4D for OCI and PL/SQL

By Noreddine Margoum, QA Engineer, 4D SA.
TN 06-09

Introduction

This Technical Note illustrates the use of routines written in PL/SQL within an application that uses 4D for OCI.

PL/SQL

The language PL/SQL (Procedural Language/Structured Query Language) is an extension of the SQL language. It integrates the SQL language, to which it adds the power of a procedural language (control structures, object oriented programming, etc.).

The purpose of this Technical Note is not to describe the PL/SQL language, so we highly recommend you conduct your own research on the subject. However, the examples that are described here are very simple and do not require advanced knowledge of PL/SQL. The main purpose of these examples is to illustrate how you can use PL/SQL with 4D for OCI.

In the examples provided with this Technical Note, we illustrate the creation of a function and a procedure and how to use them with 4D for OCI.

Factorial function:

```
-- The concept of recursion is used to calculate the factorial
-- Reminder: factorial N = N ! = 1*2x*3…*N
-- Reminder: factorial 0 = 0 ! = 1 (by definition)

FUNCTION Factorial (facto IN NUMBER)
RETURN NUMBER IS
BEGIN
-- stop condition for the recursive function: parameter is 0
IF facto = 0 THEN
RETURN 1;
        ELSE
-- recursive call:  factorial N is N*Factorial N-1

RETURN facto*Factorial(facto-1);
END IF;
END;
```

Comments on the function 'Factorial' :

In PL/SQL comments are noted by two leading dashes.
The header of the function indicates the name of the function, the parameter passed and its type, and the returned parameter and its type. In this case, the parameters passed are both of type **NUMBER**, which matches roughly 4D's real type. The keyword **IN** indicates that the parameter is an input parameter. Other possible parameter keywords are **OUT**, for an output parameter, and **IN OUT**, for an input/output parameter.

The 'body' of the factorial function is comprised between the keywords **BEGIN** and **END**.

Please also note each PL/SQL instruction has to end with a semicolon (;).

Procedure READINFO:

```
PROCEDURE READINFO (vUser OUT VARCHAR2,
vDate OUT VARCHAR2, vTime OUT VARCHAR2) IS
BEGIN
-- returns the name of the current user
SELECT USER INTO vUser FROM DUAL;
-- returns the date of the Oracle server
SELECT TO_CHAR(SYSDATE,'DD-MM-YYYY') INTO vDate FROM DUAL;
-- returns the time of the Oracle server
SELECT TO_CHAR(SYSDATE,'HH24:MI:SS') INTO vTime FROM DUAL;
END;
```

Comments on the READINFO procedure

The procedure **READINFO** returns the current user name, the date and the time of the server as strings that are placed into the passed parameters.

The Oracle SQL function **SYSDATE**, returns the date and time of the system.

We then convert the date and time, using the appropriate format. The conversion is performed by the function **TO_CHAR**, which is an Oracle function.

Implementing PL/SQL in 4D for OCI

Implementing the 'Factorial' PL/SQL function

Project method 'Ex_Factorial Function':

```
`Method   'Ex_Function_Factorial'
   `executes a 'factorial'  PL/SQL function  which returns  a  factorial  for  a number

C_LONGINT(vl_Envhp;vl_Errhp;vl_Svchp)
```

```
C_LONGINT(vl_Stmthp_Create;vl_Stmthp_Exec;vl_Stmthp_Del;vl_Bind)
C_TEXT(vt_UserName;vt_Password;vt_HostName;vt_SQL;vt_ErrorMessage)
C_LONGINT(vl_Status;vl_ErrorCode)
C_LONGINT(vl_Input)
C_REAL(vr_Output)
C_POINTER(vp_Null_Ind1;vp_Null_Ind2;vp_Null_Ind3)

    `connection parameters (to be modified)
vt_UserName:="Scott" `User name
vt_Password:="Tiger" `password
vt_HostName:="ORA8QA" `connection string

vl_Input:=0 `number whose factorial will be returned
vr_Output:=0 `result
 `Allocation of the environment handle
vl_Status:=OCIEnvCreate (vl_Envhp;OCI_DEFAULT )
If (vl_Status=OCI_SUCCESS )
    `Allocating an error handle
  vl_Status:=OCIHandleAlloc (vl_Envhp;vl_Errhp;OCI_HTYPE_ERROR_)
  If (vl_Status=OCI_SUCCESS )
     `  Ouverture de la connexion et de la session
   vl_Status:=OCILogon (vl_Envhp;vl_Errhp;vl_Svchp;vt_UserName;vt_Password;vt_HostName)
   If (vl_Status=OCI_SUCCESS )
      `Allocation of the statement handle.
      `This handle will allow us to create the function PL/SQL 'Factorial' stored on Oracle
     vl_Status:=OCIHandleAlloc (vl_Envhp;vl_Stmthp_Create;OCI_HTYPE_STMT )
     If (vl_Status=OCI_SUCCESS )

        `Defining the functon 'Factorial' in PL/SQL
       vt_SQL:=""
       vt_SQL:=vt_SQL+"CREATE OR REPLACE FUNCTION Factorial (facto IN NUMBER)"+Char(LF ASCII code )
       vt_SQL:=vt_SQL+"RETURN NUMBER IS"+Char(LF ASCII code )
       vt_SQL:=vt_SQL+"BEGIN"+Char(LF ASCII code )
       vt_SQL:=vt_SQL+"IF facto = 0"+Char(LF ASCII code )
       vt_SQL:=vt_SQL+"THEN RETURN 1;"+Char(LF ASCII code )
       vt_SQL:=vt_SQL+"ELSE"+Char(LF ASCII code )
       vt_SQL:=vt_SQL+"RETURN facto*Factorial(facto-1);"+Char(LF ASCII code )
       vt_SQL:=vt_SQL+"END IF;"+Char(LF ASCII code )
       vt_SQL:=vt_SQL+"END;"

       vl_Status:=OCIStmtPrepare (vl_Stmthp_Create;vl_Errhp;vt_SQL;OCI_DEFAULT )
       vl_Status:=OCIStmtExecute (vl_Svchp;vl_Stmthp_Create;vl_errhp;1;0;0;0;OCI_DEFAULT )
          `the function Factorial is now stored on the Oracle DB. We can now test it...

        `Allocating the statement handle.
        ` This handle will allow us to use the PL/SQL function 'Factorial' stored on Oracle
       vl_Status:=OCIHandleAlloc (vl_Envhp;vl_Stmthp_Exec;OCI_HTYPE_STMT )
       If (vl_Status=OCI_SUCCESS )
         vt_SQL:=""
         vt_SQL:=vt_SQL+"BEGIN"+Char(LF ASCII code )
         vt_SQL:=vt_SQL+":DataOutput:=Factorial(:DataInput);"+Char(LF ASCII code )
         vt_SQL:=vt_SQL+"END;"
```

```
        vl_Input:=Num(Request("Calculate  factorial  for  :  "))
        vl_Status:=OCIStmtPrepare    (vl_Stmthp_Exec;vl_Errhp;vt_SQL;OCI_DEFAULT  )

          `  Bind  by name to call 'Factorial'
        vl_Status:=OCIBindByName    (vl_Stmthp_Exec;vl_Bind;vl_Errhp;":DataOutput";-
>vr_Output;SQLT_FLT ;vp_Null_Ind1;vp_Null_Ind2;vp_Null_Ind3;OCI_DEFAULT  ;BIND_OUT )
        vl_Status:=OCIBindByName    (vl_Stmthp_Exec;vl_Bind;vl_Errhp;":DataInput";->vl_Input;SQLT_INT
;vp_Null_Ind1;vp_Null_Ind2;vp_Null_Ind3;OCI_DEFAULT  ;BIND_IN )
        vl_Status:=OCIStmtExecute    (vl_Svchp;vl_Stmthp_Exec;vl_Errhp;1;0;0;0;OCI_DEFAULT  )
        If (vl_Status#OCI_SUCCESS )
          vl_Status:=OCIErrorGet    (vl_Errhp;1;vl_ErrorCode;vt_ErrorMessage)
          ALERT("Erreur  Oracle "+String(vl_ErrorCode)+"    :  "+vt_ErrorMessage)
        End  if
        ALERT(String(vl_Input)+"!    =  "+String(vr_Output))
          `Allocating  the  statement  handle.
        `This handle will allow us to delete the PL/SQL function  'Factorial'  stored on Oracle
        vl_Status:=OCIHandleAlloc    (vl_Envhp;vl_Stmthp_Del;OCI_HTYPE_STMT  )
        If  (vl_Status=OCI_SUCCESS )
          vt_SQL:=""
          vt_SQL:=vt_SQL+"DROP  FUNCTION FACTORial"
          vl_Status:=OCIStmtPrepare    (vl_Stmthp_Del;vl_Errhp;vt_SQL;OCI_DEFAULT  )
          vl_Status:=OCIStmtExecute    (vl_Svchp;vl_Stmthp_Del;vl_errhp;1;0;0;0;OCI_DEFAULT  )
              `freeing statement  handle used to delete the Factorial function
          vl_Status:=OCIHandleFree    (vl_Stmthp_Del)
        End  if
            `freeing statement  handle used to test  the Factorial function
        vl_Status:=OCIHandleFree    (vl_Stmthp_Exec)
      End  if
          `freeing statement  handle used to create the Factorial function
      vl_Status:=OCIHandleFree    (vl_Stmthp_Create)
    End  if
    vl_Status:=OCILogoff    (vl_Svchp;vl_Errhp)
  End  if
  vl_Status:=OCIHandleFree    (vl_Errhp)
End  if
vl_Status:=OCIHandleFree    (vl_Envhp)
vl_Status:=OCICleanUp
End  if
```

Comments on the project method 'Ex Factorial Function':

The main algorithm of the project method '*Ex_Factorial_Function*' can be broken down into three main steps:

• Define and create the PL/SQL function 'Factorial': The instruction CREATE OR REPLACE is placed before the definition so that the function will always be stored on the Oracle server. Please note that we use a line feed character to end each line of the PL/SQL function.
• Dynamic call of the function: To test the function, we call it with the input and output parameter within a PL/SQL BEGIN..END block. This block is called the 'anonymous' block since it does not reside in the Oracle database, and is compiled and executed on the fly.

• Delete the function: here we execute a simple static SQL request that deletes the function stored on the Oracle server.

The most important part of this method is the binding of 4D variables to the parameters of the PL/SQL function using the command OCIBindByName. What is interesting to note here is the value (BIND_IN or BIND_OUT) of the last parameter for this command to indicate, respectively, if the data is supplied by the 4D variable (input parameter) or retrieved by the 4D variable 4D (output parameter).

Implementing the PL/SQL procedure 'READINFO'

Project Method 'Ex_Procedure_READINFO':

```
    `project method   Ex_Procedure_READINFO
    `implements procedure 'READINFO'.
    `executes a PL/SQL procedure that returns    the user currently connected
    `to the Oracle DB as well as the date and time of the Oracle server machine

C_LONGINT(vl_Envhp;vl_Errhp;vl_Svchp)
C_LONGINT(vl_Stmthp_Create;vl_Stmthp_Exec;vl_Stmthp_Del;vl_Bind)
C_TEXT(vt_UserName;vt_Password;vt_HostName;vt_SQL;vt_ErrorMessage)
C_LONGINT(vl_Status;vl_ErrorCode)
C_LONGINT(vl_Input)
C_LONGINT(vl_Output)
C_POINTER(vp_Null_Ind1;vp_Null_Ind2;vp_Null_Ind3)
C_TEXT(vt_MyUser;vt_MyDate;vt_MyTime)

    `connection parameters (to  be modified)
vt_UserName:="Scott"  `User   name
vt_Password:="Tiger"  `password
vt_HostName:="ORA8QA"  `connection   string

    `data to retrievethrough call to procedure PL/SQL 'READINFO'
vt_MyUser:=""
vt_MyDate:=""
vt_MyTime:=""

    `allocating environment handle
vl_Status:=OCIEnvCreate (vl_Envhp;OCI_DEFAULT )
If (vl_Status=OCI_SUCCESS )
      `allocating error handle
   vl_Status:=OCIHandleAlloc (vl_Envhp;vl_Errhp;OCI_HTYPE_ERROR )
   If (vl_Status=OCI_SUCCESS )
        `   opening connection  and session
      vl_Status:=OCILogon (vl_Envhp;vl_Errhp;vl_Svchp;vt_UserName;vt_Password;vt_HostName)
      If (vl_Status=OCI_SUCCESS )

           `allocating statement  handle. This handle is used to create the procedure PL/SQL 'READINFO' on
Oracle
        vl_Status:=OCIHandleAlloc (vl_Envhp;vl_Stmthp_Create;OCI_HTYPE_STMT )
```

```
If (vl_Status=OCI_SUCCESS )

    `defining 'READINFO' in PL/SQL
  vt_SQL:=""
  vt_SQL:=vt_SQL+"CREATE OR REPLACE PROCEDURE READINFO (vUser OUT VARCHAR2,"+Char(LF ASCII code )
  vt_SQL:=vt_SQL+"vDate OUT VARCHAR2, vTime OUT VARCHAR2) IS"+Char(LF ASCII code )
  vt_SQL:=vt_SQL+"BEGIN"+Char(LF ASCII code )
  vt_SQL:=vt_SQL+"SELECT USER INTO vUser FROM DUAL;"+Char(LF ASCII code )
  vt_SQL:=vt_SQL+"SELECT TO_CHAR(SYSDATE,'DD-MM-YYYY') INTO vDate FROM DUAL;"+Char(LF ASCII code )
  vt_SQL:=vt_SQL+"SELECT TO_CHAR(SYSDATE,'HH24:MI:SS') INTO vTime FROM DUAL;"+Char(LF ASCII code )
  vt_SQL:=vt_SQL+"END;"

  vl_Status:=OCIStmtPrepare (vl_Stmthp_Create;vl_Errhp;vt_SQL;OCI_DEFAULT )
  vl_Status:=OCIStmtExecute (vl_Svchp;vl_Stmthp_Create;vl_errhp;1;0;0;0;OCI_DEFAULT )
    `the function READINFO is now stored on Oracle. we can now test it.
  vl_Status:=OCIHandleAlloc (vl_Envhp;vl_Stmthp_Exec;OCI_HTYPE_STMT )
  If (vl_Status=OCI_SUCCESS )
    vt_SQL:=""
    vt_SQL:=vt_SQL+"BEGIN"+Char(LF ASCII code )
    vt_SQL:=vt_SQL+"READINFO(:MyUser,:MyDate,:MyTime);"+Char(LF ASCII code )
    vt_SQL:=vt_SQL+"END;"


    vl_Status:=OCIStmtPrepare (vl_Stmthp_Exec;vl_Errhp;vt_SQL;OCI_DEFAULT )

     `Bind by name to call 'READINFO'
    vl_Status:=OCIBindByName(vl_Stmthp_Exec;vl_Bind;vl_Errhp;":MyUser";->vt_MyUser;SQLT_STR
;vp_Null_Ind1;vp_Null_Ind2;vp_Null_Ind3;OCI_DEFAULT ;BIND_OUT )
    vl_Status:=OCIBindByName(vl_Stmthp_Exec;vl_Bind;vl_Errhp;":MyDate";->vt_MyDate;SQLT_STR
;vp_Null_Ind1;vp_Null_Ind2;vp_Null_Ind3;OCI_DEFAULT ;BIND_OUT )
    vl_Status:=OCIBindByName(vl_Stmthp_Exec;vl_Bind;vl_Errhp;":MyTime";->vt_MyTime;SQLT_STR
;vp_Null_Ind1;vp_Null_Ind2;vp_Null_Ind3;OCI_DEFAULT ;BIND_OUT )

    vl_Status:=OCIStmtExecute (vl_Svchp;vl_Stmthp_Exec;vl_Errhp;1;0;0;0;OCI_DEFAULT )

    If (vl_Status#OCI_SUCCESS )
      vl_Status:=OCIErrorGet (vl_Errhp;1;vl_ErrorCode;vt_ErrorMessage)
      ALERT("Erreur Oracle "+String(vl_ErrorCode)+" : "+vt_ErrorMessage)
    End if

    MyData:=""
    MyData:=MyData+"Current User : "+vt_MyUser+Char(Carriage return )
    MyData:=MyData+"Server Date : "+vt_MyDate+Char(Carriage return )
    MyData:=MyData+"Server Time : "+vt_MyTime+Char(Carriage return )
    ALERT(MyData)
     `Statement handle. This handle is used to delete the PL/SQL 'READINFO' procedure stored on
Oracle
    vl_Status:=OCIHandleAlloc (vl_Envhp;vl_Stmthp_Del;OCI_HTYPE_STMT )
    If (vl_Status=OCI_SUCCESS )
      vt_SQL:=""
```

```
            vt_SQL:=vt_SQL+"DROP PROCEDURE READINFO"
            vl_Status:=OCIStmtPrepare (vl_Stmthp_Del;vl_Errhp;vt_SQL;OCI_DEFAULT )
            vl_Status:=OCIStmtExecute (vl_Svchp;vl_Stmthp_Del;vl_errhp;1;0;0;0;OCI_DEFAULT )
               `freeing the statement handle
            vl_Status:=OCIHandleFree (vl_Stmthp_Del)
          End if
             `freeing statement handled used to test 'READINFO'
          vl_Status:=OCIHandleFree (vl_Stmthp_Exec)
        End if
           `freeing statement handled used to create 'READINFO'
        vl_Status:=OCIHandleFree (vl_Stmthp_Create)
      End if
      vl_Status:=OCILogoff (vl_Svchp;vl_Errhp)
    End if
    vl_Status:=OCIHandleFree (vl_Errhp)
  End if
  vl_Status:=OCIHandleFree (vl_Envhp)
  vl_Status:=OCICleanUp
End if
```

Comments on the project method '*Ex_Procedure_READINFO*' :

The main algorithm of the project method '*Ex_Procedure_READINFO*' is identical to that of '*Ex_Factorial_Function*' described previously. The main purpose of this method is to demonstrate the use of a PL/SQL procedure after seeing the use of a function.

Here, all 4D variables are passed as output parameters (BIND_OUT), because we only retrieve data (user name, date and time) form the Oracle database to 4D. There are of course better ways to achieve the same results of this procedure (for example using a simple SQL static call), but the idea here is to illustrate the use of PL/SQL.

Using the test database

The test database has only the two methods described above '*Ex_Factorial_Function*' and '*Ex_Procedure_READINFO*'. These two methods can be used from the Design environment. Before using any of those methods, you have to modify the values of the connection parameters and the headers of the methods. Obviously, you will have to use your own values to be able to use the test database. Connection parameters are described below:

```
   `connection parameters
vt_UserName:="Scott"    `User name
vt_Password:="Tiger"    `Password
vt_HostName:="ORA8QA"  `Connection string
```

Conclusion

This Technical Note illustrated the use of the PL/SQL language from within a 4D application that uses 4D for OCI. The examples provided included both a PL/SQL procedure and a PL/SQL function. PL/SQL is a powerful language as it offers features like dynamic SQL and Object Oriented Programming and it brings great flexibility to a 4D application that uses 4D for OCI.