IS 475/675
SQL First Lab Exercise: 2/3/2015

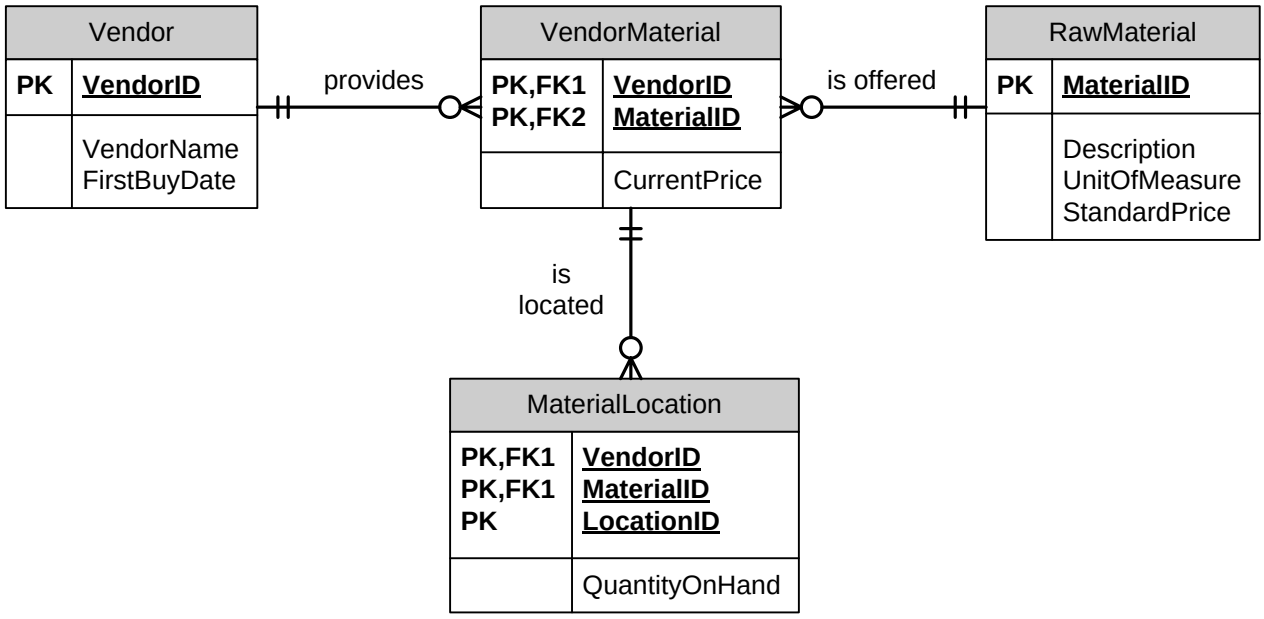
The goal of this exercise is to learn how to create and populate tables with SQL Server 2012 using SQL code only. There are no wizards, functions, or special help. We are just going to write simple SQL code.

This lab is not a homework assignment – it is not graded. There is nothing to turn in from this lab. It is to help you learn SQL. I recommend taking your time with each step to learn about both the programming environment and SQL code.

The database used for this exercise keeps track of the prices and quantity on hand of raw material available from different vendors. A raw material can be purchased from multiple vendors and a vendor can provide multiple raw materials. A vendor stores the raw material at varying locations, so the quantity on hand of a given material from a given vendor is different depending on a given location. VendorMaterial intersects Vendor and RawMaterial – but it does not serve as the intersection for MaterialLocation.

The entity on the 'one' side of the relationship is referred to as the "parent" entity. The entity on the 'many' side of the relationship is referred to as the "child" entity.

VendorMaterial is the parent in the relationship between VendorMaterial and MaterialLocation. Please note that VendorID and MaterialID is a concatenated foreign key in MaterialLocation. Here is the ERD of the tables you will create:



Task 1. Create a table.

General information:

1) Do NOT use spaces in the names of your tables or fields. Using spaces in table and field names will make life more difficult throughout the use of SQL. Don't do it!!!

2) SQL Server does not automatically "refresh" information in Object Explorer. To see new database objects in Object Explorer, make Object Explorer your active window by clicking on the name of your database and pressing the F5 key. F5 will refresh the information in that window.

3) Don't seek out special objects or functions to complete the tasks. Just type SQL code and test it.

4) Ignore indications of spelling/syntax errors for now. Learn how to find your own spelling/syntax errors.

Task instructions:

1) Click the "new query" button.

2) Type the following:

```
CREATE TABLE xtblVendor
(VendorID        char(4)          PRIMARY KEY,
  VendortName          varchar(30)      NOT NULL,
  FirstBuyDate         datetime);
```

3) Click the "execute" button.

4) Look at the table in Object Explorer: Expand the table and look at the columns and keys. Note that in the "columns" area, that the VendorID is indicated as the primary key. Note that in the "keys" area, the name of the primary key constraint was created by SQL Server and is something long and ugly like: PK_tblVendo_Fc8618D379F2524f. Your name will not be an exact replica of that name since SQL Server creates new names for all unnamed objects as they are created.

Task 2. Delete a table.

1) Click the "new query" button to open a new query tab. Don't type on top of the old query tab.

2) Type:

```
DROP TABLE xtblVendor;
```

3) Make Object Explorer your active window and refresh with F5. Note that the xtblVendor table is deleted.

Task 3. Create the table again with a manually named primary key constraint.

1) Return to the query tab where you created the table and modify the code to read as follows:

```
CREATE TABLE xtblVendor
(VendorID        char(4),
  VendorName          varchar(30)   NOT NULL,
  FirstBuyDate         datetime
  CONSTRAINT pkVendor PRIMARY KEY (vendorID));
```

2) Execute the code.

3) Make Object Explorer your active window and refresh with F5. Look at the name of the primary key constraint for the table in the "keys" area for the table in Object Explorer. The name should be pkVendor.

You have the option of naming your constraints or letting SQL Server name your constraints. We will discuss the difference in class on Tuesday. Just be aware of the difference now.

The remainder of this exercise assumes that you now know how to access a new query tab and how to execute the SQL code you type. I will not explicitly tell you to click on "new query" or "execute" your code again.

Task 4. Populate a table.

Each INSERT statement below will produce one row in xtblVendor. The commands can be typed individually and executed, or you can type them all and then execute them at one time. Make sure to enter quotes around the VendorID in the third row of '0062' and not enter quotes around the VendorID for the fourth row. I want you to see the difference in VendorID between the two rows. Also, enter the dates in the format provided. I want you to see the different ways dates may be entered into SQL Server.

```
INSERT INTO xtblVendor VALUES
('7819', 'Martinson Concrete and Supply', '04/15/2013');
INSERT INTO xtblVendor VALUES
('2745', 'Johnson Plating', '14-oct-2014');
INSERT INTO xtblVendor VALUES
('0062', 'Evergreen Surface Products', '07-12-2012');
INSERT INTO xtblVendor VALUES
(0062, 'Touchstone Materials', '05-16-13');
```

Task 5. Look at the data in a table.

1) Type the following SQL code into a new query tab:

```
SELECT      *
FROM        xtblVendor;
```

2) Execute the code. The output is referred to as a "result table" and is what is produced from any SQL SELECT statement. By using the * you are telling SQL to display all columns and all rows in the table. The result table should look like this:

	VendorID	VendorName	FirstBuyDate
1	0062	Evergreen Surface Products	2012-07-12 00:00:00.000
2	2745	Johnson Plating	2014-10-14 00:00:00.000
3	62	Touchstone Materials	2013-05-16 00:00:00.000
4	7819	Martinson Concrete and Supply	2013-04-15 00:00:00.000

Note that the data is displayed in a different order than it was input. SQL automatically "orders" the output by primary key, if no "order' statement is included.

3) Modify the SQL code typed in step #2 to add a "WHERE" clause as follows and see the new result table that is produced:

```
SELECT      VendorID,
            VendorName
FROM        xtblVendor
WHERE VendorID = '0062';
```

Change the "WHERE" clause to take out the quotation marks around '0062' execute the code and then look at the result table.

Task 6. Save the CREATE and INSERT SQL Code.

1) Copy the INSERT statements from the query tab where they are located and paste them under the CREATE TABLE code in its query tab. You should now have the code required to create and populate the

table in a single query tab. The query tab code should include both CREATE and INSERT code and should look like what you see below.

```
CREATE TABLE        xtblVendor
(VendorID           char(4),
 VendorName         varchar(30) NOT NULL,
 FirstBuyDate       datetime
 CONSTRAINT pkVendor PRIMARY KEY (vendorID));

INSERT INTO xtblVendor VALUES
('7819', 'Martinson Concrete and Supply', '04/15/2013');
INSERT INTO xtblVendor VALUES
('2745', 'Johnson Plating', '14-oct-2014');
INSERT INTO xtblVendor VALUES
('0062', 'Evergreen Surface Products', '07-12-2012');
INSERT INTO xtblVendor VALUES
(0062, 'Touchstone Materials', '05-16-13');
```

2) Go to the File menu selection at the top of the SQL Server Management Studio screen. Select the Save queryname as option and name the query something you will remember. Save the query on either your student location on the u: drive or to your personal flash drive.

You have now saved your SQL code. SQL code is not a database object; if you want to save your code, you must physically save it as we did in this step. I recommend that you save the code you write to create and populate tables for homework assignment #3, just as we are doing in class.

Task 7. Create the Raw Material table.

1) The next table to create is the Raw Material table, which I want you to call xtblRawMaterial. Here is the structure in a "tabular" format. This is the format I will provide on HW#3 for tables.

Table: xtblRawMaterial				
Attribute Name	Data Type & Size	Primary Key	Foreign Key and Referential Integrity	Other constraints
MaterialID	char(3)	Yes	no	no null value
Description	varchar(50)	No	no	
UnitOfMeasure	char(8)	No	no	
StandardPrice	money	No	no	must be > 0

2. Here is the SQL code to type in a new query window and then execute to create the table. Note that this table uses the CHECK constraint and some different data types.

```
CREATE TABLE      xtblRawMaterial
(MaterialID       char(3),
 Description      varchar(50),
 UnitOfMeasure    char(8),
 StandardPrice    money  CHECK (standardprice > 0),
 CONSTRAINT pkRawMaterial PRIMARY KEY (MaterialID));
```

3) Now, populate that table with the data shown below in tabular format. This is the format I will provide on HW#3 for table population.

xtblRawMaterial

MaterialID	Description	UnitOfMeasure	StandardPrice
255	Concrete Overlay Polymer	pound	13.75
240	Mortar Mix	null	.23
271	Graphite Isomolded Sheet	each	46.70

We are going to use a little different format for the INSERT statements this time. It is possible to use a single INSERT statement to populate multiple rows as shown below:

```
Insert into xtblRawMaterial values
('255', 'Concrete Overlay Polymer', 'pound', 13.75),
('240', 'Mortar Mix', null, .23),
('271', 'Graphite Ismolded Sheet', 'each', 46.70);
```

4) Do a SELECT * statement to see the contents of the table that you created:

```
SELECT      *
FROM        xtblRawMaterial
```

You should see the following result table:

	MaterialID	Description	UnitOfMeasure	StandardPrice
1	240	Mortar Mix	NULL	0.23
2	255	Concrete Overlay Polymer	pound	13.75
3	271	Graphite Ismolded Sheet	each	46.70

Task 8. Create the VendorMaterial table and learn about Referential Integrity.

Go back to the first page of this exercise and look at the ERD. First, note that the primary key for this entity is concatenated. You must create a concatenated primary key separate from the actual declaration of the attributes.

Second, note that there are two foreign keys in the VendorMaterial entity. One foreign key is used to relate the VendorMaterial table to the Vendor table, and the other foreign key is used to relate the VendorMaterial table to the RawMaterial table. A foreign key is just an attribute in one table that is copied to another table. A foreign key can be created without creating a SQL constraint.

1) Let's first create the VendorMaterial table without creating a SQL constraint for the foreign key.

Table: xtblVendorMaterial				
Attribute Name	Data Type & Size	Primary Key	Foreign Key and Referential Integrity	Other constraints
VendorID	char(4)	Yes	no	no null value
MaterialID	char(3)	Yes	no	no null value
CurrentPrice	money	No	no	no null value

```
CREATE TABLE   xtblVendorMaterial
(VendorID      char(4),
 MaterialID          char(3),
 CurrentPrice  money          NOT NULL,
 CONSTRAINT pkVendorMaterial
 PRIMARY KEY   (VendorID, MaterialID));
```

2) Let's put just one row of data into this table to understand what it means to enforce or not enforce referential integrity.

```
INSERT INTO xtblVendorMaterial VALUES
('7819', '288', 12.95);
```

Please note that while the vendorID exists in the Vendor table, the RawMaterial ID does not exist in the Raw Material table. It is possible, as shown in this example, to enter a foreign key in a child table that does not exist in a parent table.

3) Let's now change the CREATE TABLE statement to enforce referential integrity on the foreign key for the vendor ID, as depicted in the table below.

Table: xtblVendorMaterial				
Attribute Name	Data Type & Size	Primary Key	Foreign Key and Referential Integrity	Other constraints
VendorID	char(4)	yes	Yes – references xtblVendor	no null value
MaterialID	char(3)	yes	no	no null value
CurrentPrice	money	no	no	no null value

First, drop the table with the following SQL statement: DROP TABLE xtblVendorMaterial:

Then, modify the SQL CREATE TABLE statement to add a new constraint:

```
CREATE TABLE    xtblVendorMaterial
(VendorID       char(4),
 MaterialID          char(3),
 CurrentPrice   money      NOT NULL,
 CONSTRAINT pkVendorMaterial
 PRIMARY KEY    (VendorID, MaterialID),
 CONSTRAINT fkVendor
 FOREIGN KEY (vendorID) references xtblvendor (VENDORID));
```

Now, execute the INSERT statement again.

```
INSERT INTO xtblVendorMaterial VALUES
('7819', '288', 12.95);
```

All goes well.

4) Let's try and add another row – one that is for a vendor that isn't in the VENDOR table:

```
INSERT INTO xtblVendorMaterial VALUES
('1224', '288', 15.95);
```

You should get an error:

```
Msg 547, Level 16, State 0, Line 1
The INSERT statement conflicted with the FOREIGN KEY constraint
"fkVendor". The conflict occurred in database "test01", table
"dbo.xtblVendor", column 'VendorID'.
The statement has been terminated.
```

5) Drop the VendorMaterial table.

6) Create the VendorMaterial table again, this time with referential integrity constrained for both the VendorID and MaterialID foreign keys.

Table: xtblVendorMaterial				
Attribute Name	Data Type & Size	Primary Key	Foreign Key and Referential Integrity	Other constraints
VendorID	char(4)	yes	Yes – references xtblVendor	no null value
MaterialID	char(3)	yes	Yes – references xtblRawMaterial	no null value
CurrentPrice	money	no	no	no null value

```
CREATE TABLE     xtblVendorMaterial
(VendorID        char(4),
 MaterialID           char(3),
 CurrentPrice    money       NOT NULL,
 CONSTRAINT pkVendorMaterial
 PRIMARY KEY     (VendorID, MaterialID),
 CONSTRAINT      fkVendor
 FOREIGN KEY     (VendorID) REFERENCES xtblVendor (vendorID),
 CONSTRAINT      fkRM
 FOREIGN KEY     (MaterialID) REFERENCES xtblRawMaterial (MaterialID));
```

7) Try to execute the INSERT statement that worked correctly in step #3:

```
INSERT INTO xtblVendorMaterial VALUES
('7819', '288', 12.95);
```

You should get an error. You are now telling the DBMS to reference the Vendor table when it tries to input a vendorID, and the RawMaterial table when it tries to input a materialID. Both must pre-exist in those tables (the parent tables) before the data can be input into the VendorMaterial table (the child table).

8) Write the INSERT statement (or statements) to input the following data into the VendorMaterial table. Look back at Task #7, step #3 if you forgot the syntax to input multiple rows with a single INSERT statement.

VendorID	MaterialID	CurrentPrice
7819	255	14.25
62	255	13.95
0062	271	46.70
7819	240	.26

Task 9. Create the MaterialLocation table and learn about concatenated foreign keys.

Go back to the first page of this exercise and look at the ERD. Note that VendorMaterial is the parent table for the MaterialLocation table. Also note that the foreign key to link the two tables is a concatenation of the vendorID and material ID. The two attributes together represent ONE foreign key, not two foreign keys.

Table: tblMaterialLocation				
Attribute Name	Data Type & Size	Primary Key	Foreign Key and Referential Integrity	Other constraints
VendorID	char(4)	yes	yes – references xtblVendorMaterial	no null value
MaterialID	char(3)	yes	yes – references xtblVendorMaterial	no null value
LocationID	char(3)	yes	no	no null value
QuantityOnHand	decimal(8,3)	no	no	

1) Create the MaterialLocation table with the following SQL code.

```
CREATE TABLE      xtblMaterialLocation
(VendorID         char(4),
 MaterialID       char(3),
 LocationID       char(3),
 QuantityOnHand   decimal(8,3),
 CONSTRAINT pkMaterialLocation
 PRIMARY KEY      (VendorID, MaterialID, LocationID),
 CONSTRAINT fkVendorMaterial
 FOREIGN KEY      (VendorID, MaterialID)
 REFERENCES       xtblVendorMaterial (VendorID, MaterialID));
```

2) Write the INSERT statement (or statements) to input the following data into the MaterialLocation table. Look back at Task #7, step #3 if you forgot the syntax to input multiple rows with a single INSERT statement.

VendorID	MaterialID	LocationID	QuantityOnHand
7819	255	12	700.25
7819	255	15	600.88
7819	240	12	124
0062	271	81	5505
0062	240	81	6

You should receive an error when trying to enter the last row of the MaterialLocation table. What does the error mean? Why did it happen? What would you have to do to fix the error?

Task 10. Reflection.

1) What is the difference between a database object and SQL code?

2) How do you see the database objects you have created?

3) How do you type in SQL code?

4) How do you execute SQL code?

5) Where do you see the results from SQL code that you execute?

6) How do you save SQL code?

7) How do you save a database object?

8) How do you create a database table?

9) How do you put data into a database table? This is also called "populating" a table.

10) What is a database constraint?

11) What types of database constraints can you create when you create a table?

12) What is the difference between a foreign key and a referential integrity constraint?

13) How do you create a concatenated primary key?

14) How do you create a concatenated foreign key?

15) If you are maintaining referential integrity in a database, should you create the parent table or the child table first? Or does it matter?

16) If you are maintaining referential integrity in a database, should you populate the parent table or the child table first? Or does it matter?