Triggers

```
create or replace trigger t1
before update on emple
begin
        dbms_output.put_line('La tabla Emple va a ser modificada');
end;
```

- Esto muestra en pantalla un mensaje.
- Lo muestra antes (before) de hacer una actualización (update) en una tabla específica (emple).
- Básicamente le estamos indicando la acción que tiene que realizar el trigger cuando se dispare la condición.

```
create or replace trigger t2
before update on emple
for each row
begin
        dbms_output.put_line('El empleado nº' || :old.emp_no || ' va a ser modificado.');
end;
```

- Con :old.emp_no lo que estamos mostrando es el número de empleado antes de la modificación.
- for each row nos permite realizar una acción por cada fila que vayamos a modificar.

```
create or replace trigger t3
after update on emple
begin
        dbms_output.put_line('La tabla Emple ha sido modificada');
end;
```

- Lo mismo que el t1 pero con after.

```
create or replace trigger t4
after update on emple
for each row
begin
  dbms_output.put_line('El empleado nº' || :old.emp_no || ' ha sido modificado.');
end;
```

- Lo mismo que el t2 pero con after.

- Para ver los triggers que tenemos creados tendremos que hacer lo siguiente:

SELECT TRIGGER_NAME FROM USER_TRIGGERS;

- Para borrar los triggers que queramos:

DROP TRIGGER ___ ;

- Hay que borrar los triggers después de usarlos para que no se acumulen.
- Un trigger solo puede afectar a una tabla.

```
create or replace trigger t5
after update or delete on emple
for each row
begin
        if updating then
                dbms_output.put_line('El empleado nº' || :old.emp_no || ' ha sido
modificado');
        else
                dbms_output.put_line('El empleado nº' || :old.emp_no || ' ha sido
borrado');
        end if;
end;
```

- Aquí estamos dando la orden para un update o delete y podemos crear una condición para que muestre un mensaje u otro con las diferentes opciones que se pueden disparar.

```
create or replace trigger t6
after update or delete on emple
for each row when (old.dept_no = 10)
begin
        if updating then
                dbms_output.put_line('El empleado nº' || :old.emp_no || ' ha sido
modificado');
        else
                dbms_output.put_line('El empleado nº' || :old.emp_no || ' ha sido
borrado');
        end if;
end;
```

- En esta ocasión el trigger solo se disparará si el departamento del empleado es el nº 10.
- Además en el paréntesis de detrás del when no habrá que poner los **:** para old.

- En caso de que esté borrando no puedo utilizar el :new, ya que no tendrá ningún valor después del borrado.

- El :old y el :new solo se pueden utilizar cuando usemos el **for each row**.

```
create or replace trigger t7
before update on emple
for each row
declare
        v1 number;
begin
        select count(*) into v1
        from emple;
end;
```

- Este trigger da un error, ya que la tabla sobre la que estamos trabajando se bloquea y no podemos acceder a ella.

Crear registros de una tabla

```
declare
        type t_persona is record
        (nombre VARCHAR2(10),
         edad NUMBER(2));

        p1 t_persona;
begin
        p1.nombre := 'Ana';
        p1.edad := 20;

        dbms_output.put_line('Nombre: ' || p1.nombre || '. Edad: ' || p1.edad);
end;
```

- Con type creamos un registro con campos que nosotros indiquemos.
- Con la variable t_persona podemos acceder a los campos del registro y cambiar sus valores.

v_array

```
declare
        type t_persona is record
        (nombre VARCHAR2(10),
         edad NUMBER(2));

        type tipo1 is varray(4) of t_persona;
        v1 tipo1 := tipo1(null, null, null, null);
begin
        v1(1).nombre := 'Ana';
        v1(1).edad := 20;
        v1(2).nombre := 'Carlos';
        v1(2).edad := 19;
        v1(3).nombre := 'Andrés';
        v1(3).edad := 22;
        v1(4).nombre := 'Guillermo';
        v1(4).edad := 20;

        for i in 1..v1.count loop
                dbms_output.put_line('Nombre: ' || v1(i).nombre || '. Edad: ' || v1(i).edad);
        end loop;
end;
```

- Podemos crear un array del tipo de los registros.

v_array dinámico

```
declare
        type t_persona is record
        (nombre VARCHAR2(10),
         edad NUMBER(2));

        type tipo2 is table of t_persona;
        v1 tipo2 := tipo2();
begin
        v1.extend;
        v1(1).nombre := 'Ana';
        v1(1).edad := 20;
        v1.extend;
        v1(2).nombre := 'Carlos';
        v1(2).edad := 19;
        v1.extend;
        v1(3).nombre := 'Andrés';
        v1(3).edad := 22;
        v1.extend;
        v1(4).nombre := 'Guillermo';
```

```
        v1(4).edad := 20;

        for i in 1..v1.count loop
                dbms_output.put_line('Nombre: ' || v1(i).nombre || '. Edad: ' || v1(i).edad);
        end loop;
end;
```

- Cada vez que queramos introducir una nueva posición en el *array* deberemos utilizar el .extend;

Tabla indexada (Vista enlazada)

```
declare
        type t_persona is record
        (nombre VARCHAR2(10),
         edad NUMBER(2));

        type tipo3 is table of t_persona INDEX BY BINARY_INTEGER;
        v1 tipo3;

        i number;
begin
        v1(1).nombre := 'Ana';
        v1(1).edad := 20;

        v1(5).nombre := 'Carlos';
        v1(5).edad := 19;

        v1(10).nombre := 'Andrés';
        v1(10).edad := 22;

        v1(3).nombre := 'Guillermo';
        v1(3).edad := 20;

        i := v1.first;

        while i is not null loop
                dbms_output.put_line('Nombre: ' || v1(i).nombre || '. Edad: ' || v1(i).edad);
                i := v1.next(i);
        end loop;
end;
```

- Para esto tendremos que utilizar el while para mostrarlo. Hay que crear previamente una variable para ello.
- Además hay que añadir INDEX BY BINARY_INTEGER para que funcione. Tras esto solo tendremos que declarar v1 como tipo3.

Paquetes

- **Cabecera (todo lo que esté dentro es público)**

```
create or replace package pk1 as
	type t_persona is record(
		nombre varchar2(10),
		edad number(2));
	procedure mostrar(p1 t_persona);
end;
```

- **Cuerpo**

```
create or replace package body pk1 as
	procedure mostrar(p1 t_persona) is
	begin
		dbms_output.put_line(p1.nombre || ' ' || p1.edad);
	end mostrar;
end pk1;
```

- **Uso**

```
declare
	v1 pk1.t_persona;
begin
	v1.nombre := 'Ana';
	v1.edad := 20;
	pk1.mostrar(v1);
end;
```

Ámbito

```
create table auditoria(
	cambios		varchar2(30)
);
```

```
create or replace package pk1 as
        contador number;
end;

create or replace trigger t1
before insert or update or delete on emple
begin
        pk1.contador := 0;
end;

create or replace trigger t2
after insert or update or delete on emple
for each row
begin
        pk1.contador := pk1.contador + 1;
end;

create or replace trigger t3
after insert or update or delete on emple
begin
        if inserting then
                insert into auditoria values ('Se han insertado ' || pk1.contador || ' filas');
        elsif updating then
                insert into auditoria values ('Se han actualizado ' || pk1.contador || 'filas');
        else
                insert into auditoria values ('Se han borrado ' || pk1.contador || ' filas');
        end if;
end;
```

- Aquí aprovechamos que la variable contador del paquete pk1 se mantiene mientras que dure la sesión, por lo que podemos utilizarlo para incluir en la tabla auditoria el número de filas que se actualizan.

SQL Dinámico

```
create or replace procedure crear_tabla(n_tabla varchar2, n_columna varchar2, longitud
number)
as
        id_cursor number;
        v_comando varchar2(100);
begin
        id_cursor := dbms_sql.open_cursor;
        v_comando := 'CREATE TABLE ' || n_tabla || ' ( '  || n_columna || ' varchar2, '
                                        || longitud || 'number);';
        dbms_sql.parse(ID_CURSOR, V_COMANDO, DBMS_SQL.V7);
        dbms_sql.CLOSE_CURSOR(ID_CURSOR);
end;
```

```
create or replace procedure crear_tabla2(n_tabla varchar2, n_columna varchar2, longitud
number)
as
	v_comando varchar2(100);
begin
	v_comando := 'CREATE TABLE ' || n_tabla || ' ( '  || n_columna || ' varchar2('||
longitud || '))';
	execute immediate v_comando;
end;
```