4D for OCI and PL/SQL

By Noreddine Margoum, QA Engineer, 4D SA.

TN 06-09

Introduction

このテクニカルノートでは、4D for OCI を使用したアプリケーションで、PL/SQL で記述されたルーチンを使用する方法を説明しています。

PL/SQL

PL/SQL(Procedural Language/Structured Query Language)は、SQL 言語を拡張したもののひとつで、SQL 言語に手続き型言語(procedural language)の特徴が加えられています。

このテクニカルノートでは PL/SQL 言語を解説することに主眼は置かれていないため、その論題については別の文献を参照する必要がありますが、掲載されている例題はどれも簡単なものなので PL/SQL の知識がなくても理解できるようになっています。このテクニカルノートは、PL/SQL を 4D for OCI と組み合わせて使用する方法を説明することを目的としています。

このテクニカルノート例題では、関数およびプロシージャの作成し、4D for OCI で使用するところまでが説明されています。

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;
```

'Factorial'関数についての注釈：

PL/SQL では、コメントアウトはスラッシュふたつによって示されます。
関数のヘッダでは、関数の名前、受け取るパラメータおよびタイプ、返されるパラメータおよびタイプが説明されています。上記の例では、受け取るパラメータはいずれも **NUMBER**、つまり 4D の Real タイプとほぼ一致するタイプのものです。**IN** というキーワードは、このパラメータが入力パラメータであることを示しています。**OUT** は出力パラメータ、**IN OUT** は入出力パラメータであることを意味します。

関数の本体は、**BEGIN** および **END** というキーワードで囲まれた部分です。

なお、PL/SQL の命令文は、セミコロンで区切られます。

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;
```

'READINFO'関数についての注釈：

READINFO というプロシージャは、カレントユーザ名、サーバの現在時刻および日付を文字列としてパラメータに返します。

システム日時の取得には Oracle SQL 関数の **SYSDATE** が使用されています。

日時を適切なフォーマットに変換するために使用している Oracle 関数は **TO_CHAR** です。

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)
v t_UserName:="Scott" `User name
v t_Password:="Tiger" `password
v t_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:=" "
            v t_SQL:=vt_SQL+"CREATE OR REPLACE FUNCTION Factorial (facto IN NUMBER)"+Char(LF
            ASCII code )
            v t_SQL:=vt_SQL+"RETURN NUMBER IS"+Char(LF ASCII code )
            v t_SQL:=vt_SQL+"BEGIN"+Char(LF ASCII code )
            v t_SQL:=vt_SQL+"IF facto = 0"+Char(LF ASCII code )
            v t_SQL:=vt_SQL+"THEN RETURN 1;"+Char(LF ASCII code )
            v t_SQL:=vt_SQL+"ELSE"+Char(LF ASCII code )
            v t_SQL:=vt_SQL+"RETURN facto*Factorial(facto-1);"+Char(LF ASCII code )
            v t_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:=" "
```

```
            v t_SQL:=vt_SQL+"BEGIN"+Char(LF ASCII code )
            v t_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_O
            UT )
            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
```

*'Ex Factorial Function'*メソッドについての注釈：

*'Ex Factorial Function'*メソッドの主な処理は、次のステップに分けることができます：

• PL/SQL 関数'Factorial'の定義と作成：CREATE OR REPLACE FUNCTION を使用することに

より、関数が Oracle サーバに保存されるようにしています。PL/SQL の各行の終わりには、ラインフィードのコードを挿入している点に注目してください。

• 関数に対するダイナミックコール：関数をテストするために、入出力パラメータを使用してコールします。PL/SQL BEGIN..END ブロックが'anonymous'ブロックと呼ばれているのは、それが Oracle データベースには常駐せず、逐次コンパイルされて実行されるためです。

• 関数の削除：簡単なスタティックの SQL リクエストを実行して、Oracle サーバに保存された関数を削除しています。

このメソッドでもっとも重要なポイントは、OCIBindByName コマンドを使用して PL/SQL 関数のパラメータを 4D の変数にバインドしている箇所です。コマンドの最後のパラメータの値(BIND_IN または BIND_OUT)でそのデータが 4D 変数からのもの(入力パラメータ)なのか 4D に返されたもの(出力パラメータ)なのかを判別している点に注目することができます。

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
`t o 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)
v t_UserName:="Scott" `User name
v t_Password:="Tiger" `password
v t_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:=" "
        v t_SQL:=vt_SQL+"CREATE OR REPLACE PROCEDURE READINFO (vUser OUT
        VARCHAR2,"+Char(LF ASCII code )
        v t_SQL:=vt_SQL+"vDate OUT VARCHAR2, vTime OUT VARCHAR2) IS"+Char(LF ASCII
        code )
        v t_SQL:=vt_SQL+"BEGIN"+Char(LF ASCII code )
        v t_SQL:=vt_SQL+"SELECT USER INTO vUser FROM DUAL;"+Char(LF ASCII code )
        v t_SQL:=vt_SQL+"SELECT TO_CHAR(SYSDATE,'DD-MM-YYYY') INTO vDate FROM
        DUAL;"+Char(LF ASCII code )
        v t_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:=" "
            v t_SQL:=vt_SQL+"BEGIN"+Char(LF ASCII code )
            v t_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
```

*'Ex_Procedure_READINFO'*メソッドについての注釈：

*'Ex_Procedure_READINFO'*メソッドの基本的な構造は、前述の'Ex_Factorial_Function'メソッドのものと同じです。関数の使用方法が分かったところで、今度は P/SQL プロシージャを使用する方法を例証するためにこのメソッドを掲載しました。

ここでは、すべての 4D 変数が出力パラメータ(BIND_OUT))として渡されています。この場合、Oracle からのデータ(ユーザ名、時刻、日付)を 4D に取り込むだけだからです。いうまでもなく、この処理自体は簡単な SQL コールでも実現できますが、ここでは PL/SQL を使用する方法を示すための分かりやすい例としてこのような方法をとっています。

Using the test database

テストデータベースは、ここで取り上げたプロジェクトメソッド*'Ex_Procedure_READINFO'*および'Ex_Factorial_Function'を含んだだけの簡単なものです。メソッドを実行する前に、冒頭の接続パラメータを書き換える必要があります。接続パラメータの意味は次のとおりです：

```
vt_UserName:="Scott"    `ユーザ名
vt_Password:="Tiger"    `パスワード
vt_HostName:="ORA8QA"   `接続ストリング
```

Conclusion

このテクニカルノートでは、4D for OCI を使用する 4D アプリケーションの中から PL/SQL を使用する方法を説明しました。付属のサンプルデータベースは、PL/SQL プロシージャおよび PL/SQL 関数の両方を取り上げています。PL/SQL は、ダイナミック SQL やオブジェクト指向プログラミングなどの特徴を有しており、4D for OCI アプリケーションの開発に柔軟性を付加する有用な言語です。