SDN Contribution

SQL TOOL for ABAPers

Applies to:

4.7 and above

Summary

SE16 is the best for 1 table inspection & you can open several sessions if you have more than 1 table.

You may "hate joins" and prefer looping matches on Internal Tables.

However if you wish to see the relationships in DATA VISIBLE format NOTHING succeeds like JOINs

I came in with a Strong Oracle TOAD background and feel comfortable in seeing DATA together

Created on: 28 April 2006

Author Bio

* Author Jayanta Narayan Choudhuri

* Flat 302

* 395 Jodhpur Park

* Kolkata 700 068

* Email sss@cal.vsnl.net.in

* URL: **http://www.geocities.com/ojnc**

Old ABAPer from Kolkata India


```
REPORT Yes4SQL.

* DV1K905202

* SQL tool for SAP ABAP Programmers - BOTH OPEN & NATIVE SQLs
*            very light - approx. 20KB
*
* Objective - to see JOINS in SAP to confirm or discover relationships
*           and to see Data side by Side

* Read "SAP Table and Field search strategies"
*   in http://sapabap.iespana.es/sapabap/sap/info/search_fields_tables.htm
*  Use SAP_TABLES.exe Document in http://www.sap-img.com
*  and many other excellent resources to navigate the cryptic tables & Colums of SAP
*
* SE16 is the best for 1 table inspection & you can open several sessions if you have more than
1 table.
* You may "hate joins" and prefer looping matches on Internal Tables.
* However if you wish to see the relationships in DATA VISIBLE format NOTHING succeeds like
JOINs
*  I came in with a Strong Oracle TOAD background and feel comfortable in seeing DATA
together

* The decision to use JOIN or use iterative Internal Table match with single select
* does not detract from the visibility of tracking relationships

* SQL Must be SELECT
* List of Columns Selected before 1st FROM Must Have
*      1 Column per line in format TABLE~COLUMN if Open SQL
*      1 Column per line in format TABLE.COLUMN if Native SQL
```



```
* Naturally tables & Columns must exist
* as this used to dynamically create Internal Table for ALV Grid
*
* Count( * ) is NOT SUPPORTED but you could Use Native COUNT( any NOT NULL NUMERIC
COLUMN )


* SUM MIN MAX AVG supported
* SUM( table~COLUMN ) or SUM( table.COLUMN )
* but there must be 1 space as indicated after ( and before ) -- even for native


*  If you use NATIVE SQL make sure you have :SY-MANDT filter in WHERE Clause


* You CAN pick 2 COLUMNS having same name - this is important for inspection
* Program creates right aliases as you can see in c:\jnc.ab4
* jnc.ab4 is the generated ABAP program for diagnostics and possible reuse


* JOINs and SUBQUERIES are NOT ALLOWED for
*          Pooled Tables, Clustered Tables & Projection Views
*   Even AGGREGATE Functions are NOT ALLOWED!   -- thse restrictions are inherent in SAP


* So this tool is useful for TRANSPARENT TABLES only!


*  Author Jayanta Narayan Choudhuri
*         Flat 302
*         395 Jodhpur Park
*         Kolkata 700 068
*       Email sss@cal.vsnl.net.in
*       URL:  http://www.geocities.com/ojnc


*  TextEdit Control Tool Code Copied from SAP Standard Example saptextedit_demo_3
*  This is FREE software with FULL responsibility on USER & anyone changing sourcecode!
```



```
DATA:
*     reference to wrapper class of control based on OO Framework
      g_editor TYPE REF TO cl_gui_textedit,
*     reference to custom container: necessary to bind TextEdit Control
      g_editor_container TYPE REF TO cl_gui_custom_container,
*     other variables
      g_ok_code LIKE sy-ucomm,        " return code from screen
      g_repid LIKE sy-repid.

Data: ROWS      type I,
     ISOPEN    type C,
     DELIM     type C.

DATA: code      Type Table of rssource-line,
     prog(8)   TYPE c,
     msg(120)  TYPE c,
     lin(3)    TYPE c,
     wrd(10)   TYPE c,
     off(3)    TYPE c.

Data: OneLineCode like LINE of Code.

Type-Pools : Slis.

DATA : fcat TYPE SLIS_T_FIELDCAT_ALV.
DATA : wcat LIKE LINE OF FCAT.

CONSTANTS: c_line_length TYPE i VALUE 80.
```

```
* define table type for data exchange
 TYPES: BEGIN OF mytable_line,
         line(c_line_length) TYPE c,
        END OF mytable_line.

* table to exchange text
 DATA g_mytable TYPE TABLE OF mytable_line.

 DATA: myLine like LINE of g_mytable.

* necessary to flush the automation queue
 CLASS cl_gui_cfw DEFINITION LOAD.

START-OF-SELECTION.
 Move 'X' to ISOPEN.
 Move 100 to Rows.

 CALL SCREEN 100.

*********************************************************************
*   P B O
*********************************************************************
MODULE pbo OUTPUT.

 SET PF-STATUS 'MAIN100'.
 SET TITLEBAR  'TITLEYES4SQL'.

 IF g_editor IS INITIAL.

*   initilize local variable with sy-repid, since sy-repid doesn't work
```

```
*    as parameter directly.
   g_repid = sy-repid.

*   create control container
   CREATE OBJECT g_editor_container
      EXPORTING
         container_name = 'MYEDIT'
      EXCEPTIONS
         cntl_error = 1
         cntl_system_error = 2
         create_error = 3
         lifetime_error = 4
         lifetime_dynpro_dynpro_link = 5.
   IF sy-subrc NE 0.
*     add your handling
   ENDIF.


*   create calls constructor, which initializes, creats and links
*    a TextEdit Control
   CREATE OBJECT g_editor
     EXPORTING
       parent = g_editor_container
       wordwrap_mode = cl_gui_textedit=>wordwrap_at_fixed_position
       wordwrap_to_linebreak_mode = cl_gui_textedit=>true
     EXCEPTIONS
        others = 1.
   IF sy-subrc NE 0.
     CALL FUNCTION 'POPUP_TO_INFORM'
         EXPORTING
```



```
            titel = g_repid
            txt2  = 'Create Object Failed'
            txt1  = 'to make TextEditor Control'.
      Leave Program.
    ENDIF.

  ENDIF.                        " Editor is initial

* remember: there is an automatic flush at the end of PBO!

ENDMODULE.                      " PBO



*----------------------------------------------------------------------*
*   P A I
*----------------------------------------------------------------------*
MODULE pai INPUT.

  CASE g_ok_code.

    WHEN 'EXIT'.
      PERFORM exit_program.

    WHEN 'EXEC'.
*     retrieve table from control
      Clear g_mytable.

      CALL METHOD g_editor->get_text_as_r3table
        IMPORTING
          table = g_mytable
```

```
      EXCEPTIONS
        OTHERS = 1.
    IF sy-subrc NE 0.
     CALL FUNCTION 'POPUP_TO_INFORM'
        EXPORTING
            titel = g_repid
            txt2  = 'Get_Text_As_R3Table Failed'
            txt1  = 'Unable to Store SQL'.
     Leave Program.
    ENDIF.

*    if you would like to work with the table contents
*    perform a explicit flush here allthough the method
*    flushes internally (at least up to release 4.6D).
*    The reason: don't rely on internal flushes of control
*    wrappers. These might vanish in the future leading to a
*    malfunction of your transaction. The additional flush here
*    does no harm. The autmation queue is empty and NO additional
*    roundtrip to the frontend will be triggered.
    CALL METHOD cl_gui_cfw=>flush
      EXCEPTIONS
        OTHERS = 1.
    IF sy-subrc NE 0.
     CALL FUNCTION 'POPUP_TO_INFORM'
        EXPORTING
            titel = g_repid
            txt2  = 'cl_gui_cfw=>flush Failed'
            txt1  = 'Exiting Program'.
     Leave Program.
    ENDIF.
```



```
    Perform F_RUNSQL.

 ENDCASE.

 CLEAR g_ok_code.
ENDMODULE.                    " PAI


*---------------------------------------------------------------------*
*  F O R M S
*---------------------------------------------------------------------*

*&---------------------------------------------------------------------*
*&      Form  EXIT_PROGRAM
*&---------------------------------------------------------------------*
FORM exit_program.
* Destroy Control.
  IF NOT g_editor IS INITIAL.
    CALL METHOD g_editor->free
      EXCEPTIONS
        OTHERS = 1.
    IF sy-subrc NE 0.
      CALL FUNCTION 'POPUP_TO_INFORM'
          EXPORTING
              titel = g_repid
              txt2  = 'g_editor->free Failed'
              txt1  = 'Exiting Program'.
          Leave Program.
    ENDIF.
```

```
*   free ABAP object also
    FREE g_editor.
  ENDIF.


* destroy container
  IF NOT g_editor_container IS INITIAL.
    CALL METHOD g_editor_container->free
      EXCEPTIONS
        OTHERS = 1.
    IF sy-subrc <> 0.
*       MESSAGE E002 WITH F_RETURN.
    ENDIF.
*   free ABAP object also
    FREE g_editor_container.
  ENDIF.


* finally flush
  CALL METHOD cl_gui_cfw=>flush
     EXCEPTIONS
        OTHERS = 1.
  IF sy-subrc NE 0.
    CALL FUNCTION 'POPUP_TO_INFORM'
       EXPORTING
          titel = g_repid
          txt2  = 'cl_gui_cfw=>flush Failed'
          txt1  = 'Exiting Program'.
      Leave Program.
  ENDIF.
```



```
  LEAVE PROGRAM.

ENDFORM.                    " EXIT_PROGRAM

*&---------------------------------------------------------------------*
*&      Form  F_RUNSQL
*&---------------------------------------------------------------------*
FORM F_RUNSQL.

Data: first     type I,
      numCols   type I,
      aggfun    type I,
      Pos       Type I,
      Off       Type I,
      Len       Type I,
      NumRows   Type I,
      RowNum    Type I,
      MyString  Type String,
      MyString2 Type String,
      CRows(8)  Type C.

data : begin of TBLCOL_TAB occurs 0,
          TBL type String,
          COL type String.
data : end of TBLCOL_TAB.

    If Rows is Initial.
      Move 100 to Rows.
    EndIf.
```



```
Clear Code.

Move 0 to : first, numCols, aggFun.

Loop At g_mytable Into myLine.
  Concatenate ' ' myLine ' ' into myLine SEPARATED BY SPACE..

  If strlen( myLine ) = 0.
    continue.
  EndIf.

  If first = 0.
    Find ' Select ' in myLine Ignoring Case.
    If SY-SubRc <> 0.
      CALL FUNCTION 'POPUP_TO_INFORM'
        EXPORTING
          titel = g_repid
          txt2  = 'SELECT DMLs Only Please!'
          txt1  = 'Correct & Retry'.
      Return.
    EndIf.
    Move 1 to first.
  EndIf.

  Find ' From ' in MyLine Ignoring case.
  If SY-SubRc = 0.
    Exit.
  EndIf.
```

```
If IsOpen = 'X'.
  move '~' to delim.
Else.
  move '.' to delim.
EndIf.

Find delim in myLine Match Offset off.
If SY-SubRc <> 0.
  Continue.
EndIf.

Add 1 to NumCols.

Compute Pos = Off - 1.

Do.
 If MyLine+Pos(1) = Space.
   Exit.
 EndIf.
 Subtract 1 from Pos.
EndDo.

Add 1 to Pos.
Compute Len = Off - Pos.
Move MyLine+Pos(Len) to TBLCOL_TAB-TBL.
Move MyLine+Pos(Len) to myString.

Compute Pos = Off + 1.

Do.
```

```
  If MyLine+Pos(1) = Space OR MyLine+Pos(1) = ','.
    Exit.
  EndIf.
  Add 1 to Pos.
EndDo.

Subtract 1 from Pos.
Compute Len = Pos - Off.
Compute Pos = Off + 1.
Move MyLine+Pos(Len) to TBLCOL_TAB-COL.
Append TBLCOL_TAB.

Concatenate myString delim MyLine+Pos(Len) into myString.

Write numCols to CRows.
Concatenate 'WFLD' CRows Into MyString2.
Condense MyString2 No-Gaps.

Find ')' in myLine.
If Sy-SubRC = 0.
   Move 1 to AggFun.
   Concatenate ') as' myString2 into myString2 Separated by Space.
   Replace ')' in myLine with myString2.
Else.
   Concatenate myString 'as' myString2 into myString2 Separated by Space.
   Replace myString in myLine with myString2.
EndIf.

Modify g_mytable from MyLine.
```

```
 EndLoop.

If Lines( TBLCOL_TAB ) = 0.
  CALL FUNCTION 'POPUP_TO_INFORM'
      EXPORTING
         titel = g_repid
         txt2  = 'Table~Column Open SQL is MUST'
         txt1  = 'Table.Column Native SQL is MUST'.
  Return.
EndIf.

Append 'Program SubPool.' to Code.
Append '' to Code.

Append 'data : begin of I_TAB occurs 0,' to Code.

NumRows = Lines( TBLCOL_TAB ).

Move 0 to RowNum.

Loop At TBLCOL_TAB.
   Add 1 to RowNum.
   Write RowNum to CRows.
   Concatenate 'WFLD' CRows Into MyString.
   Condense MyString no-gaps.

   Concatenate MyString 'Like' TBLCOL_TAB-TBL  into MyString SEPARATED BY SPACE.

   If RowNum = NumRows.
     Concatenate MyString '-' TBLCOL_TAB-COL '.' into MyString.
```

```
    Else.
      Concatenate MyString '-' TBLCOL_TAB-COL ',' into MyString.
    EndIf.
    Append MyString to Code.
EndLoop.
Append 'data : end of I_TAB.' to Code.
Append '' to Code.
Append 'DATA : R_TAB LIKE LINE OF I_TAB,' to Code.
Append '      L_KOUNT    type I.' to Code.
Append '' to Code.

Append 'DATA : MyString type STRING.'         to Code.
Append 'DATA : MyTitle  type LVC_TITLE.'      to Code.
Append '' to Code.

Append 'Data: ROWS     type I.' to Code.
Append '' to Code.

Append 'Form DoSQL.' to Code.
Append '' to Code.
Write Rows to CRows.
Replace all Occurrences of ',' in Crows With ''.
Concatenate 'Move' Crows 'to ROWS.' into MyString SEPARATED BY SPACE.
Append MyString to Code.
Append '' to Code.

Move 0 to RowNum.
Loop At TBLCOL_TAB.
    Add 1 to RowNum.
    If RowNum = 1.
```



```
        Concatenate '    Move ''' TBLCOL_TAB-COL  ''' to MyString.' into MyString.
        Append MyString to Code.
      Else.
        Concatenate '    Concatenate MyString ',' ''' TBLCOL_TAB-COL  ''' Into MyString.' into
MyString.
        Append MyString to Code.
      EndIf.
      Append '    Move MyString to MyTitle.' to Code.
      Append '' to Code.
  EndLoop.


  Append '' to Code.
  Append 'Try.' to Code.
  Append '' to Code.


  If IsOpen <> 'X'.
    Append 'Move 0 to L_KOUNT.'  to Code.
    Append 'EXEC SQL.'  to Code.
    Append ' open c1 for ' to Code.
  EndIf.


  Move 0 to first.
  Loop At g_mytable into MyLine.
     If IsOpen = 'X' and first = 0.
       Find ' From ' in myLine ignoring case.
       If SY-SubRC = 0.
         Append 'Into Table I_TAB' to Code.
         If aggFun = 0.
           Append 'Up To ROWS rows' to Code.
         EndIf.
         move 1 to first.
```

```
      EndIf.
    EndIf.
    Append MyLine to Code.
EndLoop.


If IsOpen = 'X'.
  If first = 0.
    CALL FUNCTION 'POPUP_TO_INFORM'
        EXPORTING
            titel = g_repid
            txt2  = 'Open SQL Without a FROM'
            txt1  = 'Correct & retry'.
    Return.
  EndIf.
  Append '.' to Code.
Else.
  Append 'ENDEXEC.'  to Code.
  Append '' to Code.
  Append 'DO.'  to Code.
  Append '  EXEC SQL.'  to Code.
  Append '    fetch next c1 INTO :R_TAB '  to Code.
  Append '  ENDEXEC.'  to Code.
  Append '  IF sy-subrc <> 0.'  to Code.
  Append '    EXIT.'  to Code.
  Append '  ENDIF.'  to Code.
  Append '  Append R_TAB to I_TAB.'  to Code.
  Append '  Add 1 to L_KOUNT.'  to Code.
  Append '  If L_KOUNT >= ROWS.'  to Code.
  Append '    Exit.'  to Code.
  Append '  EndIf.'  to Code.
```

```
  Append 'ENDDO.'  to Code.
  Append '' to Code.
  Append 'EXEC SQL.'  to Code.
  Append '   close c1'  to Code.
  Append 'ENDEXEC.'  to Code.
  Append '' to Code.
EndIf.

Append '' to Code.
Append 'PERFORM zjnc_dump_list  USING ''I_TAB[]'' ''I_TAB'' MyTitle.' to Code.
Append '' to Code.
Append 'Catch CX_ROOT.' to Code.
If IsOpen <> 'X'.
  Append 'EXEC SQL.'  to Code.
  Append '   close c1'  to Code.
  Append 'ENDEXEC.'  to Code.
EndIf.
Append 'CALL FUNCTION ''POPUP_TO_INFORM''' to Code.
Append '   EXPORTING' to Code.
Append '       titel = ''jncDynamicSub''' to Code.
Append '       txt2  = ''Generate SUBROUTINE POOL Succeeded BUT SQL failed''' to Code.
Append '       txt1  = ''Possible Wrong SQL - see c:\jnc.ab4''.' to Code.
Append 'EndTry.' to Code.
Append 'EndForm.  "DoSQL.' to Code.

Append '' to Code.
Append '*&---------------------------------------------------------------------*' to Code.
Append '*&      Form  ZJNC_DUMP_LIST  Our Good Old ALV list -  RECOMMENDED!' to Code.
Append '*&---------------------------------------------------------------------*' to Code.
Append 'FORM zjnc_dump_list  USING value(p_it_name) TYPE c' to Code.
```

```
Append '                    value(p_wa_name) TYPE c' to Code.
Append '                    value(p_heading) TYPE c.' to Code.
Append '' to Code.
Append '  TYPE-POOLS: slis.' to Code.
Append '' to Code.
Append '  DATA:' to Code.
Append '    stru_ref    TYPE REF TO cl_abap_structdescr,' to Code.
Append '    comp_tab    TYPE abap_compdescr_tab,' to Code.
Append '    one_comp    TYPE abap_compdescr,' to Code.
Append '    one_name    TYPE string,' to Code.
Append '    type_ref    TYPE REF TO cl_abap_typedescr,' to Code.
Append '    is_ddic     TYPE abap_bool,' to Code.
Append '    lt_ddic     TYPE dd_x031l_table,' to Code.
Append '    wa_ddic     TYPE x031l,' to Code.
Append '    lt_fcat     TYPE slis_t_fieldcat_alv,' to Code.
Append '    wa_fcat     TYPE slis_fieldcat_alv,' to Code.
Append '    ls_layo     TYPE slis_layout_alv,' to Code.
Append '    l_alv       TYPE REF TO cl_gui_alv_grid.' to Code.
Append '' to Code.
Append '  FIELD-SYMBOLS: <fs_type>  TYPE ANY,' to Code.
Append '                 <fs_table> TYPE STANDARD TABLE,' to Code.
Append '                 <fs_line>  TYPE ANY.' to Code.
Append '' to Code.
Append '  ASSIGN (p_it_name) TO <fs_table>.' to Code.
Append '' to Code.
Append '  ASSIGN (p_wa_name) TO <fs_line>.' to Code.
Append '' to Code.
Append '  ls_layo-colwidth_optimize = ''X''.' to Code.
Append '  ls_layo-zebra = ''X''.' to Code.
Append '  ls_layo-window_titlebar = p_heading.' to Code.
```

```
Append '  ls_layo-box_tabname   = p_it_name.' to Code.
Append '' to Code.
Append '  stru_ref ?= cl_abap_structdescr=>describe_by_data( <fs_line> ).' to Code.
Append '' to Code.
Append '  comp_tab = stru_ref->components.' to Code.
Append '' to Code.
Append '  LOOP AT comp_tab INTO one_comp.' to Code.
Append '    CLEAR wa_fcat.' to Code.
Append '    wa_fcat-tabname   = p_it_name.' to Code.
Append '    wa_fcat-fieldname = one_comp-name.' to Code.
Append '' to Code.
Append '    CONCATENATE p_wa_name ''-'' one_comp-name INTO one_name.' to Code.
Append '' to Code.
Append '    ASSIGN (one_name) TO <fs_type>.' to Code.
Append '' to Code.
Append '    type_ref ?= cl_abap_typedescr=>describe_by_data( <fs_type> ).' to Code.
Append '' to Code.
Append '    is_ddic = type_ref->is_ddic_type( ).' to Code.
Append '' to Code.
Append '    IF is_ddic = abap_true.' to Code.
Append '     lt_ddic = type_ref->get_ddic_object( ).' to Code.
Append '' to Code.
Append '     LOOP AT lt_ddic INTO wa_ddic.' to Code.
Append '       CLEAR wa_ddic-tabname.' to Code.
Append '       SELECT SINGLE' to Code.
Append '            dd03l~tabname' to Code.
Append '         INTO wa_ddic-tabname' to Code.
Append '         FROM dd03l' to Code.
Append '        WHERE dd03l~fieldname = wa_ddic-fieldname' to Code.
Append '          AND dd03l~tabname NOT LIKE ''/%''.  " only normal namespace' to Code.
```

```
Append '' to Code.
Append '' to Code.
Append '       wa_fcat-ref_tabname    = wa_ddic-tabname.' to Code.
Append '       wa_fcat-ref_fieldname = wa_ddic-fieldname.' to Code.
Append '' to Code.
Append '       SELECT SINGLE' to Code.
Append '            dd04t~scrtext_s' to Code.
Append '            dd04t~scrtext_m' to Code.
Append '            dd04t~scrtext_l' to Code.
Append '        INTO (wa_fcat-seltext_s, wa_fcat-seltext_m, wa_fcat-seltext_l)' to Code.
Append '        FROM dd04t' to Code.
Append '       WHERE dd04t~rollname   = wa_ddic-fieldname' to Code.
Append '         AND dd04t~ddlanguage = sy-langu.' to Code.
Append '' to Code.
Append '     ENDLOOP.' to Code.
Append '    ELSE.' to Code.
Append '     MOVE one_comp-name TO: wa_fcat-seltext_s, wa_fcat-seltext_m,' to Code.
Append '                            wa_fcat-seltext_l.' to Code.
Append '    ENDIF.' to Code.
Append '' to Code.
Append '    APPEND wa_fcat TO lt_fcat.' to Code.
Append '' to Code.
Append '  ENDLOOP.' to Code.
Append '' to Code.
Append '  CALL FUNCTION ''REUSE_ALV_GRID_DISPLAY''' to Code.
Append '    EXPORTING' to Code.
Append '     is_layout   = ls_layo' to Code.
Append '     it_fieldcat = lt_fcat' to Code.
Append '    TABLES' to Code.
Append '     t_outtab    = <fs_table>.' to Code.
```

```
Append '' to Code.
Append 'ENDFORM.                "ZJNC_DUMP_LIST' to Code.

CALL FUNCTION 'GUI_DOWNLOAD'
  EXPORTING
    FILENAME                = 'c:\jnc.ab4'
  TABLES
    DATA_TAB                = CODE.


IF SY-SUBRC <> 0.
  MESSAGE ID SY-MSGID TYPE SY-MSGTY NUMBER SY-MSGNO
         WITH SY-MSGV1 SY-MSGV2 SY-MSGV3 SY-MSGV4.
ENDIF.


GENERATE SUBROUTINE POOL code   NAME    prog
                        MESSAGE msg
                        LINE    lin
                        WORD    wrd
                        OFFSET  off.

IF sy-subrc <> 0.
  CALL FUNCTION 'POPUP_TO_INFORM'
      EXPORTING
          titel = g_repid
          txt2  = 'Generate SUBROUTINE POOL Failed'
          txt1  = 'Possible Nonconformant SQL - see c:\jnc.ab4'.
ELSE.
  PERFORM DoSQL IN PROGRAM (prog).
```

```
    IF sy-subrc <> 0.
      CALL FUNCTION 'POPUP_TO_INFORM'
        EXPORTING
          titel = g_repid
          txt2  = 'Generate SUBROUTINE POOL Succeeded BUT Call failed'
          txt1  = 'Possible Wrong SQL - see c:\jnc.ab4'.
    ENDIF.
  ENDIF.

ENDFORM.       "F_RUNSQL
```

Related Content

1.

Disclaimer and Liability Notice