IS475/675 HW#6: Creating and Populating Tables in a Database using SQL

Objectives

The purpose of this assignment is to create tables on SQL Server using the SQL Server Management Studio (SSMS) client. Tables will be created with constraints and then populated with data to produce a test database for a pre-defined application scenario.

Deliverables

Please upload to WebCampus the following on one Word (or .pdf) document for grading:

1) **The SQL CREATE TABLE statements for each table.** The SQL CREATE TABLE statements must reflect the tables shown at the end of this document. All table structures, field names, and data types are provided in this document. Do not make up your own data types for the fields in this assignment. Please use the prefix tbl for all table names. For example, the Vendor table should be created with the name tblVendor.

2) **The content of all tables. Execute a "SELECT * from table_name" in SQL for each table in the database.** Example:

```
SELECT      *
FROM        tblVendor;
```

Use a tool like the Windows Snipping Tool/Snip&Sketch to copy the result tables generated from the SELECT statements from SQL Server into your deliverable document.

3) **A very brief explanation of how you populated the tables.** If you used the Import/Export wizard, then say you used the Import/Export wizard. If you typed SQL INSERT statements, then say that. Just explain how you populated the tables.

Application Scenario

Assume that the company you are working with to produce this database is called MountainDesign, LLC, a sporting goods manufacturing and supply company. MountainDesign specializes in designing and manufacturing sporting goods for hiking, camping, and backpacking. MountainDesign makes and sells sporting goods for the public and distributed through general sporting goods stores, such as Scheels, REI, and Cabela's.

The application for this assignment is part of a purchase order system. The purchase order system is used to place orders with vendors for the products (also called items or materials) that MountainDesign uses to manufacture its sporting goods. This system is **NOT** a customer ordering system; this is **not** the system that MountainDesign uses to sell sporting goods to its customers. This is the system MountainDesign uses to buy materials from its vendors. MountainDesign usually refers to the materials it buys from vendors as products.

Your job for this assignment is to create nine tables in your existing database as described in this document and populate those tables with the data available in an Excel workbook called HW6Data.xlsx available on WebCampus and also on the k: drive in the student terminal server remote desktop environment. The folder it k:\IS475\MountainDesign. Please do not make up different data – you must use the data in the Excel workbook. I want everyone in class to be working from the same test dataset. You will use the same tables for the remaining homework assignments so you must complete this homework assignment in order to do the remaining homework assignments.

Methods of Table Creation and Population

You must write SQL CREATE TABLE commands to create your tables. This will help you understand the use of constraints (especially the referential integrity constraint) within a database.

The method of populating (inserting rows of data) tables is up to you. A simple, but somewhat tedious, method of table population is to code SQL INSERT commands. You must write one INSERT statement

for each row of each table you wish to populate. If you save your INSERT statements in a script file (a script file is just a plain old ASCII file created via a text editor such as WordPad, NotePad or EditPad), then you will be able to repopulate your tables quickly and easily in case there are problems with the data in the future.

You could use the Import/Export utility to import the data into SQL Server. The first SQL Lab exercise demonstrates how to use the Import/Export utility. I recommend that you populate your tables one at a time. Don't try to populate many tables at once – the process will fail due to the referential integrity constraints on the tables.

It is also OK to use the SQL Server Designer Utility to populate your tables. This utility is also demonstrated in the first SQL Lab exercise. The problem with this method of data input is that it will not work with a table that uses an IDENTITY value to populate the primary key. Thus, you will NOT be able to use the SQL Server Designer Utilities to populate either the Receiver or the Purchase History tables.

Feel free to use whatever method you want - just make sure you understand what you are doing to create and populate the tables accurately.

Note: Inaccurate output on the next assignments will be graded as wrong - even if the error is a result of a simple data input error on this assignment. You are responsible for the accuracy of your test data set!!

Assignment Issues to Review

A few issues concerning this assignment are discussed below.

1. **Do not create a database.** Each student has his/her own database already created – so there is no need to issue a create database SQL command in the COBA labs. For this assignment, you are creating tables in an existing database. Your starting place for this part of the assignment is to issue CREATE TABLE commands in SQL to create each of the nine tables.
2. **Don't use spaces in table names or column (field) names.** Really. Don't do it. If you love the look of spaces, use underscores. If you put a blank space in a field or table name, you will have to use square brackets ('[') every time you use that field or table name. Feel free to use upper and/or lower case in your table and field names. SQL is **not** case sensitive in table and field names.
3. **Enter the capitalization in the data values exactly as shown.** Some of the data are upper case and other are lower. Please enter the data exactly as provided in the Excel workbook – we will fix these data in a future assignment. SQL **is** case sensitive in data values.
4. **A null value and a blank value are not the same thing.** All blank data in Excel should be stored as NULL values in SQL Server.
5. **Plan how to populate the tables.** Data must be entered into strong (parent) tables before it is entered into weak (child) tables. The tables are not presented in this handout in the order they must be created and populated in the database; they are not presented in any particular order. To create and populate the tables, think about the concept of referential integrity and its implementation through foreign keys. An example of the concept is discussed below.

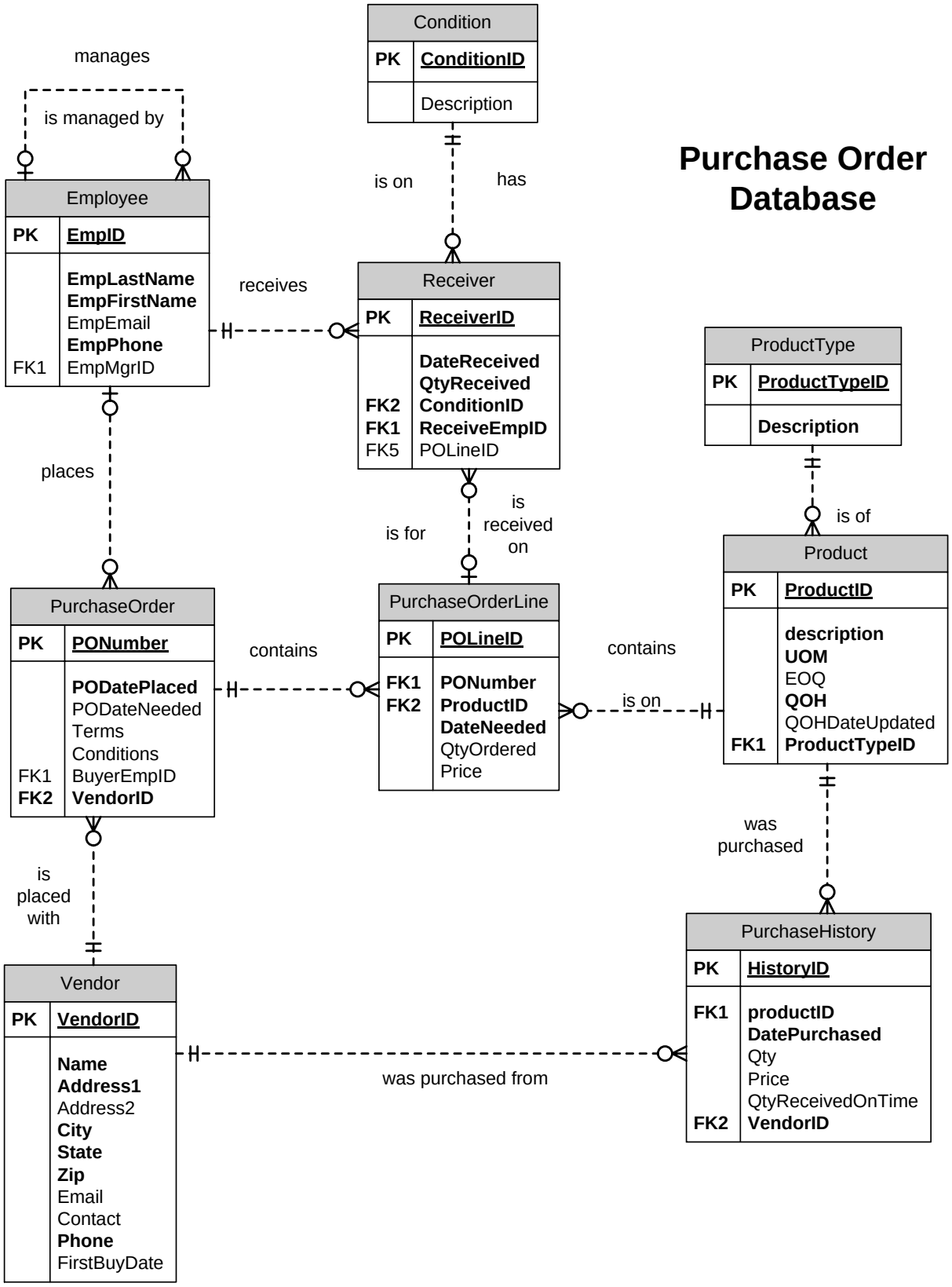
   > Many of the tables in this database are related to each other. One of those relationships, for example, is between the Vendor and PurchaseOrder tables. The business rule for this relationship is: A given purchase order must be placed with only one existing vendor (mandatory cardinality). Thus, a vendor must be entered into the Vendor table before a purchase order placed with that vendor can be entered into the PurchaseOrder table. The vendor is the parent table and the purchase order is the child table in this relationship. This business rule is established through a foreign key of the VendorID in the PurchaseOrder table. When a new purchase order is entered in the PurchaseOrder table (the child), the DBMS will check the Vendor table (the parent) to make sure that the vendor exists before permitting the entry into the PurchaseOrder table. If the vendor does not exist, the DBMS will not allow the purchase order to be entered into the PurchaseOrder table. This is a referential integrity constraint placed on the PurchaseOrder table.

A solution to this issue is to create the tables without referential integrity constraints, populate the tables, and then add the referential integrity constraints. A problem with that solution is that the DBMS will not validate the data as you populate the tables. For example, if you enter a purchase order with an invalid VendorID (a typo), the DBMS will not validate the VendorID and will let you enter the invalid data. Thus, you are responsible for validating the data as it is entered. Not the world's best solution.

A better solution is to differentiate parent tables and child tables by looking at the ERD (provided in this document). Sometimes it can be difficult to differentiate parent from child because a table may be a parent to some tables while also being a child to other tables. For example, the PurchaseOrderLine table is a child table when related to the PurchaseOrder table and Product table, but it is a parent table when related to the Receiver table. The PurchaseOrderLine table must be populated after the PurchaseOrder and Product tables, but before the Receiver table. This will take planning on your part, but it works!

6. **Be careful with quotation marks.** Microsoft Word and other word processing packages use "smart quotes," which are pretty quotation marks that curl forwards and backwards. The SQL programming language does not like smart quotes – so check your work to make sure that the quotation marks are straight and not curved. SQL prefers single, rather than double, quotation marks for almost all of the syntax.
7. **Be wary of the data types declared in Excel.** When data is imported into Excel (as I did with the data for this assignment) it is usually imported as a general data type. Even though the data looks correct, it may not be correct because of the miscommunication of data types between Excel and SQL Server. Be sure the types are correct in Excel before trying to use the Import/Export utility in SQL Server.
8. **Enter character fields as character strings.** Computers differentiate character strings from numbers, even if numeric characters are stored in a field declared as a character data type. For example, the PONumber in this database is a character field (char) type. If you enter a number in a character field (meaning that you enter the data without quotation marks surrounding it) then SQL Server will store it as a number. So if the PONumber is 025974 and you enter it as a number when using the INSERT command (without quotes) then SQL Server will store it as 25,974. If you enter it as a character string '025974,' then SQL Server will store the leading zero. Be consistent – either enter all data as character strings to retain the leading zeros or enter it as numeric and eliminate the leadings zeros. Be consistent!

Purchase Order Database



Structures for Tables

Table: tblVendor				
Attribute Name	Data Type & Size	Primary Key	Foreign Key	Other Constraints
VendorID	char(5)	Yes	no	no null value
Name	Varchar(30)	No	No	No null value
Address1	varchar(30)	No	no	no null value
Address2	varchar(30)	No	no	
City	varchar(20)	No	no	no null value
State	char(2)	No	no	no null value
Zip	varchar(12)	No	no	no null value
Email	varchar(30)	No	no	
Contact	varchar(30)	No	no	
Phone	char(15)	No	no	no null value
FirstBuyDate	Datetime	No	no	no null value

Table: tblEmployee				
Attribute Name	Data Type & Size	Primary Key	Foreign Key	Other Constraints
EmpID	char(6)	Yes	No	no null value
EmpLastName	varchar(30)	No	No	no null value
EmpFirstName	varchar(30)	No	No	no null value
EmpEmail	varchar(30)	No	No	
EmpPhone	char(15)	No	No	no null value
EmpMgrID	char(6)	No	yes – reference Employee table	

Table: tblCondition				
Attribute Name	Data Type & Size	Primary Key	Foreign Key	Other Constraints
ConditionID	char(2)	Yes	no	no null value
Description	varchar(30)	No	no	no null value

Table: tblPurchaseOrder				
Attribute Name	Data Type & Size	Primary Key	Foreign Key	Other Constraints
PONumber	char(6)	Yes	no	no null value
PODatePlaced	Datetime	No	no	no null value
PODateNeeded	Datetime	No	no	
Terms	varchar(15)	No	no	
Conditions	varchar(15)	No	no	
BuyerEmpID	char(6)	No	yes – reference Employee table	
VendorID	char(5)	No	yes – reference Vendor table	no null value

Table: tblProductType				
Attribute Name	Data Type & Size	Primary Key	Foreign Key	Other Constraints
ProductTypeID	char(2)	Yes	no	no null value
Description	varchar(30)	No	no	no null value

Table: tblProduct				
Attribute Name	Data Type & Size	Primary Key	Foreign Key	Other Constraints
ProductID	char(5)	Yes	no	no null value
Description	varchar(30)	No	no	no null value
UOM	char(10)	No	no	valid values are “each”, “feet”, “inches”, “meters”, “cm”, “sheet,” “case”.
EOQ	decimal(6,2)	No	no	
QOH	decimal(8,2)	No	no	no null value
QOHDateUpdated	datetime	no	no	
ProductTypeID	char(2)	No	yes – reference ProductType table	no null value

Table: tblReceiver				
Attribute Name	Data Type & Size	Primary Key	Foreign Key	Other Constraints
ReceiverID	int	Yes	No	This is a surrogate primary key, so it should have an identity characteristic. Seed = 1, increment = 1
DateReceived	datetime	No	No	no null value
QtyReceived	decimal(6,2)	No	No	must be greater than 0
ConditionID	char(2)	No	yes – reference Condition table	no null value
ReceiveEmpId	Char(6)	No	Yes – reference Employee table	
POLineID	int	No	Yes – reference PurchaseOrderLine	

Table: tblPurchaseOrderLine				
Attribute Name	Data Type & Size	Primary Key	Foreign Key	Other Constraints
POLineID	int	Yes	no	no null value
PONumber	char(6)	No	yes – reference PurchaseOrder table	no null value
ProductID	char(5)	No	yes – reference Product table	no null value
QtyOrdered	decimal(6,2)	No	No	must be greater than 0
Price	money	No	No	no null value
DateNeeded	datetime	No	No	No null value

Table: tblPurchaseHistory				
Attribute Name	Data Type & Size	Primary Key	Foreign Key	Other Constraints
HistoryID	int	Yes	No	This is a surrogate primary key, so it should have an identity attribute. Seed = 1, increment = 1
ProductID	char(5)	no	yes – reference Product table	no null value
DatePurchased	datetime	no	No	no null value
Qty	decimal(8,2)	no	No	no null value
Price	money	no	No	no null value
QtyReceivedOnTime	decimal(8,2)	no	No	
VendorID	char(5)	no	yes – reference Vendor table	