examen-PLSQL-2024-SOLUCION-DEL-P...

miau_33

Administración de Bases de Datos

3º Grado en Ingeniería Informática

Escuela Técnica Superior de Ingeniería Informática
Universidad de Málaga

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ADMINISTRACIÓN DE BASES DE DATOS.

21 DE JUNIO DE 2024

LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN
UNIVERSIDAD DE MÁLAGA

NOTA: Es MUY IMPORTANTE que sigas exactamente la especificación de este ejercicio y que llames a cada objeto EXACTAMENTE como se pide.

- **Para comenzar el examen conéctate a la dirección: https://docenciabd.lcc.uma.es/**
- **Utiliza el usuario que comienza por UBDXXX para conectarte a la aplicación. La aplicación suministrará un usuario y contraseña con la que debes conectarte a Apolo en Afrodita. Ese usuario, con el que tienes que realizar el examen, NO es el que empieza por UBDXXX**
- **NO OLVIDES finalizar las transacciones si fuera necesario.**

En tu usuario de examen se han creado las tablas `CENTRO`, `CLIENTE`, `ENTRENA`, `ENTRENADOR` y `USUARIO` con algunos datos.

Ejercicio 1 (0,3 puntos). Crear una función denominada `CALCULA_NUM_CLIENTES` que reciba como parámetro el código de un **entrenador** y devuelva el número de clientes que tiene ese entrenador. Se obtiene el número de clientes de un entrenador contando los clientes de ese entrenador de la tabla `ENTRENA`.

```
function CALCULA_NUM_CLIENTES
(p_entrenador ENTRENADOR.ID%TYPE) RETURN NUMBER
```

```
create or replace FUNCTION CALCULA_NUM_CLIENTES
(  P_ENTRENADOR IN VARCHAR2 ) RETURN NUMBER AS
res number;
BEGIN
    select count (*) into res from entrena
      where entrenador_id= p_entrenador;
  RETURN res;
END CALCULA_NUM_CLIENTES;
```

Ejercicio 2 (0,4 puntos). Crear un procedimiento denominado `CALCULA_CLIENTES_CENTRO` que reciba como parámetro el código de un **centro** y actualice la columna `NUM_CLIENTES` de la tabla `ENTRENADOR` con el número de clientes de los entrenadores de ese centro. Se obtiene el número de clientes de un entrenador llamando a la función anterior para cada uno de los entrenadores cuyo código de centro coincida con el parámetro.

```
procedure CALCULA_CLIENTES_CENTRO (p_centro ENTRENADOR.CENTRO_ID%TYPE)
```

NOTA: Utilizar `SELECT … FOR UPDATE;` en el cursor. No olvides confirmar la transacción antes de terminar el procedimiento.

```
create or replace PROCEDURE CALCULA_CLIENTES_CENTRO
(  P_CENTRO IN NUMBER ) AS
cursor c_centro(pp_centro number) is select id from entrenador where
centro_id = pp_centro for update;
BEGIN
for v in c_centro(p_centro) loop
  update entrenador set num_clientes = calcula_num_clientes(v.id)
where id = v.id;
     --current of c_centro; (También sirve)
end loop;
commit;
END CALCULA_CLIENTES_CENTRO;
```

Ejercicio 3 (0,4 puntos). Crear un trigger denominado TR_NUM_CLIENTES para la tabla ENTRENA que al insertarse, modificarse o borrarse cada una de las filas actualice el número de clientes de la tabla entrenador. Así, por ejemplo, si se inserta una fila en ENTRENA con el código de entrenador, 5, se debe aumentar en 1 el número de clientes que entrena el entrenador de código 5 en la tabla ENTRENADOR.

```
create or replace TRIGGER TR_NUM_CLIENTES
BEFORE DELETE OR INSERT OR UPDATE ON ENTRENA FOR EACH ROW
BEGIN
  if inserting then
    update entrenador set num_clientes = num_clientes +1
      where id = :new.entrenador_id;
  elsif deleting then
    update entrenador set num_clientes = num_clientes -1
      where id = :old.entrenador_id;
  else
    update entrenador set num_clientes = num_clientes +1
      where id = :new.entrenador_id;
    update entrenador set num_clientes = num_clientes -1
      where id = :old.entrenador_id;
  end if;
END;
```

Ejercicio 4 (0,2 puntos). Crear un procedimiento denominado CREA_VISTA_ENTRENADOR que reciba como parámetro un código de entrenador y crea una vista denominada V_ENTRENADOR_cc siendo cc el código de entrenador pasado como parámetro. Así, por ejemplo, si el código es ***1***, la vista será:

```
create view v_entrenador_1 (e_nombre, e_apellidos, c_nombre, c_apellidos)
as

select u1.nombre, u1.apellidos, u2.nombre, u2.apellidos

from entrenador e join usuario u1 on (u1.id = e.id) left join

entrena en on e.id = en.entrenador_id

left join cliente c on en.cliente_id = c.id

left join usuario u2 on c.id = u2.id

where e.id = 1;
```

El procedimiento debe tener la siguiente declaración:

```
procedure CREA_VISTA_ENTRENADOR
(p_entrenador ENTRENADOR.ID%TYPE)
```

NOTA: Se recomienda mostrar por DBMS_OUPUT el código de la sentencia a ejecutar antes de la llamada con EXECUTE IMMEDIATE para asegurarse de que se ha construido bien. Ejemplo:

```
Sentencia :=  'CREATE VIEW v_entrenador_'||p_entrenador||'(e_nombre,
e_apellidos, c_nombre, c_apellidos) AS ……. ';

DBMS_OUTPUT.PUT_LINE (Sentencia);
```

```
EXECUTE IMMEDIATE Sentencia;
```

```
create or replace procedure CREA_VISTA_ENTRENADOR
(p_entrenador ENTRENADOR.ID%TYPE) as
sentencia varchar2(1000);
begin

Sentencia :=  'CREATE VIEW v_entrenador_'||p_entrenador||'(e_nombre,
e_apellidos, c_nombre, c_apellidos) AS select u1.nombre, u1.apellidos,
u2.nombre, u2.apellidos
from entrenador e join usuario u1 on (u1.id = e.id) left join
entrena en on e.id = en.entrenador_id
left join cliente c on en.cliente_id = c.id
left join usuario u2 on c.id = u2.id
where e.id = '||p_entrenador;
DBMS_OUTPUT.PUT_LINE (Sentencia);
EXECUTE IMMEDIATE Sentencia;

end;
```

Ejercicio 5 (0,2 puntos). Crear un procedimiento denominado `BORRA_VISTA_ENTRENADOR` que reciba como parámetro el código del entrenador y borre la vista creada en el apartado anterior.

```
procedure BORRA_VISTA_ENTRENADOR
(p_entrenador ENTRENADOR.ID%TYPE)
```

Así, por ejemplo, si el código de entrenador es ***1***, borrará la vista v_entrenador_1.

```
create or replace procedure borra_VISTA_ENTRENADOR
(p_entrenador ENTRENADOR.ID%TYPE) as
sentencia varchar2(1000);
begin
Sentencia :=  'drop VIEW v_entrenador_'||p_entrenador;
DBMS_OUTPUT.PUT_LINE (Sentencia);
EXECUTE IMMEDIATE Sentencia;
end;
```

Ejercicio 6 (0,5 puntos). Crear un paquete denominado `PK_EXAMEN24` y declarar e implementar un procedimiento denominado `CREA_VISTAS_CENTRO` que reciba como parámetro el código de un centro y vaya llamando sucesivamente a los procedimientos `BORRA_VISTA_ENTRENADOR` y `CREA_VISTA_ENTRENADOR` para cada uno de los entrenadores del centro. El procedimiento debe funcionar capturando las excepciones posibles y continuando el bucle. Si el centro no existiera debe elevar la excepción `ESC.PK_PRUEBA_EXAMEN.CENTRO_NO_EXISTE` declarado en el paquete correspondiente del usuario ESC.

```
procedure CREA_VISTAS_CENTRO (p_centro centro.id%type);
```

NOTA: El ejercicio 6 debe ser implementado dentro del body de `PK_EXAMEN24` (si no es así, no será corregido ni evaluado).

```
create or replace PACKAGE PK_EXAMEN24 AS
procedure CREA_VISTAS_CENTRO (p_centro centro.id%type);
END PK_EXAMEN24;
---------------

create or replace PACKAGE BODY PK_EXAMEN24 AS
  procedure CREA_VISTAS_CENTRO (p_centro centro.id%type) AS
  cursor c_centro is select id from entrenador
      where centro_id = p_centro;
  num number;
  BEGIN
    select count(*) into num from centro where id = p_centro;
    if num = 0 then
        raise esc.pk_prueba_examen.CENTRO_NO_EXISTE;
    else
    for v in c_centro loop
        begin
            borra_vista_entrenador (v.id);
            exception when others then null;
        end;
        begin
            crea_vista_entrenador (v.id);
            exception when others then null;
        end;
    end loop;
    end if;
  END CREA_VISTAS_CENTRO;

END PK_EXAMEN24;
```