



Programación de Base de Datos

Documentación de práctica 2



Trabajo realizado por:

Carlos Guillén Moreno

Para el profesor: Andrés Caro Lindo (correo: andresc@unex.es)

Contenido

INTRODUCCIÓN	3
EJERCICIOS SIN EXCEPCIONES.....	3
InsertarClases()	3
VerClases()	4
ModificarClases()	5
ConsultarClases()	6
BorrarClases()	7
EJERCICIOS CON EXCEPCIONES.....	8
InsertarClases2()	8
ModificarClases2()	9
ConsultarClases2()	10
BorrarClases2()	11
SALIDA POR CONSOLA.....	12

INTRODUCCIÓN

Para esta práctica, se usan las herramientas instaladas en la sesión anterior. Se utilizará el gestor de base de datos (Oracle 11g Express Edition) y el entorno (SQL Developer).

EJERCICIOS SIN EXCEPCIONES

InsertarClases()

Método que dado un código de clase (clave primaria), un nombre de la clase y los ingresos, inserta en la tabla clases una tupla con esos valores.

```
PROCEDURE InsertarClases(cod NUMBER, nombre VARCHAR2, ingresos NUMBER)
IS
BEGIN
    DBMS_OUTPUT.PUT_LINE( 'SE VA A INSERTAR UNA CLASE' );
    INSERT INTO clases VALUES(cod, nombre, ingresos);
    DBMS_OUTPUT.PUT_LINE( 'CLASE INSERTADA' );
    DBMS_OUTPUT.PUT_LINE( '-----' );
    DBMS_OUTPUT.PUT_LINE( '-----' );
END;
```

Entre paréntesis se muestran los valores que se van a añadir a la tupla, siendo **cod** el código de la clase, **nombre** el nombre de la clase e **ingresos** el valor de los ingresos de la clase. En el BEGIN se escribe el código necesario para hacer la inserción. Uso dos mensajes por pantalla para saber que inicio la inserción y que la finalizo, para ello uso la función **DBMS_OUTPUT.PUT_LINE('mensaje')** y entre las comillas escribo el mensaje a mostrar. La función de **insert** es la que se encarga de añadir la tupla a la tabla en la base de datos con los datos que introducimos por parámetros de entrada. El END indica que acaba el procedimiento.

Se adjunta una imagen de lo que saldría por consola al ejecutar el método:



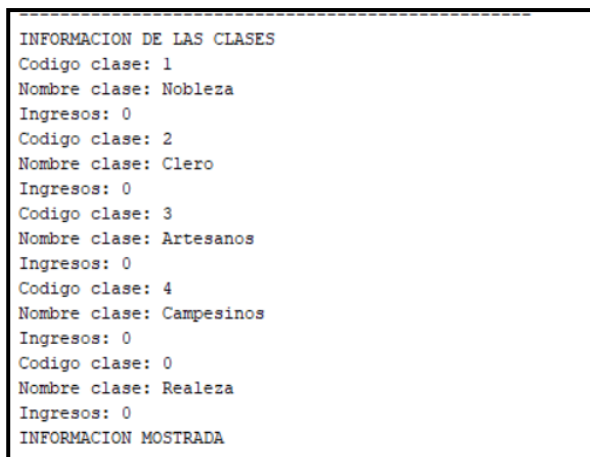
VerClases()

Método que se encarga de mostrar la información (los atributos) de cada tupla que hay insertada en la tabla clases.

```
PROCEDURE VerClases IS
    --Creo un cursor para almacenar las tuplas y poder recorrerlas en
    un bucle después
    CURSOR c_clase IS SELECT * FROM clases;
    --Declaro una variable auxiliar para almacenar cada clase, la
    llamaré aux
    aux clases%ROWTYPE;
BEGIN
    DBMS_OUTPUT.PUT_LINE( 'INFORMACION DE LAS CLASES' );
    OPEN c_clase;
    LOOP
        FETCH c_clase INTO aux;
        EXIT WHEN c_clase%notfound;
        dbms_output.put_line('Codigo clase: ' || aux.cod);
        dbms_output.put_line('Nombre clase: ' || aux.nombre);
        dbms_output.put_line('Ingresos: ' || aux.ingresos);
    END LOOP;
    CLOSE c_clase;
    DBMS_OUTPUT.PUT_LINE( 'INFORMACION MOSTRADA' );
    DBMS_OUTPUT.PUT_LINE( '-----' );
END;
```

En este método utilizamos un cursor (**c_clase**) en el que cargaremos todas las instancias de la tabla clases utilizando la **sentencia SQL en color verde**. Muestro un mensaje en el BEGIN de que comienzo a mostrar la información de las clases usando la función **DBMS_OUTPUT.PUT_LINE('mensaje')**, abro el cursor usando el OPEN seguido del nombre de la variable cursor e inicio un bucle usando la palabra reservada LOOP y usando FETCH en cada iteración, inserto el valor de una tupla de las que hay en el cursor en la variable local **aux** (**una variable del mismo tipo que clase**) y voy mostrando cada campo de la tupla (**código, nombre e ingresos**). El bucle se cierra con END LOOP y en la línea siguiente, cerramos el cursor. Muestro un **mensaje al final** para indicar que se ha acabado el procedimiento y se ha mostrado la información.

Se adjunta una imagen de lo que saldría por consola al ejecutar el método:



```
-----
INFORMACION DE LAS CLASES
Codigo clase: 1
Nombre clase: Nobleza
Ingresos: 0
Codigo clase: 2
Nombre clase: Clero
Ingresos: 0
Codigo clase: 3
Nombre clase: Artesanos
Ingresos: 0
Codigo clase: 4
Nombre clase: Campesinos
Ingresos: 0
Codigo clase: 0
Nombre clase: Realeza
Ingresos: 0
INFORMACION MOSTRADA
```

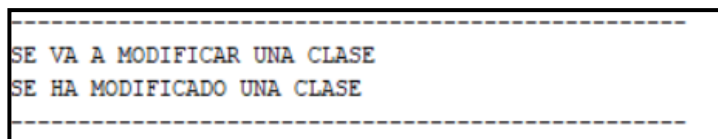
ModificarClases()

Método que se encarga de actualizar los valores de la tupla en la tabla clases usando el código de la clase que se corresponde a su clave primaria.

```
PROCEDURE ModificarClases(cod2 NUMBER, nombre2 VARCHAR2, ingresos2
NUMBER) IS
BEGIN
    DBMS_OUTPUT.PUT_LINE( 'SE VA A MODIFICAR UNA CLASE' );
    UPDATE CLASES
    SET NOMBRE = nombre2, INGRESOS=ingresos2
    WHERE COD=cod2;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE( 'SE HA MODIFICADO UNA CLASE' );
    DBMS_OUTPUT.PUT_LINE( '-----' );
END;
```

Los parámetros de entrada del método hacen referencia al **código de la clase**, al **nombre de la clase** y a los **ingresos de la clase**. Dentro del BEGIN muestro un **mensaje** para indicar que voy a modificar la clase y utilizo la sentencia **UPDATE** que actualiza los valores de la tupla que tenga de código de la clase igual al que he pasado por parámetros (cod2). Realizo un **COMMIT** para marcar el final de la transacción y muestro un **mensaje** para indicar que ya he modificado la clase. Con el END marco el final del método.

Se adjunta una imagen de lo que saldría por consola al ejecutar el método:



```
SE VA A MODIFICAR UNA CLASE
SE HA MODIFICADO UNA CLASE
-----
```

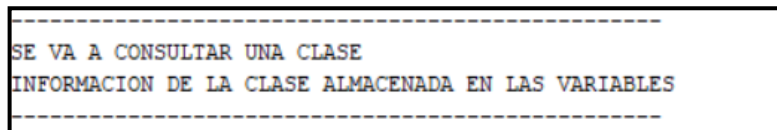
ConsultarClases()

Método que obtiene la información de una clase y la almacena en las variables de entrada/salida del procedimiento para mostrarlas una vez finalizado el método con un mensaje por pantalla.

```
PROCEDURE ConsultarClases(cod2 NUMBER, nombre2 OUT VARCHAR2, ingresos2
OUT NUMBER) IS
BEGIN
    DBMS_OUTPUT.PUT_LINE( 'SE VA A CONSULTAR UNA CLASE' );
    SELECT NOMBRE, INGRESOS INTO nombre2, ingresos2 FROM CLASES where
COD=cod2;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE( 'INFORMACION DE LA CLASE ALMACENADA EN LAS
VARIABLES' );
    DBMS_OUTPUT.PUT_LINE( '-----' );
END;
```

En este método las variables son de entrada-salida, para ello es necesario poner **OUT** entre el nombre de la variable y el tipo de la variable. El valor retornado se almacena por si se quisiese trabajar con esos datos más adelante. Dentro del BEGIN muestro un **mensaje** para indicar que se va a realizar la **consulta** de la clase usando su **código de clase** (clave primaria) y almaceno los valores del nombre y los ingresos en las variables **nombre2** e **ingresos2** que son las variables de entrada-salida. Realizo un **COMMIT** para mostrar la finalización de la transacción. Para finalizar el método muestro por pantalla un **mensaje** indicando que he almacenado la información.

Se adjunta una imagen de lo que saldría por consola al ejecutar el método:



```
-----
SE VA A CONSULTAR UNA CLASE
INFORMACION DE LA CLASE ALMACENADA EN LAS VARIABLES
-----
```

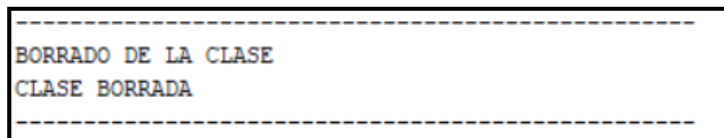
BorrarClases()

Método que, dado un código de clase, elimina la clase de la tabla clases usando el identificador.

```
PROCEDURE BorrarClases (cod2 NUMBER) IS
BEGIN
    DBMS_OUTPUT.PUT_LINE('BORRADO DE LA CLASE');
    DELETE FROM clases WHERE cod=cod2;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('CLASE BORRADA');
    DBMS_OUTPUT.PUT_LINE('-----');
END;
```

Este método tiene una variable de entrada que indica el código de la clase que se intenta eliminar. En el BEGIN muestro un **mensaje** de que se procede a borrar la clase. Seguidamente uso la **sentencia** DELETE para borrar de la tabla clases, la tupla usando su código de clase. Realizo el **COMMIT** para finalizar la transacción y muestro un **mensaje** final para indicar que he acabado de borrar la clase. Finalizo el método con el END.

Se adjunta una imagen de lo que saldría por consola al ejecutar el método:



```
-----
BORRADO DE LA CLASE
CLASE BORRADA
-----
```

EJERCICIOS CON EXCEPCIONES

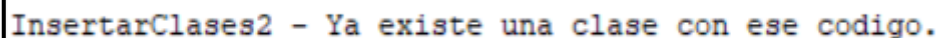
Estos ejercicios son iguales que los anteriores pero se les ha añadido excepciones para controlar posibles errores. Se va a detallar en cada método que hace la excepción añadida.

InsertarClases2()

```
PROCEDURE InsertarClases2(cod2 NUMBER, nombre VARCHAR2, ingresos
NUMBER) IS
    x NUMBER;
    Excep1 EXCEPTION;
BEGIN
    SELECT COUNT(*) INTO x FROM CLASES WHERE COD=cod2;
    --Salta la excepción si ya existe una clase con esa clave primaria
    IF(x=1) THEN
        RAISE Excep1; --lanza la excepcion
    END IF;
    DBMS_OUTPUT.PUT_LINE( 'SE VA A INSERTAR UNA CLASE' );
    INSERT INTO clases VALUES(cod2, nombre, ingresos);
    DBMS_OUTPUT.PUT_LINE( 'CLASE INSERTADA' );
    EXCEPTION --captura la excepcion
        WHEN Excep1 THEN DBMS_OUTPUT.PUT_LINE( 'InsertarClases2 - Ya
        existe una clase con ese codigo.' );
END;
```

Declaro dos variables locales, `x` y `Excep1`. La variable `x` la uso para almacenar el número de clases que hay con el código de clase igual a `cod2`, para ello, uso la **sentencia** SQL para recuperar el número de tuplas que tienen de código de clase el introducido por parámetro de entrada. Después, compruebo que no haya ninguna clase con ese código de clase y eso indicará que se permite insertar esta nueva clase y que no salte la excepción. En el caso de que hubiese una clase con ese código, saltaría la excepción `Excep1` que hemos creado previamente y mostraría un mensaje que informaría acerca de que existe una clase ya con ese código.

Se adjunta una imagen de lo que saldría por consola al saltar la excepción:



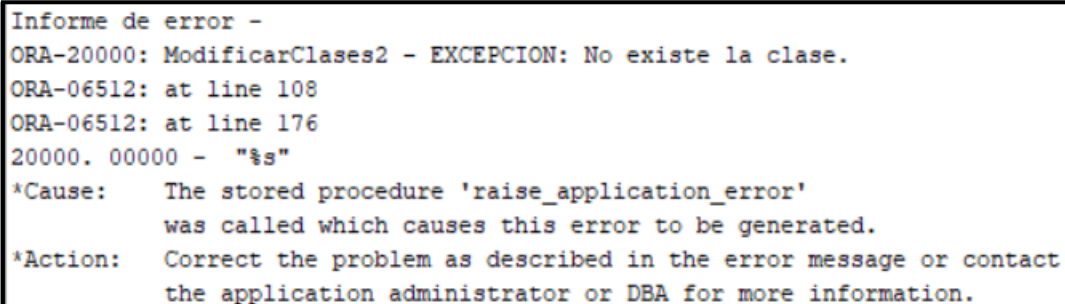
```
InsertarClases2 - Ya existe una clase con ese codigo.
```


ModificarClases2()

```
PROCEDURE ModificarClases2(cod2 NUMBER, nombre2 VARCHAR2, ingresos2
NUMBER) IS
    Cont NUMBER;
BEGIN
    SELECT COUNT(*) INTO Cont FROM CLASES WHERE COD=cod2;
    IF (Cont=0) THEN
        RAISE_Application_Error( -20000, 'ModificarClases2 -
EXCEPCION: No existe la clase.' );
    END IF;
    DBMS_OUTPUT.PUT_LINE( 'SE VA A MODIFICAR UNA CLASE' );
    UPDATE CLASES
    SET NOMBRE = nombre2, INGRESOS=ingresos2
    WHERE COD=cod2;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE( 'SE HA MODIFICADO UNA CLASE' );
    DBMS_OUTPUT.PUT_LINE( '-----' );
    DBMS_OUTPUT.PUT_LINE( '-----' );
END;
```

Declaro la variable local (**Cont**) antes del BEGIN de tipo numérico para almacenar el número de clases que hay con el código de clase que se introduce por parámetros (**cod2**). Después hago un IF comparando el valor almacenado en **Cont** con 0, con eso me aseguro de que si no hay una clase con ese código, salto la excepción No_Data_Found y se mostraría el mensaje de que el código no existe.

Se adjunta una imagen de lo que saldría por consola al saltar la excepción:



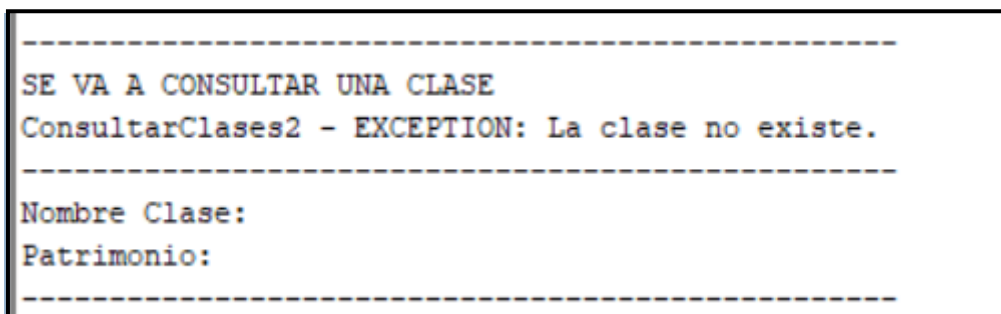
```
Informe de error -
ORA-20000: ModificarClases2 - EXCEPCION: No existe la clase.
ORA-06512: at line 108
ORA-06512: at line 176
20000. 00000 - "%s"
*Cause:      The stored procedure 'raise_application_error'
              was called which causes this error to be generated.
*Action:     Correct the problem as described in the error message or contact
              the application administrator or DBA for more information.
```

ConsultarClases2()

```
PROCEDURE ConsultarClases2(cod2 NUMBER, nombre2 OUT VARCHAR2,
ingresos2 OUT NUMBER) IS
BEGIN
    DBMS_OUTPUT.PUT_LINE( 'SE VA A CONSULTAR UNA CLASE' );
    SELECT NOMBRE, INGRESOS INTO nombre2, ingresos2 FROM CLASES where
COD=cod2;
    --DBMS_OUTPUT.PUT_LINE( 'Nombre de la clase: ' || nombre2 );
    --DBMS_OUTPUT.PUT_LINE( 'Ingresos: ' || ingresos2 );
    COMMIT;
    DBMS_OUTPUT.PUT_LINE( 'INFORMACION DE LA CLASE ALMACENADA EN LAS
VARIABLES' );
    EXCEPTION
        WHEN No_Data_Found THEN DBMS_OUTPUT.PUT_LINE(
'ConsultarClases2 - EXCEPTION: La clase no existe.' );
        DBMS_OUTPUT.PUT_LINE( '-----' );
END;
```

Se ejecuta el método y en el caso de que no exista una clase con código cod2, saltaría la excepción de abajo y mostraría que la clase no existe.

Se adjunta una imagen de lo que saldría por consola al saltar la excepción:



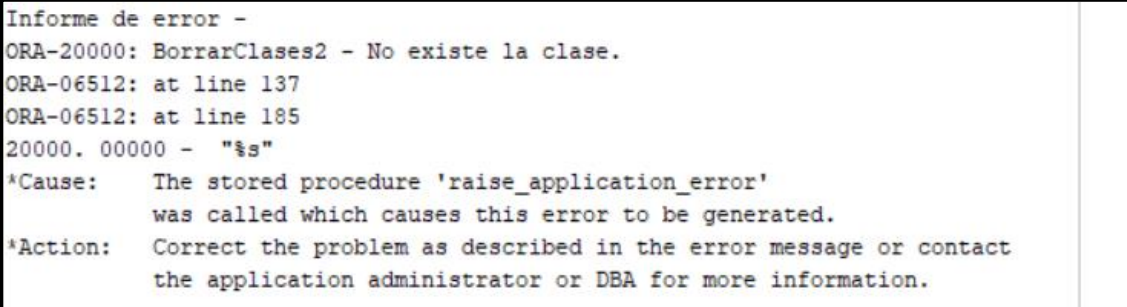
```
-----
SE VA A CONSULTAR UNA CLASE
ConsultarClases2 - EXCEPTION: La clase no existe.
-----
Nombre Clase:
Patrimonio:
-----
```

BorrarClases2()

```
PROCEDURE BorrarClases2(cod2 NUMBER) IS
aux2 NUMBER;
BEGIN
    SELECT COUNT(*) INTO aux2 FROM CLASES WHERE COD=cod2;
    IF(aux2=0) THEN
        RAISE_Application_Error(-20000, 'BorrarClases2 - No existe la
clase. ');
    END IF;
    DBMS_OUTPUT.PUT_LINE('BORRADO DE LA CLASE');
    DELETE FROM CLASES WHERE COD=cod2;
    COMMIT;
    DBMS_OUTPUT.PUT_LINE('CLASE BORRADA');
    DBMS_OUTPUT.PUT_LINE('-----');
    -----' );
END;
```

Declaro la variable local **aux2** para almacenar con la **sentencia SQL** el número de clases que hay con ese código de clase y en el caso de que no hubiese saltaría la **excepción** de tipo **Application_Error**.

Se adjunta una imagen de lo que saldría por consola al saltar la excepción:



```
Informe de error -
ORA-20000: BorrarClases2 - No existe la clase.
ORA-06512: at line 137
ORA-06512: at line 185
20000. 00000 - "%s"
*Cause:      The stored procedure 'raise_application_error'
              was called which causes this error to be generated.
*Action:      Correct the problem as described in the error message or contact
              the application administrator or DBA for more information.
```

SALIDA POR CONSOLA

Se comentan los métodos con excepciones para que no se termine la ejecución. Esta es la salida de los métodos sin excepciones:

SE VA A INSERTAR UNA CLASE

CLASE INSERTADA

INFORMACION DE LAS CLASES

Codigo clase: 1

Nombre clase: Nobleza

Ingresos: 0

Codigo clase: 2

Nombre clase: Clero

Ingresos: 0

Codigo clase: 3

Nombre clase: Artesanos

Ingresos: 0

Codigo clase: 4

Nombre clase: Campesinos

Ingresos: 0

Codigo clase: 0

Nombre clase: Realeza

Ingresos: 0

INFORMACION MOSTRADA

SE VA A MODIFICAR UNA CLASE

SE HA MODIFICADO UNA CLASE

INFORMACION DE LAS CLASES

Codigo clase: 1

Nombre clase: Nobleza

Ingresos: 0

Codigo clase: 2

Nombre clase: Clero

Ingresos: 0

Codigo clase: 3

Nombre clase: Artesanos

Ingresos: 0

Codigo clase: 4

Nombre clase: Campesinos

Ingresos: 0

Codigo clase: 0

Nombre clase: Desconocido

Ingresos: 0

INFORMACION MOSTRADA

SE VA A CONSULTAR UNA CLASE

INFORMACION DE LA CLASE ALMACENADA EN LAS VARIABLES

Nombre Clase: Desconocido

Patrimonio: 0

BORRADO DE LA CLASE

CLASE BORRADA

INFORMACION DE LAS CLASES

Codigo clase: 1

Nombre clase: Nobleza

Ingresos: 0

Codigo clase: 2

Nombre clase: Clero

Ingresos: 0

Codigo clase: 3

Nombre clase: Artesanos

Ingresos: 0

Codigo clase: 4

Nombre clase: Campesinos

Ingresos: 0

INFORMACION MOSTRADA

Fin de programa

Procedimiento PL/SQL terminado correctamente.