SYSTEMS DESIGN / CAPSTONE PROJECT
MIS 413

User Guide 2.2

Creating Update, Insert and Delete Stored Procedures

This guide will demonstrate building Insert, Update and Delete Stored Procedures with a few input parameters using the Query Editor Tool. They all assume you have a person table with a personID (auto number), FN, LN, Email, PWD attributes.

Example A: Update Stored Procedure

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

1) Open 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) 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:
personUpdate

5) Add your name of the procedure (without the < > brackets) as in:
CREATE PROCEDURE personUpdate in the first row of the stored procedure.

6) To build your first UPDATE, use the Design in 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 dialog box will appear and select the name of the table to be updated. A general rule is you will only **UPDATE one table at a time**. Do not try to UPDATE 2 or more tables with one UPDATE statement. If you need to update more than one table, you can do two separate UPDATE statements in the same Stored Procedure.

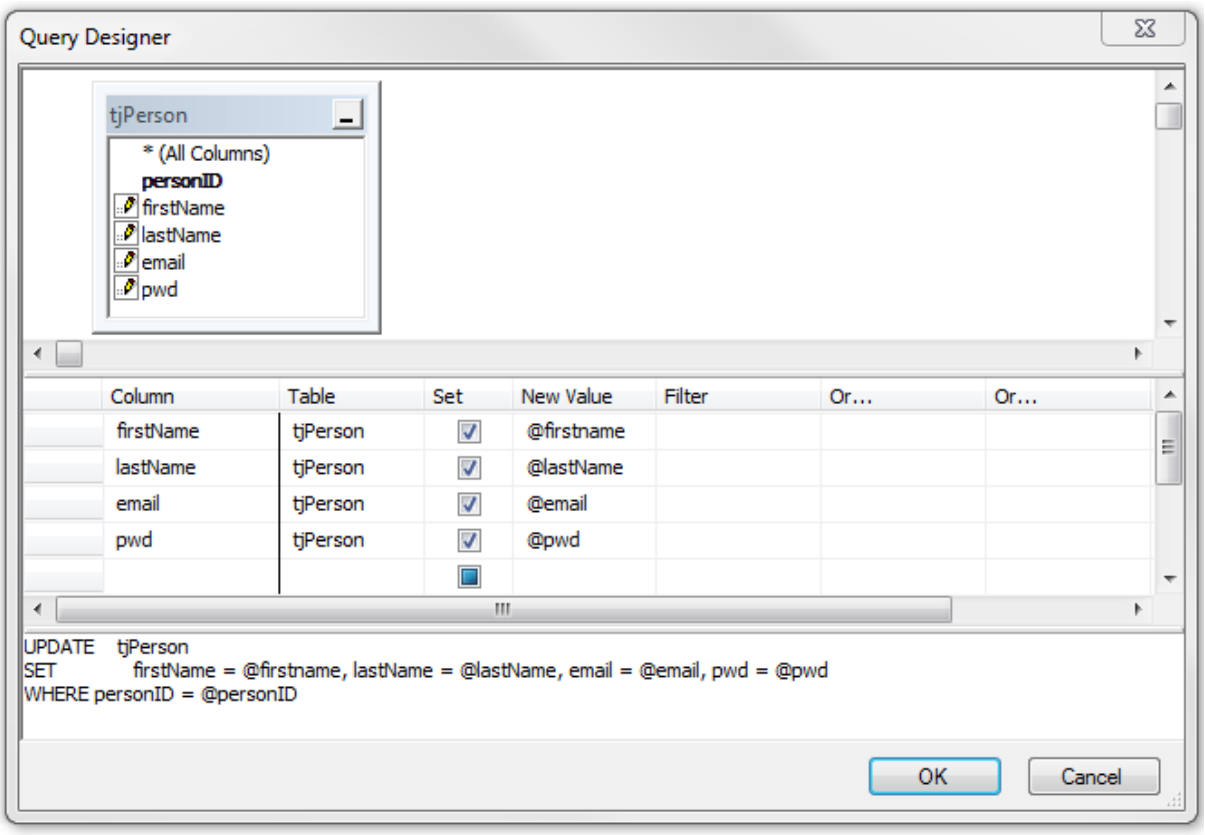
8) Right click in the space to the right of the table in your diagram and select the **CHANGE TYPE** and then **UPDATE** option.

9) Select the Columns to be Updated, notice the Primary Key will be bolded; you SHOULD NOT update the Primary Key as this will become the WHERE clause.

10) In the NEW VALUE column repeat the name of the column name with an @ as the first character, as in: @firstname

11) In the SQL code box, add in the WHERE clause based on the Primary Key as in: WHERE personID = @personID.

12) When complete your Query Editor should look like the figure below:



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

14) Now we need to tell the stored procedure that we will be passing it the values to UPDATE through the parameter list. Below the comment in green that states, add parameters, add your five input parameters as in:

@personID int,
@firstname varchar(30)

HINTS:
There needs to be a comma at the end of each row except the last parameter, in addition the parameter must define the field type and size.

To easiest way to see the field types and size is to expand the PLUS sign beside the table name if the left column menu, then beside the column names, and you will see the names of all fields, and their types and sizes.

15) When complete your Stored Procedure should look like the next diagram.

```
create PROCEDURE [dbo].[personUpdate]
  -- Add the parameters for the stored procedure here
       @personID int,
       @firstname varchar(30),
       @lastname varchar(50),
       @email varchar(75),
       @pwd varchar(12)
AS
BEGIN
  -- SET NOCOUNT ON added to prevent extra result sets from
  -- interfering with SELECT statements.
  SET NOCOUNT ON;

    -- Insert statements for procedure here
  UPDATE    person
  SET       firstName = @firstname, lastName = @lastName, email =
@email, pwd = @pwd
  WHERE     (personID = @personID)
END
```

16) Click the Red Exclamation Point to CREATE/SAVE your stored procedure

17) 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. You will need to provide a valid PK and the new values for your record. To see if your procedure worked; open the table and display 200 rows to see if the values have changed.

Use Case Exception to consider.

On some UPDATES you will need to see if that item already exists before you change a record. For example if you are modifying an email address to a new address, perhaps that address already exists in the database person table. We will discuss implementation of addition indexes in the table design to help solve this Use Case Exception.

Example B: Insert Stored Procedure

In this example we assume the primary key for our person table is auto generated and thus we will not input a value for the ID field. It also assumes you have a UPDATE stored procedure built, so we will use it to help create the INSERT stored procedure quicker.

1. Locate your UPDATE stored procedure in the Object Explorer list, and right click on it and then MODIFY

2. Immediately change the ALTER name of the stored procedure to enable it to CREATE a new INSERT procedure as in:

   CREATE procedure personINSERT

3. Highlight the UPDATE statements, down to and including the WHERE clause and right click to go into QUERY EDITOR

4. In the top panel right click and select **CHANGE TASK** and then **INSERT VALUES**

5. Notice it will then convert the UPDATE code to INSERT using the names of your parameters you typed while doing the UPDATE procedure

6. Click OK and your code will be done.

7. Finally, since the PK is generated automatically, delete the @personID int row from the Input Parameter List and you are ready to save this completed INSERT procedure.

```
create PROCEDURE [dbo].[personInsert]
     -- Add the parameters for the stored procedure here

          @firstname varchar(30),
          @lastname varchar(50),
          @email varchar(75),
          @pwd varchar(12)
AS
BEGIN
     -- SET NOCOUNT ON added to prevent extra result sets from
     -- interfering with SELECT statements.
     SET NOCOUNT ON;

    -- Insert statements for procedure here
          INSERT INTO Person
       (firstName, lastName, email, pwd)
          VALUES     (@firstname,@lastName,@email,@pwd)
END
```

8. Test the procedure before proceeding.

Use Case Exception to consider.

On INSERTS you will need to see if that item already exists before you change a record. For example if you are inserting a new STATE, perhaps that state already exists in the validStates table. We will discuss creating additional indexes in tables to help solve this Use Case Exception.

Example C: Delete Stored Procedure

Deletes are the easiest stored procedures to write and generally you will not need the Query Editor. You would normally have ONE input parameter, the PK. Following is an example of a typical delete:

```
create PROCEDURE [dbo].[personDelete]
  -- Add the parameters for the stored procedure here

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

    -- Insert statements for procedure here
  DELETE from Person
    WHERE personID = @personID
END
```

Use Case Exception to consider.

On Deletes (especially with valid tables) you need to consider any Foreign Keys to other table that rely on the entry you are deleting for description or other values. For example if you are delete from the validMajors Table and you delete the ACG major, there may be students who have the major of ACG, thus if you deleted the ACG major they would have a pointer to a row in the validMajors table that no longer exists. We will discuss other means to help solve this Use Case Exception.