Creating a Relational Database Using Microsoft SQL Code

Farrokh Alemi, Ph.D.

The objective of this note is to help you understand how a relational database is organized as a collection of tables, linked to each other. In this note, we show how Microsoft SQL Server software can be used to create tables of data. Then, we show how these tables can be linked. A separate note, posted to the class web site, demonstrates the same concepts but uses Microsoft Access code to do so.

We demonstrate the concepts behind relational databases through showing a simple electronic health record with three tables: a table on patients, another table on providers and a third table on medical encounters/visits that link the patient and the provider.

The easiest way to create a table and insert values into it, is to import the table and values created elsewhere by other software. Here, however, we show how SQL can be used to create a table inside Microsoft SQL Management Studio.

Relationships

Make sure that you have reviewed the note on creating entities from a list of fields. In database design, every entity has its own table. So in our example, patients, providers, and encounters are considered entity. For each of these three entities we need to create separate tables. Figure 1 shows the three entities and their relationship to each other. The line that connects the three images of tables shows how the tables are inter-related. Here the patient ID in the encounter table is the same as the ID in the patient table. The provider ID in the encounter table is the same as the ID in the provider table. The relationship among any two tables is preserved through having the same field in each of the tables. A join command specifies which two variables are the same across the two tables. We will discuss this more, when we cover the join types and code in a later section.

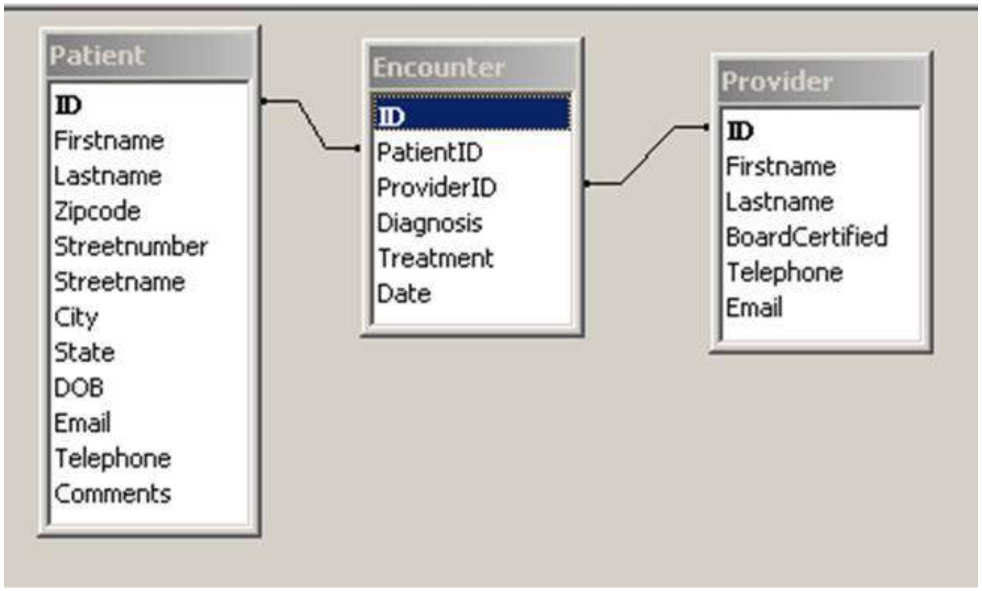

Figure 1: Relationship among Three Entities/Tables

Creating Tables

Each table contains a series of fields. Fields are data we collect about various entities. These fields provide the data for the attributes of the entity. To create a table, the command syntax is:

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

The column parameters specify the names of the fields of the table. The “datatype” parameter specifies the type of data the column can hold. Data types are specified on the web but the most common are varchar, integer, float, date, text. Always consult the web for exact data types allowed in your implementation of SQL code, as there are variations in different implementations.

The patient attributes are shown in Figure 1 and are assumed to be first name, last name, date of birth, address (street name, street number, city, State and zip code) and email. First name is a string of maximum size 20. Last name is a string of maximum size 50. Street number and zip code are integer numbers with no decimals. Date of birth is a date put in the format DD MMM YY (e.g. 19 Jan 04). The possible values for the State are Maryland, Virginia, District of Columbia and other. Patient's telephone number could be text or number. A patient ID (auto-number) should be used as the primary key for the table.

Here is a code that will accomplish these requirements. Note that field names are put in brackets because they contain spaces. Also note that the # before the table name indicates that the table is a temporary table which will disappear once the SQL window is closed:

```
CREATE TABLE #Patient (
   [First Name] char(20),
   [Last Name] char(50),
   [Street Number] Int,
   [Street] Text,
   [Zip Code] Int,
   [Birth Date] Date,
   [Email] text,
   [State] Text,
   [Phone Number] Text,
   [Patient ID] int IDENTITY(1,1) PRIMARY KEY

)
```

The provider attributes are assumed to be first name (text of size 20), last name (text of size 50), whether they are board certified (a yes/no value), date of hire (in format DD MMM YY), telephone (text or number) and email (text of size 75). Employer's ID number should be the primary key for the table. In SQL server there is no Yes/No field but the closet data type is a bit type, which assigns it a value of 1, 0, or Null. Here is the code that will create this table:

```
CREATE TABLE #Provider (
   [First Name] char(20),
   [Last Name] char(50),
   [Board Certified] bit,
   [Date of Hire] Date,
   [Phone] Text,
   [Email] char(75),
   [Patient ID] int IDENTITY(1,1) PRIMARY KEY
);
```

The encounter entity is assumed to have the following attributes: patient ID, provider ID, diagnosis (text of size 50), treatment (text of size 50) and date of encounter (date entered in the format DD MMM YY) and encounter ID as a primary key. Each encounter should have its own ID number. Here is the code that will create this table:

```
CREATE TABLE #Encounter (
   [Patient ID] Int,
   [Provider ID] Int,
   [Diagnoses] char(50),
   [Treatment] char(50),
   [Date of Encounter] Date,
   [Encounter ID] int IDENTITY(1,1) PRIMARY KEY
);
```

This completes the creation of Patient, Provider, and Encounter tables. Since the encounter table shares the patient ID with the patient table and shares the provider ID with the provider table, these tables are all related to each other. We have thus created a relational database with 3 tables.

Where is my Table?

There are different types of table prefixes in Microsoft SQL server. The place where a table is written is dictated by its prefix. A prefix of dbo indicates that the table should be permanently written to the database. A prefix of # indicates the table is temporary and should disappear if the window with the SQL code that created it is closed. A prefix of ## indicates that the table is temporary but should be available to all open windows of SQL code, not just the window for the session that created it. Thus #file is a temporary file, ##file is a global temporary file and dbo.file is a permanent file.

Entering Data into Tables

In the previous step, we created three tables. Now we show how values can be inserted into these tables. Table 1 shows the data that should be entered into the Patient table.

First Name	Last Name	Zip code	Date of birth	Email	Phone of the patient
Farrokh	Alemi	22101	08-Jan-54	Test2@gmu.edu	703 9934226
George	Smith	22102	09-Sep-60	t@tes.com	703 8884545
Jill	Smith	22103	01-Aug-89	test@test.com	703 9934226

Table 1: Three Rows of Data for Patient Table

The syntax for inserting values into fields is provided on the web and is as follows:

```
INSERT INTO table_name (column1, column2, column3, ...)
VALUES (value1, value2, value3, ...);
```

In this code, columns refer to fields in the table. Values refer to data that should be inserted. So to insert the values in Table 1 into the Patient table we would use the following commands:

```
INSERT INTO #Patient ([First Name], [Last Name], [Street
Number],[Street], [Zip Code], [Birth Date], [Email], [State], [Phone
Number])

VALUES
('Farrokh', 'Alemi',Null,Null, 22101, '08/01/1954',
'Test2@gmu.edu',Null, '7039934226'),

('George', 'Smith', Null, Null, 22102, '09/09/1960','t@tes.com',
Null,'7038884545'),

('Jill', 'Smith', Null, Null, 22103, '01/08/1989', 'test@test.com',
Null,'7039934226');
```

Did you notice that the street name, street number, and state were entered as null values? Also note that patient ID was not entered. The software will assign a unique number for patient ID. It will automatically increment by 1 each time a new record is entered. Finally, note that text fields are in quotes, dates are in quotes, but numbers and null values are not. Putting the null value in quotes will enter it as if was a text, which defeats the purpose.

The code will produce the following table of data:

First Name	Last Name	Street Number	Street	Zip Code	Birth Date	Email	State	Phone Number	Patient ID
Farrokh	Alemi	NULL	NULL	22101	8/1/1954	Test2@gmu.edu	NULL	7039934226	1
George	Smith	NULL	NULL	22102	9/9/1960	t@tes.com	NULL	7038884545	2
Jill	Smith	NULL	NULL	22103	1/8/1989	test@test.com	NULL	7039934226	3

Table 3: Data as Entered by Insert Command Into #Patient Table

Note how the patient ID shows up anyway without us entering its values.

Relational Database

The tables for Provider and Encounter are created in a similar fashion by using the “Create Table” and “Insert Values” commands. Once all three tables have been created then a relational database has been specified and Microsoft SQL server Management Studio can be used to analyze the data across all three tables. To analyze data across two or more tables one needs a Join command. Join commands are described in more detail in a later section.