Who Needs Relationships (in a Database)?

by Clark Anderson

I have taken over maintenance of applications with no relationships defined in the database. All of the control was the responsibility of the Visual Basic front end. In some cases, I gradually incorporated the most important relationship definitions. I still proceeded with caution because they were production applications and I believe in the philosophy: “If it ain`t broke don`t fix it”. Applications created in this way are difficult to maintain and sometimes unstable.

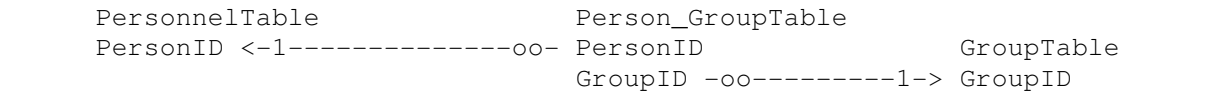
Fortunately I have also had the opportunity to work with my customers to build from the ground up a data model with all of the important entity tables and many intermediate relationship tables. Once I had designed the tables with their relationships, I could design the queries with SQL. I , then designed the Visual Basic User interface, with queries for all transactions: Select, Append, Delete and Update. The application has been very stable, first in VB3, and now in VB5. I do not anticipate any problems in migrating from Access 2.0 to Access 8.0 or beyond.

One of the main tables is a Personnel List with a counter, PersonID, as primary key. Another is The Group List, with GroupID as primary key. A third table is the Assignment List. This table is used to assign Personnel to Groups and contains the two “foreign keys”, PersonID and GroupID. In the table definition, these foreign keys are selected and marked as a two field primary key for the Person_Group Assignment table.

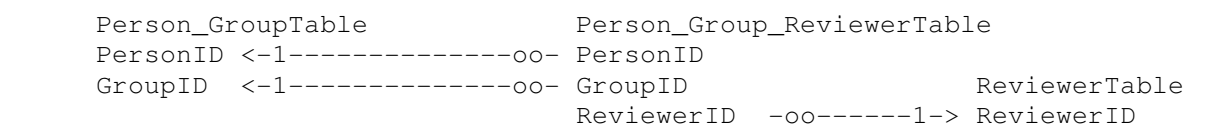
A person can be assigned to many groups (one-many) and a group can have many persons assigned to it (one-many). The Assignment list should only include the Persons that have been assigned and only the Groups that have assignees.

A primary purpose of this application is to maintain a list of personnel assignments. But, the main purpose is to record performance reviews. For this we need a table containing a list of Reviewers with a primary key, ReviewerID. The next step is to combine the composite foreign key [PersonID-GroupID] from the Assignment table with the ReviewerID foreign key to form the primary key in a “Person_Group_Reviewer” table (what a name).

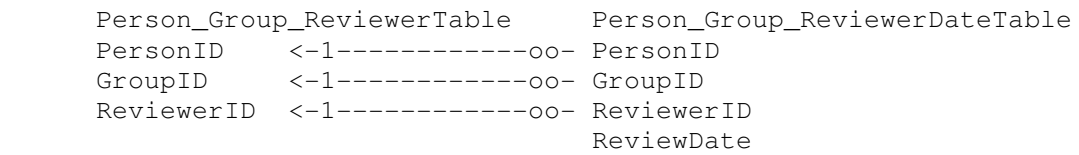
The Person_Group Assignment representing that person’s efforts for a particular group can be reviewed by many reviewers (one-many), but you want to include only the Person_Group with Reviewers that actually performed the reviews for that individual in that group. A reviewer may review many assignees, but you would only include the assignees that reviewer evaluated.

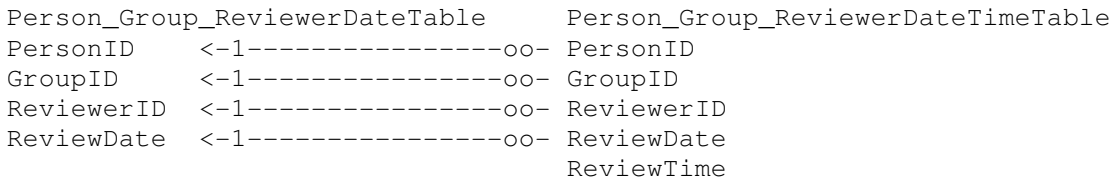


The mechanics of defining the relationship involving a composite key had me fooled at first. In the Relationships window I dragged key from the source table to the related table key. In this case, I dragged the PersonID from the Assignment table to the PersonID in the a “Person_Group_Reviewer” table. The Edit Relationship form lists the primary Assignment table with the key PersonID and the related table with the key PersonID. Under PersonID in each list there is a box that I clicked on. It became a drop-down box where I selected the other part of the composite key, GroupID, for one table then the other table.



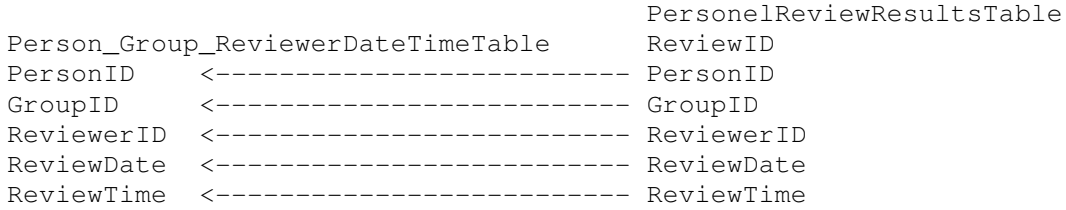
It was easier once I became accustomed to this process. The next table combines this three field key with another foreign key that is a virtual entity time(date). Time is always a many to many relationship. The key for this table is the ReviewDate because the Person_Group_Reviewer combination may occur on many dates providing a table “Person_Group_Reviewer_Date” with a primary key consisting of 4 fields. To really nail this down I took it a step further and created a “Person_Group_Reviewer_Date_Time” table with a fifth field in its primary key, the ReviewTime.





In defining all of these relationships, I enforced referential integrity by specifying Cascade Update of Related Fields and Cascade Delete of Related Records. This is easy to specify before there is any data. Notice that the arrows in the Relationships Window point from the “many” tables to the “one” Lookup tables. Before one person can be assigned many times they must be “looked up” in the Personnel table.

This set of tables and relationships provides a tightly controlled definition for creating reviews of personnel. I created another table, PersonelReviewResults. This table has a simple counter for a primary key, ReviewID. It also has the 5 field foreign key that has been so carefully constructed. The referential integrity for this 5 field relationship is intentionally NOT enforced.



It was important to create a unique Review, but once the review is created, one or more of the key elements may go away. A reviewer may leave or a group may be disbanded or a person may be transferred from that group to another.

In my first application of relational data modeling, I depended too much on intuition and skipped a step or two. I have learned that it is important to build each table and combine the foreign keys one at a time. Many of the tables contain nothing but their primary key. They can be referred to as relational tables. The tables that contain entities and attributes can be referred to as entity tables.

The Visual Basic front end for this application must append the data records to the tables in bottom-up order Person_Group_Reviewer, Person_Group_Reviewer_Date, Person_Group_Reviewer_Date_Time. Record deletions must be done in top-down order.

It is very difficult to wade into an application that has no relationships defined. Discrepancies with lookup tables must be resolved some how. Some data can be translated to consolidate into controlled choices and some data must be thrown out.

Who Needs Relationships? I need good solid relationships from day one or at least early in the game! Everything else is a compromise that may or may not be worth it.