

SHADE GRL - Export to Excel via SQL

DESCRIPTION

This document describe a routine that exports to MS-Excel a SQL command, without creating a CSV file. The routine opens a Excel worksheet automatically and import all the data from the SQL command.

INITIAL CONSIDERATIONS

- This routine **MUST** run within a form and **NOT** from a database.
- The MS-Excel software **MUST BE INSTALLED** in the PC you run this routine. If not, the routine will show an error.
- The Export has 2 parts:
 - The procedure **EXPORT_EXCEL_bySQL**
 - A trigger that calls the procedure (for example: WHEN-BUTTON-PRESSED)
- The procedure EXPORT_EXCEL_bySQL uses the **OLE2** and **EXEC_SQL** form packages. This is excellent, because we don't have to download or install anything. All is included in ORACLE FORMS.

EXPORT_EXCEL_bySQL

This procedure has 2 parameters:

- Data_SQL: the SQL command that we want to export to Excel
- Worksheet_name: [optional] the name of the Excel worksheet. Is not the name of the Excel file, but the worksheet.

```
PROCEDURE Export_Excel_bySQL
```

```
(data_SQL IN VARCHAR2,  
 worksheet_name IN VARCHAR2 DEFAULT 'Hoja1')
```

```
IS
```

```
APPLICATION OLE2.OBJ_TYPE;  
WORKBOOKS OLE2.OBJ_TYPE;  
WORKBOOK OLE2.OBJ_TYPE;  
WORKSHEETS OLE2.OBJ_TYPE;  
WORKSHEET OLE2.OBJ_TYPE;  
Arglist OLE2.LIST_TYPE;  
CELL OLE2.OBJ_TYPE;  
r INTEGER;  
c INTEGER;  
i INTEGER;  
tit_long INTEGER;  
ini INTEGER;  
pos INTEGER;  
file_name_cl VARCHAR2 (32767);  
user_cancel EXCEPTION;
```

```

Workfont OLE2.OBJ_TYPE;
WorkInterior OLE2.OBJ_TYPE;
SQL_cmd VARCHAR2(2000);
-- Definir datos del Cursor que se ingresa por parametro
conn_id EXEC_SQL.conntype;
cursor_id EXEC_SQL.curstype;
colNpos PLS_INTEGER;
colName VARCHAR2(30);
colLen PLS_INTEGER;
colType PLS_INTEGER;
auxCursor PLS_INTEGER;
TYPE table_rec IS RECORD(title VARCHAR2(30),
                        type VARCHAR2(10),
                        long NUMBER
                        );
TYPE table_def IS TABLE OF table_rec
                INDEX BY BINARY_INTEGER;
table_exp table_def;
colNumber NUMBER;
colVarchar VARCHAR2(200);
colDate DATE;

```

--Inner Proc.

```

PROCEDURE put_cell (Row_num NUMBER,
                  Col_num NUMBER,
                  put_value VARCHAR2,
                  font_name VARCHAR2 DEFAULT NULL ,
                  font_size BINARY_INTEGER DEFAULT NULL ,
                  font_style VARCHAR2 DEFAULT NULL ,

```

/*here you can pass BOLD for

bold, ITALIC for italic etc*/

```

                  font_color BINARY_INTEGER DEFAULT NULL
                )
IS
BEGIN
    Arglist := OLE2.create_arglist;
    OLE2.add_arg (Arglist, row_num);
    OLE2.add_arg (Arglist, col_num);
    cell := OLE2.get_obj_property (Worksheet, 'Cells', Arglist);
    OLE2.destroy_arglist (Arglist);
    OLE2.set_property (cell, 'Value', put_value);

```

```
Workfont      := OLE2.get_obj_property (cell, 'Font');
WorkInterior  := OLE2.get_obj_property (cell, 'Interior');
```

```
IF font_name IS NOT NULL THEN
  OLE2.set_property (Workfont, 'Name', font_name);
END IF;
```

```
IF font_size IS NOT NULL THEN
  OLE2.set_property (Workfont, 'Size', font_size);
END IF;
```

```
IF font_style IS NOT NULL THEN
  OLE2.set_property (Workfont, font_style, 1);
END IF;
```

```
IF font_color IS NOT NULL THEN
  OLE2.set_property (Workfont, 'ColorIndex', font_color);
END IF;
```

```
OLE2.RELEASE_OBJ (workinterior);
OLE2.RELEASE_OBJ (workfont);
OLE2.RELEASE_OBJ (cell);
END;
```

```
BEGIN
```

```
--Open Excel Application and make it visible
```

```
APPLICATION := OLE2.CREATE_OBJ ('Excel.Application');
OLE2.SET_PROPERTY (APPLICATION, 'Visible', TRUE);
```

```
-----
--Open a specified file where i want to export data
```

```
/* Arglist := OLE2.create_arglist;
OLE2.add_arg(Arglist,'C:\temp\test.xls');
WORKBOOKS := OLE2.GET_OBJ_PROPERTY(APPLICATION, 'WORKBOOKS');
WORKBOOK := OLE2.INVOKE_OBJ(WORKBOOKS, 'Open',Arglist);
OLE2.destroy_arglist(Arglist); */
```

```
/*open new file*/
```

```
WORKBOOKS := OLE2.GET_OBJ_PROPERTY (APPLICATION, 'WORKBOOKS');
WORKBOOK := OLE2.INVOKE_OBJ (WORKBOOKS, 'Add');
```

```
-----
---Initilize work sheet
```

```
Worksheet := OLE2.get_obj_property(Application,'Activesheet');
OLE2.set_property(Worksheet,'Name',worksheet_name);
```

```

/*Print Titles and Data*/
-- Crear y abrir un cursor y ejecutar el SQL que se envi3 por par3metro en data_SQL
BEGIN
  cursor_id := EXEC_SQL.OPEN_CURSOR;
  EXEC_SQL.PARSE(cursor_ID, data_SQL);

  -- Cargar los titulos en el Excel y definir (DEFINE_COLUMN) los campos que se van a leer en el SQL
  c := 0;
  LOOP
    c := c + 1;

    BEGIN
      -- Obtener datos de la columna
      EXEC_SQL.DESCRIBE_COLUMN(cursor_ID,
                               c,          -- nro. de columna
                               colName,   -- npmbre de la columna dada por el SQL
                               colLen,    -- longitud en bytes
                               colType    -- datatype
                              );
      -- Setear Datos
      table_exp(c).title := colName;
      table_exp(c).type := colType;
      table_exp(c).long := colLen;

      -- Definir la variable que contendra el valor de esta columna luego de ejecutar el SQL
      IF colType = EXEC_SQL.NUMBER_TYPE THEN
        EXEC_SQL.DEFINE_COLUMN(cursor_ID, c, colNumber);

      ELSIF colType = EXEC_SQL.VARCHAR2_TYPE THEN
        EXEC_SQL.DEFINE_COLUMN(cursor_ID, c, colVarchar, colLen);

      ELSIF colType = EXEC_SQL.DATE_TYPE THEN
        EXEC_SQL.DEFINE_COLUMN(cursor_ID, c, colDate);

      ELSE
        RAISE EXEC_SQL.package_error;

      END IF;

      -- Grabar en el Excel el Titulo
      put_Cell (1, -- fila 1 de la planilla excel
                c,
                colName,
                font_style => 'BOLD'
               );
    END LOOP;
  END;

```

```

        );
    EXCEPTION
    WHEN EXEC_SQL.INVALID_COLUMN_NUMBER THEN
        EXIT;
    END;

END LOOP;

c := c - 1;

EXCEPTION
WHEN exec_sql.package_error THEN
    Message('ERROR GRL-0001 (E): error al llamar al MS-Excel. (PRUTIL01.Export_Excel_byCursor). '
        || CHR(10) || exec_sql.last_error_mesg
        || CHR(10) || ' Rows:' || TO_CHAR(exec_sql.last_Row_Count)
    );
END ;

-- Cargar los datos en el Excel
BEGIN

auxCursor := EXEC_SQL.EXECUTE(cursor_ID);
r := 2;
WHILE EXEC_SQL.FETCH_ROWS(cursor_ID) > 0
LOOP

    FOR i IN 1 .. c
    LOOP

        -- Determinar el datatype del dato a leer y exportar a Excel
        IF table_exp(i).type = EXEC_SQL.NUMBER_TYPE THEN

            EXEC_SQL.COLUMN_VALUE(cursor_ID, i, colNumber);
            put_Cell (r, i, colNumber);

        ELSIF table_exp(i).type = EXEC_SQL.VARCHAR2_TYPE THEN

            EXEC_SQL.COLUMN_VALUE(cursor_ID, i, colVarchar);
            put_Cell (r, i, colVarchar);

        ELSIF table_exp(i).type = EXEC_SQL.DATE_TYPE THEN

            EXEC_SQL.COLUMN_VALUE(cursor_ID, i, colDate);
            put_Cell (r, i, colDate);

```

```

ELSE
  NULL;

END IF;

END LOOP;

r := r + 1;

END LOOP;

EXCEPTION
WHEN exec_sql.package_error THEN
  Message('ERROR GRL-0001 (E): error al llamar al MS-Excel. (PRUTIL01.Export_Excel_byCursor). '
    || CHR(10) || exec_sql.last_error_mesg
    || CHR(10) || ' Rows:' || TO_CHAR(exec_sql.last_Row_Count)
  );

END;

EXEC_SQL.CLOSE_CURSOR(cursor_ID);

/*
-- Save the Excel file created
Arglist := OLE2.Create_Arglist;
OLE2.Add_Arg (Arglist, 'c:\temp\test.xls');
OLE2.Invoke (workbook, 'Save', Arglist);
OLE2.Destroy_Arglist (Arglist);
*/
-- release workbook
OLE2.RELEASE_OBJ (worksheet);
OLE2.RELEASE_OBJ (workbook);
OLE2.RELEASE_OBJ (workbooks);
OLE2.RELEASE_OBJ (application);
END;
```

TRIGGER: WHEN-BUTTON-PRESSED

Besides the procedure, we need a trigger to call the EXPORT_EXCEL_bySQL procedure. We can use a PUSH_BUTTON item with a WHEN-BUTTON-PRESSED trigger, or We can use an IMAGE item with a WHEN-IMAGE-PRESSED trigger.

The coding for this trigger could be as follows:

```

-- Example for Export to Excel trigger
DECLARE
    SQL_cmd VARCHAR2(7000);
BEGIN
    SQL_cmd := 'select ....';

    Export_Excel_bySQL(SQL_cmd, 'Datos_SQL');

    Go_Block(:GLOBAL.cur_block);

EXCEPTION
WHEN OTHERS THEN
    Msg_alert('ERROR GRL-0001 (E): error calling MS-Excel <form name>.Export_Excel_bySQL.'
        || SQLERRM,
        'E',
        TRUE
    );
END;

```

Some important considerations when coding the SQL command in the trigger

- First of all, it has to be coded within a VARCHAR2 variable
- All the apostrophes (') must be in pairs. If not, the compiler will consider them as text separators.

Example:

```
AND prv_fecha = TO_DATE('01-10-2013', 'dd/mm/yyyy')
```

Must be replace by:

```
AND prv_fecha = TO_DATE(''01-10-2013'', ''dd/mm/yyyy'')
```

- We **MUST** convert all the bind variables we used in the WHERE clause

Examples

a) *NUMERIC bind variable:*

From: AND cliente_ID = :CLI.cliente_ID

To: 'AND cliente_ID = ' || TO_CHAR(:CLI.cliente_ID)

b) *DATE bind variable:*

From: AND pro_fecha >= :CBTE.pro_fecha

To: 'AND TO_CHAR(pro_fecha, "yyyy-mm-dd") >= ' || TO_CHAR(:CBTE.pro_fecha, 'yyyymmdd') || ''

- Pay attention to the amount of apostrophes that it has to be use
- The date is converted to text, using the YEAR-MONTH-DAY format (yyyymmdd)
- The SQL **MUST NOT** finish with the classic ; (semi-colon).

SQL EXAMPLE

The following SQL is only an example of coding a medium complexity SQL into a VARCHAR2 variable.

```
SQL_cmd := 'select MIN(SEN.sen_descripcion)           as Señal,'
|| ' MIN(TO_CHAR(pdd_hora, "yyyy-mm-dd hh24:mi:ss")) as ONAIR,'
|| ' PDD.MAT_CODIGO                               as Material,'
|| ' PDD.mde_codigo                               as Capitulo,'
|| ' MAX(TO_CHAR(RMA.emp_codigo, "99") || "-" || RMA.sop_codigo || "-" || RMA.tro_codigo || "-" ||
LTRIM(TO_CHAR(RMA.rol_codigo))) as SOPORTE,'
|| ' MAX(MAT.mat_titulo_origen_principal)         as MAT_TIT_ORIGEN,'
|| ' MAX(DET.MDE_TITULO_ORIGEN)                   as CAP_Tit_Origen,'
|| ' MAX(RMA.rol_codigo_juego)                    AS Juego_Rollos,'
|| ' MAX(RMA.rma_house_dal)                       AS HouseDAL,'
|| ' MAX(DECODE(HDAL.lho_marca, "S", "Si", "No")) AS Ingestado,'
|| ' MAX(RMA.rma_timecode_inicio)                 AS TimeCode_Inicio,'
|| ' MAX(RMA.rma_duracion)                       AS Duracion,'
|| ' MAX(TO_CHAR(RMA.rma_numero_parte, "99") || " de" ||
TO_CHAR(RMA.rma_cantidad_partes, "99")) as Partes,'
|| ' MAX(RMA.etc_codigo)                          as E_Tecnico,'
|| ' MAX(ETC.etc_descripcion)                     as ET_Descripcion,'
|| ' MAX(RMA.nor_codigo)                          as Norma,'
|| ' MAX(PRO_DOB_SUB_IDIOMAS(PDD.emp_codigo, RMA.sop_codigo, RMA.tro_codigo,
RMA.rol_codigo, MAT.mat_codigo, DET.mde_codigo, "D")) as Canales_Audio,'
|| ' MAX(PRO_DOB_SUB_IDIOMAS(PDD.emp_codigo, RMA.sop_codigo, RMA.tro_codigo,
RMA.rol_codigo, MAT.mat_codigo, DET.mde_codigo, "S")) as Subtitulos,'
|| ' MAX(DECODE((SELECT NVL(SUM(1), 0) '
|| '           from pro_rollo_mat_subtit rms '
|| '           where rms.emp_codigo = PDD.emp_codigo_rma '
|| '           and rms.sop_codigo = PDD.sop_codigo '
|| '           and rms.tro_codigo = PDD.tro_codigo '
|| '           and rms.rol_codigo = PDD.rol_codigo '
|| '           and rms.mat_codigo = PDD.mat_codigo_rma '
|| '           and rms.mde_codigo = PDD.mde_codigo_rma '
|| '           and rms.tsu_codigo = "I"), '
|| '           0, "Normal", "Alta") '
|| ' ) AS Prioridad_Subtit,'
|| ' MAX(PRO_DOB_SUB_IDIOMAS(PDD.emp_codigo, RMA.sop_codigo, RMA.tro_codigo,
RMA.rol_codigo, MAT.mat_codigo, DET.mde_codigo, "C")) as Cortes,'
|| ' MAX(PRO_DOB_SUB_IDIOMAS(PDD.emp_codigo, RMA.sop_codigo, RMA.tro_codigo,
RMA.rol_codigo, MAT.mat_codigo, DET.mde_codigo, "G")) as Grafica,'
|| ' MAX(PRO_ULTIMO_MOV_SOPORTE(PDD.emp_codigo, "BDIG", RMA.tro_codigo,
RMA.rol_codigo, "TE"))                          as Ultimo_Movimiento '
|| ' from pro_programacion_dia_detalles PDD,'
```



```

||'   pro_materiales MAT, '
||'   pro_mat_detalles DET, '
||'   pro_tipo_materiales TIP, '
||'   pro_señales SEN, '
||'   pro_rollo_materiales RMA, '
||'   pro_estado_tecnicos ETC, '
||'   pro_log_house_dal HDAL '
||' where PDD.emp_codigo  = ' || TO_CHAR(:PARAMETER.emp_cod)
||' and PDD.sen_codigo  = "" || :GUIA_TX.sen_codigo ||""
||' and TO_CHAR(PDD.pro_fecha, "yyyymmdd") >= "" || TO_CHAR(:GUIA_TX.pro_fecha_desde,
'yyyymmdd') || ""
||' and TO_CHAR(PDD.pro_fecha, "yyyymmdd") <= "" || TO_CHAR(:GUIA_TX.pro_fecha_hasta,
'yyyymmdd') || ""
||' and MAT.mat_codigo  = PDD.mat_codigo '
||' and TIP.tma_codigo  = MAT.tma_codigo '
||' and DET.mat_codigo  = PDD.mat_codigo '      -- PRO_MAT_DETALLES (capitulos)
||' and DET.mde_codigo  = PDD.mde_codigo '
||' and SEN.emp_codigo  = PDD.emp_codigo '      -- PRO_SEÑALES (señales)
||' and SEN.sen_codigo  = PDD.sen_codigo '
||' and RMA.emp_codigo(+) = PDD.emp_codigo_rma ' -- PRO_ROLLO_MATERIALES
(obtengo los soportes del material)
||' and RMA.sop_codigo(+) = PDD.sop_codigo '
||' and RMA.tro_codigo(+) = PDD.tro_codigo '
||' and RMA.rol_codigo(+) = PDD.rol_codigo '
||' and RMA.mat_codigo(+) = PDD.mat_codigo_rma '
||' and RMA.mde_codigo(+) = PDD.mde_codigo_rma '
||' and RMA.rma_estado(+) = "V" '
||' and ETC.etc_codigo(+) = RMA.etc_codigo '    -- PRO_ESTADO_TECNICO
||' and HDAL.rma_house_dal(+) = RMA.rma_house_dal ' -- PRO_LOG_HOUSE_DAL
||' and HDAL.lho_profile(+) = "XP34" '          -- ID del sistema de tx iTX.
||' group by RMA.emp_codigo, '
||'   RMA.sop_codigo, '
||'   RMA.tro_codigo, '
||'   RMA.rol_codigo, '
||'   PDD.mat_codigo, '
||'   PDD.mde_codigo';

```

And this SQL generates the following MS-Excel worksheet:

Libro1 - Microsoft Excel

Inicio Insertar Diseño de página Fórmulas Datos Revisar Vista

Cortar Copiar Copiar formato Portapapeles Fuente Alineación Personalizada Formato condicional Dar formato como tabla Estilos de celda Insertar Eliminar Form Celdas

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	SEÑAL	ONAIR	MATERIAL	CAPITULO	SOPORTE	MAT_TIT_OR	CAP_TIT_OR	JUEGO_ROLL	HOUSEDAL	INGESTADO	TIMECODE	DURACION	PARTES
2	GHFL	01/08/2013 07:30	919891	13	16-FIB1-TX-1	SEXY URBAN	LOVE HURTS		XXX0000182	SI	01:00:00:00	00:27:57:21	1 de 1
3	GHFL	01/08/2013 22:58	919815	1	16-FIB1-TX-1	PLAYMATE V	THE DAHM TRIPLETS		XXX0000193	SI	01:00:00:00	00:59:39:12	1 de 1
4	GHFL	01/08/2013 09:30	922680	11	16-FIB1-TX-1	TOUR GIRLS	TOUR GIRLS II		XXX0000170	SI	01:00:00:00	00:25:52:01	1 de 1
5	GHFL	01/08/2013 15:30	922967	1	16-FIB1-TX-1	NAUGHTY A	NAUGHTY AMATEUR HOM		XXX0000159	SI	01:00:00:00	00:25:52:03	1 de 1
6	GHFL	01/08/2013 16:00	922967	2	16-FIB1-TX-1	NAUGHTY A	NAUGHTY AMATEUR HOM		XXX0000159	SI	01:00:00:00	00:25:46:24	1 de 1
7	GHFL	01/08/2013 16:30	922967	3	16-FIB1-TX-1	NAUGHTY A	NAUGHTY AMATEUR HOM		XXX0000160	SI	01:00:00:00	00:25:54:16	1 de 1
8	GHFL	01/08/2013 15:00	923153	7	16-FIB1-TX-1	CONSULTA E	CONSULTA EROTICA CON		XXX0000156	SI	01:00:00:00	00:21:13:10	1 de 1
9	GHFL	01/08/2013 08:00	923552	1	16-FIB1-TX-1	PLAYBOY'S V	PLAYBOY'S WORLD SOCCE		XXX0000172	SI	01:00:00:00	00:53:30:03	1 de 1
10	GHFL	02/08/2013 01:00	924796	1	16-FIB1-TX-1	LADY SCARF	LADY SCARFACE		XXX0000157	SI	01:00:00:00	01:17:01:08	1 de 1
11	GHFL	02/08/2013 04:30	925063	1	16-FIB1-TX-1	THE PLEASU	F CHOCOLATE		XXX0000142	SI	01:00:00:00	00:23:37:02	1 de 1