Ken's SQL Server Stored Procedure Notes

A quick refresher through common code examples

Contents

Using Stored Procedures

Declaring a Stored Procedure

```
/*--- Start test_proc ---*/
If Exists (Select * From Sysobjects
Where Id = Object_ID(N'[test_proc]') And
Objectproperty(Id, N'IsProcedure') = 1)
Drop Procedure test_proc
Go
Create Procedure test_proc( @dummy Int )
As
/* Prolog */
Declare @Retc Int

Set @Retc = 5

Return @Retc
Go
/*--- End test_proc ---*/
```

If no parms passed

```
If Exists (Select * From Sysobjects
Where Id = Object_ID(N'[test_proc]') And
Objectproperty(Id, N'IsProcedure') = 1)
Drop Procedure test_proc
Go
Create Procedure test_proc
As
/* Prolog */
Declare @Retc Int

Set @Retc = 5

Return @Retc
Go
/*--- End test_proc ---*/
```

Calling a Stored Procedure

```
Exec ValLogError @S_ID, @ErrID,  @Labels

Exec ValLogCellHandleErr @SessionID,'NF',@Label_ID, @Lang_ID

Exec  @Ret = GetTestSwitchesValue @Test_ID = @Test_ID,
                                  @Test_Version = @Test_Version,
                                  @MeasLeg = @MeasLeg Output
```

Calling a Stored Procedure from VBA

```
'*************************************************************
' Note: To use this you must include
'       Microsoft Active X 2.8 Data Objects Library
'       via Tools->References from the VB Editor
'*************************************************************
Function run_sql(stSql) As Variant
Dim return_obj As Variant

   Dim thisCon As ADODB.Connection
   Set thisCon = New ADODB.Connection

   thisCon.Open gODBC_NAME, gUSER, gPW
   Set return_obj = thisCon.Execute(stSql)
   thisCon.Close

   Set run_sql = return_obj  'returns an object

End Function 'run_sql
```

Caller

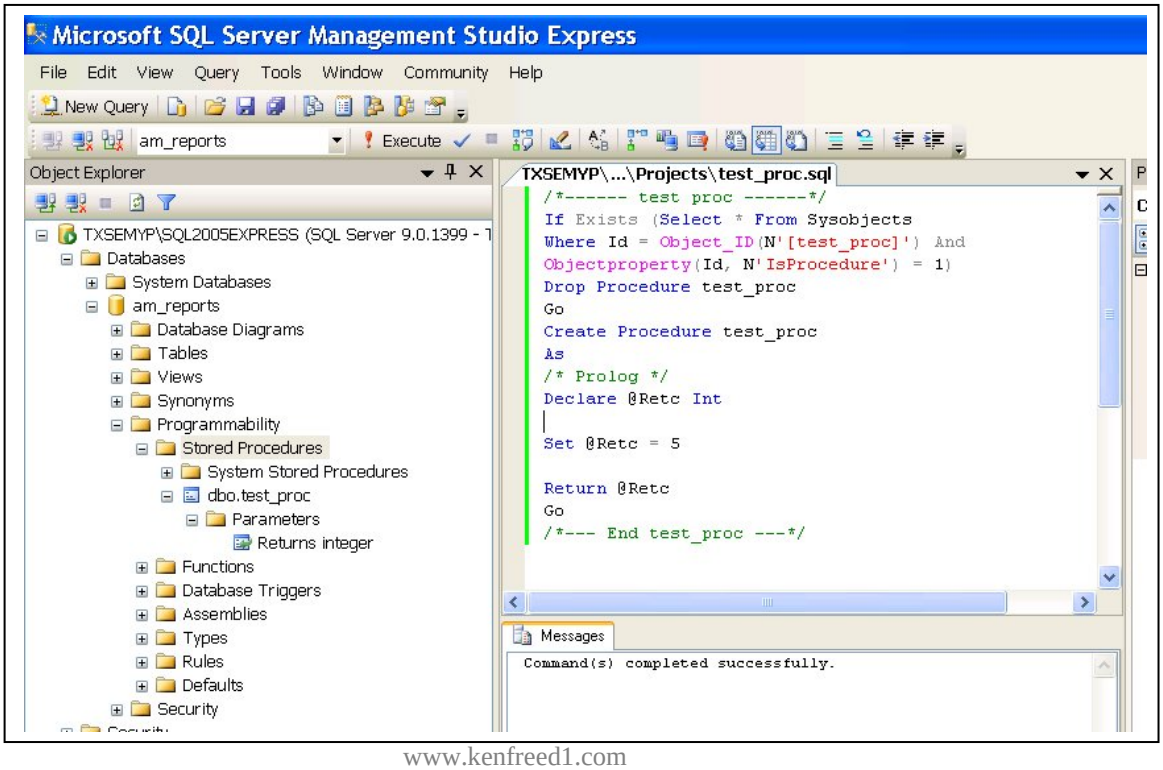
```
Dim stSql As String
Dim return_obj as Variant

    stSql = "dbo.test_proc"
    Set return_obj = run_sql (stSql)
```

Todo: how to get the value returned out of the return_obj

Proc Definition in Sql Server Manager

Hitting the Execute button will execute the proc locally



Constructing a Stored Procedure Call Dynamically

```
/*-------------------------------------
  Call the query we just created
    @Qry holds (eg):
     exec @Okay = ValDupPins @Test_ID=22909,
                             @Test_Version=1,
                             @Tester_ID=10015,

@Labels='101369,101370,101371,101372,101373,112002',
                             @ScreenID=105003,
                             @Single_Test=1,
                             @Session_ID=695465993,
                             @Lang_ID=1
  ------------------------------------------------*/
  Set @Qry =  N'exec @Okay = ' +@Proc + ' ' + @ArgsStr
  Exec sp_executesql  @Qry, N'@Okay INT OUTPUT', @Okay OUTPUT
  Return @Okay
```

Setting the Default Database

```
USE [AM_REPORTS]
GO
```

Comments

```
/* here is a comment block */
-- here is a rest of line comment
```

Misc

Starts-Stops the message that shows the count of the number of rows affected by a Transact-SQL statement or stored procedure from being returned as part of the result set.

```
Set NOCOUNT On
Set NOCOUNT Off   /* runs faster */

/* If any of our error msgs are concatenated together and have a Null in them,
   the whole message turns into a null without the next setting */
  SET CONCAT_NULL_YIELDS_NULL OFF
```

Variables/Data Types

- Anything not explicitly set at declaration time is automatically set to Null.
- Types are case insensitive.

```
Declare @a_sql_variant Sql_variant,
        @Ext_Supply Bit = 0,
        @CRlen smallint = 0,
        @idx Int = 0,
        @fValue float = 0.00,
        @aString varchar(32)    /* = Null */
```

Type Converting

```
Set @DependentCellValue = Cast(@a_sql_variant As Varchar(64))
```

IsNumeric

```
If ISNUMERIC(@fValue) = 1

Blank Is Numeric
Null Is Not Numeric
$5.06 Is Numeric
i.e.:

declare @Value float = ''  /* '' is numeric, null is not numeric, $5.06 is
numeric */

    If IsNumeric(@Value) = 1
      Select 'IsNumeric' = @Value
    Else
      Select 'NOT Numeric' = @Value
```

Table Variables

Returning a Table variable: See Split

```
Declare @TmpTbl      Table (StrVal Nvarchar(255),
                            Idx Int)
```

Decisions

If

```
If @Ret <> 0
Begin
  Set @CountInvalids = @CountInvalids + 1
  If @Single_Test = 1
    Return -1              /* invalid */
End
Else
  Return 0               /* valid */


If (Select CPMpins+PAMpins
      From Tester_HardwareProfiles
      Where Tester_ID=@TesterID) < 8
Begin
  Set @Ret='No CPM or PAM matrix pins'
End
Else
  Set @Ret=''

/* not equals */
If (Len(@CellHandlingCode)<>@CRLen)

/* Use In rather than Or */
If @Valtype In ('SYS$I','SYS$IALL','SYS$ISTRESS','SYS$WLRI')
```

Case

```
Select @Okay  = Case When( @Value = 0 ) And (IsNumeric(@FloatVal)=1))
                    Then 1 Else 0 End

Select @Num=
      Case Cast(Right(@orgnum,1) As Varbinary(8))
        When Cast('f' As Varbinary(8)) Then @num*1e-15
        When Cast('p' As Varbinary(8)) Then @num*1e-12
        When Cast('n' As Varbinary(8)) Then @num*1e-9
        Else @Num
      End /* Case */

Select @Ret=
      Case @Meter
        When 'DMM1' Then (Select DMM1Type From Tester_HardwareProfiles
                         Where Tester_ID=@TesterID)
        When 'DMM2' Then (Select DMM2Type From Tester_HardwareProfiles
                          Where Tester_ID=@TesterID)
        Else ''
      End /* Case */
```

Handling Nulls

```
If @ErrID Is Null Set @ErrID=100139

Set @sCellValue = LTRIM(RTRIM(CAST(@CellValue As varchar(128))))
If LEN(@sCellValue) = 0        /* if empty */
    Select @CellValue = Null   /* make sure it is a Null */

If (@CellValue Is Not Null)

/* Exit if cell value is null */
  If ((IsNull(@Value,'') = '') Or ( @Value = '' ))
      Return 0
```

Strings

Case, Trim, SubString

```
If (upper(@Field_Name) like '%RANGE%')  /* upper, lower case */

Set @Valtype = Upper(@Valtype)
Set @Valtype = Lower(@Valtype)
Set @strlist =
  Ltrim(Substring(@strlist,Charindex(@delim, @strlist)+1, 4000))
```

Concatenation

```
Set @Ret=@Meter + ' missing'
Select @PinCnt = @instCPM_Pins + @instPAM_Pins

If (Select CPMpins+PAMpins From Tester_HardwareProfiles
        Where Tester_ID=@TesterID) < 8
```

Indexing, Checking for Characters

```
Set @CellCode=Left(@CellHandlingCode,2)

/* Does the numeric Value contain $ or comma */
    If ((CharIndex('$',@Value)>0) Or (CharIndex(',',@Value)>0))
      Return –1

Set @pos = Charindex(@Num, 'HR')
If ((@pos > 0) And (Len(@Num) <> 2))

/* Contains : H1-H8,HR
If (Left(@PinValue, 1) = 'H')
  If (Len(@PinValue) = 2) And
     (CharIndex(Upper(Right(@PinValue, 1)),'12345678R') > 0 )

/* Check If Character And valid */
  If ((Len(@Value) = 1)
        And (CharIndex(Upper(@Value),'ABCDEFGHIJKLMNOPQRSTUVWXYZ')>0))

Select @HISMUpins = Case When (Left(@Value, 1) = 'H') Then 1 Else 0 End
```

Split

```
/*--- Start DelimStrToTable ---*/
If Exists (Select * From Sysobjects
             Where Id = Object_ID(N'[DelimStrToTable]') And
                        Objectproperty(Id, N'IsTableFunction') = 1)
  Drop Function DelimStrToTable
Go

/* Begin of DelimStrToTable */
Create Function DelimStrToTable ( @delimStr Nvarchar(4000),
                                  @delimchar Char )

/* Returning a table variable */
Returns @StrValTable Table (StrVal Nvarchar(100),
                            Idx Int )
As
Begin
  Declare @strlist Nvarchar(4000)
  Declare @pos Int
  Declare @delim Char
  Declare @lstr Nvarchar(100)
  Declare @idx Int = 0

  Set @strlist = @delimStr
  Set @delim = @delimchar

While ((len(@strlist) > 0) And (@strlist Is Not Null))
Begin
  Set @idx=@idx + 1
  Set @pos = Charindex(@delim, @strlist)

    If @pos > 0
    Begin
        Set @lstr = Substring(@strlist, 1, @pos-1)
        Set @strlist =
Ltrim(Substring(@strlist,Charindex(@delim, @strlist)+1, 4000))
    End
    Else
    Begin
        Set @lstr = @strlist
        Set @strlist = Null
    End
    Insert @StrValTable Values (@lstr,@idx)
End

Return
End
Go
/*--- End of DelimStrToTable ---*/


Declare @CommaNumString varchar(12) = '1,2,3,4,5'
Declare @PinsTbl Table (StrVal Nvarchar(32),
                        Idx Int )

Insert Into @PinsTbl
    Select * From DelimStrToTable(@CommaNumString,',')
```

Basic SQL

Select

```
Set @Lbl = (Select Label
              From UI_Labels
              Where Language_ID = @Lang_ID And
                    Label_ID = @LblID)

/* check If this test in Table */
Set @ret = (Select Count(*) From ValidTests
   Where (Session_ID=@S_ID) And
         (IntraDieList_ID=@IDL_ID) And
         (IntraDieList_Version=@IDL_Ver) And
         (Test_ID=@T_ID) And (Test_Version=@T_Ver) And
         (Test_Order=@T_Ord))


Select @instCPM_Pins=CPMPins,
       @instPAM_Pins=PAMPins
  From Tester_HardwareProfiles
  Where Tester_ID=@TesterID


Select @LLimit = Cast(Val As Float) From ValConstants
                   Where EngTest = @EngTest
                   And StrRule = @Valtype
                   And Const = 'LLimit'

Select TOP 1 @MeasLeg = MeasurementLeg,
                @MeasMeter = MeasurementMeter,
                @HVSMUChk = PS1_HVSMU,
                @EngTest = EngineTest,
                @MeasType = MeasureType
      From Test_Switches
      Where Test_ID = @Test_ID
        And Test_Version = @Test_Version
```

Update

```
Update ValidTests Set Valid=@Valid
     Where (Session_ID=@S_ID)
     And   (IntraDieList_ID=@IDL_ID)
     And   (IntraDieList_Version=@IDL_Ver)
```

Inserting one Row

```
Insert Into tmp_SQLmessages (Session_ID, ErrorText, ErrorCode, MsgDate)
    Values(@S_ID, @Lbl, @ErrID, Getdate())
```

Inserting Multiple Rows

```
Insert Into tmp_SQLmessages (Session_ID, ErrorText)
   Select @sessionID, 'BLD$MINREQUIRE,'+@HdwList


Insert Into @PinsTbl
    Select * From DelimStrToTable(@Num,',')
```

Tables, Cursors

Table, Index Creation

```
If OBJECT_ID('[RIWEBSQL].[dbo].[ValCellLimits]') Is Null   /* already exist? */
  Begin
     Create Table [RIWEBSQL].[dbo].[ValCellLimits] (Label_ID int not null,
                                                    Test_ID int not null,
                                                    TesterID int not null,
                                                    Test_Version int not null,
                                                    LLimit float null,
                                                    ULimit float null,
                                                    sLRange varchar(32) null,
                                                    sURange varchar(32) null,
                                                    ErrorCode int null,
                                                    CanBeBlank bit null)

/* The TestID and Test_Version fields can make this table grow large.
   Without them we dont need this index - but well get an occasional
   error.  With them, this index is critical for performance (since
   the table can get large): */

Create Index IX_Label_ID_Test_ID
  On [RIWEBSQL].[dbo].[ValCellLimits](Label_ID,Test_ID)
  End
```

Cursors

```
Declare RecCursor Cursor Local Read_Only Forward_Only For
  Select ScreenLabels.Label_ID, Field_ID,
         ValidationType, CellHandlingCode
  From ScreenLabels, UI_Labels
  Where ((Screen_ID=@ScreenID) OR
        ((Screen_ID=115199) And
         ((@ScreenID = 115100) Or (@ScreenID = 115101) Or
          (@ScreenID = 115102) Or (@ScreenID = 115103))))
        And (ScreenLabels.Label_ID=UI_Labels.Label_ID)
        And (UI_Labels.Language_ID=@Lang_ID)
        And (Field_ID>0)
        And (CellHandlingCode Is Not Null)

/* Open And get the first record, If applicable */
Open RecCursor

Fetch From RecCursor
 Into @Label_ID, @Field_ID, @ValidationType, @CellHandlingCode

/* Handle all the records */
While @@Fetch_status=0 And @ValStat=0
Begin
…
  Fetch Next From RecCursor Into @Label_ID,
                                 @Field_ID,
                                 @ValidationType
                                 @CellHandlingCode
End /* While */
Close RecCursor      /* release recordset, unlocks rows held by cursor */
Deallocate RecCursor /* removes the structure pointed to be the cursor */
```

Temporary Local Table Cursor

```
/*-------------------------------------------------------------------------
 In order to avoid  row locking as we loop through, let's read into a local
table at one shot to be safer
-------------------------------------------------------------------------*/
If OBJECT_ID('tempdb..#tmp_ScreenLabels') Is Not Null
    Drop Table #tmp_ScreenLabels

  Create Table #tmp_ScreenLabels (Label_ID int,
                                  Label nvarchar(1000),
                                  Field_ID int,
                                  ValidationType varchar(100),
                                  CellHandlingCode varchar(10))
  Insert Into #tmp_ScreenLabels
    Select Distinct
      RIWEBSQL.dbo.UI_Labels.Label_ID,
                             Label,
                             Field_ID,
                             ValidationType,
                             CellHandlingCode
      From RIWEBSQL.dbo.UI_Labels, RIWEBSQL.dbo.ScreenLabels
      Where Screen_ID = @ScreenID
        And RIWEBSQL.dbo.UI_Labels.Label_ID =
            RIWEBSQL.dbo.ScreenLabels.Label_ID
        And (ValidationType Like 'cel%'  /* Like 'cel' is case insensitive */
          Or ValidationType Like 'pin%'
          Or ValidationType Like 'tst%'
          Or ValidationType Like 'sys%')
        And Language_ID=@Lang_ID
        And Field_ID>0
        And Field_ID Is Not Null
        And CellHandlingCode Is Not Null
      Order By RIWEBSQL.dbo.UI_Labels.Label_ID

  Declare @ScreenLabelsCursor Cursor

  Set @ScreenLabelsCursor = Cursor FORWARD_ONLY STATIC For
    Select Label_ID, Label, Field_ID, ValidationType, CellHandlingCode
      From #tmp_ScreenLabels

  Open @ScreenLabelsCursor
  Fetch Next From @ScreenLabelsCursor Into @Label_ID,
                                           @Label,
                                           @Field_ID,
                                           @ValidationType,
                                           @CellHandlingCode
  /*------------- Looping  --------------*/
  While @@Fetch_status=0
  Begin
    ...
    Fetch Next From @ScreenLabelsCursor Into @Label_ID,
                                             @Label,
                                             @Field_ID,
                                             @ValidationType,
                                             @CellHandlingCode

  End                              /* While somethig for the cursor to fetch */
  Close @ScreenLabelsCursor   /* release recordset, unlocks cursor rows */
  Deallocate @ScreenLabelsCursor  /* remove structure pointed to by cursor */
    If OBJECT_ID('tempdb..#tmp_ScreenLabels') Is Not Null
    Drop Table #tmp_ScreenLabels
```

Using Attributes of One Table to Reference Another

```
/*-------------------------------------------------------
     Where (in which table and column) the value is located
     for particular Test_ID and Test_Version is in the
     columns of the FieldNames table:
     e.g. Here is a sample of the UI_Labels, FieldNames join
          below:

       Field_Name             Table_Name
       = the column           = the table that
       name in the table      has this column
       that has this value
       ----------             -----------------
       High_Pin               Test_Facts
       Engine_Test            Test_Switches
  -------------------------------------------------------------*/
  Select  @Field_Name = FieldNames.Field_Name,
          @Table_Name = FieldNames.Table_Name
    From  UI_Labels, FieldNames
    Where FieldNames.Field_ID=UI_Labels.Field_ID
      And UI_Labels.Label_ID = @Label_ID
      And CellHandlingCode Is Not Null
      And UI_Labels.Field_ID>0
      And UI_Labels.Language_ID=@Lang_ID

Select @aCount = Count(@Field_Name)
If @aCount > 0   /* If anything there */
Begin

  Select @Column_Name = @Field_Name    /* for readability */

  If Object_ID('TempDB..#CellValTmp','U') Is Not Null
    Delete From #CellValTmp
  Else
    Create Table #CellValTmp (CellValue Sql_variant)

  /*-------------------------------------------------------
     Taking the above Table_Name and Column_Name, we setup
     a query which gives us results similar to (e.g.):
          Table_Name=    Column_Name=  Test_ID  Test_Version
          Test_Facts     High_Pin
                         -----------   -------  ------------
                         5             10033    1   <== so the CellValue is 5
  -------------------------------------------------------------*/
  Set @Qry = 'Insert into #CellValTmp Select ' +
      @Column_Name + ' From ' +
      @Table_Name + ' Where ' +
      'Test_ID' + ' = ' + Cast(@Test_ID As Varchar(10)) + ' And ' +
      'Test_Version' + ' = ' + Cast(@Test_Version As Varchar(10))

  Exec (@Qry)
  Set @CellValue = (Select CellValue From #CellValTmp) /*here is the CellValue*/


...
```

Try-Catch Error Handling

```
Begin Try
        Select @idx = CAST(@CellValue As Integer) /* eventually winds up in
html as enumerated Value= */
      End Try
      Begin Catch
        Select @idx = -1
      End Catch
```

Small Stored Procedure to Test a Bigger One

```
USE [RIWEBSQL]
GO

DECLARE @return_value Int = 0,
        @Test_ID Int = 0,
        @ScreenCount Int = 0,
        @startDateTime DateTime = GetDate(),
        @ScreenLabelsCursor Cursor

  Set @ScreenLabelsCursor = Cursor For
    Select Test_ID
    From RIWEBSQL.dbo.IntraDieList_Facts
    Where IntraDieList_ID = 10561
    Order By Test_ID


  Open @ScreenLabelsCursor
  Fetch Next From @ScreenLabelsCursor Into @Test_ID
  While @@Fetch_status=0
  Begin
    SELECT  @ScreenCount = @ScreenCount + 1

    Begin Try
      EXEC   @return_value = [dbo].[ValidateScreen]
                  @SessionID = 154197077,
                  @Lang_ID = 1,
                  @Test_ID = @Test_ID,
                  @Test_Version = 1,
                  @Test_Order = 1,
                  @TesterID = 10015,
                  @ITDListID = 10561,
                  @ITDListVer = 1,
                  @Single_Test = 1
       End Try
       Begin Catch
         Select 'ScreenCount' = @ScreenCount, ' Error @Test_ID' = @Test_ID
       End Catch

    /* SELECT  'Return Value' = @return_value, 'ScreenCount' = @ScreenCount */
    Fetch Next From @ScreenLabelsCursor Into @Test_ID
  End                             /* While */
  Close @ScreenLabelsCursor      /* release recordset, locks on cursor rows */
  Deallocate @ScreenLabelsCursor  /* removes structure pointed to by cursor */

SELECT 'Return Value' = @return_value, 'Count' = @ScreenCount
SELECT 'RunTime (mS)' = DATEDIFF(MS, @StartDateTime, GetDate() )
```