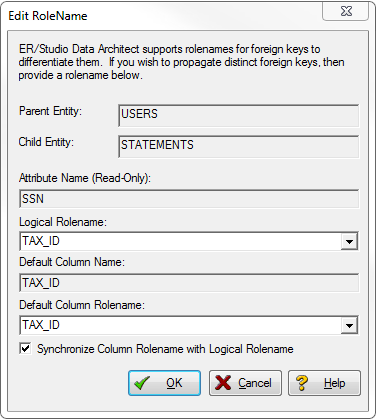
At this point, the OBJECTID column in the STATEMENTS table looks a little funky. We’ll fix that.



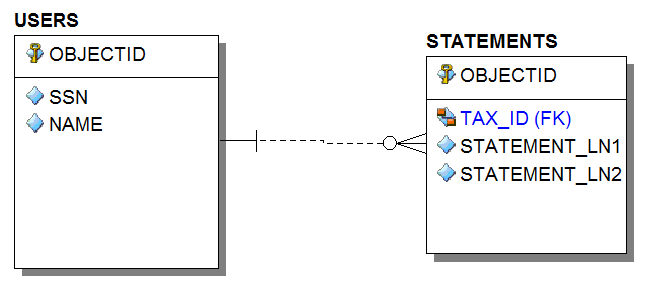
Double-click on the relationship to launch the Relationship Editor. In the Parent Key drop down, select the Alternate Key we created earlier. This is your chance to change the Relationship Type, Existence, and Cardinality if you desire. Press OK.



Next, right-click on the relationship and select Edit Rolenames. Since we specified the Alternate Key, the SSN column from the parent table will automatically be used. All we need to do is select the TAX\_ID column from the child table, press OK and we’re done!



Our ERD now looks as we have intended all along.



And if you double click on the relationship, we can see the correct columns for the correct tables are used in our relationship.



## Conclusion

There should be an easier way to do this. It would be nice if ER/Studio let you drop a relationship between two entities and ask which columns you want to participate in the relationship.