PAC de desarrollo UF3

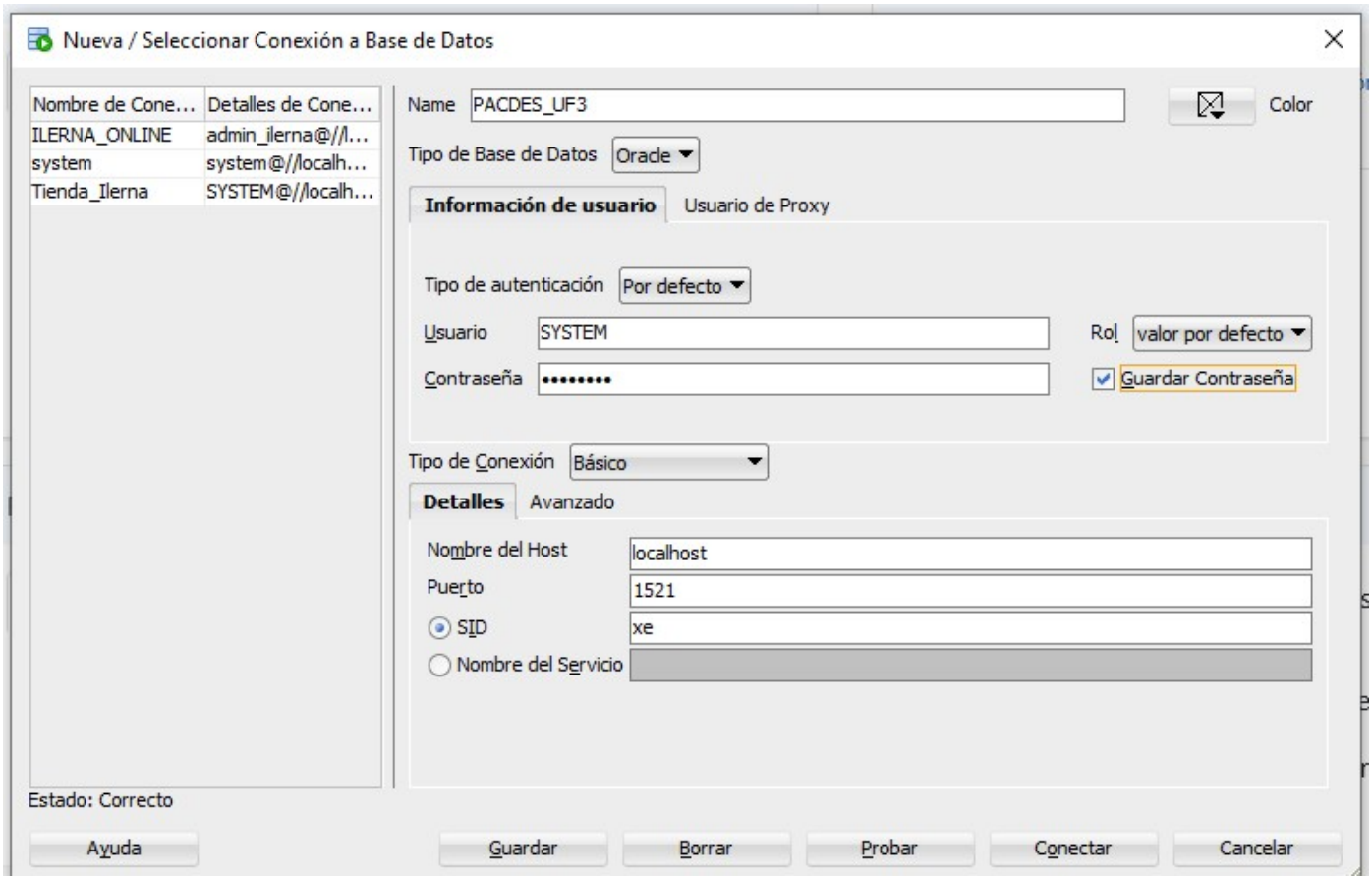
Base de Datos B

Sonia Ujaque Ortiz

Índice Página

CONFIGURACIÓN INICIAL

Primero creamos la nueva conexión **PACDES_UF3** de la siguiente forma utilizando el usuario **SYSTEM**:



Después ejecutamos el **script** que nos han proporcionado y se nos creará el usuario ILERNA_PAC con todos los privilegios y todas las tablas que vamos a necesitar para los ejercicios

1. GESTIÓN DE USUARIOS Y ROLES

En la siguiente imagen creo el “**ROL_GESTOR**”. Con “**GRANT CREATE SESSION**” le damos el permiso para crear la conexión y la siguiente instrucción es para darle permiso de modificar la estructura (**ALTER**), insertar y modificar registros (**INSERT**, **UPDATE**) y seleccionar (**SELECT**). Como se puede ver en la salida se crean correctamente.

Hoja de Trabajo | Generador de Consultas

```
ALTER SESSION SET "_ORACLE_SCRIPT" = true;

SET SERVEROUTPUT ON

CONN ILERNA_PAC / i1234;

-------------------------------------------------------------
-- 1)   GESTIÓN DE USUARIOS Y ROLES -------------------------
-------------------------------------------------------------
-- CREAR ROLE "ROL_GESTOR"

CREATE ROLE rol_gestor;
GRANT CREATE SESSION TO rol_gestor;
GRANT ALTER, INSERT, UPDATE, SELECT ON asignaturas_pac TO rol_gestor;

-- CREAR USUARIO "GESTOR"
CREATE USER GESTOR IDENTIFIED BY g1234;
-- ASIGNAR ROL A USUARIO
GRANT rol_gestor TO gestor;
-- CONECTAR CON EL NUEVO USARIO
CONN GESTOR / g1234;

-- REALIZAR LAS MODIFICACIONES DEL EJERCICIO
--Elimino campo creditos
ALTER TABLE asignaturas_pac DROP COLUMN creditos;
--Añadir campo ciclo
ALTER TABLE asignaturas_pac ADD ciclo VARCHAR(3);
--Insertar registro
INSERT INTO asignaturas_pac VALUES ('DAX_M02B','MP2. Bases de datos B','Emilio Saurina','DAX');
```

Salida de Script | Tarea terminada en 0,043 segundos

```
Conectado.
La conexión creada por el comando de script CONNECT se ha desconectado

Role ROL_GESTOR creado.


Grant correcto.


Grant correcto.
```

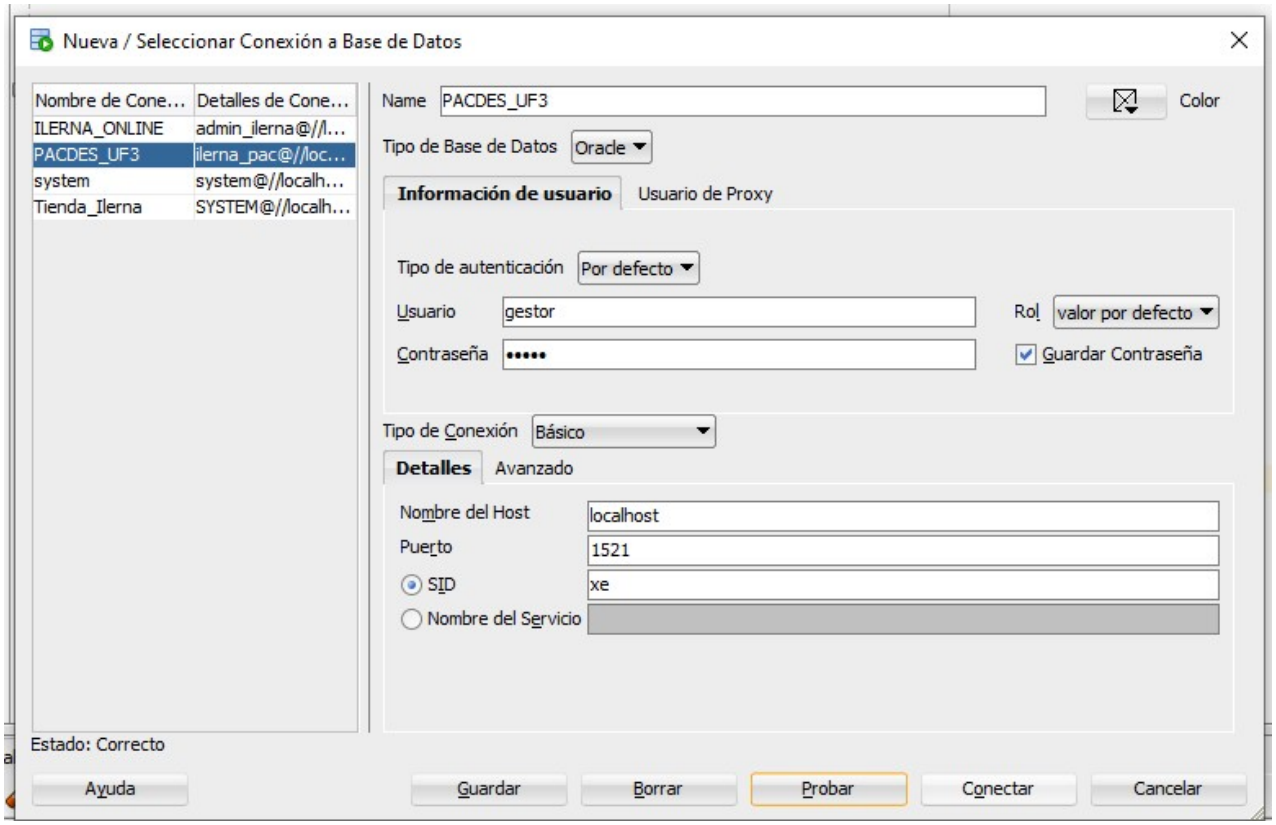
Creo el usuario **GESTOR**, le asigno el rol **ROL_GESTOR** y conectamos con el usuario **GESTOR**

```
-- CREAR USUARIO "GESTOR"
CREATE USER GESTOR IDENTIFIED BY gl234;
-- ASIGNAR ROL A USUARIO
GRANT rol_gestor TO gestor;
-- CONECTAR CON EL NUEVO USARIO
CONN GESTOR / gl234;

-- REALIZAR LAS MODIFICACIONES DEL EJERCICIO
--Elimino campo creditos
ALTER TABLE asignaturas_pac DROP COLUMN creditos;
--Añadir campo ciclo
ALTER TABLE asignaturas_pac ADD ciclo VARCHAR(3);
--Insertar registro
INSERT INTO asignaturas_pac VALUES ('DAX_M02B','MP2. Bases de datos B','Emilio Saurina','DAX');
```

Salida de Script

Tarea terminada en 0,092 segundos

```
User GESTOR creado.


Grant correcto.

Conectado.
La conexión creada por el comando de script CONNECT se ha desconectado
```



En las siguientes sentencias, eliminamos el campo **CREDITOS** y le añadimos el campo **CICLO**

Insertamos un registro en la tabla **asignaturas_pac**. La primera vez, como se puede ver, me ha dado un error porque en el **nombre_asignatura** había un carácter más de los permitidos, por lo que le he quitado un espacio y ya me ha dejado introducirlo. Otra opción es cambiar el **VARCHAR** de **nombre_asignatura** introduciendo que pueda tener más caracteres.

En la última sentencia modificamos el dato del campo **ciclo** por **DAM** no le pongo el **WHERE** porque solo hay un registro y si hubiera más en el ciclo serian todos los registros iguales **DAM**

Pongo delante de la tabla **ilerna_pac** porque si no, no me encontraba la tabla ya que **ilerna_pac** es el usuario que la ha creado

Después volvemos a conectar con el usuario **ilerna_pac**

```
-- REALIZAR LAS MODIFICACIONES DEL EJERCICIO
--Elimino campo creditos
ALTER TABLE ilerna_pac.asignaturas_pac DROP COLUMN creditos;
--Añadir campo ciclo
ALTER TABLE ilerna_pac.asignaturas_pac ADD ciclo VARCHAR(3);
--Insertar registro
INSERT INTO ilerna_pac.asignaturas_pac VALUES ('DAX_M02B','MP2.Bases de datos B','Emilio Saurina','DAX');
--Modificar dato de Ciclo
UPDATE ilerna_pac.asignaturas_pac SET ciclo = 'DAM';

SHOW USER;

-- CONECTAR DE NUEVO CON EL USUARIO ILERNA_PAC
CONN ILERNA_PAC / il234;
```

```
Salida de Script
Tarea terminada en 0,033 segundos

Table ILERNA_PAC.ASIGNATURAS_PAC alterado.


Table ILERNA_PAC.ASIGNATURAS_PAC alterado.


Error que empieza en la línea: 29 del comando :
INSERT INTO ilerna_pac.asignaturas_pac VALUES ('DAX_M02B','MP2. Bases de datos B','Emilio Saurina','DAX')
Informe de error -
ORA-12899: el valor es demasiado grande para la columna "ILERNA_PAC"."ASIGNATURAS_PAC"."NOMBRE_ASIGNATURA" (real: 21, máximo: 20)


1 fila insertadas.


1 fila actualizadas.
```

Vemos que los cambios se han realizado correctamente:

Ordenar... | Filtrar:

	ID_ASIGNATURA	NOMBRE_ASIGNATURA	NOMBRE_PROFESOR	CICLO
1	DAX_M02B	MP2.Bases de datos B	Emilio Saurina	DAM

2. PROCEDIMIENTO

```
-- CREAR PROCEDIMIENTO "RANKING_JUGADOR"
CREATE OR REPLACE PROCEDURE ranking_jugador (
    id_jugad IN jugadores_pac.id_jugador%TYPE,
    num_puntos IN jugadores_pac.puntos%TYPE,
    nombre_par OUT jugadores_pac.nombre%TYPE,
    apellido_par OUT jugadores_pac.apellidos%TYPE,
    puntos_totales_par OUT jugadores_pac.puntos%TYPE,
    ranking_act OUT ranking_pac.nombre_ranking%TYPE
) IS
    valor_nombre     jugadores_pac.nombre%TYPE;
    valor_apellido   jugadores_pac.apellidos%TYPE;
    valor_puntos     jugadores_pac.puntos%TYPE;
    ranking          ranking_pac.nombre_ranking%TYPE;
BEGIN

SELECT
    nombre,
    apellidos,
    puntos
INTO
  valor_nombre,
  valor_apellido,
  valor_puntos

FROM
    jugadores_pac
WHERE
    id_jugador = id_jugad;

    valor_puntos:=valor_puntos + num_puntos;
    nombre_par:=valor_nombre;
    apellido_par:=valor_apellido;
    puntos_totales_par:=valor_puntos;
IF valor_puntos<=1000
THEN ranking := 'Bronce';
ranking_act:=ranking;

ELSIF valor_puntos<=2000
THEN ranking :='Plata';
ranking_act:=ranking;

ELSIF valor_puntos<=3000
THEN ranking :='Oro';
ranking_act:=ranking;

ELSIF valor_puntos<=4000
THEN ranking :='Platino';
ranking_act:=ranking;

ELSIF valor_puntos<=99999
THEN ranking:='Diamante';
ranking_act:=ranking;

END IF;

 END ranking_jugador;
/
```

Salida de Script

Tarea terminada en 0,296 segundos

```
Procedure RANKING_JUGADOR compilado
```

En el procedimiento que he creado **RANKING_JUGADOR**, le pongo los parámetros de entrada **id_jugad** y **num_puntos** para después poderlos introducir en el bloque anónimo, también le pongo parámetros de salida para que me saque el nombre, apellido, puntos y ranking y poderlos utilizar después en el bloque e imprimir los datos que necesitamos. Creo 4 variables para poder guardar, el nombre, apellido, los puntos y el ranking.

Hago un **SELECT** para encontrar el nombre, apellido y puntos de la tabla **jugadores_pac** y guardarlos con el **INTO** en las variables que he creado del jugador cuyo **id** coincida con el que hemos introducido.

Después en la variable **valor_puntos** que es donde nos ha guardado los puntos que tiene el jugador le digo que es igual a la suma de la misma variable con la variable **num_puntos** que va a ser donde introducimos los nuevos puntos. Después para los parámetros de salida les asigno las variables donde se ha guardado cada dato (nombre, apellido y puntos totales)

Para saber en qué **ranking** se encuentran esos puntos ya sumados, hago un **IF ELSIF** para que se guarde en la variable ranking a que ranking pertenece comparando los puntos totales con los puntos máximos que tiene cada ranking y después se lo asigno al parámetro de salida **ranking_act** para poder utilizarla en el bloque y hago lo mismo para cada caso.

3. FUNCIÓN

```
-- CREAR FUNCION "JUGADORES_POR_RANKING"
CREATE OR REPLACE FUNCTION jugadores_por_ranking (
    nom_ranking IN ranking_pac.nombre_ranking%TYPE
)RETURN INT
AS
total_jugadores INT;
valor_puntos_min jugadores_pac.puntos%TYPE;
valor_puntos_max jugadores_pac.puntos%TYPE;
BEGIN
    SELECT
    puntos_max,
    puntos_min
INTO
  valor_puntos_max,
  valor_puntos_min

FROM
    ranking_pac
WHERE
    nombre_ranking=nom_ranking;

    SELECT COUNT(*)INTO total_jugadores FROM jugadores_pac WHERE puntos<=valor_puntos_max AND puntos>=valor_puntos_min;

RETURN total_jugadores;
END jugadores_por_ranking;
/
```

```
Salida de Script
Tarea terminada en 0,047 segundos

Function JUGADORES_POR_RANKING compilado
```

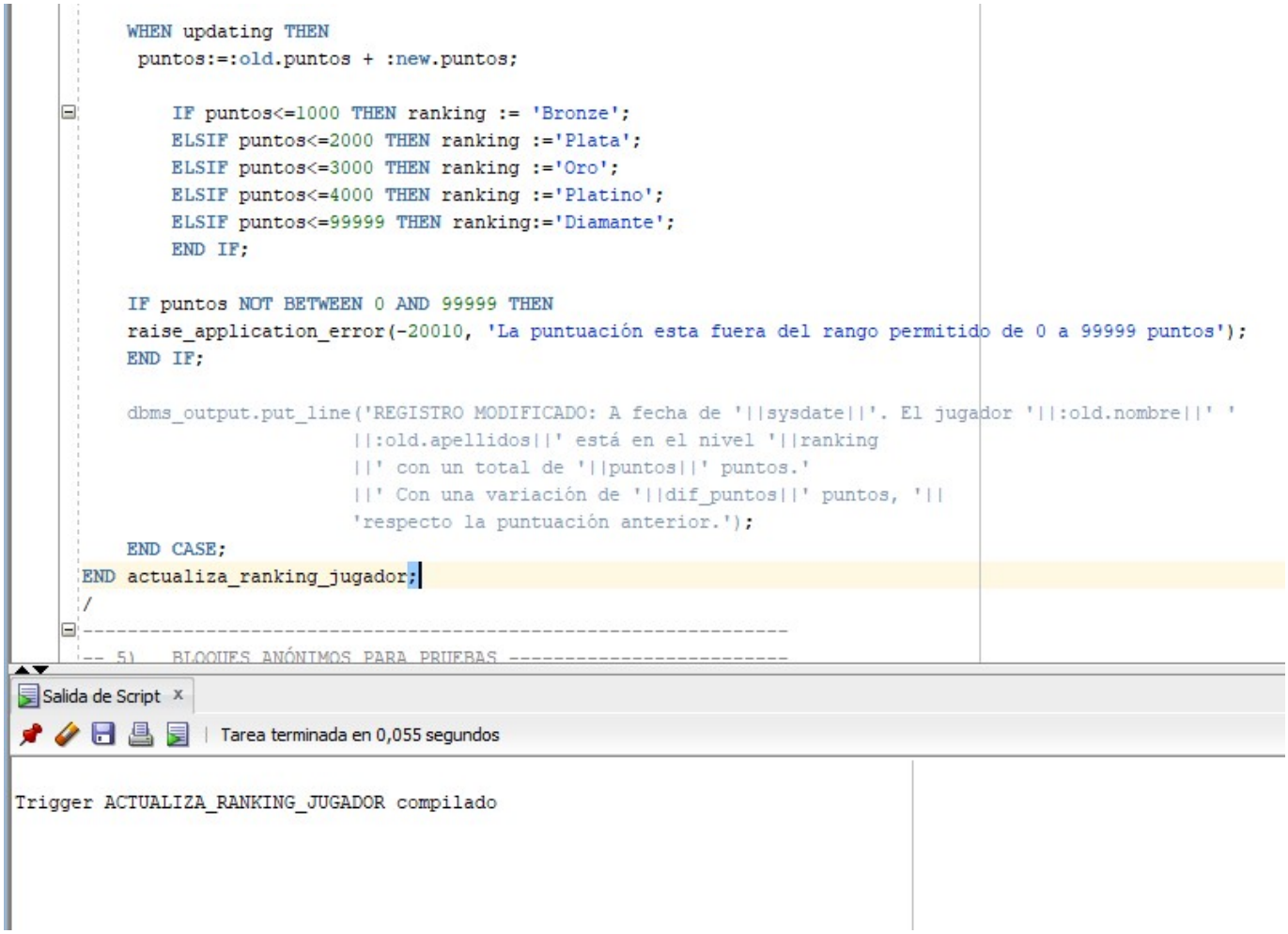
Para la función **jugadores_por_ranking**, he puesto un parámetro **nom_ranking** para entrar el nombre del ranking que queremos. He creado 3 variables para almacenar el valor de puntos mínimo, el valor de puntos máximos y el total de jugadores.

El primer **SELECT** es para almacenar los puntos máximos y mínimos en las variables creadas **valor_puntos_max**, **valor_puntos_min** con el **INTO**, los cuales el **nombre_ranking** debe ser igual al **ranking** que se le ha pasado por parámetro **nom_ranking**.

En el segundo **SELECT** con el comando **COUNT (*)** vamos a contar todos los jugadores de la tabla **jugadores_pac** que tengan el rango de puntos entre el mínimo y el máximo que se ha almacenado en las variables del **SELECT** anterior, en las cuales se habrán guardado el rango de puntos del nombre del ranking que le hemos pasado por parámetro, lo guardamos con **INTO** en la variable **total_jugadores** y con el **RETURN** nos va a devolver el número total de jugadores del ranking que elijamos.

4. TRIGGER

```
-- CREAR TRIGGER "CAMBIO_RANKING_JUGADOR"
CREATE OR REPLACE TRIGGER actualiza_ranking_jugador
AFTER INSERT OR UPDATE ON jugadores_pac
FOR EACH ROW
DECLARE
puntos      jugadores_pac.puntos%TYPE;
ranking     ranking_pac.nombre_ranking%TYPE;
dif_puntos  jugadores_pac.puntos%TYPE;

BEGIN
   puntos:= :new.puntos;
   dif_puntos:= :new.puntos - :old.puntos;

    CASE
    WHEN inserting THEN
         IF puntos<=1000 THEN ranking := 'Bronze';
         ELSIF puntos<=2000 THEN ranking :='Plata';
         ELSIF puntos<=3000 THEN ranking :='Oro';
         ELSIF puntos<=4000 THEN ranking :='Platino';
         ELSIF puntos<=99999 THEN ranking:='Diamante';
         END IF;

    dbms_output.put_line('REGISTO INSERTADO: A fecha de '||sysdate||'. El jugador '||:new.nombre||' '
                         ||:new.apellidos||' está en el nivel '||ranking
                         ||' con un total de '||:new.puntos||' puntos');
```



```
    WHEN updating THEN
     puntos:=:old.puntos + :new.puntos;

        IF puntos<=1000 THEN ranking := 'Bronze';
        ELSIF puntos<=2000 THEN ranking :='Plata';
        ELSIF puntos<=3000 THEN ranking :='Oro';
        ELSIF puntos<=4000 THEN ranking :='Platino';
        ELSIF puntos<=99999 THEN ranking:='Diamante';
        END IF;

    IF puntos NOT BETWEEN 0 AND 99999 THEN
    raise_application_error(-20010, 'La puntuación esta fuera del rango permitido de 0 a 99999 puntos');
    END IF;

    dbms_output.put_line('REGISTRO MODIFICADO: A fecha de '||sysdate||'. El jugador '||:old.nombre||' '
                         ||:old.apellidos||' está en el nivel '||ranking
                         ||' con un total de '||puntos||' puntos.'
                         ||' Con una variación de '||dif_puntos||' puntos, '||
                         'respecto la puntuación anterior.');
    END CASE;
END actualiza_ranking_jugador;
/
------------------------------------------------------------
-- 5)   BLOQUES ANÓNIMOS PARA PRUEBAS -------------------------
```

Salida de Script ×

Tarea terminada en 0,055 segundos

```
Trigger ACTUALIZA_RANKING_JUGADOR compilado
```

Para el **TRIGGER actualizar_ranking_jugador** he puesto que se lance después de insertar o modificar un dato con **AFTER**, ya que si lo hago antes de insertar los datos todavía no estarán en la tabla.

He puesto la instrucción **FOR EACH ROW** para que lo haga por cada sentencia que encuentre

Declaro 3 variables para poder almacenar los puntos que se introducen, para guardar el ranking de la puntuación y otra para calcular la diferencia de puntos con los que ya tenía.

He hecho un **CASE** para que haga según sea un **INSERT** o un **UPDATE**. Dentro del caso **inserting** como el de **updating** comparo los puntos para saber que ranking le corresponde e imprimo el mensaje por pantalla con la información, en el caso del **INSERT** con los datos nuevos **:NEW** y en el caso del **UPDATE** con los datos antiguos **:OLD**, excepto los puntos, que los introducimos en ese momento. En el caso del **updating** los puntos van a ser los antiguos más los nuevos que se van a introducir.

Con un **IF** verifico que la puntuación este entre 0 y 99999, de lo contrario saltará el error por pantalla.

He utilizado la instrucción **sysdate** para imprimir la fecha.

5. BLOQUES ANONIMOS PARA PRUEBAS

5.1. COMPROBACIÓN GESTIÓN DE USUARIOS Y ROLES

```
-- COMPROBACIÓN GESTIÓN USUARIOS Y ROLES
EXECUTE dbms_output.put_line('-- COMPROBACIÓN GESTIÓN USUARIOS Y ROLES --');
DECLARE
    id_asig     asignaturas_pac.id_asignatura%TYPE;
    nombre_asig asignaturas_pac.nombre_asignatura%TYPE;
    nombre_profe    asignaturas_pac.nombre_profesor%TYPE;
BEGIN
    id_asig:='DAX_M02B';

SELECT
    nombre_asignatura,
    nombre_profesor
INTO
    nombre_asig,
    nombre_profe
FROM
    asignaturas_pac
WHERE
    id_asignatura=id_asig;
    dbms_output.put_line('El profesor de '||nombre_asig
                         ||' se llama '||nombre_profe);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('ID asignatura no existente');

END;

/
-- COMPROBACIÓN DE TRIGGER CAMBIO_RANKING_JUGADOR
```

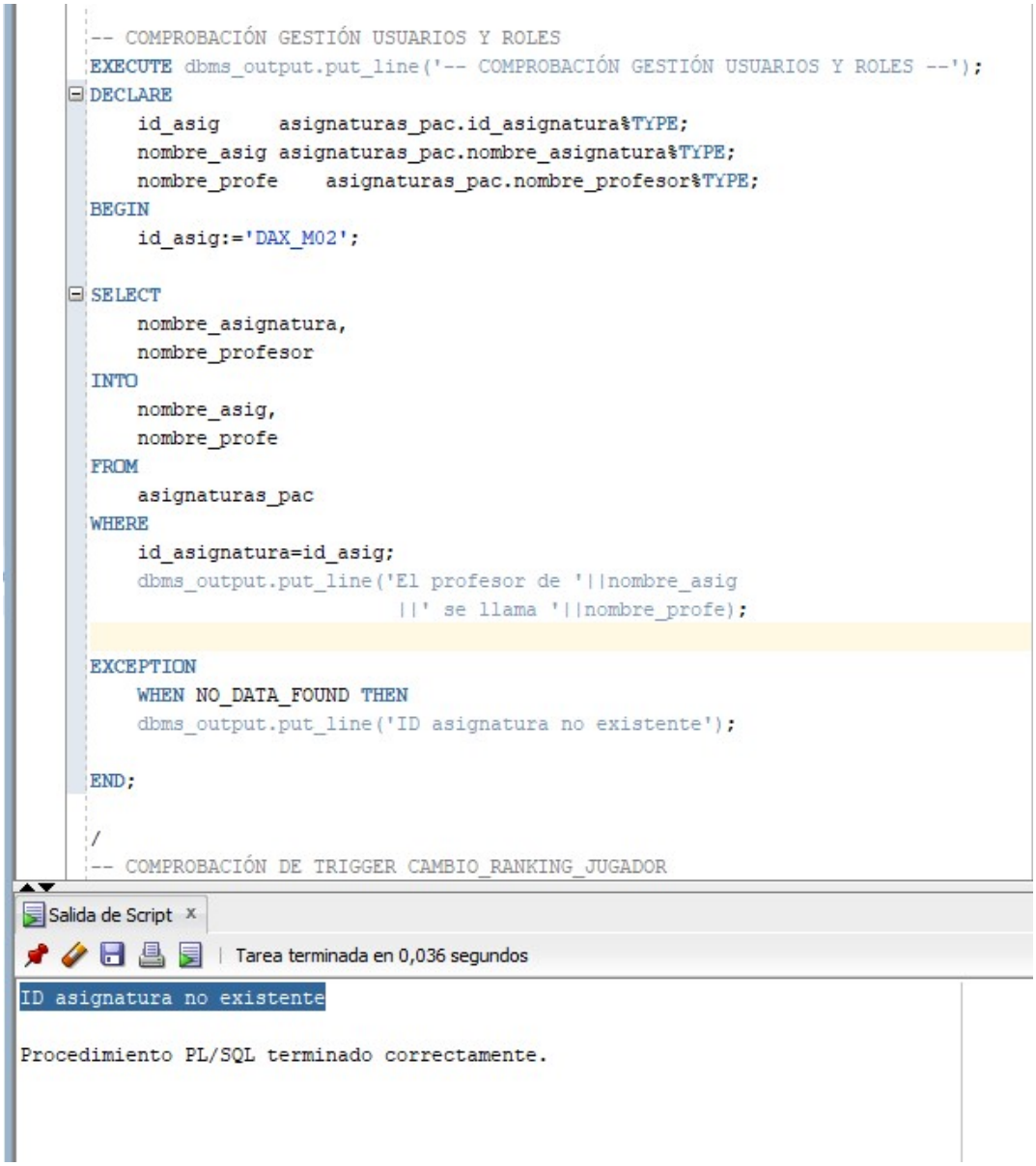
```
Salida de Script x
Tarea terminada en 0,037 segundos
El profesor de MP2.Bases de datos B se llama Emilio Saurina

Procedimiento PL/SQL terminado correctamente.
```

Para mostrar por pantalla los datos de la asignatura con el bloque anónimo, he creado 3 variables para almacenar el **id_asignatura** en **id_asig**, **nombre_asignatura** en **nombre_asig** y el **nombre_profesor** en **nombre_profe**.

Con el **SELECT** almaceno los datos en las variables que he creado donde coinciden el **id_asignatura** con el **id** que pasamos por parámetro e imprimo por pantalla el mensaje concatenado con esas variables.

Creo la **EXCEPTION** para que si no encuentra el dato que introducimos nos lance un mensaje de que el **id** de la asignatura no existe.

Podemos ver la excepción en la siguiente imagen:



```
-- COMPROBACIÓN GESTIÓN USUARIOS Y ROLES
EXECUTE dbms_output.put_line('-- COMPROBACIÓN GESTIÓN USUARIOS Y ROLES --');
DECLARE
    id_asig     asignaturas_pac.id_asignatura%TYPE;
    nombre_asig asignaturas_pac.nombre_asignatura%TYPE;
    nombre_profe    asignaturas_pac.nombre_profesor%TYPE;
BEGIN
    id_asig:='DAX_M02';

SELECT
    nombre_asignatura,
    nombre_profesor
INTO
    nombre_asig,
    nombre_profe
FROM
    asignaturas_pac
WHERE
    id_asignatura=id_asig;
    dbms_output.put_line('El profesor de '||nombre_asig
                         ||' se llama '||nombre_profe);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('ID asignatura no existente');

END;

/
-- COMPROBACIÓN DE TRIGGER CAMBIO_RANKING_JUGADOR
```

Salida de Script x

Tarea terminada en 0,036 segundos

```
ID asignatura no existente

Procedimiento PL/SQL terminado correctamente.
```

5.2. COMPROBACIÓN DE TRIGGER ACTUALIZAR_RANKING_JUGADOR

Para la comprobación del **TRIGGER** he creado un bloque anónimo donde declaro 3 variables para poder entrar los nuevos puntos por pantalla, para poder almacenar el id del jugador que queremos modificar y otra variable para comprobar si ese id que introducimos para modificar el registro existe en la tabla

Se realiza el **INSERT** y seguidamente se piden los puntos por pantalla y en la variable **n_id_jugador** almaceno el id del jugador que quiero modificar

Con un **SELECT** hago que me devuelva el id y me lo almacene en la variable **id_comprobar** de este modo si no encuentra ese **id** en la tabla, me saltará la excepción con el mensaje de que no se ha encontrado al jugador.

Si el jugador existe y la suma de los puntos está dentro del rango permitido, se ejecutará el **UPDATE** e imprimirá la frase con la información junto con la del **INSERT**. Como vemos a continuación:

```
/
-- COMPROBACIÓN DE TRIGGER CAMBIO_RANKING_JUGADOR
EXECUTE dbms_output.put_line('-- COMPROBACIÃ"N DE TRIGGER â€œCAMBIO_RANKING_JUGADOR --');
DECLARE

nuevos_puntos   jugadores_pac.puntos%TYPE;
n_id_jugador    jugadores_pac.id_jugador%TYPE;
id_comprobar    jugadores_pac.id_jugador%TYPE;
BEGIN

INSERT INTO jugadores_pac VALUES(11,'Pepito','Ortiz Pérez',0);

nuevos_puntos:=&puntos;
n_id_jugador:=11;

SELECT jugadores_pac.id_jugador INTO id_comprobar FROM jugadores_pac WHERE n_id_jugador=jugadores_pac.id_jugador;
UPDATE jugadores_pac set jugadores_pac.puntos=puntos+nuevos_puntos WHERE n_id_jugador=jugadores_pac.id_jugador;

    EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('El jugador con id '||n_id_jugador||' no existe en la base de datos');
END;


/
-- COMPROBACIÓN DEL PROCEDIMIENTO RANKING_JUGADOR
```

Salida de Script

Tarea terminada en 8,566 segundos

```
SELECT jugadores_pac.id_jugador INTO id_comprobar FROM jugadores_pac WHERE n_id_jugador=jugadores_pac.id_jugador;
UPDATE jugadores_pac set jugadores_pac.puntos=puntos+nuevos_puntos WHERE n_id_jugador=jugadores_pac.id_jugador;

    EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('El jugador con id '||n_id_jugador||' no existe en la base de datos');
END;
REGISTO INSERTADO: A fecha de 20/11/22. El jugador Pepito Ortiz Pérez está en el nivel Bronze con un total de 0 puntos
REGISTRO MODIFICADO: A fecha de 20/11/22. El jugador Pepito Ortiz Pérez está en el nivel Oro con un total de 3000 puntos. Con una variación

Procedimiento PL/SQL terminado correctamente.
```

En caso de que la suma de los puntos este fuera de los rangos permitidos nos saldrá el siguiente error:

```
-- COMPROBACIÓN DE TRIGGER CAMBIO_RANKING_JUGADOR
EXECUTE dbms_output.put_line('-- COMPROBACIÃ"N DE TRIGGER â€œCAMBIO_RANKING_JUGADOR --');
DECLARE

nuevos_puntos   jugadores_pac.puntos%TYPE;
n_id_jugador    jugadores_pac.id_jugador%TYPE;
id_comprobar    jugadores_pac.id_jugador%TYPE;
BEGIN

INSERT INTO jugadores_pac VALUES(11,'Pepito','Ortiz Pérez',0);

nuevos_puntos:=&puntos;
n_id_jugador:=11;

SELECT jugadores_pac.id_jugador INTO id_comprobar FROM jugadores_pac WHERE n_id_jugador=jugadores_pac.id_jugador;
UPDATE jugadores_pac set jugadores_pac.puntos=puntos+nuevos_puntos WHERE n_id_jugador=jugadores_pac.id_jugador;

    EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('El jugador con id '||n_id_jugador||' no existe en la base de datos');
END;
```

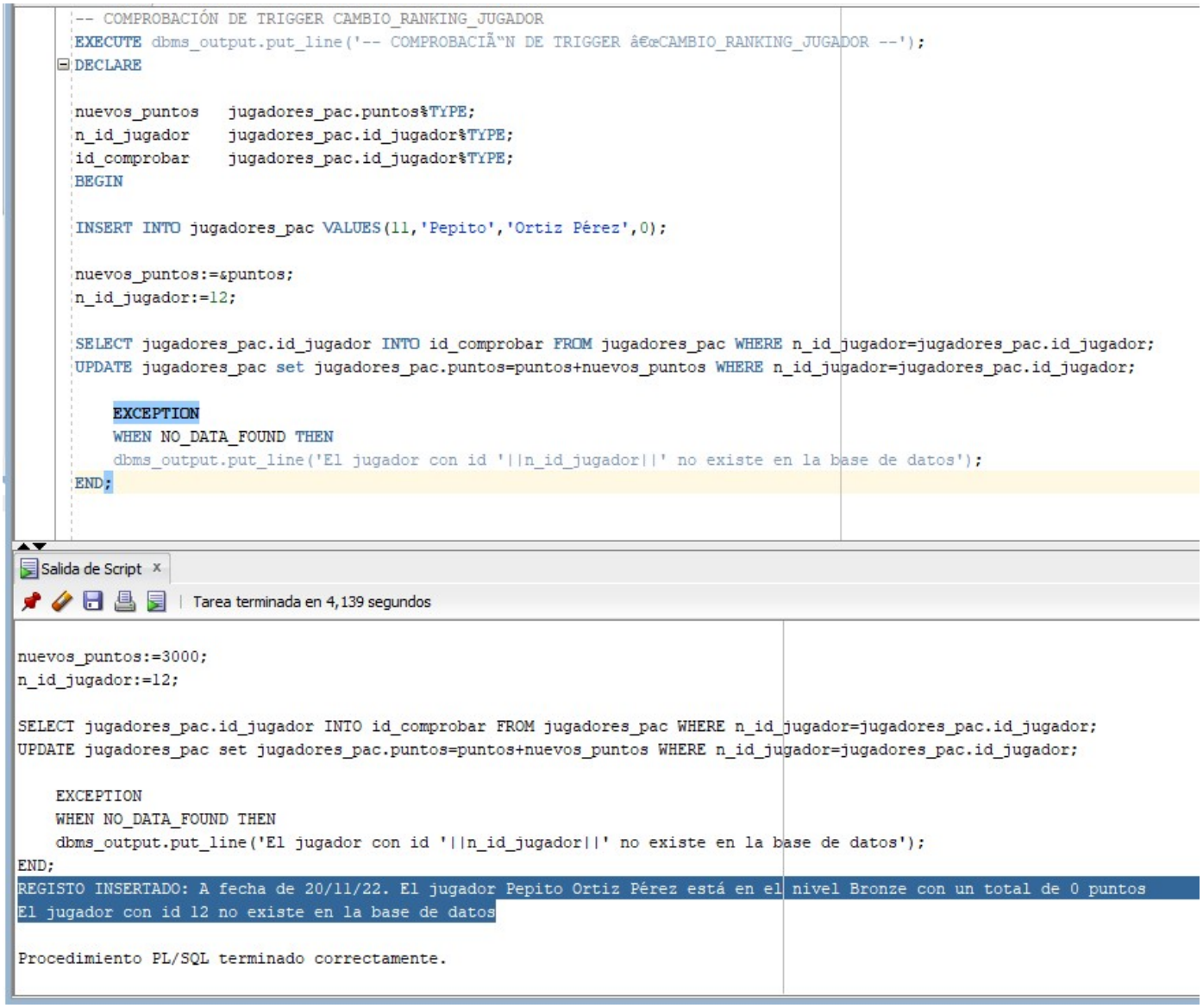
Salida de Script

Tarea terminada en 5,469 segundos

```
nuevos_puntos:=100000;
n_id_jugador:=11;

SELECT jugadores_pac.id_jugador INTO id_comprobar FROM jugadores_pac WHERE n_id_jugador=jugadores_pac.id_jugador;
UPDATE jugadores_pac set jugadores_pac.puntos=puntos+nuevos_puntos WHERE n_id_jugador=jugadores_pac.id_jugador;

    EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('El jugador con id '||n_id_jugador||' no existe en la base de datos');
END;
Informe de error -
ORA-20010: La puntuación esta fuera del rango permitido de 0 a 99999 puntos
ORA-06512: en "ILERNA_PAC.ACTUALIZA_RANKING_JUGADOR", línea 34
ORA-04088: error durante la ejecución del disparador 'ILERNA_PAC.ACTUALIZA_RANKING_JUGADOR'
ORA-06512: en línea 14
```

En el caso que el **id** del jugador que introducimos no exista nos aparece el siguiente error:

Me ejecuta el **INSERT** y me aparece el mensaje y para el **UPDATE** me dice que el jugador con el id 12 no existe.



```
-- COMPROBACIÓN DE TRIGGER CAMBIO_RANKING_JUGADOR
EXECUTE dbms_output.put_line('-- COMPROBACIÃ"N DE TRIGGER â€œCAMBIO_RANKING_JUGADOR --');
DECLARE

nuevos_puntos   jugadores_pac.puntos%TYPE;
n_id_jugador    jugadores_pac.id_jugador%TYPE;
id_comprobar    jugadores_pac.id_jugador%TYPE;
BEGIN

INSERT INTO jugadores_pac VALUES(11,'Pepito','Ortiz Pérez',0);

nuevos_puntos:=&puntos;
n_id_jugador:=12;

SELECT jugadores_pac.id_jugador INTO id_comprobar FROM jugadores_pac WHERE n_id_jugador=jugadores_pac.id_jugador;
UPDATE jugadores_pac set jugadores_pac.puntos=puntos+nuevos_puntos WHERE n_id_jugador=jugadores_pac.id_jugador;

    EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('El jugador con id '||n_id_jugador||' no existe en la base de datos');
END;
```

Salida de Script

Tarea terminada en 4,139 segundos

```
nuevos_puntos:=3000;
n_id_jugador:=12;

SELECT jugadores_pac.id_jugador INTO id_comprobar FROM jugadores_pac WHERE n_id_jugador=jugadores_pac.id_jugador;
UPDATE jugadores_pac set jugadores_pac.puntos=puntos+nuevos_puntos WHERE n_id_jugador=jugadores_pac.id_jugador;

    EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('El jugador con id '||n_id_jugador||' no existe en la base de datos');
END;
REGISTO INSERTADO: A fecha de 20/11/22. El jugador Pepito Ortiz Pérez está en el nivel Bronze con un total de 0 puntos
El jugador con id 12 no existe en la base de datos

Procedimiento PL/SQL terminado correctamente.
```

5.3. COMPROBACIÓN DEL PROCEDIMIENTO RANKING_JUGADOR

```
/
-- COMPROBACIÓN DEL PROCEDIMIENTO RANKING_JUGADOR
EXECUTE dbms_output.put_line('-- COMPROBACIÓN DEL PROCEDIMIENTO RANKING_JUGADOR --');
DECLARE
    valor_id            jugadores_pac.id_jugador%TYPE;
    puntos              jugadores_pac.puntos%TYPE;
    nombre_par          jugadores_pac.nombre%TYPE;
    apellido_par        jugadores_pac.apellidos%TYPE;
    puntos_totales_par  jugadores_pac.puntos%TYPE;
    ranking_act         ranking_pac.nombre_ranking%TYPE;
BEGIN
    valor_id:=&id;
    puntos:=&puntos;


    ranking_jugador(valor_id,puntos,nombre_par,apellido_par,puntos_totales_par,ranking_act);

    dbms_output.put_line ('El jugador: ' ||nombre_par||' ' ||apellido_par ||', tendrá '
                    ||puntos_totales_par ||' puntos y pasa al nivel de ranking '||ranking_act);

EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('No se encuentra al jugador');
END;
/
```

```
Salida de Script ×
Tarea terminada en 4,066 segundos
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('No se encuentra al jugador');
END;
El jugador: Antonio Garcia Melero, tendrá 750 puntos y pasa al nivel de ranking Bronce

Procedimiento PL/SQL terminado correctamente.
```

Para este bloque anónimo he creado dos variables para que el usuario introduzca por pantalla el **id** y los **puntos**, que son los parámetros de entrada que nos pide el procedimiento. Después creo las variables de salida que nos pide el procedimiento y donde guardaremos los datos que necesitamos. Llamo al procedimiento y le paso los parámetros de entrada y salida y nos va a imprimir el mensaje con el nombre, apellidos y los nuevos puntos que tendrá después de sumar o restar los puntos que se le ha pasado y también el nombre del ranking que le toca según esos puntos.

He creado también la **EXCEPTION** para que si no encuentra el **id** que le pasamos nos lance un mensaje diciendo que no se encuentra al jugador

```
Tarea terminada en 5,753 segundos
EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('No se encuentra al jugador');
END;
No se encuentra al jugador

Procedimiento PL/SQL terminado correctamente.
```

5.4. COMPROBACIÓN DE LA FUNCION JUGADORES_POR_RANKING

```
-- COMPROBACIÓN DE LA FUNCION JUGADORES_POR_RANKING
EXECUTE dbms_output.put_line('-- COMPROBACIÓN DE LA FUNCION JUGADORES_POR_RANKING --');

DECLARE
total INT;
ranking VARCHAR2 (20);

BEGIN
ranking:=&ranking;
total := jugadores_por_ranking(ranking);
dbms_output.put_line ('En el ranking '||ranking|| ' tenemos a ' ||total||' jugadores');
EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('El ranking introducido no existe');
END;
/
```

Salida de Script

Tarea terminada en 10,089 segundos

```
EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('El ranking introducido no existe');
END;
En el ranking Bronze tenemos a 2 jugadores

Procedimiento PL/SQL terminado correctamente.
```

En este bloque anónimo he creado dos variables, la variable **total** para almacenar el resultado que nos devuelve la función y la variable **ranking** para poder introducir por pantalla el parámetro que nos pide la función. Al introducir un ranking la función nos va a devolver el total de jugadores que tienen ese ranking, se guardará en la variable **total** e imprime el mensaje con la información.

Con la **EXCEPTION** en caso de que entremos un ranking que no existe nos va a lanzar el mensaje de que el ranking introducido no existe

Tarea terminada en 4,275 segundos

```
EXCEPTION
    WHEN NO_DATA_FOUND THEN
    dbms_output.put_line('El ranking introducido no existe');
END;
El ranking introducido no existe

Procedimiento PL/SQL terminado correctamente.
```