How do I generate CRUD stored procedures from a table in SQL Server Management Studio

Ask Question

11

5

How do I take a table, and auto-gen CRUD stored procs for it in SSMS?

sql sql-server tsql ssms crud

edited Jul 31 '18 at 17:09
Aaron Bertrand
212k 27 369 408

asked Jul 26 '12 at 16:13
The Internet
5,128 5 42 75

I don't believe it has this functionality. – Joe Jul 26 '12 at 16:15

7 Answers

SSMS doesn't have the capability to generate CRUD procedures. You can generate INSERT, UPDATE statements etc. by right-

20

clicking, Script Table As > but I think you will have better luck with Mladen Prajdic's SSMS Tools Pack.

answered Jul 26 '12 at 16:15

Aaron Bertrand
212k 27 369 408

2 This is a god send – The Internet Jul 26 '12 at 18:58

2 @DavidJohnson glad it helped. And if you think CRUD is useful, wait until SSMS crashes and doesn't recover your files. Mladen's got you covered with very flexible and powerful auto-save options. – Aaron Bertrand Jul 26 '12 at 19:05

Yea, that kind of makes me want to cry. There is no panacea. – The Internet Jul 26 '12 at 19:17

Just stumbled onto this post and want to say thank you :). This addon/tool is a brilliant find. +1 – Tay Sep 5 '13 at 10:21

20

If you are using Visual Studio you can do it:
http://weblogs.asp.net/stevewellens/archive/2009/12/11/automatically-generate-stored-procedures-with-visual-studio.aspx

answered Jul 26 '12 at 16:24

Steve Wellens
19.2k 2 20 57

This looks very helpful as well. Early bird gets the worm though – The Internet Jul 26 '12 at 18:59

7 Yes, but it's the second mouse that gets the cheese. – Steve Wellens Jul 27 '12 at 0:48

1 Ah dang it you're right. – The Internet Jul 27 '12 at 2:32

Damn good answer, giving it a link from here : stackoverflow.com/questions/3728641/... – Arjang Nov 21 '12 at 6:01

15

I have a simple TSQL script I use to do mine. It's basic but easily modified to suit your needs. It generates TSQL for an Upsert, Select, and Delete Procedure using table & view you specify.

```
    /*
This is a simple, single-table CRUD Generator. It does not have a bui
bells and whistles, and is easy to follow and modify. I wrote this t
my job easier, and I am sharing it with you to do with it as you wis

The Basics:
The TSQL below will create 3 procedures:
    1. An Upsert Procedure: Suffix _ups
    2. A Select Procedure: Suffix _sel
    3. A Delete Procedure: Suffix _del

A Little More Detail:
Things you should know:
    All 3 procedures have a parameter called @MyID which is used to
    the Context, so that my audit procedures get the validated user.
    Have no use for it, you'll need to remove the piece of generator
    that adds it as a parameter to each of the 3 procedures. You wil
    need to remove the PRINT statement for each procedure that looks

            PRINT N'  SET CONTEXT_INFO @MyID;' + CHAR(13) + CHAR(10)

    This generator expects to perform inserts, updates, and deletes
    table, and selects from a view. If you want to perform selects d
    from the table, simply use the table name in both @TableName and
    @ViewName.

The Upsert Procedure:
    If ID (Primary Key) is supplied it will perform an Update. Other
    will perform an Insert. This generator is hard-coded to avoid in
    or updating these particular fields:
                                    Created
                                    CreatedBy
                                    Modified
                                    ModifiedBy
                                    RowVersion
                                    <The Primary Key Field>
    That's because in my databases I use those field names for audit
    are never modified except internally within the database. You ca
    the part of this procedure that performs this function to suit y
    This generator always uses the Parameter name @ID to represent tl
    key defined for the table. This is my preference but you can mod

The Select Procedure:
    If ID (Primary Key) is supplied it will select a single row from
    (Table) whose name you provide. Otherwise it will select all rows
    @ISACTIVE_SEL variable is set to 1 (True), then the Select Proce
    your View (Table) to have a bit-type field named 'IsActive'. My
    standardized to this. If @ISACTIVE_SEL = 1 the Select Procedure
    additional parameter called @IsActive (bit). When @ID is not supp
    @IsActive is not supplied, the procedure selects all rows. When (
    supplied, and @IsActive is supplied, the procedure selects all ro
    the field IsActive matches the parameter @IsActive

The Delete Procedure:
    The Delete Pocedure requires that the Key value and the RowVersi
    be supplied. I use an Int type RowVersion, so if you use TimeSta
    then you will need to tweak the generator.


    --Casey W Little
    --Kaciree Software Solutions, LLC
    Version 1.00
*/


--Type Your Database Name in this Use statement:
USE [<Your Database>]
GO

SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

/*MODIFY THE VALUES BELOW TO SUIT YOUR NEEDS*/
DECLARE @DBName nvarchar(100)=N'<Your Database>';
DECLARE @ProcName nvarchar(100)=N'<Your Proc Name>';
DECLARE @DBRoleName nvarchar(100)=N'<Role that should have exec Righ
DECLARE @TableName nvarchar(100)=N'<Your Table Name>';
DECLARE @ViewName nvarchar(100)=N'<Your View Name>';
DECLARE @OrderBy nvarchar(100)=N'<Your Field Name>';
DECLARE @OrderByDir nvarchar(4)=N'ASC';
DECLARE @AUTHOR nvarchar(50) ='<Your Name & Company>';
DECLARE @DESC nvarchar(100) ='<Proc Information>'; -- Ex. 'User Data
'Description : Upsert User Data'
```

```
DECLARE @ISACTIVE_SEL bit =0; --Set to 1 if your table has a Bit fie
/*DO NOT MODIFY BELOW THIS LINE!!!*/

DECLARE @NNND char(23) ='NOT_NULLABLE_NO_DEFAULT';
DECLARE @NNWD char(22) ='NOT_NULLABLE_W_DEFAULT';
DECLARE @NBLE char(8) ='NULLABLE';
DECLARE @LEGEND nvarchar(max);
DECLARE @PRIMARY_KEY nvarchar(100);


--Set up Legend
    SET @LEGEND = N'USE [' + @DBName + N'];' + CHAR(13) + CHAR(10)
    SET @LEGEND = @LEGEND + N'GO' + CHAR(13) + CHAR(10)
    SET @LEGEND = @LEGEND + CHAR(13) + CHAR(10)
    SET @LEGEND = @LEGEND + N'SET ANSI_NULLS ON' + CHAR(13) + CHAR(1
    SET @LEGEND = @LEGEND + N'GO' + CHAR(13) + CHAR(10)
    SET @LEGEND = @LEGEND + CHAR(13) + CHAR(10)
    SET @LEGEND = @LEGEND + N'SET QUOTED_IDENTIFIER ON' + CHAR(13) +
    SET @LEGEND = @LEGEND + N'GO' + CHAR(13) + CHAR(10)
    SET @LEGEND = @LEGEND + CHAR(13) + CHAR(10)

    SET @LEGEND = @LEGEND + N'--
==================================================================='
CHAR(10)
    SET @LEGEND = @LEGEND + N'-- Author      : ' + @AUTHOR + CHAR(13
    SET @LEGEND = @LEGEND + N'-- Create date : ' + CONVERT(nvarchar(
CHAR(13) + CHAR(10)
    SET @LEGEND = @LEGEND + N'-- Revised date: ' + CHAR(13) + CHAR(1

--Get Primary Key Field
SELECT TOP 1 @PRIMARY_KEY = COLUMN_NAME
FROM INFORMATION_SCHEMA.KEY_COLUMN_USAGE
WHERE OBJECTPROPERTY(OBJECT_ID(constraint_name), 'IsPrimaryKey') = 1
@TableName AND TABLE_CATALOG = @DBName;

DECLARE TableCol Cursor FOR
SELECT c.TABLE_SCHEMA, c.TABLE_NAME, c.COLUMN_NAME, c.DATA_TYPE,
c.CHARACTER_MAXIMUM_LENGTH
    ,
IIF(c.COLUMN_NAME='RowVersion',@NBLE,IIF(c.COLUMN_NAME=@PRIMARY_KEY,(
 = 'NO' AND c.COLUMN_DEFAULT IS NULL,@NNND,IIF(c.IS_NULLABLE = 'NO' 
IS NOT NULL,@NNWD,@NBLE)))) AS [NULLABLE_TYPE]
FROM INFORMATION_SCHEMA.Columns c INNER JOIN
    INFORMATION_SCHEMA.Tables t ON c.TABLE_NAME = t.TABLE_NAME
WHERE t.Table_Catalog = @DBName
    AND t.TABLE_TYPE = 'BASE TABLE'
    AND t.TABLE_NAME = @TableName
```

```
ORDER BY [NULLABLE_TYPE], c.ORDINAL_POSITION;

DECLARE @TableSchema varchar(100), @cTableName varchar(100), @Columnl
DECLARE @DataType varchar(30), @CharLength int, @NullableType varchar

DECLARE @PARAMETERS nvarchar(max);
DECLARE @INSERT_FIELDS nvarchar(max),@INSERT_VALUES nvarchar(max);
DECLARE @UPDATE_VALUES nvarchar(max);

SET @PARAMETERS ='@MyID int,';
SET @INSERT_FIELDS ='';
SET @INSERT_VALUES ='';
SET @UPDATE_VALUES ='';

-- open the cursor
OPEN TableCol

-- get the first row of cursor into variables
FETCH NEXT FROM TableCol INTO @TableSchema, @cTableName, @ColumnName
@CharLength, @NullableType

WHILE @@FETCH_STATUS = 0
    BEGIN
        IF @ColumnName NOT IN('Created','CreatedBy','Modified','Modi
        BEGIN
            SET @PARAMETERS=@PARAMETERS + '@' +
IIF(@ColumnName=@PRIMARY_KEY,'ID',@ColumnName) + ' ' + iif(@CharLeng
NULL,@DataType,@DataType + '(' +
                CAST(@CharLength AS nvarchar(10)) + ')') +  IIF(@Nul
@NullableType=@NNWD,',','=NULL,');
            IF @ColumnName <> @PRIMARY_KEY AND @ColumnName <> N'RowV
                BEGIN
                    SET @INSERT_FIELDS=@INSERT_FIELDS + '[' + @Colum
                    SET @INSERT_VALUES=@INSERT_VALUES + '@' +
IIF(@ColumnName=@PRIMARY_KEY,'ID',@ColumnName) + ',';
                    SET @UPDATE_VALUES=@UPDATE_VALUES + '[' + @Colum
IIF(@ColumnName=@PRIMARY_KEY,'ID',@ColumnName) + ',';
                END
        END

        FETCH NEXT FROM TableCol INTO @TableSchema, @cTableName, @Co
@CharLength, @NullableType
    END;

    SET @PARAMETERS=LEFT(@PARAMETERS,LEN(@PARAMETERS)-1)
    SET @INSERT_FIELDS=LEFT(@INSERT_FIELDS,LEN(@INSERT_FIELDS)-1)
    SET @INSERT_VALUES=LEFT(@INSERT_VALUES,LEN(@INSERT_VALUES)-1)
    SET @UPDATE_VALUES=LEFT(@UPDATE_VALUES,LEN(@UPDATE_VALUES)-1)

-- ----------------
-- clean up cursor
-- ----------------
CLOSE TableCol;
DEALLOCATE TableCol;

--Print Upsert Statement
    PRINT N'/****** Object:  StoredProcedure [dbo].[' + @ProcName +
Date: ' + CAST(GETDATE() AS nvarchar(30)) + '  ******/' + CHAR(13) +
    PRINT @LEGEND;
    PRINT N'-- Description : Upsert ' + @DESC + CHAR(13) + CHAR(10)
    PRINT N'-- =====================================================
CHAR(13) + CHAR(10)
    PRINT CHAR(13) + CHAR(10)
    PRINT N'CREATE PROCEDURE [dbo].[' + @ProcName  + '_ups]' + CHAR(
    PRINT N'  (' + @PARAMETERS + N')' + CHAR(13) + CHAR(10);
    PRINT N'AS' + CHAR(13) + CHAR(10)
    PRINT N'BEGIN' + CHAR(13) + CHAR(10)
    PRINT N'  SET CONTEXT_INFO @MyID;' + CHAR(13) + CHAR(10)
    PRINT N'  IF @ID IS NULL OR @ID = 0' + CHAR(13) + CHAR(10)
    PRINT N'    BEGIN' + CHAR(13) + CHAR(10)
    PRINT N'      INSERT INTO [dbo].[' + @TableName + ']' + CHAR(13)
    PRINT N'        (' + @INSERT_FIELDS + N')' + CHAR(13) + CHAR(10)
    PRINT N'      VALUES' + CHAR(13) + CHAR(10)
    PRINT N'        (' + @INSERT_VALUES + N');' + CHAR(13) + CHAR(10
    PRINT N'      SELECT * FROM [dbo].[' + @ViewName + '] WHERE [ID]
SCOPE_IDENTITY();' + CHAR(13) + CHAR(10)
    PRINT N'    END' + CHAR(13) + CHAR(10)
    PRINT N'  ELSE' + CHAR(13) + CHAR(10)
    PRINT N'    BEGIN' + CHAR(13) + CHAR(10)
    PRINT N'      UPDATE [dbo].[' + @TableName + ']' + CHAR(13) + CH
    PRINT N'        SET ' + @UPDATE_VALUES + CHAR(13) + CHAR(10)
    PRINT N'        WHERE ([' + @PRIMARY_KEY + '] = @ID) AND ([RowVe
@RowVersion);' + CHAR(13) + CHAR(10)
    PRINT N'      SELECT * FROM [dbo].[' + @ViewName + '] WHERE [ID]
+ CHAR(10)
    PRINT N'    END' + CHAR(13) + CHAR(10)
    PRINT N'END' + CHAR(13) + CHAR(10)
    PRINT N'GO' + CHAR(13) + CHAR(10)
    PRINT CHAR(13) + CHAR(10)

----Now add GRANT and DENY permissions to the Role
    PRINT N'GRANT EXECUTE ON [dbo].[' + @ProcName + '_ups] TO [' + @
CHAR(13) + CHAR(10)
    PRINT N'GO' + CHAR(13) + CHAR(10)
    PRINT N'DENY VIEW DEFINITION ON [dbo].[' + @ProcName + '_ups] TO 
']' + CHAR(13) + CHAR(10)
    PRINT N'GO' + CHAR(13) + CHAR(10)
    PRINT CHAR(13) + CHAR(10)
    PRINT CHAR(13) + CHAR(10)

    --Print Select Statement
    PRINT N'/****** Object:  StoredProcedure [dbo].[' + @ProcName + 
Date: ' + CAST(GETDATE() AS nvarchar(30)) + '  ******/' + CHAR(13) + 
    PRINT @LEGEND;
    PRINT N'-- Description : Select ' + @DESC + CHAR(13) + CHAR(10)
    PRINT N'-- =====================================================
CHAR(13) + CHAR(10)
    PRINT CHAR(13) + CHAR(10)
    PRINT N'CREATE PROCEDURE [dbo].[' + @ProcName  + '_sel]' + CHAR(
    PRINT N'  (@MyID int, @ID int=NULL' + IIF(@ISACTIVE_SEL = 1,', @
bit=NULL','') + ')' + CHAR(13) + CHAR(10);
    PRINT N'AS' + CHAR(13) + CHAR(10)
    PRINT N'BEGIN' + CHAR(13) + CHAR(10)
    PRINT N'  SET CONTEXT_INFO @MyID;' + CHAR(13) + CHAR(10)
    PRINT N'  IF @ID IS NULL OR @ID = 0' + CHAR(13) + CHAR(10)

    IF @ISACTIVE_SEL = 1
        BEGIN
            PRINT N'    BEGIN' + CHAR(13) + CHAR(10)
            PRINT N'      IF @IsActive IS NULL' + CHAR(13) + CHAR(10
            PRINT N'        SELECT * FROM [dbo].[' + @ViewName + '] 
@OrderBy + '] ' + @OrderByDir + ';' + CHAR(13) + CHAR(10)
            PRINT N'      ELSE' + CHAR(13) + CHAR(10)
            PRINT N'        SELECT * FROM [dbo].[' + @ViewName + '] 
@IsActive ORDER BY [' + @OrderBy + '] ' + @OrderByDir + ';' + CHAR(1
            PRINT N'    END' + CHAR(13) + CHAR(10)
        END
    ELSE
        PRINT N'    SELECT * FROM [dbo].[' + @ViewName + '] ORDER BY 
' + @OrderByDir + ';' + CHAR(13) + CHAR(10)

    PRINT N'  ELSE' + CHAR(13) + CHAR(10)
    PRINT N'    SELECT * FROM [dbo].[' + @ViewName + '] WHERE [ID] = 
CHAR(10)
    PRINT N'END' + CHAR(13) + CHAR(10)
    PRINT N'GO' + CHAR(13) + CHAR(10)
    PRINT CHAR(13) + CHAR(10)

----Now add GRANT and DENY permissions to the Role
    PRINT N'GRANT EXECUTE ON [dbo].[' + @ProcName + '_sel] TO [' + @
CHAR(13) + CHAR(10)
```

```
    PRINT N'GO' + CHAR(13) + CHAR(10)
    PRINT N'DENY VIEW DEFINITION ON [dbo].[' + @ProcName +'_sel] TO
']' + CHAR(13) + CHAR(10)
    PRINT N'GO' + CHAR(13) + CHAR(10)
    PRINT CHAR(13) + CHAR(10)
    PRINT CHAR(13) + CHAR(10)

    --Print Delete Statement
    PRINT N'/****** Object:  StoredProcedure [dbo].[' + @ProcName +
Date: ' + CAST(GETDATE() AS nvarchar(30)) + '  ******/' + CHAR(13) +
    PRINT @LEGEND;
    PRINT N'-- Description : Delete ' + @DESC + CHAR(13) + CHAR(10)
    PRINT N'-- ====================================================
CHAR(13) + CHAR(10)
    PRINT CHAR(13) + CHAR(10)
    PRINT N'CREATE PROCEDURE [dbo].[' + @ProcName  + '_del]' + CHAR(
    PRINT N'  (@MyID int, @ID int, @RowVersion int)' + CHAR(13) + CH
    PRINT N'AS' + CHAR(13) + CHAR(10)
    PRINT N'BEGIN' + CHAR(13) + CHAR(10)
    PRINT N'  SET CONTEXT_INFO @MyID;' + CHAR(13) + CHAR(10)
    PRINT N'  SET NOCOUNT ON;' + CHAR(13) + CHAR(10)
    PRINT N'  DELETE FROM [dbo].[' + @TableName + '] WHERE [' + @PRI
AND [RowVersion]=@RowVersion;' + CHAR(13) + CHAR(10)
    PRINT N'  SELECT @@ROWCOUNT as [Rows Affected];' + CHAR(13) + CH
    PRINT N'END' + CHAR(13) + CHAR(10)
    PRINT N'GO' + CHAR(13) + CHAR(10)
    PRINT CHAR(13) + CHAR(10)

----Now add GRANT and DENY permissions to the Role
    PRINT N'GRANT EXECUTE ON [dbo].[' + @ProcName + '_del] TO [' + @
CHAR(13) + CHAR(10)
    PRINT N'GO' + CHAR(13) + CHAR(10)
    PRINT N'DENY VIEW DEFINITION ON [dbo].[' + @ProcName +'_del] TO
']' + CHAR(13) + CHAR(10)
    PRINT N'GO' + CHAR(13) + CHAR(10)
```

edited Jul 10 '13 at 9:51

answered Jul 10 '13 at 8:30

KacireeSoftware
678 5 18

2 This is gold. many thanks, such wow... – Luiscencio Oct 16 '15 at 15:54

3 This is great. Thanks. For anyone using this for non DBO schema tables, besides using a schema variable, you also need to modify the Get PK code to account for schema. OBJECTPROPERTY(OBJECT_ID(CONSTRAINT_SCHEMA + '.' + CONSTRAINT_NAME) – jbd Feb 6 '16 at 0:14

this one should go into the new documentation under something like "how-to generate CRUD for mssql" – Yordan Georgiev Jul 25 '16 at 6:30

13

Beside tools mentioned before, there is another free tool you can use to get the job done in a few clicks.

To do so, you need to enter prefix and suffix in the ApexSQL Complete options window, where you can choose one of the sub-tabs for each of CRUD procedure templates (Select, Insert, Update, Delete). After this is done, the CRUD procedures feature will be available by right clicking in the Object Explorer window, in database or table in the drop down menu.

Here is an article with more details about this functionality (article is a bit old though, as feature is added into the current release)

edited Jun 27 '17 at 10:36

answered Jun 27 '17 at 8:09

V.Jokinen
181 1 6

```
CREATE PROCEDURE SP_USUARIO
```

0

```
@usuario varchar(30)=null,
@clave varchar(500)=null,
@idPersona int=null,
@idRol int=null,
@idUsuario int=null,
@TIPO TINYINT=1
AS
BEGIN
    IF @TIPO=1 -- LISTAR PERSONAS
    BEGIN
        SELECT idPersona,Nombre,ApePaterno,ApeMaterno,DNI,Direccion,
Persona
    END
    IF @TIPO=2 -- LISTAR ROLES
    BEGIN
        SELECT idRol,Nombre,Descripcion FROM ROL
    END
    IF @TIPO=3 -- LISTAR USUARIOS
    BEGIN
        SELECT idUsuario,usuario,pass,idPersona,idRol from Usuario
    END
    IF @TIPO=3 -- USUARIOS
    BEGIN
        SELECT idUsuario,usuario,pass,idPersona,idRol from Usuario w
idUsuario=@idUsuario
    END
    IF @TIPO=4 -- AGREGAR USUARIO
    BEGIN
        INSERT Usuario(usuario,pass,idPersona,idRol)
values(@usuario,@clave,@idPersona,@idRol)
    END
    IF @TIPO=5 -- EDITAR USUARIO
    BEGIN
        UPDATE Usuario SET
usuario=@usuario,pass=@clave,@idPersona=@idPersona,idRol=@idRol wher
idUsuario=@idUsuario
    END
    IF @TIPO=6 -- ELIMINAR USUARIO
    BEGIN
        DELETE Usuario WHERE idUsuario=@idUsuario
    END
END

go
```

answered May 20 '18 at 3:26

user9817675
1

0

Thanks for the original script.
I improved it with the following :
- Allow to do on another database
- For select, generate dynamically the query according parameters
- Manage columns CreatedBy/Date and ModificationBy/Date
- Work even if special characters are found in schema/table/column
- Allow to add systematically the user and culture.
- Template for procedure name
And lot of options.

Note: code send in two answers as limited to 30000 characters.

```
IF OBJECT_ID('dbo.GenerateDynamicallyProceduresForTables','P') IS NO
    DROP PROCEDURE dbo.GenerateDynamicallyProceduresForTables
GO

CREATE PROCEDURE dbo.GenerateDynamicallyProceduresForTables @Databas
nvarchar(200)=NULL,
                                                            @SchemaN
NULL,
                                                            @TableNa
NULL,
                                                            @NoCount
                                                            @ManageT
                                                            @Generat
bit = 1,
                                                            @Paramet
nvarchar(20) = '@UserInP',
                                                            @Paramet
nvarchar(20) = '@CultureInP',
                                                            @FirstPa
bit=1,
                                                            @Procedu
nvarchar(100) = '[{SchemaName}].[{TableName}_Proc_{ActionType}]',
                                                            @ColumnN
nvarchar(500)= '', --(syscolumns.name LIKE ''%Creation%'' OR syscolu
(''SomeInt'',''Somebit'') )
                                                             @Creatio
nvarchar(500) = 'syscolumns.name LIKE ''%CreationUser%'' OR syscolum
''%CreationBy%''',
                                                             @Creatio
nvarchar(500) = 'syscolumns.name LIKE ''%CreationDate%'' OR syscolum
''%CreatedDate%''',
                                                             @Modific
nvarchar(500) = 'syscolumns.name LIKE ''%ModificationUser%'' OR sysc
''%ModifiedBy%''  OR syscolumns.name LIKE ''%ModifiedUser%''',
                                                             @Modific
nvarchar(500) = 'syscolumns.name LIKE ''%ModificationDate%'' OR sysc
''%ModifiedDate%'''
AS
BEGIN
DECLARE @UnCommentExecForDebug bit=0 --To set at 0 for final

DECLARE @StatementList TABLE(id INT IDENTITY(1,1) NOT NULL PRIMARY K
nvarchar(1000),StatementType nvarchar(100),Statement nvarchar(max))
DECLARE @FirstParameters nvarchar(400)='',@FirstParametersForExec nv

IF LEN(@ParameterForUser)>1
BEGIN
  SET @FirstParameters = @FirstParameters + @ParameterForUser  +' nv
WHEN @FirstParametersAreMandatory =0  THEN ' = NULL' ELSE '' END +
    '
  SET @FirstParametersForExec = @FirstParametersForExec + @Parameter
@FirstParametersAreMandatory =0  THEN ' = NULL' ELSE ' =''K2:Denalli
END +  ',
    '
END
IF LEN(@ParameterForCulture)>1
BEGIN
  SET @FirstParameters = @FirstParameters + @ParameterForCulture  +
CASE WHEN @FirstParametersAreMandatory =0  THEN ' = NULL' ELSE '' EN
    '
  SET @FirstParametersForExec = @FirstParametersForExec + @Parameter
WHEN @FirstParametersAreMandatory =0  THEN ' = NULL' ELSE '=''en-gb'
    '
END
IF NOT(LEN(@DatabaseName)>0)
 SET @DatabaseName=DB_NAME()

 IF LEN(@SchemaName)=0
 SET @SchemaName=NULL

 IF LEN(@TableName)=0
 SET @TableName=NULL

IF NOT(LEN(@ColumnNameLimitation)>0)
  SET @ColumnNameLimitation = '1=1'

IF NOT(LEN(@CreationUserMatch)>0)
   SET @CreationUserMatch = 'syscolumns.name = ''BIDON1234567891707

IF NOT(LEN(@CreationDateMatch)>0)
   SET @CreationDateMatch = 'syscolumns.name = ''BIDON1234567891707

IF NOT(LEN(@ModificationUserMatch)>0)
   SET @ModificationUserMatch = 'syscolumns.name = ''BIDON123456789

IF NOT(LEN(@ModificationDateMatch)>0)
   SET @ModificationDateMatch = 'syscolumns.name = ''BIDON123456789




DECLARE @strSpText nVarchar(max) ='USE [' + @DatabaseName + ']'
IF @DatabaseName!=DB_NAME()
INSERT INTO @StatementList (FullTableName,StatementType,Statement) V
current database',@strSPText)

 DECLARE @sqlstatementForTables nvarchar(max) = -- Not test with USE
+ '] ISSUE ON Table iDENTITY.Identity: 'Could not complete cursor op
set options have changed since the cursor was declared
      N'
      DECLARE Tables_cursor CURSOR FOR
       SELECT TABLE_SCHEMA,TABLE_NAME
         FROM [' + @DatabaseName + '].INFORMATION_SCHEMA.TABLES
        WHERE TABLE_TYPE=''BASE TABLE''
          AND (TABLE_SCHEMA=@pSchemaName OR @pSchemaName IS NULL)
          AND (Table_Name=@pTableName OR @pTableName IS NULL)'

     --EXEC loopbackServerForDebug.[K2FranceDebugDB].dbo.K2FranceDeb
'@sqlstatementForColumns',@sqlstatementForColumns
      exec sp_executesql @sqlstatementForTables, N'@pSchemaName
nvarchar(200),@pTableName  nvarchar(200)', @pSchemaName=@SchemaName,
@pTableName=@TableName;

OPEN Tables_cursor
DECLARE @CurrentSchemaName nvarchar(100),@CurrentFullTableName
nvarchar(1000),@CurrentTableName nVarchar(1000),
       @DropStatement nvarchar(max)=''
Fetch next
```

```
from Tables_cursor
INTO @CurrentSchemaName,@CurrentTableName
WHILE @@FETCH_STATUS=0
BEGIN

    SET @CurrentFullTableName='['+@CurrentSchemaName+'].['+@CurrentT
    --PRINT @CurrentFullTableName


    Declare @dbName nVarchar(50)
    Declare @insertSPName nVarchar(4000), @updateSPName nVarchar(400
nVarchar(4000), @listSPName nVarchar(4000)--, @ReadSPName nVarchar(5
    Declare @ColumnParametersInsert nVarchar(max), @ColumnDefForInse
nVarchar(max),@ColumnInValueForInsert nVarchar(max),
            @ColumnParametersList nVarchar(max),@ColumnParametersLis
nVarchar(max),
            @tableColumnForWhereInList nvarchar(max),
            @tableColumnForWhereInListVariables nVarchar(max),
@tableColumnForWhereInListAffectVariables nVarchar(max),@DebugVariab
nvarchar(max)='',
            @ColumnParametersInsertForExec nvarchar(max)
    Declare @tableCols nVarchar(max), @ColumnParametersUpdate
nVarchar(max),@ColumnParametersUpdateForExec nVarchar(max);
    Declare @space nVarchar(50) = REPLICATE(' ', 4) ;
    Declare @colName nVarchar(max) ;
    Declare @DataType nvarchar(200),@colVariable nVarchar(200),@colV
nVarchar(200);
    Declare @colParameter nVarchar(max) ;
    Declare @colAllowNull nvarchar(15), @colIsPrimaryKey INT,@ColIsI
INT,@ColLength INT,@ColIsComputed INT,@ColMatchCreationUser INT,@Col
INT,@ColMatchModificationUser INT,@ColMatchModificationDate INT;

    Declare @updCols nVarchar(max);
    Declare @ColumnParametersDelete nVarchar(max),@ColumnParametersD
nVarchar(max),
            @LastPrimaryKey nvarchar(max),@NbPrimaryKey INT=0,@ColNu
    Declare @whereCols nVarchar(2000);
    DECLARE @SetVariablesForExec nvarchar(max)='',@SetVariablesForEx
nvarchar(max)='', @SetVariablesForExecDelete nvarchar(max)=''

    DECLARE @strBegin nvarchar(1000)=' AS' + CHAR(13) + CHAR(10) +
'BEGIN',@spaceForTrans nvarchar(10)=''
    IF @NoCount=1
      Set @strBegin = @strBegin + CHAR(13) + CHAR(10) + @space + 'SE

    IF @ManageTransaction = 1
    BEGIN
```

```
        Set @strBegin = @strBegin + CHAR(13) + CHAR(10) + @space + ':
if a Transact-SQL statement raises a run-time error, the entire trans
terminated and rolled back.'
      Set @strBegin = @strBegin + CHAR(13) + CHAR(10) + ''
      Set @strBegin = @strBegin + CHAR(13) + CHAR(10) + @space + 'BE
      SET @spaceForTrans= @space;
    END
    DECLARE @strEnd nvarchar(1000)=''
    IF @ManageTransaction = 1
      Set @strEnd = @strEnd + CHAR(13) + CHAR(10) + @space + 'COMMIT
    SET @strEnd = @strEnd + CHAR(13) + CHAR(10) + 'END'
    Set @strEnd = @strEnd + CHAR(13) + CHAR(10) + 'GO'
    Set @strEnd = @strEnd + CHAR(13) + CHAR(10) + ''
    IF @UnCommentExecForDebug = 0 Set @strEnd = @strEnd + CHAR(13) +

    IF @ProcedureTemplateName IS NULL
      BEGIN
        Set @insertSPName = '['+@CurrentSchemaName+'].[sp_' + @Curre
+'_insert]' ;
        Set @updateSPName = '['+@CurrentSchemaName+'].[sp_' + @Curre
+'_update]' ;
        Set @deleteSPName = '['+@CurrentSchemaName+'].[sp_' + @Curre
+'_delete]' ;
        set @listSPName =  '['+@CurrentSchemaName+'].[sp_' + @Curren
;
      END
    ELSE
      BEGIN
         DECLARE @ProcedureName
nvarchar(200)=REPLACE(REPLACE(@ProcedureTemplateName,'{SchemaName}',

         Set @insertSPName = REPLACE(@ProcedureName,'{ActionType}','
         Set @updateSPName = REPLACE(@ProcedureName,'{ActionType}','
         Set @deleteSPName = REPLACE(@ProcedureName,'{ActionType}','
         Set @listSPName = REPLACE(@ProcedureName,'{ActionType}','Li
      END

SET @DropStatement = @DropStatement+ '
DROP PROCEDURE ' + @insertSPName + '
DROP PROCEDURE ' + @updateSPName + '
DROP PROCEDURE ' + @deleteSPName +'
DROP PROCEDURE ' + @listSPName


    Set @ColumnParametersInsert = @FirstParameters ;
    SET @ColumnParametersInsertForExec = @FirstParametersForExec
    Set @ColumnParametersUpdate=@FirstParameters
    SET @ColumnParametersUpdateForExec=@FirstParametersForExec
    Set @ColumnParametersDelete = @FirstParameters ;
    SET @ColumnParametersDeleteForExec = @FirstParametersForExec ;
    SET @ColumnParametersList = @FirstParameters;
    SET @ColumnParametersListForExec = @FirstParametersForExec

    SET @tableColumnForWhereInList= ''
    SET @tableColumnForWhereInListVariables =''
    SET @tableColumnForWhereInListAffectVariables =''
    SET @DebugVariablesForList ='';
    Set @ColumnDefForInsert = '' ;
    Set @ColumnInValueForInsert = '' ;
    Set @strSPText = '' ;
    Set @tableCols = '' ;
    Set @updCols = '' ;

    Set @whereCols = '' ;

    SET NOCOUNT ON


    CREATE TABLE #tmp_Structure (colid int,ColumnName nvarchar(max),
                                 ColumnVariable nvarchar(max),
                                 DataType nvarchar(max),
                                 ColumnParameter nvarchar(max),
                                 AllowNull int,
                                 IsPrimaryKey int,
                                 IsIdentityAutoIncrement int,
                                 ColLength int,
                                 IsIsComputedColumn int,
                                 ColMatchCreationUser int,ColMatchCr
                                 ColMatchModificationUser INT,ColMat
INT)

    DECLARE @sqlstatementForColumns nvarchar(max) =
      N'USE [' + @DatabaseName + ']
      SELECT distinct
           --sysobjects.name as ''Table'',
           syscolumns.colid ,
           ''['' + syscolumns.name + '']'' as ''ColumnName'',
           ''@''+syscolumns.name as ''ColumnVariable'',
           systypes.name +
           Case When systypes.xusertype in (165,167,175,231,239 ) Th
Convert(varchar(10),Case When syscolumns.length=-1 Then ''max'' else
CAST(syscolumns.length AS nvarchar(10)) end) +'')'' Else '''' end as
           systypes.name +  Case When systypes.xusertype in (165,167
''('' + Convert(varchar(10),Case When syscolumns.length=-1 Then ''ma
CAST(syscolumns.length AS nvarchar(10)) end) +'')'' Else '''' end as
''ColumnParameter'',
           COLUMNPROPERTY(OBJECT_ID(@pFullTableName),syscolumns.name
AllowNull,
           (SELECT COUNT(*)
                    FROM [' + @DatabaseName + '].INFORMATION_SCHEMA.
Tab,
                         [' + @DatabaseName +
'].INFORMATION_SCHEMA.CONSTRAINT_COLUMN_USAGE Col
                   WHERE Col.Constraint_Name = Tab.Constraint_Name
                    AND Col.Table_Name = Tab.Table_Name
                    AND Constraint_Type = ''PRIMARY KEY''
                    AND Col.Table_Name = @pTableName
                    AND Tab.TABLE_SCHEMA=@pSchemaName
                    AND Col.Column_Name = syscolumns.name
                        ) AS IsPrimaryKey,
           SC.is_identity AS IsIdentityAutoIncrement,
           syscolumns.length,
           (SELECT COUNT(*)
              FROM sys.computed_columns
             WHERE computed_columns.object_id=sysobjects.id
               AND computed_columns.Name=syscolumns.name) AS IsCompu
            CASE WHEN ' + @CreationUserMatch +' THEN 1 ELSE 0 END AS
ColMatchCreationUser,
            CASE WHEN ' + @CreationDateMatch +' THEN 1 ELSE 0 END AS
ColMatchCreationDate,
            CASE WHEN ' + @ModificationUserMatch +' THEN 1 ELSE 0 EN
ColMatchModificationUser,
            CASE WHEN ' + @ModificationDateMatch +' THEN 1 ELSE 0 EN
ColMatchModificationDate
    FROM sysobjects
         LEFT JOIN syscolumns ON syscolumns.id=sysobjects.id
         LEFT JOIN systypes ON systypes.xusertype=syscolumns.xuserty
         LEFT JOIN sys.columns SC ON SC.object_id = sysobjects.id
                           AND SC.name=syscolumns.name
    Where sysobjects.xtype = ''u''
      and sysobjects.id = OBJECT_ID(@pFullTableName)
      AND (' + @ColumnNameLimitation + ')
    Order by syscolumns.colid'

     --PRINT @sqlstatementForColumns
     --EXEC loopbackServerForDebug.[K2FranceDebugDB].dbo.K2FranceDeb
'@sqlstatementForColumns',@sqlstatementForColumns

      INSERT INTO #tmp_Structure
      exec sp_executesql @sqlstatementForColumns, N'@pSchemaName
nvarchar(200),@pTableName  nvarchar(200),@pFullTableName nvarchar(10
@pSchemaName=@CurrentSchemaName,
@pTableName=@CurrentTableName,@pFullTableName=@CurrentFullTableName;


      --SELECT * FROM #tmp_Structure

    /* Read the table structure and populate variables*/
    DECLARE SpText_Cursor CURSOR FOR
     SELECT ColumnName, ColumnVariable, DataType, ColumnParameter, A
IsPrimaryKey, IsIdentityAutoIncrement,ColLength,
IsIsComputedColumn,ColMatchCreationUser,ColMatchCreationDate,ColMatc

       FROM #tmp_Structure
    OPEN SpText_Cursor

    FETCH NEXT FROM SpText_Cursor INTO @colName, @colVariable,  @Dat
@colParameter, @colAllowNull,@colIsPrimaryKey, @ColIsIdentityAutoInc
@ColIsComputed,@ColMatchCreationUser,@ColMatchCreationDate,@ColMatch

    WHILE @@FETCH_STATUS = 0
    BEGIN
       SET @ColNumber=@ColNumber+1

       SET @SetVariablesForExec = @SetVariablesForExec  + CASE WHEN (
THEN ''
                                                         ELSE  CASE 
('datetime','datetime2','smalldatetime','date') AND @SetVariablesFor
'%@Date%' THEN CHAR(13) +CHAR(10) + 'DECLARE @Date datetime =GetDate

@DataType  IN ('uniqueidentifier') AND @SetVariablesForExec NOT LIKE
CHAR(13) +CHAR(10) + 'DECLARE @TheGuid uniqueidentifier  =NEWID()'


                                                         END

       --RegEx to keep only alphanumeric characters:
       DECLARE @MatchExpression nvarchar(20) =  '%[^a-z0-
9]%',@DateTypeWithoutSpecialCharacters nvarchar(100)=@DataType;

       WHILE PatIndex(@MatchExpression, @DateTypeWithoutSpecialChara
        SET @DateTypeWithoutSpecialCharacters = Stuff(@DateTypeWitho
PatIndex(@MatchExpression, @DateTypeWithoutSpecialCharacters), 1, ''


       --Remove Special characters (like space...) for variable name
       WHILE PatIndex(@MatchExpression, @colVariable) > 0
        SET @colVariable = Stuff(@colVariable, PatIndex(@MatchExpres
1, '')

       SET @colVariableProc = '@p'+ @colVariable
       SET @colVariable = '@'+ @colVariable

       SET @colParameter = @colVariable + ' ' + @colParameter







       DECLARE @AffectationForExec nvarchar(max)=@colVariable + CASE
=1 THEN ' = NULL'
                     ELSE ' = ' +  CASE WHEN @DataType IN ('Text','s
@DataType LIKE '%char%'  THEN '''' + SUBSTRING ( CAST(ABS(@ColLength
'TEST' + @DateTypeWithoutSpecialCharacters,0,CASE WHEN @ColLength < 
@DataType LIKE 'nchar%' THEN @ColLength/2+1 ELSE @ColLength END) + '
                                        WHEN @DataType  IN
('int','numeric','bigint','tinyint') THEN CAST(@ColNumber AS nvarcha
                                        WHEN @DataType  IN ('bit') T
                                        WHEN @DataType  IN ('float')
CAST(@ColNumber AS nvarchar(10))   +  '.' + CAST(@ColNumber+1 AS nva
                                        WHEN @DataType  IN
('datetime','datetime2','smalldatetime','date') THEN '@Date'
                                        WHEN @DataType  IN ('uniquei
'@TheGuid'
                                        WHEN @DataType  IN ('xml') T
<value name="test">' + CAST(@ColNumber AS nvarchar(10)) + '</value><
                                        ELSE '''1''--Currently Not m
                                   END
                END +  ', --Type ' + @DataType  + CHAR(13) + CHAR(10

       IF @ColIsIdentityAutoIncrement = 0 AND @ColIsComputed = 0
       BEGIN
          IF @ColMatchModificationUser = 0  AND @ColMatchModificatio
            Set @ColumnDefForInsert = @ColumnDefForInsert + @colName
CHAR(10) + @space + @space + @spaceForTrans ;

          IF @ColMatchCreationUser= 0 AND @ColMatchCreationDate = 0 
@ColMatchModificationUser = 0  AND @ColMatchModificationDate = 0
          BEGIN
            Set @ColumnParametersInsert = @ColumnParametersInsert + 
WHEN @colAllowNull =1  THEN ' = NULL' ELSE '' END +  ','  + CHAR(13)
;
            SET @ColumnParametersInsertForExec = @ColumnParametersIn
@AffectationForExec
```

```
          END

          IF @ColMatchCreationUser= 1
          BEGIN
            IF LEN(@ParameterForUser)>1
               Set @ColumnInValueForInsert = @ColumnInValueForInsert
@ParameterForUser + ',SYSTEM_USER)'
            ELSE
               Set @ColumnInValueForInsert = @ColumnInValueForInsert
          END
          ELSE
          BEGIN
            IF @ColMatchCreationDate= 1
              Set @ColumnInValueForInsert = @ColumnInValueForInsert
            ELSE
              IF @ColMatchModificationUser = 0  AND @ColMatchModific
                Set @ColumnInValueForInsert = @ColumnInValueForInser
          END
          IF @ColMatchCreationUser= 1 OR @ColMatchCreationDate= 1 OR
@ColMatchModificationUser = 0  AND @ColMatchModificationDate = 0
        SET @ColumnInValueForInsert =@ColumnInValueForInsert + ',' +
+ @space + @space+ @spaceForTrans


          Set @tableCols = @tableCols + @colName + ',' ;


          IF @ColMatchModificationUser = 1
          BEGIN
             IF LEN(@ParameterForUser)>1
               Set @updCols = @updCols + @colName + ' = ISNULL(' + @
',SYSTEM_USER)';
            ELSE
               Set @updCols = @updCols + @colName + ' = SYSTEM_USER'
          END
          ELSE
          BEGIN
            IF @ColMatchModificationDate = 1
               Set @updCols = @updCols + @colName + ' = GETDATE()';
            ELSE
              IF @ColMatchCreationUser=0 AND @ColMatchCreationDate=0
              Set @updCols = @updCols + @colName + ' = ' + @colVaria
          END
          IF @ColMatchModificationUser = 1 OR @ColMatchModificationD
@ColMatchCreationUser=0 AND @ColMatchCreationDate=0
        SET @updCols =@updCols + ',' + CHAR(13) + CHAR(10) + @space
@spaceForTrans
```

```
       END

       SET @ColumnParametersList = @ColumnParametersList + @colParam
',' + CHAR(13) + CHAR(10) + @space ;
       SET @ColumnParametersListForExec = @ColumnParametersListForEx
' = NULL, --Type ' + @DataType  + CHAR(13) + CHAR(10) + @space

       IF @ColIsIdentityAutoIncrement = 1 AND @DataType='int'
          BEGIN
            SET @SetVariablesForExecUpdate =   CHAR(13)+CHAR(10)+'DE
@PrimaryKeyValue INT= (SELECT MIN(' + @colName + ') FROM ' + @Curren
            SET @AffectationForExec = @colVariable  + '= @PrimaryKey
@DataType  + CHAR(13) + CHAR(10) + @space
          END

       IF @ColIsComputed = 0 AND @ColMatchCreationUser=0 AND @ColMat
@ColMatchModificationUser=0 AND @ColMatchModificationDate=0
       BEGIN
         Set @ColumnParametersUpdate = @ColumnParametersUpdate + @co
CHAR(13) + CHAR(10) + @space ;
         SET @ColumnParametersUpdateForExec = @ColumnParametersUpdat
@AffectationForExec
       END

       IF @DataType NOT IN ('text')
       BEGIN
         IF @DataType NOT IN ('Xml')
         BEGIN
             SET @tableColumnForWhereInList = @tableColumnForWhereIn
                 IF ' + @colVariable + ' IS NOT NULL
                   BEGIN
                     SET @Statement= @Statement+ @Separator + ''' +
REPLACE(@colName,'''','''''') + '= '+  @colVariableProc +'''
                     SET @Separator = @SeparatorAnd
                   END'
             SET @tableColumnForWhereInListVariables =
@tableColumnForWhereInListVariables + @space + @space + @space + @sp
@colVariableProc + ' ' + @DataType +',
        '
             SET @tableColumnForWhereInListAffectVariables =
@tableColumnForWhereInListAffectVariables + @space + @space + @space
@colVariableProc + '=' + @colVariable + ',
        '
         END
          SET @DebugVariablesForList = @DebugVariablesForList+ CHAR(
@space + @space + @space + @space +@space+ @spaceForTrans
          IF @DataType IN ('Xml')
            SET @DebugVariablesForList = @DebugVariablesForList+  'I
@colVariableProc + ' ' + @DataType+' = CAST('''''' + REPLACE(CAST('
nvarchar(max)),''''''',''''''''''') + '''''AS XML);''+CHAR(13)+CH
          ELSE
            SET @DebugVariablesForList = @DebugVariablesForList+  'I
@colVariableProc + ' ' + @DataType+' = '''''' + REPLACE(' +@colVaria
',''''''',''''''''''') + '''''';''+CHAR(13)+CHAR(10) ,'''') + '
       END


       IF @colIsPrimaryKey= 1
       BEGIN
          IF @ColIsIdentityAutoIncrement = 1 AND @DataType='int'
          BEGIN
            SET @SetVariablesForExecDelete =   CHAR(13)+CHAR(10)+'DE
@PrimaryKeyValue INT= (SELECT MAX(' + @colName + ') FROM ' + @Curren
          END

         SET @ColumnParametersDelete = @ColumnParametersDelete + @co
CHAR(13) + CHAR(10) + @space ;

         SET @ColumnParametersDeleteForExec = @ColumnParametersDelet
@AffectationForExec
         SET @whereCols = @whereCols + @colName + ' = ' + @colVariab
         SET @NbPrimaryKey = @NbPrimaryKey +1
         SET @LastPrimaryKey = @colName
       END
    FETCH NEXT FROM SpText_Cursor INTO @colName, @colVariable, @Data
@colAllowNull,@colIsPrimaryKey,@ColIsIdentityAutoIncrement,@ColLengt

    END
    CLOSE SpText_Cursor
    DEALLOCATE SpText_Cursor

    IF @ColumnDefForInsert IS NULL
      RAISERROR('@ColumnDefForInsert IS NULL',16,1)
    IF @ColumnParametersInsert IS NULL
      RAISERROR('@ColumnParametersInsert IS NULL',16,1)
    IF @ColumnParametersInsertForExec IS NULL
      RAISERROR('@ColumnParametersInsertForExec IS NULL',16,1)
    IF @ColumnInValueForInsert IS NULL
      RAISERROR('@ColumnInValueForInsert IS NULL',16,1)
    IF @tableCols IS NULL
      RAISERROR('@tableCols IS NULL',16,1)
    IF @updCols IS NULL
```

```
      RAISERROR('@updCols IS NULL',16,1)

    IF @ColumnParametersDelete IS NULL
      RAISERROR('@ColumnParametersDelete IS NULL',16,1)
    IF @whereCols IS NULL
      RAISERROR('@whereCols IS NULL',16,1)

    DECLARE @LastPosOfComma INT

    If (LEN(@ColumnParametersUpdate)>0)
    BEGIN
      Set @ColumnParametersUpdate =
LEFT(@ColumnParametersUpdate,LEN(@ColumnParametersUpdate)-3) ;
      SET @LastPosOfComma = LEN(@ColumnParametersUpdateForExec) - CH
,',REVERSE(@ColumnParametersUpdateForExec))
      SET @ColumnParametersUpdateForExec =
LEFT(@ColumnParametersUpdateForExec,@LastPosOfComma+3) +
SUBSTRING(@ColumnParametersUpdateForExec,@LastPosOfComma+5,40000);
    END
     --See next post for the end of procedure
```

edited Dec 3 '18 at 12:00

answered Dec 3 '18 at 9:25

Olivier Chatagnon

1 1

0

```
-- Here end end of the procedure (started on another post)
    If (LEN(@ColumnParametersInsert)>0)
    Begin


      Set @ColumnParametersInsert =
LEFT(@ColumnParametersInsert,LEN(@ColumnParametersInsert)-3) ;
      SET @LastPosOfComma = LEN(@ColumnParametersInsertForExec) - CH
,',REVERSE(@ColumnParametersInsertForExec))
```

```
      SET @ColumnParametersInsertForExec =
LEFT(@ColumnParametersInsertForExec,@LastPosOfComma+3) +
SUBSTRING(@ColumnParametersInsertForExec,@LastPosOfComma+5,40000);

      Set @ColumnParametersDelete =
LEFT(@ColumnParametersDelete,LEN(@ColumnParametersDelete)-4) ;
      SET @LastPosOfComma = LEN(@ColumnParametersDeleteForExec) - CH
,',REVERSE(@ColumnParametersDeleteForExec))
      SET @ColumnParametersDeleteForExec =
LEFT(@ColumnParametersDeleteForExec,@LastPosOfComma+3) +
SUBSTRING(@ColumnParametersDeleteForExec,@LastPosOfComma+5,40000);



      SET @ColumnParametersList =
LEFT(@ColumnParametersList,LEN(@ColumnParametersList)-3) ;
      SET @LastPosOfComma = LEN(@ColumnParametersListForExec) - CHAR
,',REVERSE(@ColumnParametersListForExec))
      SET @ColumnParametersListForExec =
LEFT(@ColumnParametersListForExec,@LastPosOfComma+3) +
SUBSTRING(@ColumnParametersListForExec,@LastPosOfComma+5,40000);

      IF LEN(@ColumnInValueForInsert)>0
        Set @ColumnInValueForInsert =
LEFT(@ColumnInValueForInsert,LEN(@ColumnInValueForInsert)-3) ;
      IF LEN(@ColumnDefForInsert)>0
        Set @ColumnDefForInsert = LEFT(@ColumnDefForInsert,LEN(@Colu
      IF LEN(@tableCols)>0
        Set @tableCols = LEFT(@tableCols,LEN(@tableCols)-1) ;
      IF LEN(@updCols)>0
        Set @updCols = LEFT(@updCols,LEN(@updCols)-3) ;
      SET @tableColumnForWhereInListVariables =
LEFT(@tableColumnForWhereInListVariables,LEN(@tableColumnForWhereInL
      SET @tableColumnForWhereInListAffectVariables =
LEFT(@tableColumnForWhereInListAffectVariables,LEN(@tableColumnForWh
 ;


    END


    If (LEN(@whereCols)>0)
      Set @whereCols = 'WHERE ' + LEFT(@whereCols,LEN(@whereCols)-4)
    ELSE
       Set @whereCols = 'WHERE 1=0 --Too dangerous to do update or d
table'
```

```
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '--
============================================='
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '-- Author :
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '-- Create da
Convert(varchar(20),Getdate())
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '-- Descripti
Procedure for ' + @CurrentTableName
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '--
============================================='

    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'IF OBJECT_II
REPLACE(@insertSPName,'''','''''') + ''',''P'') IS NOT NULL'
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '   DROP PRO
@insertSPName
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'GO'
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + ''
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + ''

    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'CREATE PROCI
@insertSPName
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + '' -
@ColumnParametersInsert
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @strBegin
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @spa
'INSERT INTO ' + @CurrentFullTableName + '('
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @spa
+ '' + @ColumnDefForInsert + ')'
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @spa
'VALUES ('
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @spa
+ '' + @ColumnInValueForInsert + ')'

    IF @NbPrimaryKey =1 --No return if 2 or 0 primarykeys
      Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @
'SELECT SCOPE_IDENTITY() AS ' + @LastPrimaryKey

    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @strEnd

    Set @strSPText = @strSPText + @SetVariablesForExec
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'EXEC ' + @in
@ColumnParametersInsertForExec
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'GO'
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'SELECT * FRO
@CurrentFullTableName + ' ORDER BY 1 DESC'
    IF @UnCommentExecForDebug = 0 Set @strSPText = @strSPText + CHAR
'*/'
```

```
    INSERT INTO @StatementList (FullTableName,StatementType,Statemen
(@CurrentFullTableName,'Insert',@strSPText)



    Set @strSPText = ''
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '--
=============================================='
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '-- Author :
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '-- Create d
Convert(varchar(20),Getdate())
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '-- Descript
Procedure for ' + @CurrentTableName
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '--
=============================================='

    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'IF OBJECT_I
REPLACE(@updateSPName,'''','''''') + ''',''P'') IS NOT NULL'
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '   DROP PRO
@updateSPName
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'GO'
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + ''
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + ''


    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'CREATE PROC
@updateSPName
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + '' 
@ColumnParametersUpdate
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @strBegin
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @sp
' + @CurrentFullTableName
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @sp
' + @updCols
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @sp
@whereCols
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @strEnd
    Set @strSPText = @strSPText + @SetVariablesForExec  + @SetVariab
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'EXEC ' + @u
@ColumnParametersUpdateForExec
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'SELECT * FR
@CurrentFullTableName + ' '
    IF @UnCommentExecForDebug = 0 Set @strSPText = @strSPText + CHAR
'*/'

    INSERT INTO @StatementList (FullTableName,StatementType,Statemen
(@CurrentFullTableName,'Update',@strSPText)
```

```
    Set @strSPText = ''
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '--
=============================================='
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '-- Author :
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '-- Create da
Convert(varchar(20),Getdate())
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '-- Descripti
Procedure for ' + @CurrentTableName
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '--
=============================================='

    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'IF OBJECT_II
REPLACE(@deleteSPName,'''',''''''') + ''',''P'') IS NOT NULL'
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '   DROP PRO
@deleteSPName
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'GO'
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + ''
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + ''

    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'CREATE PROCE
@deleteSPName
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + '' +
@ColumnParametersDelete
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @strBegin
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @spa
@spaceForTrans + 'DELETE FROM ' + @CurrentFullTableName
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @spa
@spaceForTrans + ' ' + @whereCols
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @strEnd
    Set @strSPText = @strSPText + @SetVariablesForExecDelete
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'EXEC ' + @de
@ColumnParametersDeleteForExec
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'GO'
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'SELECT * FRO
@CurrentFullTableName + ' ORDER BY 1 DESC'
    IF @UnCommentExecForDebug = 0 Set @strSPText = @strSPText + CHAR
'*/'

    INSERT INTO @StatementList (FullTableName,StatementType,Statemen
(@CurrentFullTableName,'Delete',@strSPText)




    Set @strSPText = ''
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '--
=============================================='
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '-- Author :
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '-- Create da
Convert(varchar(20),Getdate())
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '-- Descripti
for ' + @CurrentFullTableName
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '--
=============================================='

    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'IF OBJECT_II
REPLACE(@listSPName,'''',''''''') + ''',''P'') IS NOT NULL'
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + '   DROP PROC
@listSPName
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'GO'
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + ''


    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'CREATE PROCE
@listSPName
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + '' +
@ColumnParametersList
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @strBegin
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space +'   I
nvarchar(20) =''
     WHERE '''
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space +'   I
@SeparatorAnd  nvarchar(20) =''
       AND '''
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space +'   I
nvarchar(max) =''SELECT *
      FROM ' + REPLACE(@CurrentFullTableName,'''',''''''') + '''' +
@tableColumnForWhereInList

    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @spa
@spaceForTrans + ' --PRINT @Statement'
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @spa
@spaceForTrans + ' BEGIN TRY'

    IF @GenerateDebugScriptForList=1
    BEGIN
      Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @s
@spaceForTrans + '    IF 1=0--DEBUG --TODO: verify if not set for fi
      Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @s
@spaceForTrans + '      BEGIN'
      Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @s
@spaceForTrans + '          DECLARE @FullQueryForDebug nvarchar(max)=
@DebugVariablesForList
      Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @
@spaceForTrans + '            CHAR(13) +CHAR(10) + @Statement'
      Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @
@spaceForTrans + '  '
      Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @
@spaceForTrans + '         --PRINT @FullQueryForDebug'
      Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @
@spaceForTrans + '         --EXEC [K2FranceDebugDB].dbo.K2FranceDebu
''@FullQueryForDebug DIRECT'', @FullQueryForDebug'
      Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @
@spaceForTrans + '         --EXEC loopbackServerForDebug.
[K2FranceDebugDB].dbo.K2FranceDebug  ''@FullQueryForDebug loopback''
@FullQueryForDebug'
      Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @
@spaceForTrans + '      END'
    END
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @spa
@spaceForTrans + '     exec sp_executesql @Statement ,N''' +
@tableColumnForWhereInListVariables + ''',' + @tableColumnForWhereIn
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @spa
@spaceForTrans + ' END TRY'
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @spa
@spaceForTrans + ' BEGIN CATCH'

    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @spa
@spaceForTrans + '   DECLARE @ErrorToDisplay nvarchar(max)= ''Error
Query Error number:'' + CAST(ERROR_NUMBER() AS nvarchar(max)) +
                        --'' Error severity:'' + ISNULL(CAST(ERROR_S
nvarchar(max)),'''') +
                        --'' Error state:'' + ISNULL(CAST(ERROR_STAT
nvarchar(max)),'''')  +
                        --'' Error procedure:'' + ISNULL(CAST(ERROR_
nvarchar(max)),'''')  +
                        --'' Error line:'' + ISNULL(CAST(ERROR_LINE(
nvarchar(max)),'''')  +
                        '' Error message:'' + ISNULL(CAST(ERROR_MESS
nvarchar(max)),'''')
'
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @spa
@spaceForTrans + '   RAISERROR(@ErrorToDisplay, 16, 1);'
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @space + @spa
@spaceForTrans + ' END CATCH'
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + @strEnd
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'EXEC ' + @l
@ColumnParametersListForExec
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'GO'
    Set @strSPText = @strSPText + CHAR(13) + CHAR(10) + 'SELECT * FR
@CurrentFullTableName + ' ORDER BY 1'
    IF @UnCommentExecForDebug = 0 Set @strSPText = @strSPText + CHAR
'*/'


    INSERT INTO @StatementList (FullTableName,StatementType,Statemen
(@CurrentFullTableName,'List',@strSPText)





    Drop table #tmp_Structure
    Fetch next from Tables_cursor INTO @CurrentSchemaName,@CurrentTa
END
CLOSE Tables_cursor
DEALLOCATE Tables_cursor



SET @DropStatement = '

------------------------------------------- TO CLEAN COMPLETELY THE 
-------------------------------
/*' + @DropStatement +'
*/'
INSERT INTO @StatementList (FullTableName,StatementType,Statement) V
('Common','Drop statement to put at the end of final script',@DropSt


SELECT * FROM @StatementList
ORDER BY 1


END
GO

--For all tables of schema dbo of database "OlivierDb":
EXEC dbo.GenerateDynamicallyProceduresForTables 'OlivierDB','dbo'

--With all possible parameters:
EXEC dbo.GenerateDynamicallyProceduresForTables @DatabaseName
'OlivierDB',
                                                @SchemaName
                                                @TableName
                                                @NoCount
                                                @ManageTransaction
                                                @GenerateDebugScript
                                                @ParameterForUser
'@UserInP',
                                                @ParameterForCulture
'@CultureInP',
                                                @FirstParametersAreM
                                                @ProcedureTemplateNa
'[{SchemaName}].[{TableName}_Proc_{ActionType}]',
                                                @ColumnNameLimitatio
(syscolumns.name LIKE ''%Creation%'' OR syscolumns.name IN (''SomeIn
                                                @CreationUserMatch
'syscolumns.name LIKE ''%CreationUser%'' OR syscolumns.name LIKE ''%
                                                @CreationDateMatch
'syscolumns.name LIKE ''%CreationDate%'' OR syscolumns.name LIKE ''%
                                                @ModificationUserMat
'syscolumns.name LIKE ''%ModificationUser%'' OR syscolumns.name LIKE
OR syscolumns.name LIKE ''%ModifiedUser%''',
                                                @ModificationDateMat
'syscolumns.name LIKE ''%ModificationDate%'' OR syscolumns.name LIKE
```

answered Dec 3 '18 at 9:32

Olivier Chatagnon
1 1