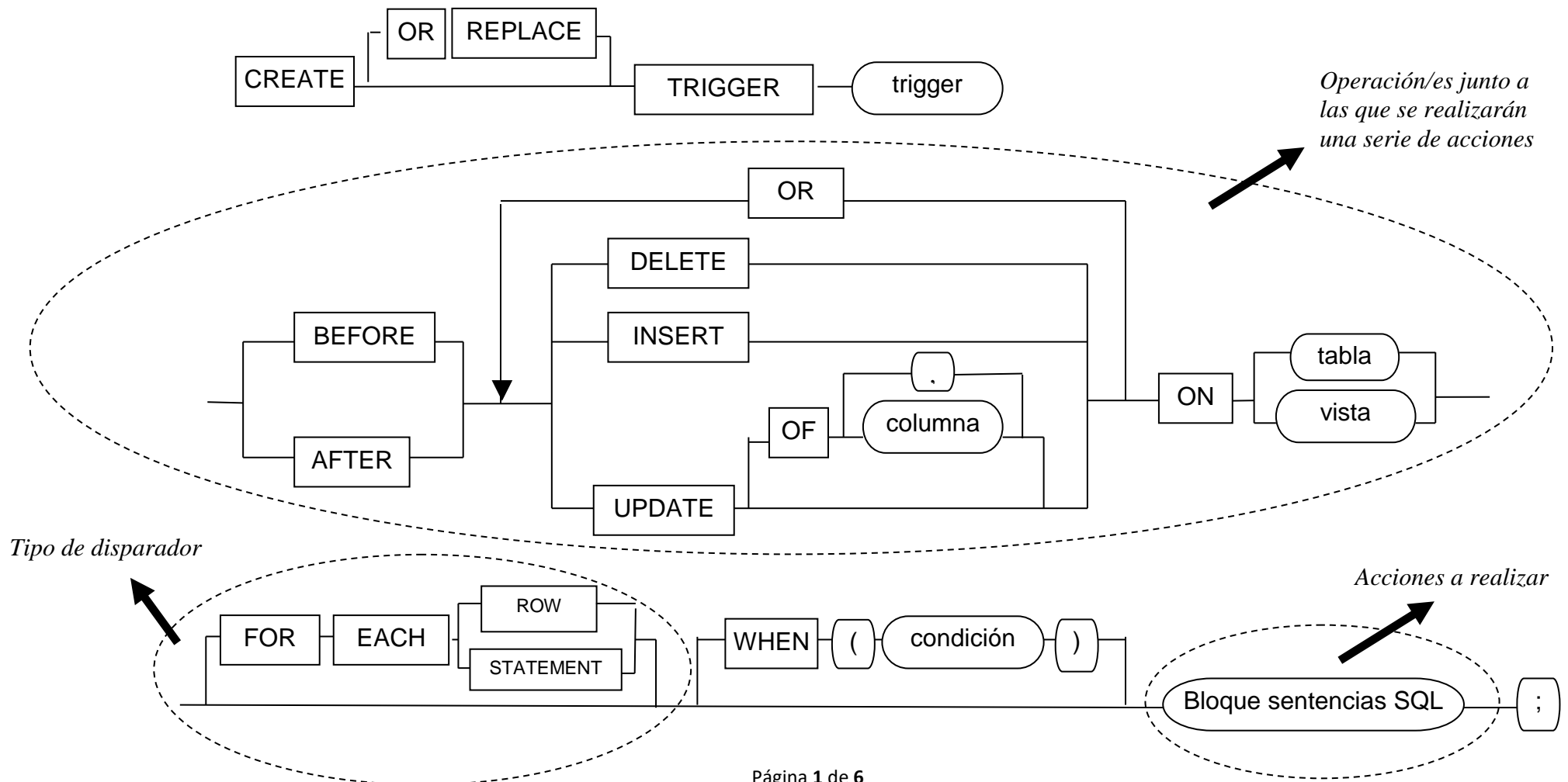


Sesión 11: CREATE TRIGGER

Crear un trigger (o disparador) asociado a una operación de manipulación de datos sobre una tabla T, significa definir una serie de acciones que se desencadenarán automáticamente cuando se realice una operación de inserción, borrado, o actualización sobre la tabla T. Puede ser útil para completar los valores de columnas derivadas (cuyo valor se calcula partiendo del de otras columnas) o para establecer restricciones complejas (no se han podido establecer mediante restricciones CHECK).



¿Cuándo se ejecutan las acciones asociadas a un disparador?

Podemos indicar que estas acciones se lleven a cabo justo antes que la operación sobre la tabla T (BEFORE), o justo después (AFTER).

¿Cuántas veces se ejecutan estas operaciones?

- Si ponemos FOR EACH ROW: una vez por cada fila afectada por la operación que desencadena el trigger .
- Si no ponemos nada o ponemos FOR EACH STATEMENT se ejecutarán sólo una vez por operación.

Dentro del trigger se pueden usar unos predicados condicionales para ejecutar distinto código según el tipo de sentencia que ha activado el trigger.

- IF INSERTING devuelve verdad si el trigger se ha activado por una sentencia INSERT
- IF DELETING devuelve verdad si el trigger se ha activado por una sentencia DELETE
- IF UPDATING devuelve verdad si el trigger se ha activado por una sentencia UPDATE
- IF UPDATING(nom_colu) devuelve verdad si el trigger se ha activado por una sentencia UPDATE y nom_colu se ha actualizado.

Cuando realizamos operaciones **INSERT** o **UPDATE**, Oracle controla **automáticamente** una variable **:new** que nos sirve para referirnos a las nuevas filas insertadas o a las filas después de ser modificadas.

Por ejemplo, si estamos insertando en usuarios y queremos referirnos al identificador de cada fila insertada, dentro de FOR EACH ROW nos referiremos al identificador con :new.identificador .

Al realizar operaciones **DELETE** o **UPDATE**, para hacer referencia a los valores que hemos eliminado utilizaremos **:old**. (:old.identificador)

Orden de ejecución:**BEFORE + FOR EACH STATEMENT trigger**

Antes de iniciar la sentencia que dispara el trigger, y se ejecuta UNA sola VEZ.

BEFORE + FOR EACH ROW trigger

Se ejecuta para cada fila afectada antes de hacer la instrucción sobre la fila.

Permiten manejar el contenido de los campos tanto actuales como futuros de cada tupla afectada

:old.campo -> valor antes de la ejecución del disparador (deletes y updates). SOLO LECTURA

:new.campo -> valor que tomará después de la ejecución (inserts y updates). LECTURA Y MODIFICACIÓN

EJECUCION DE LA SENTENCIA (por fila afectada)**AFTER + FOR EACH ROW trigger**

Se ejecuta para cada fila afectada después de hacer la instrucción en la fila.

Permiten leer el contenido de los campos tanto actuales como futuros de cada tupla afectada

:old.campo -> valor antes de la ejecución del disparador (deletes y updates). SOLO LECTURA

:new.campo -> valor que tomará después de la ejecución (inserts y updates). SOLO LECTURA

AFTER + FOR EACH STATEMENT trigger

Después de finalizar toda la sentencia que dispara el trigger, y se ejecuta UNA sola VEZ

Ejemplo 1

Triggers para controlar que la generalización de RECURSO sea DISJUNTA.

Se muestra como ejemplo los triggers que controlan que los recursos que insertemos en RECURSO no estén simultáneamente en las tablas RECURSO_GRATUITO y RECURSO_PAGO.

```
create or replace trigger disj_recurso_grt
before insert or update of codigo
on RECURSO_GRATUITO
for each row
declare cuantos number :=0;
begin
  select count(*) into cuantos from RECURSO_PAGO where codigo=:new.codigo;
  if (cuantos=1) then
    raise_application_error(-20601,'El recurso ' || :new.codigo || ' es un recurso de pago');
  end if;
end;

create or replace trigger disj_recurso_pg
before insert or update of codigo
on RECURSO_PAGO
for each row
declare cuantos number :=0;
begin
  select count(*) into cuantos from RECURSO_GRATUITO where codigo=:new.codigo;
  if (cuantos=1) then
    raise_application_error(-20601,'El recurso ' || :new.codigo || ' es un recurso gratuito');
  end if;
end;
```

Al crear un trigger, **si se ha creado con errores de compilación debéis corregirlos.**

Para consultar los errores cometidos podéis proceder como con las funciones y los procedimientos

Algunas observaciones:

- ❑ No debemos ejecutar el código de creación de un disparador combinado con otras sentencias (crear tablas, insert ...) , nos pueden dar errores de sintaxis.
- ❑ Dentro de un disparador no se pueden ejecutar algunas sentencias como CREATE TABLE.
- ❑ Cuando utilicemos IF UPDATING(nom_colum), el nombre de la columna debe ir entre comillas simples.
- ❑ Si ponemos WHEN condición, en la condición si usamos las variables :old y :new, éstas aparecen sin los :

Ejemplo 2

Imaginemos que tenemos una tabla

COMPROBAR_ESTUDIOS(NIF char(9), estudios varchar(50))

C.P.: NIF

Clave ajena: NIF → EMPLEADO

Queremos que cuando se inserten los datos de un empleado y tenga valor en la columna estudios, insertemos en la tabla COMPROBAR_ESTUDIOS el NIF y los estudios que indica el empleado.

create or replace trigger control_estudios

after insert on empleado

for each row

when (new.estudios is not null)

begin

insert into comprobar_estudios values(:new.nif, :new.estudios);

end;

Comprueba lo que ocurre si en
lugar de AFTER ponemos BEFORE

Otras operaciones con TRIGGERS

- Se puede borrar un trigger con la sentencia:

```
drop trigger nombretigger;
```

- Se pueden desactivar temporalmente:

```
alter trigger nombretigger disable;  
alter table nombretable disable all triggers;
```

- Y volver a activarlos:

```
alter trigger nombretigger enable;  
alter table nombretable enable all triggers;
```

Se pueden definir varios triggers sobre la misma tabla

- Hasta la versión 10g de Oracle no se podía conocer el orden en el que se disparaban. A partir de la versión 11g, podemos definir en un TRIGGER cuál es su precedente, usando la opción **FOLLOWS**.

