SYSTEMS DESIGN / CAPSTONE PROJECT
MIS 413

User Guide 2.1
Creating a SELECT Stored Procedure

Stored procedures are an effective way to access and modify data in database tables. Not only do they run faster than standard SQL queries, they **provide an increased level of security**. This is because stored procedures utilize 'parameters' for the actual modification of data in tables, while standard SQL coding in programs expose the database to potential injection of harmful SQL code.

Example One: Select All Rows from One Table Only

The easiest manner to create a stored procedure is to let the wizard in MS SQL assist you.

1) Open MS SQL and the proper database
2) Expand the 'programmability' tab under your database
3) Right click on stored procedures and click 'new stored procedure'
4) Revise the comments section for your name, date and description.
5) Replace the default name with the name of your new stored procedure (remove the < > in the create procedure command line)

```
-- ============================================
-- Author:      <MIS 413 Class>
-- Create date: <August 2012>
-- Description: <This procedure will select all the valid Person Types>
-- ============================================
CREATE PROCEDURE validPersonTypesSelectAll
    -- Add the parameters for the stored procedure here

AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    SET NOCOUNT ON;
```

Consistent and proper naming of your stored procedure will help you and future developers understand their purpose. A guideline for this class would be:
Name of the Table, the Operation (select, update, etc.), Optional Word(s) as in:

personSelectAll
personSelectOne
personUpdate
personDeleteOne
personRolesInsert
validRolesSelectAll

6) To build your first SELECTs, UPDATEs, etc, in the Query Editor Option. Delete the following row in your Stored Procedure:

```
SELECT <@Param1, sysname, @p1>, <@Param2, sysname, @p2>
```

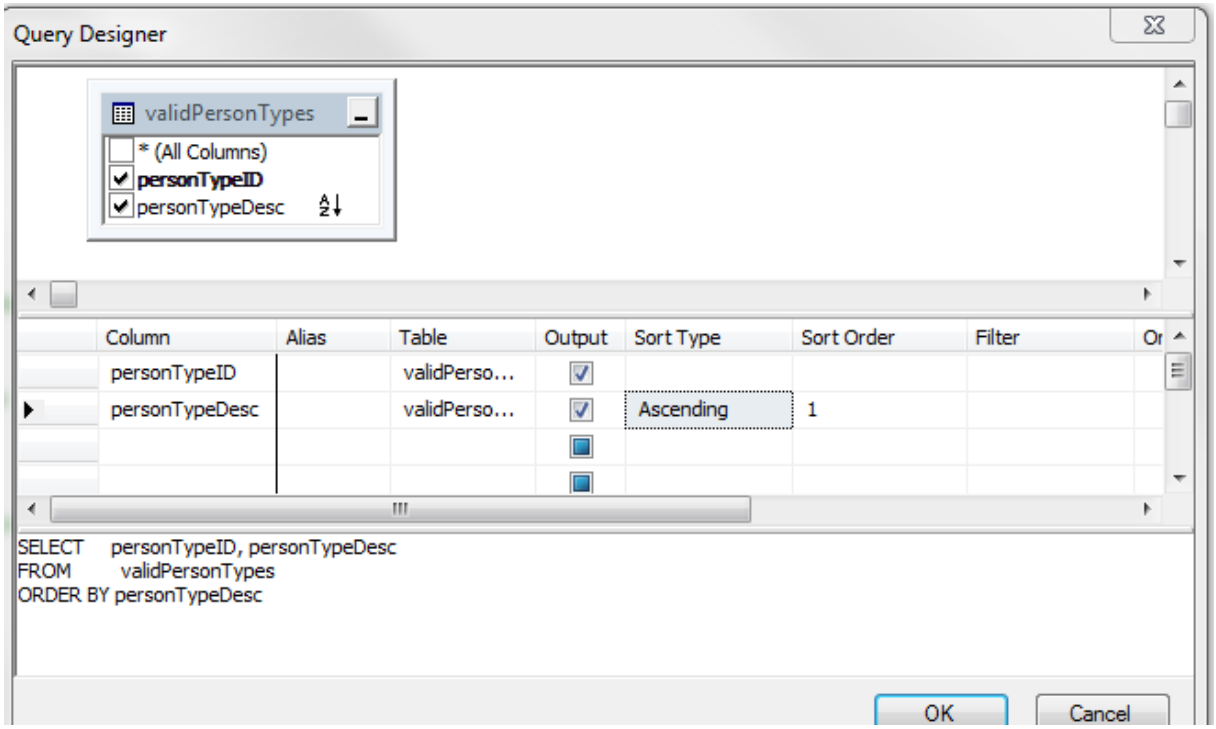
7) Then right click in the blank area and select DESIGN IN QUERY EDITOR

   A screen similar to building a ***Query by Example in Access*** will appear. A dialog box will appear with the names of the available tables. Select the table or tables for your query (you can always add more tables later).

8) Once you have the table in the designer, select the fields you desire to return to your application. **Avoid selecting the * (all columns).**

9) Also you may want to include a sort (order by) which is generally done on the description column.

10) See the diagram below for a sample validPersonTypesSelectAll Stored Procedure



11) Click OK and you will see the SQL code the wizard has created.

12) For the above select stored procedure you have no input parameters as we want to select ALL records; thus delete the parameter lines between the CREATE and the AS. (keeping the CREATE and AS statements)

13) Execute the SQL commands, (which actually RUNS the CREATE SQL statement, thus saving it in your database) by clicking the **red !** point. Modifications to the stored procedure from this point forward would be done with an ALTER name of procedure COMMAND as in **ALTER validPersonTypesSelectAll**

14) Test the stored procedure, by refreshing the list of stored procedures in the left column and then right click on the name of the stored procedure you just created and then EXECUTE. You should see the results of your stored procedure

Example Two: Select One Row based on an input parameter

In this example we want to display employer information for only one employer. It is a similar process as the previous however we will need to add a WHERE clause as well as allow for an input parameter that the ASP.NET program will feed the stored procedure:

1) Right click on stored procedures and click 'new stored procedure"
2) Revise the comments section for your name, date and description.
3) Provide an appropriate name, in our case employersSelectOne

4) Provide an input parameter. The input parameter should be the name of the column/field for the WHERE clause preceded with an @ sign and followed by the input field type,

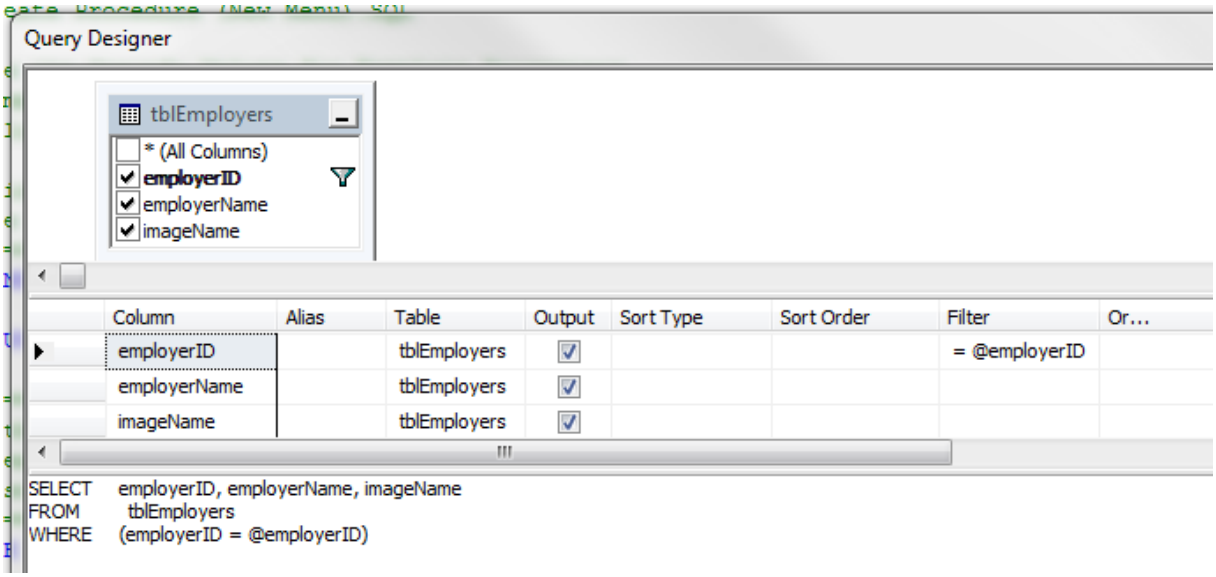
Replace the parameter place holder with the name of your input parameter.

```
-- Add the parameters for the stored procedure here
<@Param1, sysname, @p1> <Datatype_For_Param1, , int> =
<Default_Value_For_Param1, , 0>,
```

Delete the above and replace with:

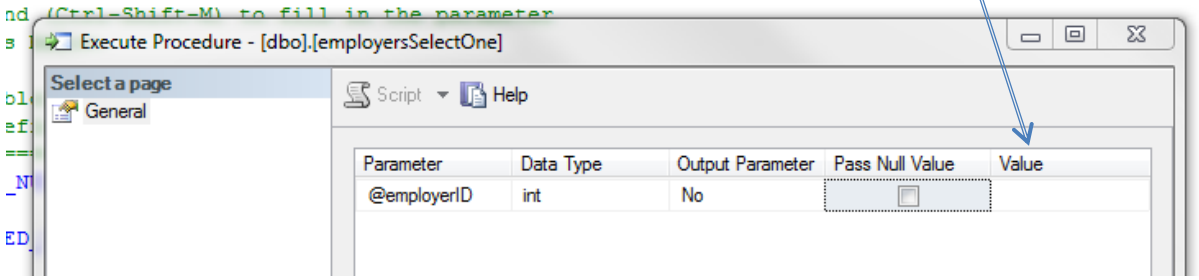
```
-- Add the parameters for the stored procedure here
```
@employerID int

5) Use the Query Editor to select the appropriate fields and add a WHERE clause as in:



6) Click the red Execute to create/save your new procedure

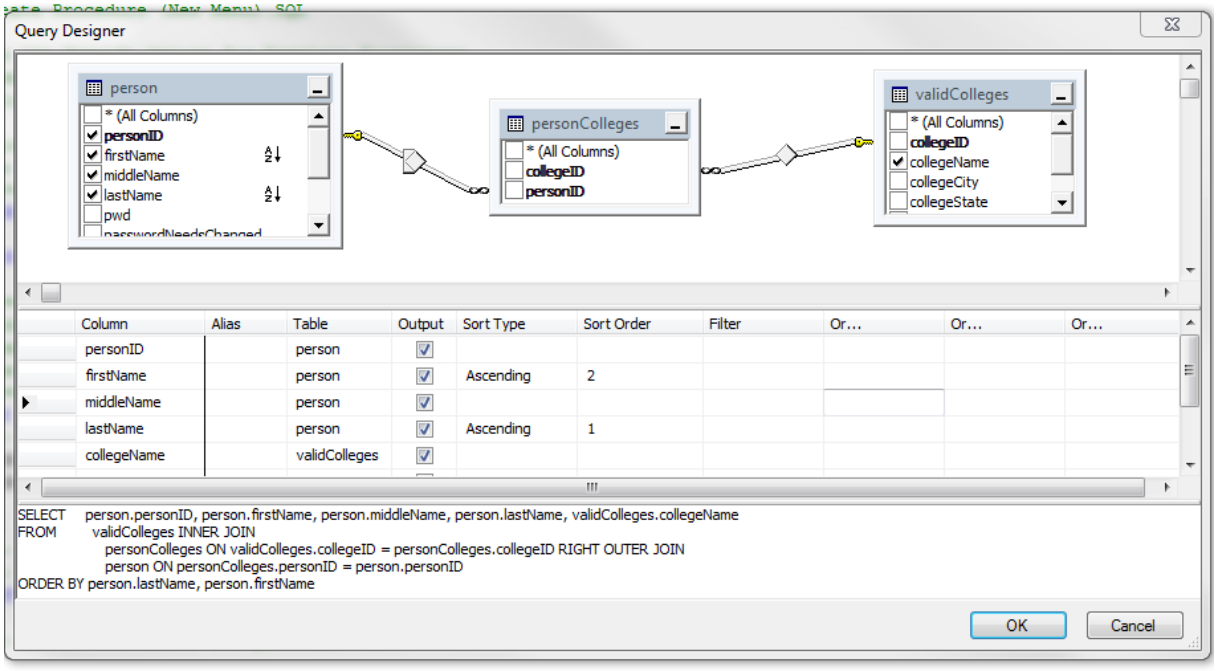
15) Test the stored procedure, by refreshing the list of stored procedures and right clicking on the name of the stored procedure you just created and then EXECUTE. However you will now need to provide an input parameter, once you hit EXECUTE the following screen will appear:



Type a valid EmployerID in the value column and the hit OK and you should see the results for one row.

Example Three: Selecting all from more than one table with a RIGHT/LEFT JOIN

In this example we will use the relationships already built to link two or more tables so that data may be shown from more than one table. Assume we have a person table and we want to know what colleges they have attended. However we might have some people who have not gone to college to date or individuals who have gone to more than one college, thus the reason for the associated table (personColleges).



Notice we have 3 tables including an associated table between colleges and person. Our goal is to show all individuals and their colleges. We also desire to show the names of all individuals in the person table even if they have not attended college.

If we use the normal process all the links would be INNER JOINS - which implies if a person has not gone to college and therefore would not be an entry in the personCollege table they would not be displayed.

The solution is to create a LEFT or RIGHT INNER JOIN. Let Query Editor assist you. Right click on the link between PERSON and PERSONCOLLEGES and select the option to Select all from the Person Table; the editor will then build the proper JOIN statement.

How do you know which table should be the ALL FROM table? You want to select the primary table where you want the data to display even if the data is not in the other tables. Thus we wanted to show all people even if they have not attended college, therefore they would be in the person table but might not be in the personCollege table.