PL/SQL Procedimientos / Funciones

Antonio Espín Herranz

Subprogramas

- Scripts.
- Procedimientos almacenados.
- Funciones.

Scripts

- Ficheros de código pl/sql con extensión sql, se interpretan y ejecutan por el motor de Oracle.

- Normalmente los scripts son bloques de código PL/SQL anónimos que se utilizan para realizar tareas de mantenimiento y explotación de la Base de datos.

- Los subprogramas son bloques de código PL/SQL identificados y almacenados en la BD.
  - Procedimientos, funciones, triggers.

Procedimientos Almacenados

- Un procedimiento almacenado es un bloque de PL/SQL nominado y almacenado en la BD.
- Se puede ejecutar haciendo referencia a su nombre, dentro SQL *Plus se puede ejecutar con EXECUTE.
- Sintaxis:
  - Create [or replace] procedure nombre_procedimiento
    - [(param1 {IN / OUT / IN OUT} tipo, …)]
    - {IS/AS} Bloque PL/SQL;

Procedimientos Almacenados

- OR Replace:
  - Si el procedimiento ya existe se sustituye, si no se produciría un error si lo intentamos crear y ya existe.
- Parámetros:
  - Por cada parámetro indicamos el nombre, si es de entrada, salida o entrada / salida y el tipo.
- Si es necesario dentro del procedure se pueden definir variables.

Ejemplo

```
Create or replace procedure elim_arti(numero in char) as
        Begin
                Delete from lineasPed where refart=numero;
                Delete from articulos where refart=numero;
        End;
/
-- Llamada al procedimiento, desde el SQL *Plus:
Execute elim_arti('AB01');

--Llamada desde un bloque:
Declare
   x char(5);
Begin
   …
   elim_arti(x);
   commit;   -- Confirmar la operación.
End;
```

Si los parámetros son de tipo char, varchar2 NO tenemos que indicar la longitud en la declaración de parámetros. Esta se calculará a partir de la longitud del parámetro indicado.

Procedimientos almacenados

- Cuando declaramos variables dentro de un procedimiento no se indica “declare”.

- **CREATE [**OR **REPLACE]**
  PROCEDURE <*procedure_name*> [(<*param1*> [**IN**|**OUT**|**IN OUT**] <*type*>,
  <*param2*> [**IN**|**OUT**|**IN OUT**] <*type*>, ...)]
  AS
  -- Declaración de variables locales
  BEGIN
  -- Sentencias
  [**EXCEPTION**]
  -- Sentencias control de excepción
  END [<*procedure_name*>];

Parámetros por defecto

- En la declaración de parámetros se pueden indicar valores por defecto.
- create or replace PROCEDURE CUADRADO
  (num IN NUMBER **default 10**)
  AS …

- En la llamada si no se indica el valor del parámetro se tomará el valor 10.

Orden de los parámetros

- En la llamada al procedimiento se pueden colocar los parámetros en un orden distinto siempre y cuando se indique el nombre de estos.
- Si no, se rellenarán de izq. a der. como siempre.

Ejemplo

```
create or replace PROCEDURE CONCATENAR
( nombre IN VARCHAR2, apellidos IN VARCHAR2) AS
BEGIN
 dbms_output.put_line('Nombre completo es: ' || nombre || ', ' || apellidos);
END CONCATENAR;

-- En la llamada:
concatenar(apellidos => ‘Gracia’, nombre=>'Andrés');
```

Sobrecarga de Procedimientos

```
-- Sobrecarga de procedimientos almacenados locales:
set serveroutput on
DECLARE
PROCEDURE LocalProc(p_Parameter1 IN NUMBER) IS
BEGIN
 DBMS_OUTPUT.PUT_LINE('In version 1, p_Parameter1 = ' || p_Parameter1);
END LocalProc;

PROCEDURE LocalProc(p_Parameter1 IN VARCHAR2) IS
 BEGIN
   DBMS_OUTPUT.PUT_LINE('In version 2, p_Parameter1 = ' || p_Parameter1);
END LocalProc;

BEGIN
 LocalProc(12345);
 LocalProc('abcdef');
 END;
 /
```

A nivel global,
No lo permite, el procedimiento se actualiza y se pierde la 1ª versión.

Funciones Almacenadas

- Al igual que los procedimientos una función es un bloque de código PL/SQL que nos devuelve un resultado.
- Sólo puede devolver un único resultado.

- Sintaxis:

  Create [or replace] function nombre_función

  [(parámetro [IN] tipo, …)]

  Return tipo {IS/AS} bloque PL/SQL;

Funciones Almacenadas

- Or replace:
  - Si existe la función se reemplaza.
- Parámetro:
  - Parámetro de entrada dentro de la función.
- Tipo:
  - Tipo del parámetro.
- Return tipo:
  - Tipo devuelto por la función.

Ejemplo

```
Create or replace function factorial (n in number) return number is
        Begin
                If n = 0 then
                        Return (1);
                Else
                        Return (n * factorial(n – 1));
                End if;
        End;
/
-- La llamada puede ser:
Select factorial(5) from dual;
-- O también podemos recoger el resultado en una variable:
miVar:=factorial(4);
```

Restricciones

- Las funciones almacenadas tienen las siguientes **restricciones**:
  - **Modificar variables globales** dentro de un Paquete.
  - Contener parámetros ***OUT o IN OUT***.
  - Ser una función de *columna o grupo. Las funciones de columna son aquéllas que toman como* entrada toda una columna de datos, como por ejemplo *SUM(), o AVG(). Por lo tanto nuestras* funciones sólo podrán comportarse como funciones de *fila.*