

PAC DESARROLLO UF3

Nombre: Liher

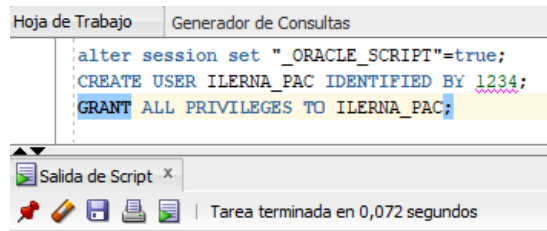
Apellidos : Arce Sanchez-Ocaña

Asignatura: Bases de Datos

Índice

1) GESTIÓN DE USUARIOS Y TABLAS	3
2) BLOQUES ANÓNIMOS	6
3) PROCEDIMIENTOS Y FUNCIONES SIMPLES	9
4) PROCEDIMIENTOS Y FUNCIONES COMPLEJAS	9
5) GESTIÓN DE TRIGGERS	10
6) BLOQUES ANÓNIMOS PARA PRUEBAS DE CÓDIGO	11

Configuración inicial



Session alterado.

User ILERNA_PAC creado.

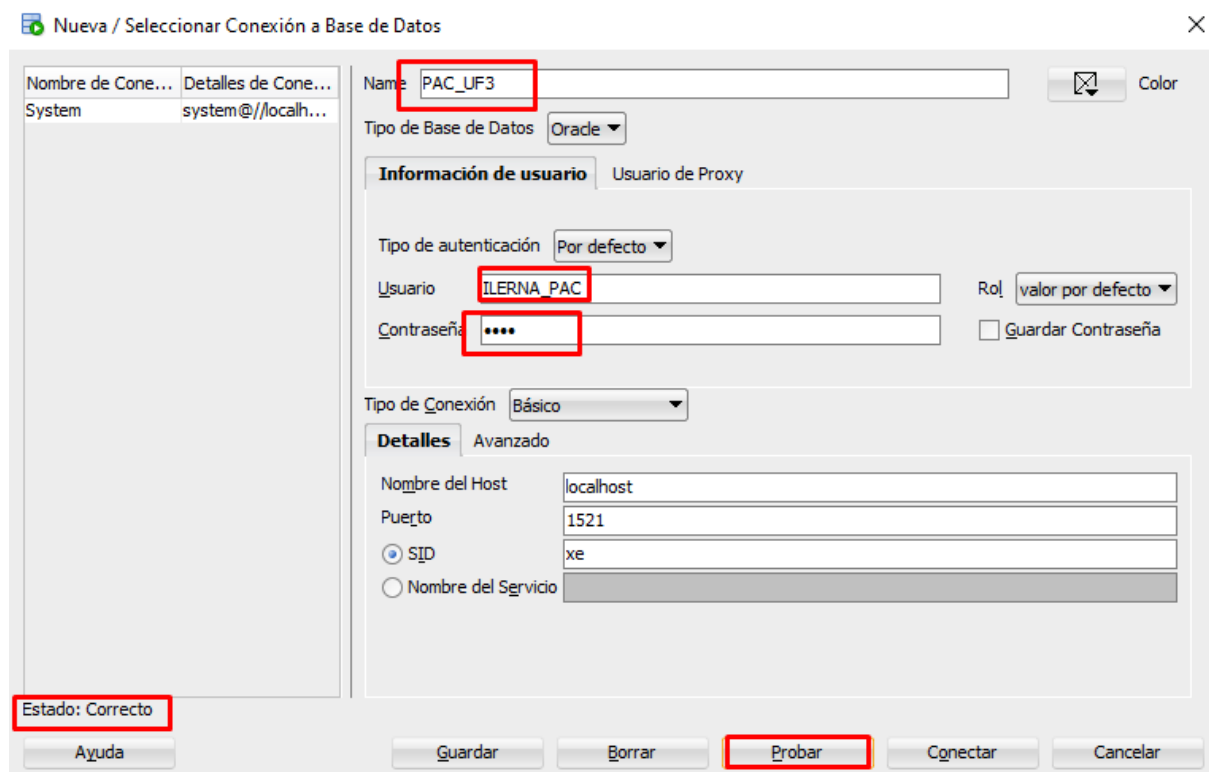
Grant correcto.

Cargamos el script proporcionado por el profesor.

- Iniciamos el Script de oracle para poder hacer modificaciones con usuarios

- Creamos el usuario ILERNA_PAC y le añadimos todos los privilegios

Añadimos la conexión llamada PAC_UF3 con el usuario recién creado (ILERNA_PAC)



Cargamos el script proporcionado por el profesor y nos dejaría esta estructura de tablas

Oracle conexiones

PAC_UF3

Tablas (Filtrado)

- ALUMNOS_PAC
 - ID_ALUMNO
 - NOMBRE
 - APELLIDOS
 - EDAD
- ASIGNATURAS_PAC
 - ID_ASIGNATURA
 - NOMBRE_ASIGNATURA
 - NOMBRE_PROFESOR
 - CREDITOS
- JUGADORES_PAC
 - ID_JUGADOR
 - NOMBRE
 - APELLIDOS
 - PUNTOS
 - RANKING
- RANKING_PAC
 - ID_RANKING
 - VALOR_BAJO
 - VALOR_ALTO
 - NOMBRE_RANKING

PAC_Desarrollo_UF3_Configuracion Inicial.sql x Página de bienvenida x JUGADORES_PAC x

Columnas Datos Model Restricciones Permisos Estadísticas Disparadores Flashback Dependencias Detalle

Ordenar... Filtrar:

	ID_JUGADOR	NOMBRE	APELLIDOS	PUNTOS	RANKING
1	1	Antonio	Garcia Melero	250	Bronze
2	2	Juan	Suarez Jimeno	2350	Diamante
3	3	Alonso	Valencia Morales	1800	Oro
4	4	Fermin	Lopez Galera	2050	Platino
5	5	Dolores	Remiro Soria	1360	Plata
6	6	Maria	Blazquez Ortiz	1520	Oro
7	7	Manuel	Soledad Niera	1060	Plata
8	8	Lurdes	Giro Bueno	960	Bronze

RANKING_PAC x

Columnas Datos Model Restricciones Permisos Estadísticas Disparadores Flashback

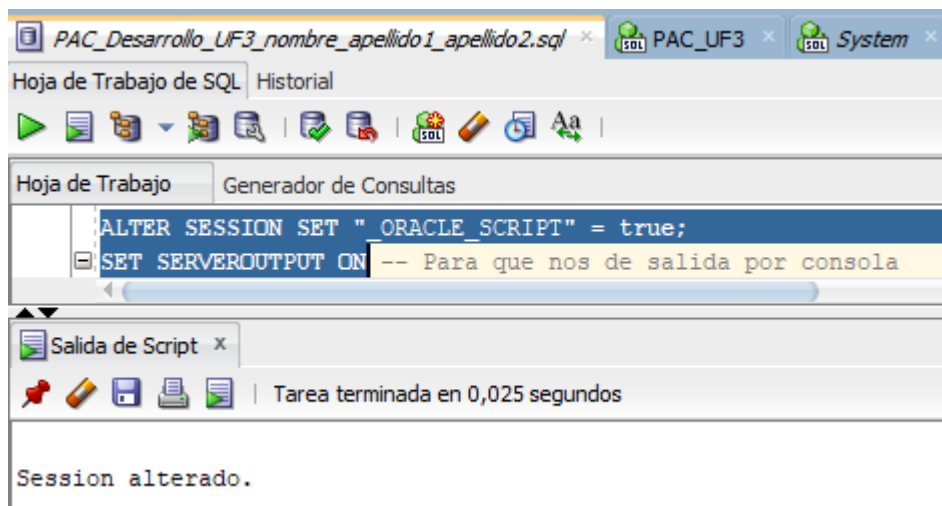
Ordenar... Filtrar:

	ID_RANKING	VALOR_BAJO	VALOR_ALTO	NOMBRE_RANKING
1	1	0	1000	Bronze
2	2	1001	1400	Plata
3	3	1401	1800	Oro
4	4	1801	2200	Platino
5	5	2201	99999	Diamante

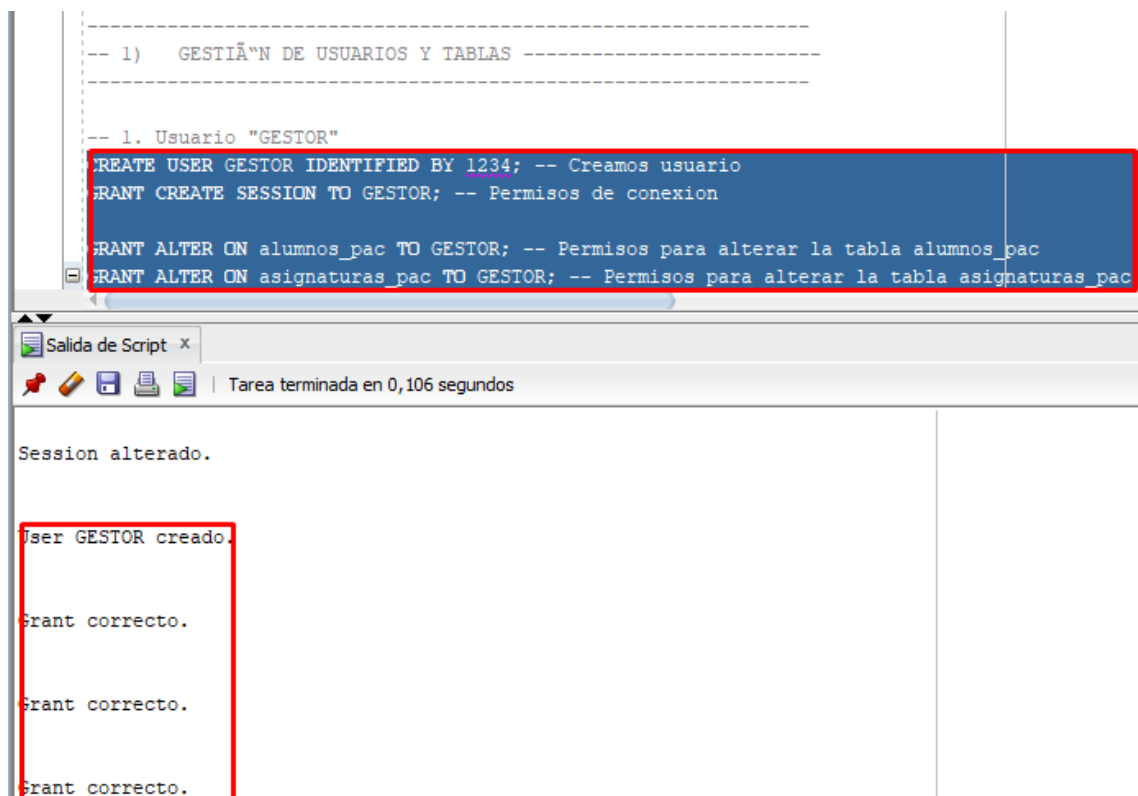
1) GESTIÓN DE USUARIOS Y TABLAS

Esto desde PAC_UF3

- Ejecutamos el script de oracle de nuevo para la gestión de usuarios por si acaso



- Creamos un usuario GESTOR y le damos acceso para iniciar sesión
- Le damos permisos a GESTOR para modificar las tablas alumnos_pac y asignaturas_pac



Nos conectamos al usuario gestor

Nueva / Seleccionar Conexión a Base de Datos

Nombre de Cone...	Detalles de Cone...
PAC_UF3	ILERNA_PAC@//l...
System	system@//localh...

Name: **GESTOR**

Tipo de Base de Datos: Oracle

Información de usuario Usuario de Proxy

Tipo de autenticación: Por defecto

Usuario: **gestor** Rol: valor por defecto

Contraseña: **** ☐ Guardar Contraseña

Tipo de Conexión: Básico

Detalles Avanzado

Nombre del Host: localhost

Puerto: 1521

☒ SID: xe

☐ Nombre del Servicio:

Estado: Correcto

Ayuda Guardar Borrar **Probar** Conectar Cancelar

Utilizamos el usuario GESTOR para realizar estas alteraciones

```
-- Nos conectamos al usuario GESTOR y realizamos todos estos cambios
ALTER TABLE ilerna_pac.alumnos_pac ADD ciudad VARCHAR(30); -- Anadimos la c
ALTER TABLE ilerna_pac.asignaturas_pac MODIFY nombre_profesor VARCHAR(50);
ALTER TABLE ilerna_pac.asignaturas_pac DROP COLUMN credits; -- Eliminamos
ALTER TABLE ilerna_pac.asignaturas_pac ADD ciclo VARCHAR(3); -- Anadimos la
```

Salida de Script x

Tarea terminada en 0,224 segundos

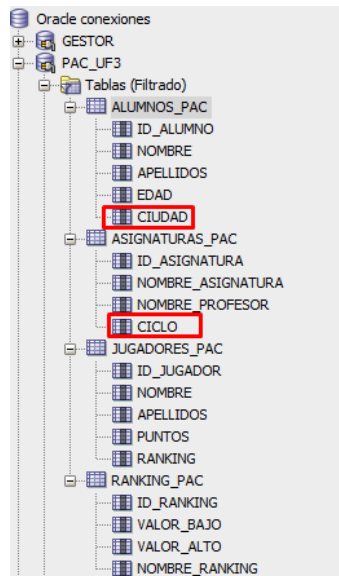
```
Table ILERNA_PAC.ALUMNOS_PAC alterado.

Table ILERNA_PAC.ASIGNATURAS_PAC alterado.

Table ILERNA_PAC.ASIGNATURAS_PAC alterado.

Table ILERNA_PAC.ASIGNATURAS_PAC alterado.
```

Comprobamos que es cierto:



Las columnas CIUDAD y CICLO se han añadido y la columna CRÉDITOS se ha borrado.

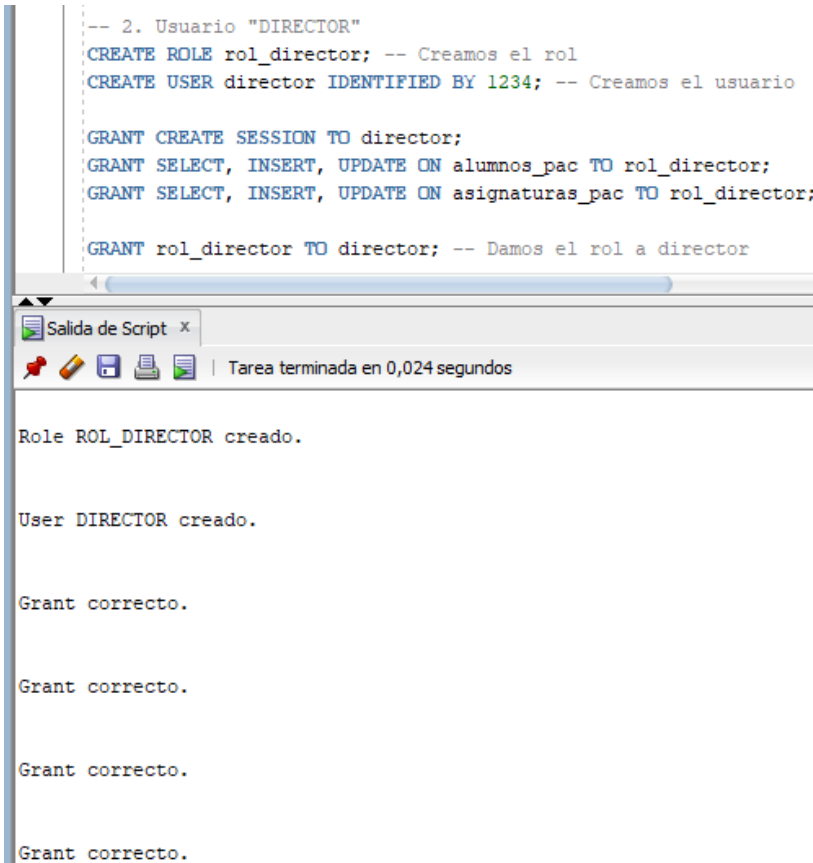
También hemos modificado (ampliado) el VARCHAR de la columna nombre profesor

COLUMN_NAME	DATA_TYPE	NULLABLE	DATA_DEFAULT	COLUMN_ID	COMMENTS
1 ID_ASIGNATURA	VARCHAR2 (11 BYTE)	No	(null)	1 (null)	
2 NOMBRE_ASIGNATURA	VARCHAR2 (20 BYTE)	Yes	(null)	2 (null)	
3 NOMBRE_PROFESOR	VARCHAR2 (50 BYTE)	Yes	(null)	3 (null)	
4 CICLO	VARCHAR2 (3 BYTE)	Yes	(null)	4 (null)	

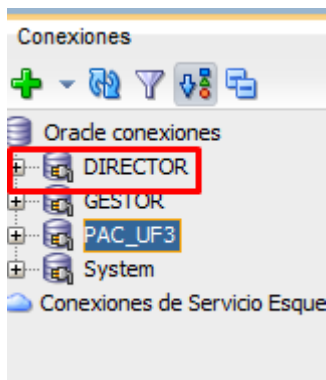
Creamos un rol llamado "ROL_DIRECTOR" y un usuario llamado "DIRECTOR"
Este lo crearemos desde PAC_UF3 con el usuario llerna_pac

Asignamos permisos de conexión, seleccion, insertar y actualizar las tablas alumnos_pac y asignaturas_pac a el rol 'rol_director'.

Le damos el rol "rol_director" a el usuario "director"



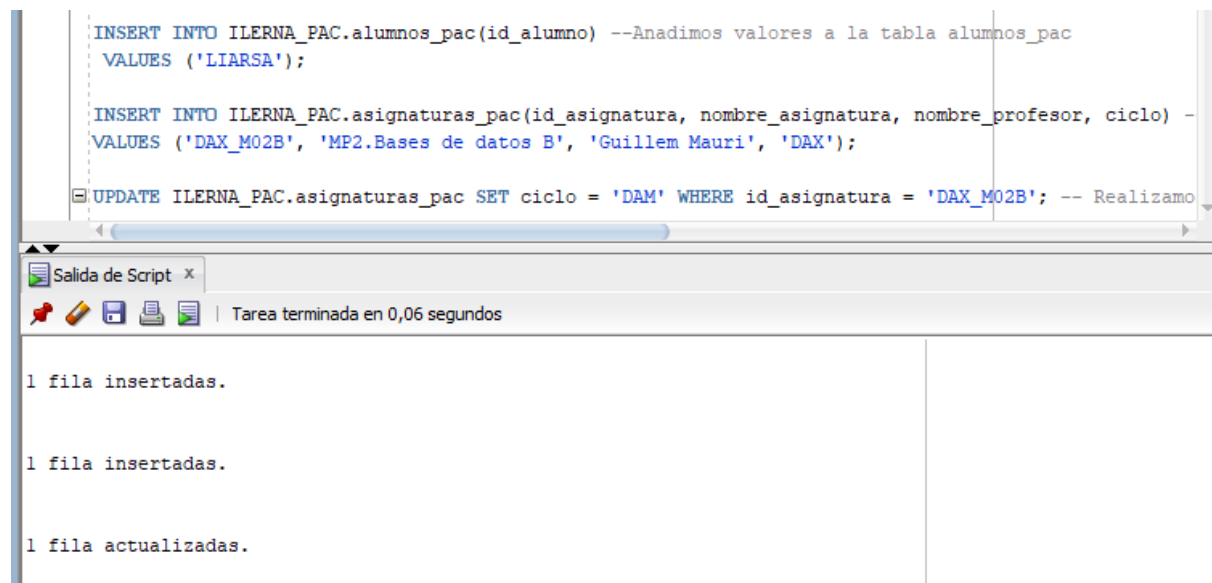
Nos conectamos a director



Inserto un registro en la tabla alumnos_pac con mis 2 primeras iniciales de mi nombre y mis 2 apellidos

Inserto un registro en la tabla asignaturas_pac

Modificó el registro que he añadido hace poco y cambio DAX por DAM que es mi ciclo y lo identifiqué con el id_asignatura.



2) BLOQUES ANÓNIMOS

Creo un bloque anónimo con 3 variables

- contador
- input para seleccionar jugador
- los puntos actuales que se irán sumando

He creado un código que pide el num de un jugador y recoge los puntos actuales y los guarda en la variable puntos_actuales

Hacemos un WHILE LOOP que se realiza 4 veces (1 para mostrar todo + repetirlo 3 veces más) Cada vez que se haga el LOOP se añadirá +1 al contador y +300 a puntos_actuales

Le añadimos condicionales IF y ELSIF para filtrar los puntos que tenemos y actualizar el ranking en caso de que sea necesario

Y sacamos por pantalla: Los puntos actuales, el ranking actual y cuanto seria en caso de el incremento de +300 a los puntos_actuales

El código es demasiado largo como para sacar una foto sola. la parto en 2

```
DECLARE
puntos_actuales NUMBER(10, 2):=0; --Variable con los puntos
v_counter NUMBER(2) :=1; -- Contador para el loop
jugador_seleccionado NUMBER(2);

BEGIN
    jugador_seleccionado := &jug_selec;
    SELECT puntos INTO puntos_actuales FROM jugadores_pac WHERE id_jugador=jugador_seleccionado;
    WHILE v_counter <= 4 LOOP -- Realizar el codigo mientras el contador sea menor o igual a 4

        -- Anadimos condicionales para los diferentes casos que pueden existir
        -- En todos los casos posibles se ejecutara una salida por consola que muestre los puntos actu

        -- Caso entre 0 y 1000 - ranking bronce
        IF puntos_actuales BETWEEN 0 AND 1000 THEN
            dbms_output.put_line('Puntos actuales: '||puntos_actuales);
            UPDATE jugadores_pac SET ranking='Bronze' WHERE id_jugador=jugador_seleccionado;
            dbms_output.put_line('Ranking: Bronze');

        -- Caso entre 1001 y 1400 - ranking plata
        ELSIF puntos_actuales BETWEEN 1001 AND 1400 THEN
            dbms_output.put_line('Puntos actuales: '||puntos_actuales);
            UPDATE jugadores_pac SET ranking='Plata' WHERE id_jugador=jugador_seleccionado;
            dbms_output.put_line('Ranking: Plata');

        -- Caso entre 1401 y 1800 - ranking oro
        ELSIF puntos_actuales BETWEEN 1401 AND 1800 THEN
            dbms_output.put_line('Puntos actuales: '||puntos_actuales);
            UPDATE jugadores_pac SET ranking='Oro' WHERE id_jugador=jugador_seleccionado;
            dbms_output.put_line('Ranking: Oro');

        -- Caso entre 1801 y 2200 - ranking platino
        ELSIF puntos_actuales BETWEEN 1801 AND 2200 THEN
            dbms_output.put_line('Puntos actuales: '||puntos_actuales);
            UPDATE jugadores_pac SET ranking='Platino' WHERE id_jugador=jugador_seleccionado;
            dbms_output.put_line('Ranking: Platino');

        -- Caso entre 2201 y 99999 - ranking diamante
        ELSIF puntos_actuales BETWEEN 2001 AND 99999 THEN
            dbms_output.put_line('Puntos actuales: '||puntos_actuales);
            UPDATE jugadores_pac SET ranking='Diamante' WHERE id_jugador=jugador_seleccionado;
            dbms_output.put_line('Ranking: Diamante');
        ELSE
            dbms_output.put_line('Tus puntos no entran dentro de los valores establecidos por el
            END IF;

        -- Fuera del IF agregamos siempre +1 al contador
        -- Tambien anadimos el incremento a los puntos actuales y los mostramos por pantalla

        puntos_actuales := puntos_actuales+300;
        UPDATE jugadores_pac SET puntos = puntos_actuales WHERE id_jugador=jugador_seleccionado;

        v_counter := v_counter + 1;
        dbms_output.put_line('Incremento de 300: '||puntos_actuales);
        dbms_output.put_line(''); --Linea para separar las salidas de consola cada vez que ejecuta

    END LOOP;
END;
```


Lo ejecutamos e introducimos como ejemplo a el user con el ID 5 y vemos el output que nos manda

Introducir Variable de Sustitución ✕

Introduzca un valor para jug_selec:

Aceptar Cancelar

Puntos actuales: 1360
Ranking: Plata
Incremento de 300: 1660

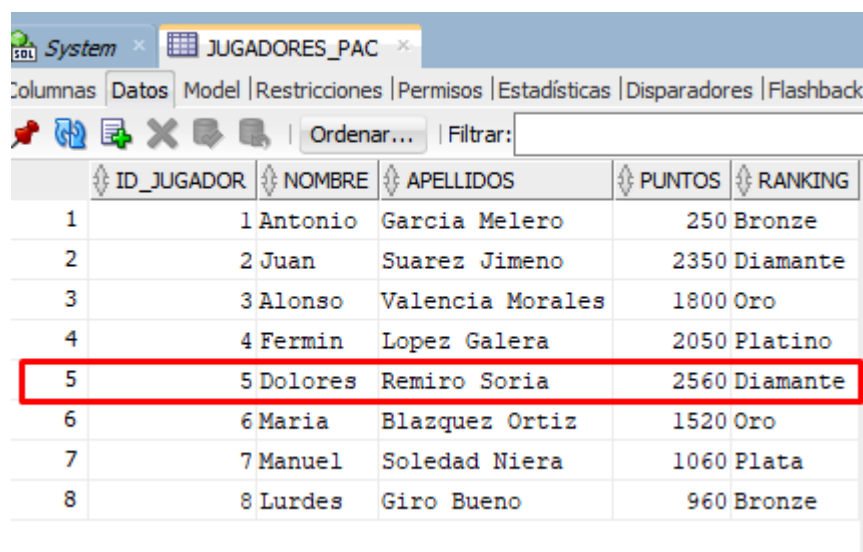
Puntos actuales: 1660
Ranking: Oro
Incremento de 300: 1960

Puntos actuales: 1960
Ranking: Platino
Incremento de 300: 2260

Puntos actuales: 2260
Ranking: Diamante
Incremento de 300: 2560

Procedimiento PL/SQL terminado correctamente.

Este es el output que nos sale y vemos en la tabla como se han cambiado los puntos y el ranking cuando lo hemos ejecutado:



ID_JUGADOR	NOMBRE	APELLIDOS	PUNTOS	RANKING
1	1 Antonio	Garcia Melero	250	Bronze
2	2 Juan	Suarez Jimeno	2350	Diamante
3	3 Alonso	Valencia Morales	1800	Oro
4	4 Fermin	Lopez Galera	2050	Platino
5	5 Dolores	Remiro Soria	2560	Diamante
6	6 Maria	Blazquez Ortiz	1520	Oro
7	7 Manuel	Soledad Niera	1060	Plata
8	8 Lurdes	Giro Bueno	960	Bronze

3) PROCEDIMIENTOS Y FUNCIONES SIMPLES

Creamos una función que de los 3 números que nos den devuelva el más grande y en caso de que se repita algún número nos aparezca por pantalla que “No se pueden repetir números en la secuencia”.

Hacemos la función que pase los 3 números por parámetro y devuelve una variable VARCHAR que puede almacenar número o letras por si tenemos que devolver texto

Utilizamos operadores IF para saber si no hay números repetidos, si no los hay entonces comparamos los números unos de otros y asignamos la variable num_mayor si una de las subcondiciones se cumple y lo devolvemos.

```
CREATE FUNCTION numero_mayor (num1 NUMBER, num2 NUMBER, num3 NUMBER)
RETURN VARCHAR
AS
    num_mayor VARCHAR(100);
BEGIN

    IF num1 != num2 OR num1 != num3 OR num3 != num2 THEN -- IF para
        -- 2 IF y un ELSE para comparar los numeros y el mas grande
        IF num1 > num2 AND num1 > num3 THEN
            num_mayor:=num1;
        ELSIF num2 > num1 AND num2 > num3 THEN
            num_mayor:=num2;
        ELSE
            num_mayor:=num3;
        END IF;
    ELSE -- ELSE por si algun numero se repite
        num_mayor := 'No se pueden repetir números en la secuencia';
    END IF;
    RETURN(num_mayor);
END;
```

4) PROCEDIMIENTOS Y FUNCIONES COMPLEJAS

Función que devuelva la cantidad de jugadores que hay en el ranking que hemos especificado.

Pasamos un solo parámetro por valor que será el ranking y devolveremos un número en una variable.

Dentro de la función se realiza un select , se realiza un count de las personas con el ranking pasado por parámetro y las almacena en número jugadores.

```
CREATE FUNCTION jugadores_por_ranking (ranking_act VARCHAR) -- Pasamos por parametro un valor que
RETURN NUMBER -- Devolvemos un numero
AS
    num_jugadores NUMBER(3); -- Variable en la que almacenar el num de jugadores por ranking
BEGIN
    SELECT COUNT(*) INTO num_jugadores FROM ilerna_pac.jugadores_pac WHERE ranking = ranking_act;
    RETURN num_jugadores; -- Devolvemos el contador con el num de personas en ese ranking
END;
```

5) GESTIÓN DE TRIGGERS

Creamos un trigger que salte cuando hacemos un update a el campo puntos de la tabla jugadores_pac.

Declaramos 4 variable

- v_puntos Almacena los puntos
- v_nombre Almacena el nombre de la persona con los puntos
- v_hora Almacena la hora de la modificación
- v_fecha Almacena la fecha de la modificación

v_nombre:= :new.nombre; Guardamos el nombre del usuario al que se la ha realizado la modificación

v_puntos:= :new.puntos; Guardamos los puntos en una variable que usaremos mas tarde con los comparadores IF

Realizamos un select de la hora y fecha por separado desde la tabla dual(tabla dummy por si no tenemos tabla de donde sacar esos datos)

```
SELECT to_char(sysdate, 'HH24:MI:ss') INTO v_hora FROM dual;
```

```
SELECT sysdate INTO v_fecha FROM dual; -- Guardamos la fecha
```

Utilizamos los comparadores IF como en el ejercicio 2 de bloque anónimos para comparar los puntos y si estan dentro de ciertos puntos cambian de un rango a otro. En caso de ser asi pondremos :new ranking que se el rango que corresponda

```
-----
IF v_puntos BETWEEN 0 AND 1000 THEN
    :new ranking := 'Bronze';

-- Caso entre 1001 y 1400 - ranking plata
ELSIF v_puntos BETWEEN 1001 AND 1400 THEN
    :new ranking := 'Plata';

-- Caso entre 1401 y 1800 - ranking oro
ELSIF v_puntos BETWEEN 1401 AND 1800 THEN
    :new ranking := 'Oro';

-- Caso entre 1801 y 2200 - ranking platino
ELSIF v_puntos BETWEEN 1801 AND 2200 THEN
    :new ranking := 'Platino';

-- Caso entre 2201 y 99999 - ranking diamante
ELSIF v_puntos BETWEEN 2001 AND 99999 THEN
    :new ranking := 'Diamante';
END IF; -- Terminamos el IF
```

Sacamos por consola un resultado con todas las variables

```
/
CREATE OR REPLACE TRIGGER cambio_ranking_jugador BEFORE
UPDATE OF puntos ON jugadores_pac -- tabla en la que se realizara
FOR EACH ROW -- Por cada linea que hagamos update
DECLARE
    v_puntos NUMBER(10,2);
    v_nombre VARCHAR2(20);
    v_hora VARCHAR2(20);
    v_fecha DATE;
BEGIN
    v_nombre:= :new.nombre; -- Guardamos el nombre del usuario de la linea que hemos hecho update
    v_puntos:= :new.puntos; -- Guardamos los puntos del usuario de la linea que hemos hecho update

    SELECT to_char(sysdate, 'HH24:MI:ss') INTO v_hora FROM dual; -- Guardamos la hora de la modificacion en una variable
    SELECT sysdate INTO v_fecha FROM dual; -- Guardamos la fecha en una variable

    -- cadena de if y elsif para filtrar los puntos actuales de la linea a la que se le ha hecho update y cambiar el ranking
    IF v_puntos BETWEEN 0 AND 1000 THEN
        :new.ranking := 'Bronze';

    -- Caso entre 1001 y 1400 - ranking plata
    ELSIF v_puntos BETWEEN 1001 AND 1400 THEN
        :new.ranking := 'Plata';

    -- Caso entre 1401 y 1800 - ranking oro
    ELSIF v_puntos BETWEEN 1401 AND 1800 THEN
        :new.ranking := 'Oro';

    -- Caso entre 1801 y 2200 - ranking platino
    ELSIF v_puntos BETWEEN 1801 AND 2200 THEN
        :new.ranking := 'Platino';

    -- Caso entre 2201 y 99999 - ranking diamante
    ELSIF v_puntos BETWEEN 2001 AND 99999 THEN
        :new.ranking := 'Diamante';
    END IF; -- Terminamos el IF
    -- Sacamos por pantalla un mensaje con las 4 variables recogidas mas el antiguo ranking
    DBMS_OUTPUT.PUT_LINE('El ranking del jugador '|| v_nombre || ' se ha modificado el dia '||v_fecha||' '||v_hora||', ante
END;
```

6) BLOQUES ANÓNIMOS PARA PRUEBAS DE CÓDIGO

1. COMPROBACIÓN GESTIÓN USUARIOS Y TABLAS

Comprobamos las tablas con un select para poder verlas

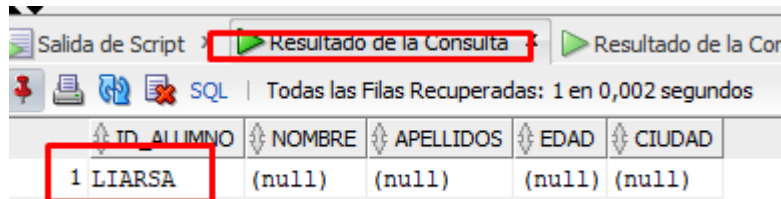
```
-- 1. COMPROBACIÓN REGISTROS DE TABLAS
/
EXECUTE dbms_output.put_line('-- 1. COMPROBACIÓN REGISTROS DE TABLAS');
SELECT ALL* FROM alumnos_pac;-- Mostramos la tabla
SELECT ALL* FROM asignaturas_pac;-- Mostramos la tabla

Salida de Script x Resultado de la Consulta x Resultado de la Consulta 1 x
Tarea terminada en 0,491 segundos

-- 1. COMPROBACIÓN REGISTROS DE TABLAS

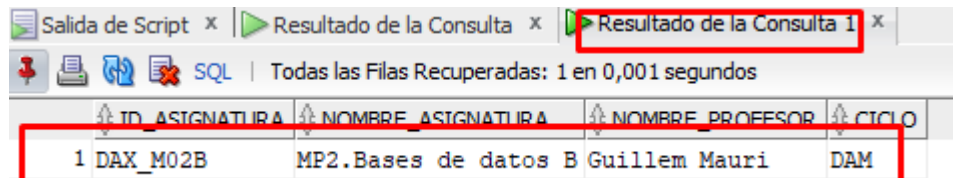
Procedimiento PL/SQL terminado correctamente.
>>Query Run In:Resultado de la Consulta
>>Query Run In:Resultado de la Consulta 1
```

Resultado de la tabla alumnos_pac



ID_ALUMNO	NOMBRE	APELLIDOS	EDAD	CIUDAD
1	LIARSA	(null)	(null)	(null)

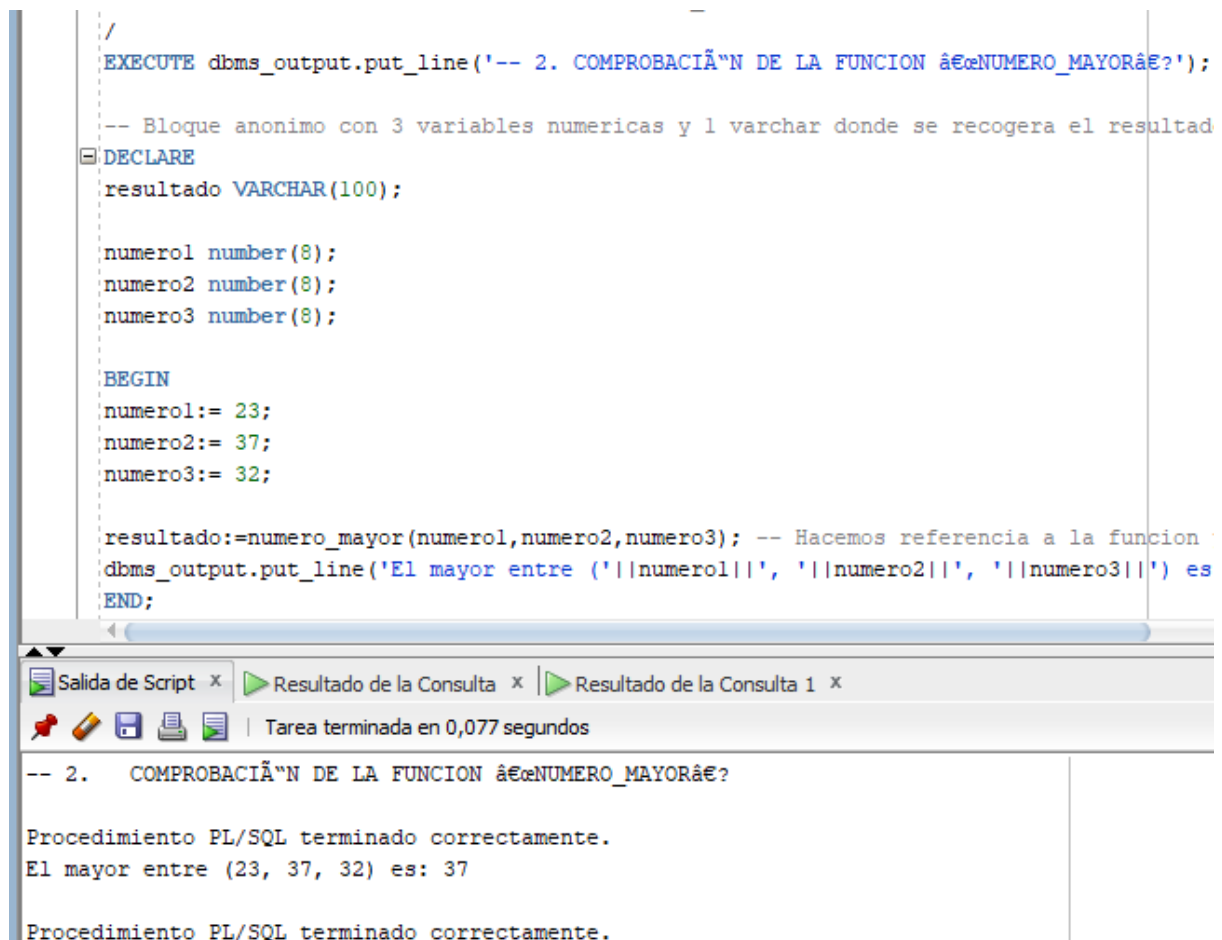
Resultado de la tabla asignaturas_pac



ID_ASIGNATURA	NOMBRE_ASIGNATURA	NOMBRE_PROFESOR	CICLO
1	DAX_M02B	MP2.Bases de datos B	Guillem Mauri
			DAM

2. COMPROBACIÓN DE LA FUNCIÓN “NUMERO_MAYOR”

Comprobamos la función número mayor creando un bloque anónimo con 3 números e invocando a la función con los 3 números. Mostrando el resultado en pantalla.



```
/
EXECUTE dbms_output.put_line('-- 2. COMPROBACIÓN DE LA FUNCION "NUMERO_MAYOR?');

-- Bloque anonimo con 3 variables numericas y 1 varchar donde se recogera el resultado
DECLARE
resultado VARCHAR(100);

numero1 number(8);
numero2 number(8);
numero3 number(8);

BEGIN
numero1:= 23;
numero2:= 37;
numero3:= 32;

resultado:=numero_mayor(numero1,numero2,numero3); -- Hacemos referencia a la funcion
dbms_output.put_line('El mayor entre ('||numero1||', '||numero2||', '||numero3||') es
END;
```

Salida de Script x | Resultado de la Consulta x | Resultado de la Consulta 1 x

Tarea terminada en 0,077 segundos

-- 2. COMPROBACIÓN DE LA FUNCION "NUMERO_MAYOR?

Procedimiento PL/SQL terminado correctamente.

El mayor entre (23, 37, 32) es: 37

Procedimiento PL/SQL terminado correctamente.

3. COMPROBACIÓN DE LA FUNCION “JUGADORES_POR_RANKING”

Creemos un bloque anónimo con 2 variables

- nombre_ranking Almacenaremos a mano el nombre del ranking
- contador_personas Almacenamos el resultado de la función

Pedimos el nombre del ranking y llamamos a la función con el.

```

/
EXECUTE dbms_output.put_line('-- 3. COMPROBACIÓN DE LA FUNCION “JUGADORES_POR_RANKING”');
/
-- Bloque anonimos que inicializa la funcion de jugadores_por_ranking
DECLARE
nombre_ranking VARCHAR(14);
contador_personas NUMBER(3);
BEGIN
    nombre_ranking := 'Plata'; --Ponemos el nombre del ranking
    contador_personas:=jugadores_por_ranking(nombre_ranking); -- Inicializamos la funcion con l
    dbms_output.put_line('En el ranking '||nombre_ranking|| ', tenemos a '||contador_personas||
END;

-- 4. COMPROBACIÓN DE LOS TRIGGERS
/

```

Salida de Script x Resultado de la Consulta x Resultado de la Consulta 1 x

Tarea terminada en 0,047 segundos

```

-- 3. COMPROBACIÓN DE LA FUNCION “JUGADORES_POR_RANKING”

Procedimiento PL/SQL terminado correctamente.
En el ranking Plata, tenemos a 2 jugadores.

Procedimiento PL/SQL terminado correctamente.

```

4. COMPROBACIÓN DE LOS TRIGGERS “CAMBIO_RANKING_JUGADOR”

Creemos un bloque anónimo que realice un update con 2 valores que nosotros le demos

Serán 2 inputs que aparecerán por pantalla uno para el id del jugador y otros para los puntos que queramos ponerle.

Introducir Variable de Sustitución X

Introduzca un valor para idjug:

Aceptar Cancelar

Introducir Variable de Sustitución X

Introduzca un valor para puntos:

Aceptar Cancelar

```
-- 4. COMPROBACIÓN DE LOS TRIGGERS

EXECUTE dbms_output.put_line('-- 4. COMPROBACIÓN DE LOS TRIGGERS');
-- Bloque anonimo en el que añadiremos los valores que deseamos para el update y salte el trigger
DECLARE
    v_puntos jugadores_pac.puntos%TYPE;
    v_idjug jugadores_pac.id_jugador%TYPE;
BEGIN
    v_idjug := &idjug;
    v_puntos:= &puntos;

    UPDATE jugadores_pac SET puntos=v_puntos WHERE id_jugador=v_idjug; -- Sentencia update de la tabla jugadores_pac pa:
END;
```

Salida de Script x Resultado de la Consulta x Resultado de la Consulta 1 x

Tarea terminada en 43,522 segundos

```
UPDATE jugadores_pac SET puntos=v_puntos WHERE id_jugador=v_idjug; -- Sentencia update de la tabla jugadores_pac para que
END;
Nuevo:DECLARE
    v_puntos jugadores_pac.puntos%TYPE;
    v_idjug jugadores_pac.id_jugador%TYPE;
BEGIN
    v_idjug := 7;
    v_puntos:= 1530;

    UPDATE jugadores_pac SET puntos=v_puntos WHERE id_jugador=v_idjug; -- Sentencia update de la tabla jugadores_pac para que
END;
El ranking del jugador Manuel se ha modificado el dia 01/05/22 19:41:21, antes era Plata y ahora es Oro

Procedimiento PL/SQL terminado correctamente.
```

PAC DESARROLLO UF3