Loading Into BW Via A Database Link

ASAP FOR BW ACCELERATOR

BUSINESS INFORMATION WAREHOUSE

Document Version 2.0

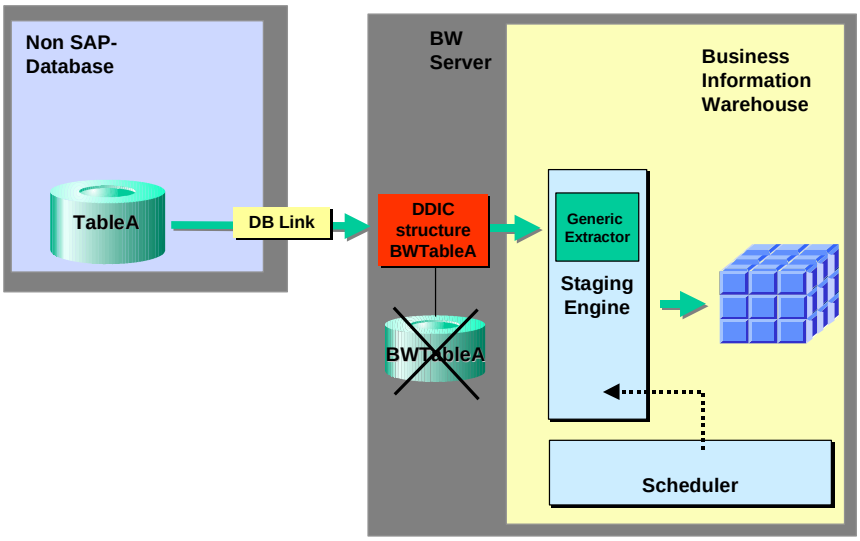
Table of Contents

1 Motivation

- Flexible way of loading Non-SAP database tables via a database link without using an external tool
- Usage in an SAP environment also possible

2 Concept

- Create table in data dictionary of BW with the same structure as the external table.
- Create and execute program ZCREATE_DB_LINK (see appendix):
  - Deletes corresponding database table on database level without deleting the DDIC structure.
  - Creates database link between this DDIC structure and the external table.
  - Creates public synonym/nickname for external table.
- Use generic extraction within BW to extract from the external table into the BW InfoSource.



3 Prerequisites

Oracle

- User SAPR3 must have the authority to maintain database links and public synonyms. With DB2 UDB the user SAPR3 must have SYSADM or DBADM authority. The database administrator must grant these authorizations to SAPR3.
- Maintenance of ORACLE-NETV2 must be guaranteed.

DB2 UDB for Unix and Windows

- User SAPR3 must have SYSADM or DBADM authority. The database administrator must grant these authorizations to SAPR3.
- If the external table is on a non-DB2 database then DB2 Relational Connect must be installed. For example if you want to access data in an Oracle 8.1 database, then an Oracle Client, Net8 and DB2 Relational Connect must be installed.
- The federated database must be enabled according to the DB2 Installation and Configuration Supplement Guide.
- In the current version of DB2 UDB V7.2 the following external data sources are supported:

Database Type	Description
DB2/UDB	IBM DB2 Universal Database
DataJoiner	IBM DB2 DataJoiner V2.1 and V2.1.1
DB2/6000	IBM DB2 for AIX
DB2/HPUX	IBM DB2 for HP-UX V1.2
DB2/NT	IBM DB2 for Windows NT
DB2/EEE	IBM DB2 Enterprise-Extended Edition
DB2/SUN	IBM DB2 for Solaris V1 and V1.2
DB2/2	IBM DB2 for OS/2
DB2/LINUX	IBM DB2 for Linux
DB2/PTX	IBM DB2 for NUMA-Q
DB2/SCO	IBM DB2 for SCO Unixware
DB2/400	IBM DB2 for AS/400
DB2/390	IBM DB2 for OS/390
DB2/MVS	IBM DB2 for MVS
DB2/VM	IBM DB2 for VM
DB2/VSE	IBM DB2 for VSE
SQL/DS	IBM SQL/DS
ORACLE	Oracle
MSSQLSERVER	Microsoft SQL Server
SYBASE	Sybase

Future releases will support additional data sources. Please check the DB2 Installation and Configuration Supplement Guide.

4 Restrictions

- Only supported for ORACLE and DB2 UDB for Unix and Windows databases.

5 Step by Step - Example

- Create table ZT100 with same structure as external table T100 (from system XYZ) in data dictionary of BW, save and activate ZT100
- Execute report ZCREATE_DB_LINK specifying system ID XYZ and table names ZT100 and T100 (report deletes corresponding database table in BW, creates database link with name of this DDIC structure to table in the other database and creates a synonym for this table)

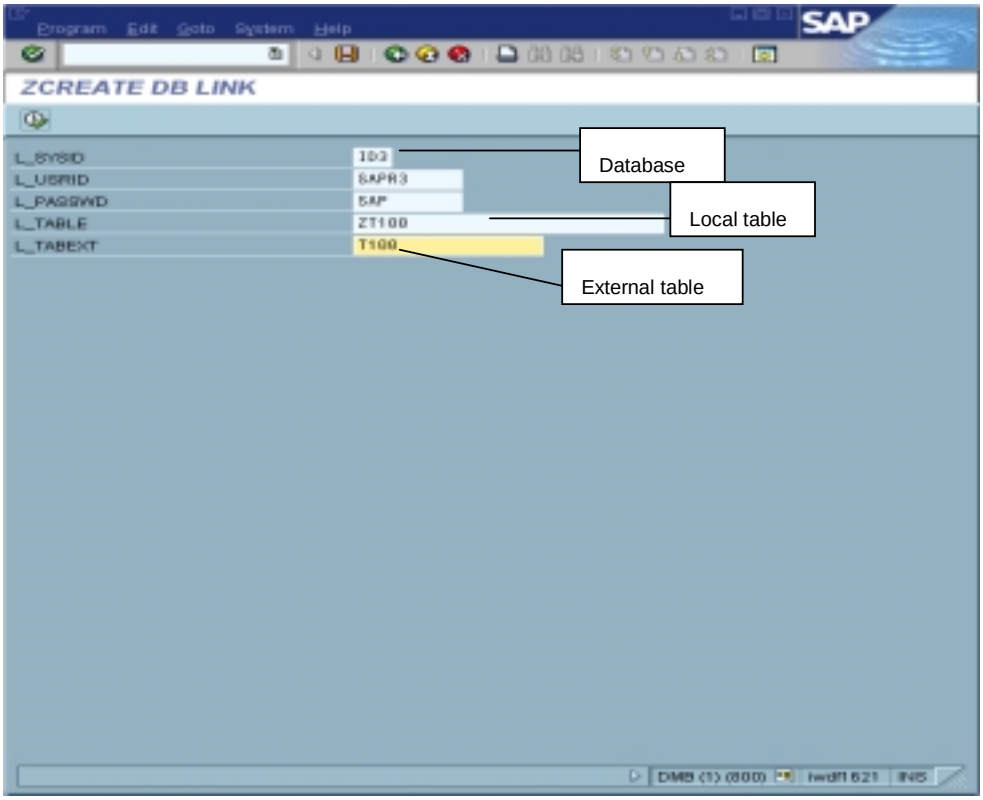


Fig. 1a: Database link from an Oracle system.

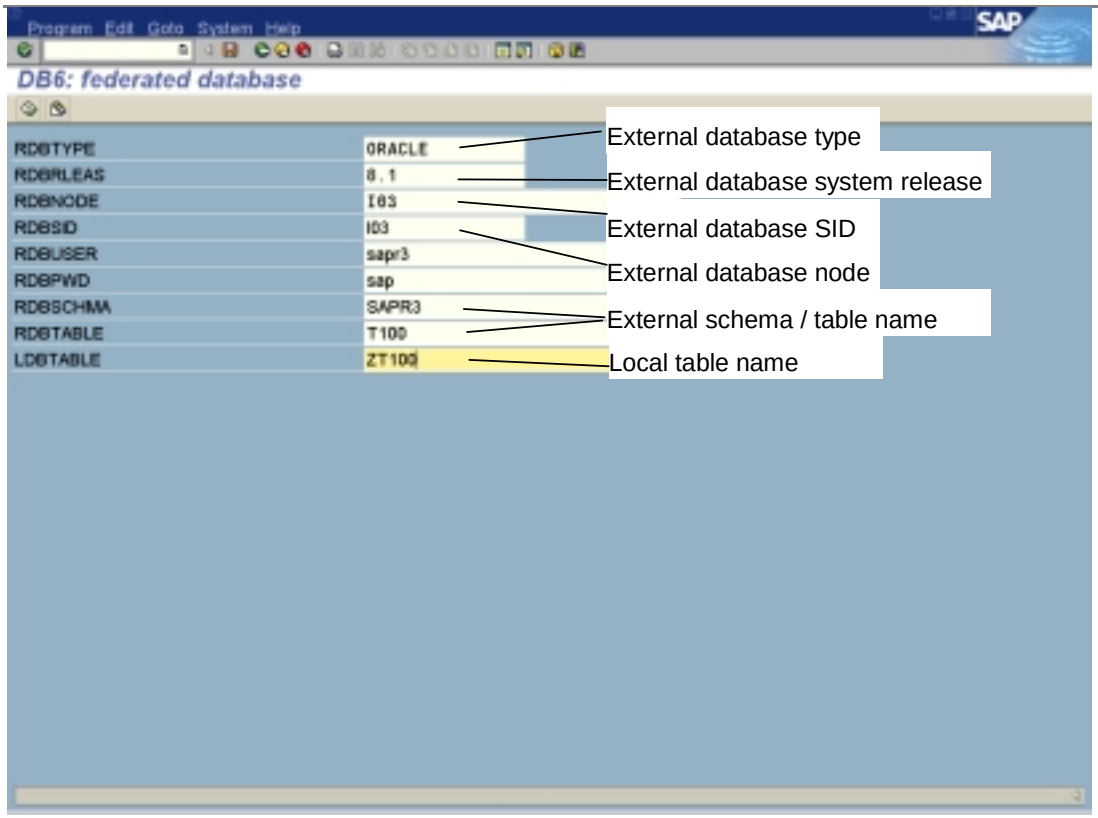


Fig. 1b: Database link from a DB2 UDB for Unix and Windows system.

- Use generic extraction within BW to extract from this table into BW InfoSource:
  - In transaction RSO2 create text data source T100_TEXT under application component SAP-R/3
    - Extract from DB view ZT100
    - Enter description and save
  - Create InfoObject T100ARBGB, data type CHAR, length 20, save and activate
  - Create an InfoObject T100_TEXT, data type CHAR, length 3, with compounding T100ARBGB, specify it as language-dependent and having long text, application component ZSC_SCENARIOS, save and activate
  - In the Administrator Workbench in the source systems tree execute ‚Replicate DataSources' for the BW system itself
    - In this scenario BW acts as a target and also as a source which has the link to the external table (DB Link)
  - Assign data source T100_TEXT to InfoSource T100_TEXT
  - Create InfoPackage for texts, load data from BW itself and check texts.

6 Appendix

6.1 Oracle Report ZCREATE_DB_LINK

```
REPORT zcreate_db_link.
*********************************************************************
***** ORACLE ONLY!!!! Database link for access of a remote table    **
***** use as datasource to load into Infocubes                      **
*********************************************************************
PARAMETERS: l_sysid(3)      TYPE c,
            l_usrid(10)     TYPE c DEFAULT 'SAPR3',
            l_passwd(10)    TYPE c DEFAULT 'SAP',
            l_table           like dd02l-tabname,
            l_tabext(18)    TYPE c.

DATA:       l_str(72)   TYPE c,
            l_sql       TYPE i,
            l_cnt       TYPE i,
            l_dblnk(6)  TYPE c,
            l_subrc     LIKE sy-subrc.
CONSTANTS:  c_tab_not_exist TYPE i VALUE 942,   "table does not exist
            c_synonym_exist TYPE i VALUE 955,   "synonym already exists
            c_dblink_exist  TYPE i VALUE 1000.  "db_link already exists

****check prerequisites for creating a database link and synonym *****
**** If table local exists on the database --> error            *****
CALL FUNCTION 'DB_EXISTS_TABLE'
  EXPORTING
    TABNAME       = l_table
  IMPORTING
    SUBRC         = l_subrc.
IF l_subrc = 0.
  select count(*) from (l_table) into l_cnt.
  if l_cnt <>  0.
    WRITE: / 'local table exists and is not empty'.
    EXIT.
  else.
    CONCATENATE 'DROP TABLE' l_table INTO l_str SEPARATED BY ' '.
    CALL FUNCTION 'RSDU_EXEC_SQL'
       EXPORTING
         i_stmt   = l_str
       IMPORTING
         e_sqlerr = l_sql
       EXCEPTIONS
         OTHERS   = 1.
    IF sy-subrc <> 0 AND l_sql <> c_tab_not_exist.
      WRITE: / 'ERROR during DROP TABLE OCCURED SQL-ERR: ', l_sql.
      EXIT.
    ENDIF.
  ENDIF.
```

```
ENDIF.
CLEAR l_subrc.

PERFORM create_dblink USING l_sysid
                            l_usrid
                            l_passwd
                   CHANGING l_dblnk
                            l_subrc.

IF l_subrc <> 0.
  WRITE: 'Create databaselink failed'.
  EXIT.
ENDIF.

PERFORM create_synonym USING l_dblnk
                             l_table
                             l_tabext
                             l_usrid
                    CHANGING l_subrc.

IF l_subrc <> 0.
  WRITE: 'Create synonym failed'.
  EXIT.
ENDIF.

*&---------------------------------------------------------------------*
*&      Form  create_dblink
*&---------------------------------------------------------------------*
*       text
*----------------------------------------------------------------------*
*      -->i_SYSID  logical oracle system
*      -->i_USRID  owner of the table
*      -->i_passwd password of owner
*      <--c_DBLNK  name of db_link
*      <--c_SUBRC  returncode of formroutine
*----------------------------------------------------------------------*
FORM create_dblink    USING i_sysid TYPE c
                            i_usrid TYPE c
                            i_passwd TYPE c
                   CHANGING c_dblnk TYPE c
                            c_subrc LIKE sy-subrc.
  DATA:       l_str(120) TYPE c,
              l_sql      TYPE i,
              l_sysid(5) TYPE c.
  CONSTANTS: c_dblink_exists TYPE i VALUE 2011.
  CONCATENATE 'TO_' i_sysid INTO c_dblnk.
  CONCATENATE '''' i_sysid '''' INTO l_sysid.
  CONCATENATE 'create database link ' c_dblnk ' connect to ' i_usrid
              'identified  by ' i_passwd ' using ' l_sysid INTO l_str
              SEPARATED BY ' '.

  CALL FUNCTION 'RSDU_EXEC_SQL'
       EXPORTING
            i_stmt    = l_str
       IMPORTING
            e_sqlerr  = l_sql
       EXCEPTIONS
            sql_error = 1
            OTHERS    = 2.
  IF sy-subrc <> 0 AND l_sql <> c_dblink_exists.
    WRITE: / 'SQL-ERROR ', l_sql,' creating database link '.
    c_subrc = sy-subrc.
  ENDIF.
ENDFORM.                                " create_dblink

*&---------------------------------------------------------------------*
*&      Form  create_synonym
*&---------------------------------------------------------------------*
```

```
*       text
*---------------------------------------------------------------------*
*      -->i_DBLNK  name of database link
*      -->i_TABLE  table name
*      -->i_USRID  owner of  the remote table
*      <--c_SUBRC  returncode
*---------------------------------------------------------------------*
FORM create_synonym USING    i_dblnk TYPE c
                             i_table TYPE c
                             i_tabext TYPE c
                             i_usrid TYPE c
                    CHANGING c_subrc LIKE sy-subrc.

  DATA: l_str(120) TYPE c,
        l_tab(30)  TYPE c,
        l_sql      TYPE i.
  CONCATENATE i_usrid '.' i_tabext '@' i_dblnk INTO l_tab.
  CONCATENATE 'create synonym' i_table ' for ' l_tab
              INTO l_str SEPARATED BY ' '.
  CALL FUNCTION 'RSDU_EXEC_SQL'
       EXPORTING
            i_stmt    = l_str
       IMPORTING
            e_sqlerr  = l_sql
       EXCEPTIONS
            sql_error = 1
            OTHERS    = 2.
  IF sy-subrc <> 0  AND l_sql <> c_synonym_exist.
    WRITE: / 'SQL-ERROR ', l_sql,' creating public synonym '.
    c_subrc = sy-subrc.
  ENDIF.
ENDFORM.                               " create_synonym
```

6.2 DB2 UDB for Unix and Windows Report ZCREATE_DB_LINK

```
************************************************************************
***** DB2 UDB for Unix and Windows ONLY!!!!                          **
***** Database link for access of a remote table                     **
***** use as datasource to load into Infocubes                       **
************************************************************************
REPORT ZCREATE_DB_LINK .
*****
* Purpose: Create a db link between DB6 and another database, e.g.
*          ORACLE, MS SQL Server, Sybase, DB2, DB4, DB6 etc.
* Prereq:  1. user <schema> must have dbadm authority
*             "db2 grant dbadm on database to user sapr3"
*          2. db2 federated system must be configured
*****
* If additional data sources are supported by DB6 you must add them in
* the section
* "->add additional database wrappers | types here"
* "->..."
* in this report.
*****

  type-pools db6cc.
  include rsdb6pmi.

  constants:
    quote(1)      type c value '''',
    quote2(1)     type c value '"'.

* available wrappers
  constants:
    oraWrapper(20)   type c value 'NET8',       "#EC NOTEXT
    db2Wrapper(20)   type c value 'DRDA',       "#EC NOTEXT
*   only supported with DB2 UDB V7.2
```

```
    sybWrapper(20)   type c value 'CTLIB',      "#EC NOTEXT
    mssWrapperNT(20) type c value 'DJXMSSQL3',  "#EC NOTEXT
    mssWrapperUX(20) type c value 'MSSQLODBC3'. "#EC NOTEXT
*   ->add additional database wrappers here
*   ->...

  parameters:
    rDBtype(15)    type c,            "remote DB type e.g. DB2/UDB
    rDBrleas(15)   type c,            "remote DB release e.g. 6.1 or 8i
    rDBnode(30)   type c,            “remote DB node
    rDBsid(15)    type c lower case,  "remote DB SID
    rDBuser(30)   type c lower case,  "remote DB user e.g. sapr3
    rDBpwd(30)    type c lower case,  "remote DB user password
    rDBschma(30)  type c lower case,  "remote DB schema
    rDBtable(30)  type c lower case,  "remote DB table
    lDBtable(30)  type c lower case.  "local DB table

  data:
    subrc         type sysubrc,
    schema(32)    type c, "local schema name
    wrapper(32)   type c, "local wrapper name
    srvname(32)   type c, "local server name
    nickname(255) type c. "local nickname (synonym)

* get local schema name
  perform get_schema in program sdb1fdb6
       changing schema.

* check existence of tbs userspace1
  perform check_userspace1 changing subrc.
  if subrc <> 0.
    exit.
  endif.

* drop local table on db if empty
  perform drop_table using schema
                           lDBtable
                  changing subrc.
  if subrc <> 0.
    exit.
  endif.

* set the library
  perform create_wrapper using rDBtype
                       changing wrapper
                                subrc.
  if subrc <> 0.
    exit.
  endif.

* set the db you want to query
  perform create_server using rDBtype
                              rDBnode
                              rDBsid
                              rDBrleas
                              rDBuser
                              rDBpwd
                              wrapper
                     changing srvname
                              subrc.
  if subrc <> 0.
    exit.
  endif.

* create the user mapping between local and remote user
  perform create_usermapping using srvName
                                   space
                                   rDBuser
```

```
                                    rDBpwd.

* how should the remote table named in the local system
  perform create_nickname using schema
                                lDBtable
                                srvName
                                rDBschma
                                rDBtable
                         changing nickname.

* commit all
  commit work.

*---------------------------------------------------------*
* drop table if existing and empty
form drop_table using schema   type c
                      lDBTable type c
              changing subrc    type sysubrc.
  data:
    cnt               type i,
    tabname           type dd02l-tabname,
    localtable(255)   type c,
    stmt(255)         type c.

  tabname = lDBTable. "without any qoutes!
  CALL FUNCTION 'DB_EXISTS_TABLE'
    EXPORTING
      TABNAME       = tabname
   IMPORTING
      SUBRC         = subrc.
  if subrc = 0. "table exists
    localtable = lDBTable.
    select count(*) from (localtable) into cnt.
    if sy-subrc > 4.
      write: / 'Error reading table ', localtable.
      subrc = 8.
    endif.
    if cnt <> 0.
      write: / 'Error: Local table ', localtable,
               'exists and is not empty'.
      subrc = 4.
    else.
      concatenate schema '.' quote2 lDBTable quote2 into localtable.
      concatenate 'DROP TABLE' localtable
                  into stmt separated by space.
      perform exec_sql using stmt.
    endif.
  else.
    subrc = 0.
  endif.
endform.


*---------------------------------------------------------*
* tbs userspace1 is needed for the following operations
form check_userspace1 changing subrc type sysubrc.
  data:
    count type i.

  EXEC SQL.
    SELECT COUNT( * ) INTO :count
           FROM SYSCAT.TABLESPACES
           WHERE TBSPACE LIKE 'USERSPACE1'
  ENDEXEC.
  if sy-subrc <> 0 or count <> 1.
    write: / 'Please create a tablespace named "USERSPACE1" first.'.
    subrc = 1.
    exit.
```

```
  endif.
  subrc = 0.
endform.


*---------------------------------------------------------*
* Create a wrapper
form create_wrapper using    rDBType  type c
                     changing wrapper type c
                              subrc   type sysubrc.
  data:
    dbopsys          type syopsys,
    wrapper_lib(100) type c,
    stmt(255)        type c.

  subrc = 0.

* Oracle's Net8
  if rDBType = 'ORACLE'.
    wrapper = oraWrapper.
    wrapper_lib = wrapper.
* DB2 family data sources
  elseif rDBType = 'DB2/UDB' or rDBType = 'DATAJOINER' or
         rDBType = 'DB2/6000' or rDBType = 'DB2/HPUX' or
         rDBType = 'DB2/NT' or rDBType = 'DB2/EEE' or
         rDBType = 'DB2/2' or rDBType = 'DB2/SUN' or
         rDBType = 'DB2/LINUX' or rDBType = 'DB2/PTX' or
         rDBType = 'DB2/SCO' or rDBType = 'DB2/400' or
         rDBType = 'DB2/390' or rDBType = 'DB2/MVS' or
         rDBType = 'DB2/VM' or rDBType = 'DB2/VSE' or
         rDBType = 'SQL/DS'.
    wrapper = db2Wrapper.
    wrapper_lib = wrapper.
* MSSQL Server from DB2/NT: DJXMSSQL3; on DB2/AIX MSSQLODBC3
  elseif rDBType = 'MSSQLSERVER'.
    CALL FUNCTION 'GET_DB_SERVER_OPSYSTEM_DB6'
     IMPORTING
       DBOPSYS              = dbopsys
     EXCEPTIONS
       ERROR_OCCURED        = 1
       OTHERS               = 99.
    case sy-subrc.
      when 0.
      when 1 or 99.
        message i002(DB6PM) with 'GET_DB_SERVER_OPSYSTEM_DB6'
                sy-subrc.
    endcase.
    if dbopsys = opsys_windows_NT.
      wrapper = mssWrapperNT.
    else.
      wrapper = mssWrapperUX.
    endif.
    wrapper_lib = wrapper.
    write: /
     'Only DB2 V7.2 and above supports Microsoft SQL Server'.
* Sybase
  elseif rDBType = 'SYBASE'.
    wrapper = sybWrapper.
    wrapper_lib = wrapper.
    write: /
     'Only DB2 V7.2 and above supports Sybase'.
* ->add additional database types here
* ->...

  else.
    write: / 'External database type ' , rDBType, ' is not supported'.
    subrc = 4.
  endif.
```

```
  concatenate 'CREATE WRAPPER' wrapper_lib
              into stmt separated by space.

  perform exec_sql using stmt.
endform.

*---------------------------------------------------------*
* Create a server
form create_server using rDBType  type c
                         rDBNode  type c
                         rDBSid   type c
                         rDBRleas type c
                         rDBUser  type c
                         rDBPwd   type c
                         wrapper  type c
                changing srvName  type c
                         subrc    type sysubrc.
  data:
    stmt(255)    type c,
    rNode(32)    type c,
    rSid(32)     type c,
    options(128) type c.

  subrc = 0.
  srvName = rDBNode.
  concatenate quote rDBNode quote into rNode.
  concatenate quote rDBSid quote into rSid.

  concatenate 'OPTIONS ( NODE ' rNode
              into options separated by space.

* ->add additional database types here in if stmt
* ->...

* option DBNAME required
  if rDBType = 'DB2/UDB' or rDBType = 'DATAJOINER' or
     rDBType = 'DB2/6000' or rDBType = 'DB2/HPUX' or
     rDBType = 'DB2/NT' or rDBType = 'DB2/EEE' or
     rDBType = 'DB2/2' or rDBType = 'DB2/SUN' or
     rDBType = 'DB2/LINUX' or rDBType = 'DB2/PTX' or
     rDBType = 'DB2/SCO' or rDBType = 'DB2/400' or
     rDBType = 'DB2/390' or rDBType = 'DB2/MVS' or
     rDBType = 'DB2/VM' or rDBType = 'DB2/VSE' or
     rDBType = 'SQL/DS' or
     rDBType = 'MSSQLSERVER' or
     rDBType = 'SYBASE'.
    concatenate options ', DBNAME ' rSid
              into options separated by space.
* no option DBNAME required for ORACLE
  elseif rDBType = 'ORACLE'.
*
  else.
    write: / 'External database type ' , rDBType, ' is not supported'.
    subrc = 4.
  endif.
* add other common options here
  concatenate options ', FOLD_ID ''N'' , FOLD_PW ''N'' )'
              into options separated by space.

  concatenate 'CREATE SERVER' srvName
                'TYPE' rDBType
                'VERSION' rDBRleas
                'WRAPPER' wrapper
                'AUTHORIZATION'  rDBUser
                'PASSWORD' rDBPwd
                options
                into stmt separated by space.
```

```
  perform exec_sql using stmt.
endform.

*---------------------------------------------------------*
* Create a user mapping
form create_usermapping using lsrvName type c
                              lusrName type c
                              rDBUser  type c
                              rDBPwd   type c.
  data:
    stmt(255)   type c,
    rUser(32)   type c,
    rPwd(32)    type c.

  concatenate quote rDBUser quote into rUser.
  concatenate quote rDBPwd quote into rPwd.
  if lusrName = space or
     lusrName is initial.
*   create the user mapping between the current user and remote user
*          -> needed to execute the command 'create nickname'
    concatenate quote rDBUser quote into rUser.
    concatenate quote rDBPwd quote into rPwd.
    concatenate 'CREATE USER MAPPING FOR USER'
                'SERVER' lsrvName
                'OPTIONS ( REMOTE_AUTHID' rUser
                         ', REMOTE_PASSWORD' rPwd ')'
                into stmt separated by space.
  else.
    concatenate 'CREATE USER MAPPING FOR' lusrName
                'SERVER' lsrvName
                'OPTIONS ( REMOTE_AUTHID' rUser
                         ', REMOTE_PASSWORD' rPwd ')'
                into stmt separated by space.
  endif.

  perform exec_sql using stmt.
endform.

*---------------------------------------------------------*
* Create a nickname
form create_nickname using schema   type c
                           lDBTable type c
                           srvName  type c
                           rDBSchma type c
                           rDBTable type c
                  changing nickname type c.
  data:
    stmt(255)       type c,
    objectName(255) type c.

  concatenate srvName '.'
              quote2 rDBSchma quote2 '.'
              quote2 rDBTable quote2
               into objectName.
  concatenate schema '.' quote2 lDBTable quote2
               into nickname.
  concatenate 'CREATE NICKNAME' nickname
              'FOR' objectName
              into stmt separated by space.

  perform exec_sql using stmt.
endform.

*---------------------------------------------------------*
* execute sql statement
form exec_sql using sql_stmt type c.
  data:
```

```
    errsql       TYPE i,
    errtxt(72)   TYPE c,
    subrc        LIKE sy-subrc.

  CALL FUNCTION 'RSDU_EXEC_SQL'
   EXPORTING
*     I_T_STMT                          =
     I_STMT                            = sql_stmt
*     I_LOWER_TRANS                      = RS_C_FALSE
*     I_ERROR_DOWNLOAD                   = RS_C_TRUE
   IMPORTING
     E_SQLERR                          = errsql
*     E_SQLERRTXT                        =
*     E_STMTSIZE                         =
   EXCEPTIONS
     SQL_ERROR                         = 1
     STATEMENT_TOO_COMPLEX             = 2
     NO_STATEMENT                      = 3
     INHERITED_ERROR                   = 4
     OBJ_NOT_FOUND_OR_DUPL_RECORD      = 5
     OTHERS                            = 6.
  subrc = sy-subrc.
  write: / sql_stmt.
  write: / 'SY-SUBRC ', subrc.
  if subrc = 1.
    write: 'SQL ERROR'.
  elseif subrc = 2.
    write: 'STATEMENT TOO COMPLEX'.
  elseif subrc = 3.
    write: 'NO STATEMENT'.
  elseif subrc = 4.
    write: 'INHERITED ERROR'.
  elseif subrc = 5.
    write: 'OBJ NOT FOUND OR DUPL RECORD'.
  endif.
  write: / 'SQL-ERROR ', errsql.
  if errsql = -601.
    write: 'The name of the object is identical to an existing one'.
  endif.
  write: / '************************************************************'.
endform.
```