1. Procediments i funcions

NF2. Procediments i funcions. Ordres per manipular les dades

UF3 - Llenguatges SQL: DCL i extensió procedimental

Desenvolupament d'Aplicacions Multiplataforma
M02 – Bases de dades. Versió 1.0
© Mª Carmen Brito Ruiz

1.1. Procedimientos y Funciones

Al empezar la unidad comentamos que había tres tipos de bloques PL/SQL:

• **Bloques Anónimos:** Estos bloques no tienen nombre y es la zona de declaraciones que comienza con la palabra reservada DECLARE.

• **Subprogramas:** Son los bloques PL/SQL que tienen un nombre y la zona de declaraciones comienza con la palabra reservada IS. Estos subprogramas pueden ser de dos tipos:

a) **Procedimientos**: Es el tipo que más se usa en PL/SQL y normalmente se almacenan en la base de datos.

b) **Funciones**: Es otro tipo que también se usa en PL/SQL y su formato genérico es similar al de los procedimientos, pero pueden devolver un valor.

1.2. Procedimientos

```
PROCEDURE <nombre_procedimiento>
[(<lista_de_parámetros>)]
IS[AS]
[<declaraciones_objetos_locales>;]
BEGIN
 <instrucciones>;
[EXCEPTION
 <excepciones>;]
END <nombre_procedimiento>;
```

Es la cabecera, donde va el nombre del procedimiento y los parámetros.

Es el cuerpo del procedimiento y es un bloque PL/SQL. En este bloque se incluye las declaraciones, instrucciones y manejo de excepciones (si son necesarias).

1.2.1. Lista de parámetros de los procedimientos

La lista de parámetros indica la declaración de cada uno de los parámetros separados por comas. La sintaxis es:

```
<nombre_variable> [IN|OUT|
IN OUT]<tipo_de_dato>[{:=|DEFAULT}<valor>]
```

donde,

nombre_variable ⇨ es el identificador de la variable, que ha de cumplir las reglas que hemos estudiado sobre los identificadores.

IN | *OUT* | *IN OUT* ⇨ son las opciones que hacen referencia al tipo de parámetro: entrada | salida | entrada / salida.

tipo de dato ⇨ es el tipo de datos de la variable.

valor ⇨ es el valor de la variable.

1.2.2. Crear y borrar un procedimiento

Para crear procedimiento:

```
CREATE [OR REPLACE] PROCEDURE <nombre_procedimiento>
```

donde:

CREATE ⇨ crea el procedimiento correspondiente.

OR REPLACE ⇨ sustituye (machaca) el procedimiento existente.

nombre_procedimiento ⇨ identifica el nombre del procedimiento.

Para borrar procedimiento:

```
DROP PROCEDURE <nombre_procedimiento>;
```

1.2.3. Ejecución de un procedimiento

➢ Se ejecuta y si 'compila' bien, da el mensaje:

```
nombre_procedimiento compilado
```

➢ Aunque está creado el procedimiento, se ha de invocar desde un bloque principal u otro subprograma.

```
SET SERVEROUTPUT ON
SET VERIFY OFF
SET ECHO OFF
[DECLARE]
BEGIN
 nombre_procedimiento();
END;
/
```

1.2.4. Invocar un procedimiento

- desde un bloque principal u otro subprograma.

- desde Sqldeveloper (sin programar un bloque principal / subprograma), escribiendo:

```
EXECUTE nombreprocedimiento;
```

- desde isqlPlus:

```
SQL> START .\procedimientos\saludo.sql
```

1.2.5. Ejemplos de procedimiento

Ejemplo1:

Crear un procedimiento que imprima por pantalla un mensaje de saludo.

```
CREATE OR REPLACE PROCEDURE saludo
IS [o bien, AS]
BEGIN
  DBMS_OUTPUT.PUT_LINE ('HOLA SOY YO');
END saludo;
/
```

El procedimiento se ha creado pero ahora se ha de llamar, por ejemplo desde otro script:

```
SET SERVEROUTPUT ON
BEGIN
  saludo;
END;
```

Ejemplo2:

Crear un procedimiento que imprima por pantalla un mensaje de saludo, saludando a una persona correspondiente (el nombre de la persona se le pasa por parámetro).

```
CREATE OR REPLACE PROCEDURE saludocon(nombre VARCHAR2)
AS
BEGIN
  DBMS_OUTPUT.PUT_LINE ('HOLA SOY YO '||nombre);
END saludocon;
/
```

Cuando se ejecuta, aparece el mensaje de Procedimiento compilado. A partir de estos momentos el procedimiento ya está preparado para que pueda ser llamado.

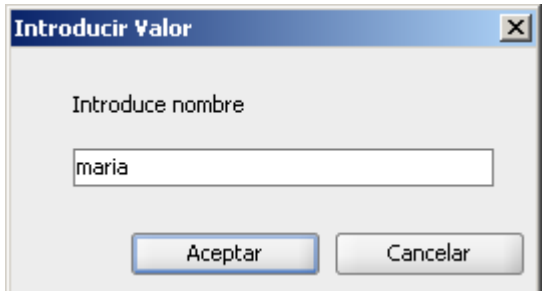
```
PROCEDURE SALUDOCON compilado
```

Bloque principal que llama al procedimiento `saludocon` pasándole como parámetro el nombre de la persona a saludar. Uno se le pasa un nombre fijo y el otro se le pregunta al usuario.

```
SET SERVEROUTPUT ON
SET ECHO OFF
SET VERIFY OFF
BEGIN
  saludocon('PEPE');
END;
```

```
bloque anónimo terminado
HOLA SOY YO PEPE
```

```
ACCEPT varnombre PROMPT 'Introduce nombre'
SET SERVEROUTPUT ON
SET ECHO OFF
SET VERIFY OFF
BEGIN
  saludocon(UPPER('&varnombre'));
END;
```




```
bloque anónimo terminado
HOLA SOY YO MARIA
```

Ejemplo3:

Crear un procedimiento que salude a una persona por su nombre y le diga quien la saluda (por su apellido). El procedimiento aprovechará el procedimiento `saludocon` creado anteriormente.

Se ha de introducir el nombre la persona a saludar y el apellido de la persona que saluda.

```
CREATE OR REPLACE PROCEDURE escribir(
nombre VARCHAR2, apellidos VARCHAR2)
AS
BEGIN
  saludocon(UPPER(nombre));
  DBMS_OUTPUT.PUT_LINE('SOY TU AMIGO, '||UPPER(apellidos));
END escribir;
/
```

Ahora se ha de llamar al procedimiento, desde un programa principal:

```
SET SERVEROUTPUT ON
SET ECHO OFF
SET VERIFY OF
ACCEPT varnombre PROMPT 'Introduce el nombre: '
ACCEPT varapellido PROMPT 'Introduce el apellido: '
BEGIN
  escribir ('&varnombre', '&varapellido');
END;
```

1.3. Funciones

```
FUNCTION <nombre_función>
[(<lista_de_parámetros>)]
RETURN <tipo_de_valor_devuelto>
IS[AS]
 [<declaraciones_objetos_locales>;]
BEGIN
  <instrucciones>;
  RETURN <expresión>;
  …
[EXCEPTION
  <excepciones>;]
END <nombre_función>;
```

Las funciones son subprogramas como los procedimientos, admite el paso de parámetros.

A diferencia de los procedimientos, las funciones devuelven un valor.

Este valor puede ser un número, carácter, un registro, etc.

1.3.1. Parámetros de una función

A la función también se la pasa parámetros como a un procedimiento. La sintaxis es:

```
<nombre_variable> [IN|OUT|
IN OUT]<tipo_de_dato>[{:=|DEFAULT}<valor>]
```

Tipo	Descripción
IN	Este parámetro permite pasar valores a un subprograma. Dentro del subprograma, el parámetro actúa como una constante (no se le puede asignar ningún valor). Y en cuanto al parámetro actual puede ser una variable, constante, literal o expresión.
OUT	Este parámetro permite devolver valores al bloque que llama al subprograma correspondiente. Dentro del subprograma, el parámetro actúa como una variable no inicializada. Y el parámetro actual debe ser una variable. Este tipo de parámetro no puede intervenir en ninguna expresión, sólo puede tomar un valor.
IN OUT	Este parámetro permite pasar un valor inicial y devolver un valor actualizado. Dentro del subprograma actúa como una variable inicializada. Y el parámetro actual debe ser una variable. Este tipo de parámetro puede intervenir en otras expresiones y puede tomar nuevos valores.

1.3.2. Crear y borrar una función

Para crear una función :

```
CREATE [OR REPLACE] FUNCTION <nombre_función>
```

donde:

CREATE ⇨ crea el procedimiento correspondiente.

OR REPLACE ⇨ sustituye (machaca) el procedimiento existente.

nombre_función ⇨ identifica el nombre de la función.

Para borrar una función:

```
DROP FUNCTION <nombre_función>;
```

1.3.3. Ejemplos de función

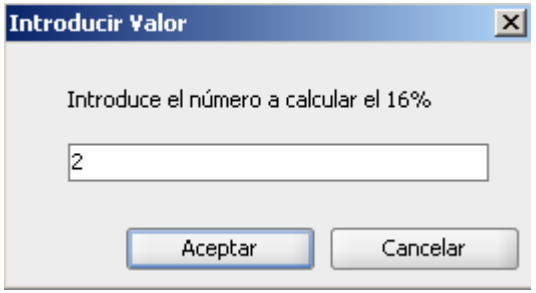
Ejemplo 1:

Crear una función que calcule el 16% de una cantidad que se le pasará como parámetro. La cantidad la introducirá el usuario en el programa principal.

```
CREATE OR REPLACE FUNCTION calculo (i NUMBER)
RETURN NUMBER
IS
 resul NUMBER;
BEGIN
 resul := i*1.16;
 RETURN(resul);
END calculo;
/
```

El programa principal que llama a la función:

```
SET VERIFY OFF
SET ECHO OFF
SET SERVEROUTPUT ON
ACCEPT var_num PROMPT 'Introduce el número a calcular el 16%'
BEGIN
 DBMS_OUTPUT.PUT_LINE ('El 16% del número es: '||calculo(&var_num));
END;
```




```
bloque anónimo terminado
El 16% del número es: 2,32
```

Ejemplo 2:

Crear una función llamada c_euro, que recibirá un número que será una cantidad en pesetas y devuelve dicha cantidad en euros.

```
CREATE OR REPLACE FUNCTION c_euro (cantidad_ptas NUMBER)
RETURN NUMBER
AS
 cantidad_euros NUMBER;
BEGIN
 IF cantidad_ptas<0 THEN
  DBMS_OUTPUT.PUT_LINE ('La cantidad no puede ser
negativa');
 ELSE
  cantidad_euros:=cantidad_ptas/166.386;
 END IF;
 RETURN (cantidad_euros);
END c_euro;
/
```

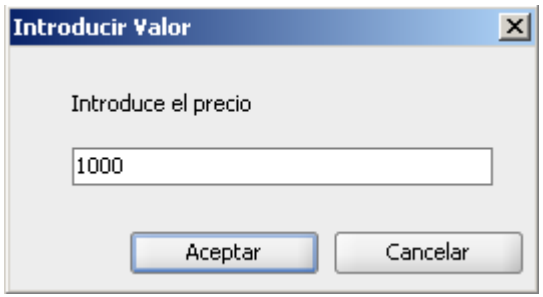
La función `c_euro`, se llamará desde el procedimiento `cambio`, donde se imprimirá con dos decimales el importe en euros.

```
CREATE OR REPLACE PROCEDURE cambio (cantidad_ptas NUMBER)
AS
 cantidad NUMBER;
BEGIN
 cantidad:=ROUND(c_euro(cantidad_ptas),2);
 DBMS_OUTPUT.PUT_LINE(cantidad_ptas||' ptas, son
'||cantidad||' euros');
END cambio;
/
```

¿Devuelve algún resultado por pantalla este procedimiento?

El procedimiento imprime el mensaje correspondiente del cálculo, pero ahora haría falta un script que llame al procedimiento `cambio`.

```
SET SERVEROUTPUT ON
SET VERIFY OFF
SET ECHO OFF
ACCEPT varmoneda PROMPT 'Introduce el precio'
BEGIN
 cambio(&varmoneda);
END;
/
```



→ `1000 ptas, son 6,01 euros`

Preguntes!!!!!