Práctica 2: El lenguaje de programación PL/SQL

Objetivo

Estudiar el lenguaje de programación procedimental PL/SQL mediante una aplicación que permita listar y comprobar algunas condiciones de consistencia. Para ello se estudiará:
- Declaración de bloques y procedimientos almacenados.
- Declaración y uso de cursores.
- Uso de instrucciones de recorrido de cursores.
- Uso de instrucciones de control de PL/SQL.
- Uso de instrucciones de entrada/salida por pantalla.
- Manejo de excepciones.

Introducción

PL/SQL es un lenguaje de programación Turing-completo, que incluye características específicas para el acceso a bases de datos. En particular, este acceso es posible mediante cursores (un objeto que representa a un conjunto de datos extraídos mediante una instrucción SELECT) y el sistema de tipos del lenguaje permite trabajar directamente con los datos persistentes. Es posible ejecutar sentencias SQL directamente en un bloque PL/SQL y asignar el resultado a una variable de programa, como en:

```
SELECT COUNT(*) FROM Clientes INTO var_número_de_clientes;
```

El código PL/SQL se puede escribir en el contexto de un bloque, que pueden ser anónimos (sin nombre y no persistente) o almacenados (con nombre y persistente). Un bloque anónimo en PL/SQL se define como:

```
DECLARE
  Declaraciones de variables y cursores
BEGIN
  Instrucciones PL/SQL
EXCEPTION
  Tratamiento de excepciones
END;
```

(Si se usa la consola SQL*Plus hay que añadir una barra de división al final (/), para indicar el final de la entrada).

Un cursor es un objeto que almacena registros (filas de las tablas) que proceden de tablas mediante consultas. El recorrido de los registros se realiza mediante un bucle en cuyo cuerpo se usa una instrucción de lectura de registros (por ejemplo, FETCH). El siguiente es un ejemplo de uso de un cursor dentro de un bloque anónimo:

```
DECLARE
  v_StudentID    students.id%TYPE;
  v_FirstName    students.first_name%TYPE;
  v_LastName     students.last_name%TYPE;
  v_Major        students.major%TYPE := 'Computer Science';
  CURSOR c_Students IS
    SELECT id, first_name, last_name
      FROM students
      WHERE major = v_Major;
BEGIN
  OPEN c_Students;
  LOOP
    FETCH c_Students INTO v_StudentID, v_FirstName, v_LastName;
    EXIT WHEN c_Students%NOTFOUND;
  END LOOP;
  CLOSE c_Students;
END;
```

En este ejemplo los tipos se escogen directamente del esquema de la base de datos (***campo*%TYPE**), aunque también se pueden escribir explícitamente (como **VARCHAR(10)**), en particular los que soporta la base de datos y que se escriben en las sentencias **CREATE TABLE**.

Los procedimientos almacenados son grupos de instrucciones PL/SQL que se pueden llamar por su nombre. Son similares a los procedimientos conocidos de los lenguajes de tercera generación, pero a diferencia de éstos, se almacenan directamente en la base de datos (de ahí que sean persistentes). También admiten parámetros. Asimismo, existen funciones almacenadas.

Un procedimiento almacenado PL/SQL se define como:

```
CREATE OR REPLACE PROCEDURE Nombre [Lista_de_parámetros] AS
 Declaraciones
BEGIN
 Instrucciones PL/SQL
EXCEPTION
 Tratamiento de excepciones
END;
```

Se pueden usar estructuras de control como **IF THEN ELSE**. La parte **EXCEPTION** maneja el control de errores mediante excepciones. Para posibilitar la salida a pantalla se usa el paquete **DBMS_OUTPUT** y los métodos **PUT**, **PUT_LINE** y **NEW_LINE**. Antes de poder escribir algo es necesario emitir la instrucción:

```
SET SERVEROUTPUT ON SIZE 1000000.
```

Ejemplo:

```
CREATE OR REPLACE PROCEDURE DetectarNulos AS
Declaraciones
BEGIN
 ...
  IF v_cp IS NULL THEN //  Si el código postal es nulo se genera una excepción
    RAISE excepcion_nulo;
  END IF;
 ...
EXCEPTION
  WHEN excepcion_nulo
    DBMS_OUTPUT.PUT_LINE ('Código postal nulo.');
    ...
END;
```

Para crear un procedimiento se puede hacer directamente en SQL Developer, o en la consola SQL o en la Consola de administración Enterprise Manager. Si se usa esta última se escoge la carpeta Esquema y se pulsa con el botón derecho sobre nuestro usuario y en Crear en el menú contextual que aparece. A partir de aquí se puede elegir el objeto a crear (en este caso, un procedimiento). Si se usa el primer método y al crear el procedimiento se obtiene un aviso de errores, se puede introducir **SHOW ERRORS** para mostrar la información acerca de los errores.

Apartado 0

Ejecuta en SQLDeveloper el siguiente bloque de PL/SQL:

```
SET SERVEROUTPUT ON SIZE 100000;
DECLARE
  V VARCHAR(10);
BEGIN
  V:='Hola';
  DBMS_OUTPUT.PUT_LINE(V);
END;
```

IMPORTANTE: Antes de crear procedimientos almacenados hay que entrar con el usuario de administración ADMINUSER, contraseña ADMINUSER y dar permiso al usuario GIICxx para crear, modificar, borrar y ejecutar procedimientos almacenados:

GRANT CREATE PROCEDURE, ALTER ANY PROCEDURE, DROP ANY PROCEDURE, EXECUTE ANY PROCEDURE TO GIICxx.

Apartado 1

Procedimiento almacenado llamado `PEDIDOS_CLIENTE` que reciba como parámetro el `DNI` de un cliente y muestre por pantalla sus datos personales, junto con un listado con los datos de los pedidos que ha realizado (código de pedido, fecha, fecha de entrega, estado e importe del pedido), ordenados crecientemente por fecha. En caso de error (DNI no existe, no hay pedidos para ese cliente, etc..), deberá mostrarse por pantalla un mensaje de advertencia explicando el error. Al finalizar el listado se deberá mostrar la suma de los importes de todos los pedidos del cliente. Incluye un bloque de código anónimo para probar el procedimiento para cada uno de los casos posibles (el cliente no existe, si existe pero no tiene pedidos y si existe y además tiene pedidos).

Apartado 2

Procedimiento almacenado llamado `REVISA_PRECIO_CON_COMISION` (sin argumentos) cuya misión es comprobar la consistencia de los datos de todos los precios con comisión en la tabla Contiene. El campo “precio con comisión” de la tabla “Contiene” debe almacenar el precio del plato incluyendo el porcentaje de la comisión de su restaurante.

El procedimiento debe verificar y actualizar estos datos de modo que resulten consistentes. Si todos los datos son correctos, se mostrará un mensaje indicando “Ningún cambio en los datos de Contiene”. En caso contrario se indicará el número de filas modificadas en Contiene.

Apartado 3

Procedimiento almacenado llamado `REVISA_PEDIDOS` (sin argumentos) cuya misión es comprobar la consistencia de los datos de todos los pedidos. El campo “importe total” de la tabla “Pedidos” debe almacenar la suma de los “precio con comisión” de los platos del pedido multiplicados por su cuantía.
Se pide usar un cursor **FOR UPDATE**
El procedimiento debe verificar y actualizar estos datos para todos los pedidos, de modo que resulten consistentes. Si todos los datos son correctos, se mostrará un mensaje indicando “Ningún cambio en los datos de la tabla Pedidos”. En caso contrario se indicará el número de filas modificadas en en la tabla Pedidos.

Apartado 4

Crea un procedimiento DATOS_CLIENTES que recorra todos los Clientes con un **FOR** y muestre todos sus datos junto con la suma de importe total de todos sus pedidos. Finalmente se mostrará la suma total de los importes de todos los pedidos de todos los clientes.

Apartado 5

Crea un procedimiento que llame a `REVISA_PRECIO_CON COMISION, REVISA_PEDIDOS y DATOS CLIENTES.` Incluye un bloque anónimo de prueba.