Práctica 5: El lenguaje de programación PL/SQL

Objetivo

Estudiar el lenguaje de programación procedimental PL/SQL mediante una aplicación que compruebe la consistencia de datos con respecto a ciertas restricciones de integridad. 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 de cuarta generación, es decir, un lenguaje de propósito general y procedimental que tiene integrados los tipos y sus operaciones asociadas correspondientes a un SGBD concreto; en este caso, a Oracle. Además es posible ejecutar sentencias SQL directamente sin necesidad de usar ningún API. El acceso a los datos de las tablas se hace mediante cursores (un objeto que representa a un conjunto de datos extraídos mediante una instrucción **`SELECT`**) o incluso directamente en variables a partir de una **`SELECT`** (son los denominados cursores en línea). Las instrucciones de PL/SQL se pueden escribir en determinados contextos: bloques, procedimientos almacenados y disparadores.

Un bloque PL/SQL se define como:

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

Un *cursor* es un objeto que almacena registros (filas con sus campos) que proceden de tablas y vistas mediante consultas. Hay dos tipos: cursores en línea y cursores explícitos.

El siguiente ejemplo muestra cómo es posible ejecutar sentencias SQL directamente en un bloque PL/SQL asignando el resultado a una variable de programa. Esto es posible gracias al concepto de cursor en línea: no es necesario definirlo, simplemente se crea antes de ejecutar la instrucción y se descarta automáticamente después de que termine.

```
SELECT COUNT(*) FROM students INTO var_número_de_estudiantes;
```

El siguiente ejemplo es un cursor explícito cuyos registros se recorren mediante un bucle. En el cuerpo del bucle se usa la instrucción de lectura de registros **`FETCH`** que devuelve una tupla con los campos del registro actual. Los campos se corresponden con la lista de proyección de la instrucción SQL que lo define.

```
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;
```

La condición de salida del bucle es **c_Students%NOTFOUND**, que es cierta cuando no haya más registros que explorar en el cursor. 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 estos, se almacenan directamente en la base de datos. 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. Para posibilitar la salida a pantalla se usa el paquete DBMS_OUTPUT y los procedimientos **PUT**, **PUT_LINE** y **NEW_LINE** (¡pero no hay equivalentes para la entrada de datos!). Un procedimiento almacenado se ejecuta con la instrucción:

```
execute NombreProcedimiento;
```

Importante: Antes de poder escribir en consola con las instrucciones anteriores es necesario habilitar la salida por consola con la siguiente instrucción (se emite una sola vez desde la consola para toda la sesión), donde el número indica el tamaño del buffer de salida:

```
SET SERVEROUTPUT ON SIZE 100000;
```

Fragmento de un procedimiento almacenado para mostrar el manejo de excepciones:

```
CREATE OR REPLACE PROCEDURE ComprobarNulos AS
...
  excepcion_nulo EXCEPTION;
...
BEGIN
 ...
  RAISE excepcion_nulo;
 ...
EXCEPTION
  WHEN excepcion_nulo
    DBMS_OUTPUT.PUT_LINE ('Código postal nulo.');
...
END;
```

Se puede crear un procedimiento almacenado desde Oracle SQL Developer o desde la consola SQL. Si se usa este último método y al crear el procedimiento se obtiene un aviso de errores, se puede introducir **SHOW ERRORS** para mostrar la información acerca de estos errores.

Ejemplo a probar

Ejecuta en SQLDeveloper el siguiente bloque de PL/SQL (se indica también la habilitación de la salida):

```
SET SERVEROUTPUT ON SIZE 100000;

DECLARE
  V VARCHAR(10);
BEGIN
  V:='Hola';
  DBMS_OUTPUT.PUT_LINE(V);
END;
```

Se debe comprobar que en la ventana de salida se muestra la cadena 'Hola' (sin las comillas). No es necesario subir nada al CV.

Ejercicios

En este apartado se llevará a cabo una práctica habitual en la empresa: la migración de datos. El objetivo es asegurar las restricciones de integridad definidas en el diseño de la base de datos que se va a cargar con datos de una fuente externa. Se asume que estos datos no son fiables y pueden ser inconsistentes. En este caso se parte de datos para las tablas Domicilios y Códigos postales almacenados en ficheros ASCII delimitados por el carácter separador ";" que se han exportado desde alguna otra fuente.

Como resultado de este apartado se debe subir al CV un documento PDF con las instrucciones usadas, procedimientos escritos y resultados de las ejecuciones para cada uno de los subapartados siguientes:

Se pide:

1) Generar manualmente los ficheros ASCII con los siguientes datos.

Códigos postales I
08050;Parets;Barcelona;
14200;Peñarroya;Córdoba;
14900;Lucena;Córdoba;
;Arganda;Sevilla;
08050;Zaragoza;Zaragoza;
28040;Arganda;Madrid;
28004;Madrid;Madrid;

Domicilios I
12345678A;Avda. Complutense;28040;
12345678A;Cántaro;28004;
12345678P;Diamante;14200;
12345678P;Carbón;14901;

2) Crear las tablas "Domicilios I" y "Códigos postales I" con el mismo esquema que Domicilios y " Códigos postales" pero sin restricciones de integridad.
3) Importar con Oracle Loader las tablas del punto 1 en estas nuevas tablas.
4) Escribir un procedimiento almacenado denominado "comprobar_NULL" que permita la detección de valores nulos en alguno de los campos de la tabla "Códigos postales I", y que emita un error por pantalla que identifique el problema. Para ello hay que definir el cursor con una instrucción SQL que devuelva las tuplas con un valor nulo en alguno de sus campos. Después se recorrerá este cursor y se mostrarán las tuplas erróneas. Si ha habido algún error, al final se emitirá una excepción con la instrucción **`RAISE`** que informará del número de tuplas incorrectas.
5) Crear un procedimiento almacenado denominado "comprobar_PK" que permita detectar la violación de clave primaria en la tabla la tabla "Códigos postales I" y que emita por pantalla un error que identifique el primer problema encontrado. Se debe proceder de manera similar al apartado 4.
6) Crear un procedimiento almacenado paramétrico denominado "comprobar_FK"que permita detectar la violación de integridad referencial en la tabla "Domicilios I" y que emita un error por pantalla que identifique el problema. Recibirá un número como entrada que indica el límite de tuplas erróneas a mostrar. Se debe proceder de manera similar al apartado 4. Para comprobar este apartado hay que reemplazar el valor nulo encontrado en el apartado 4 por el valor **`'14900'`**.