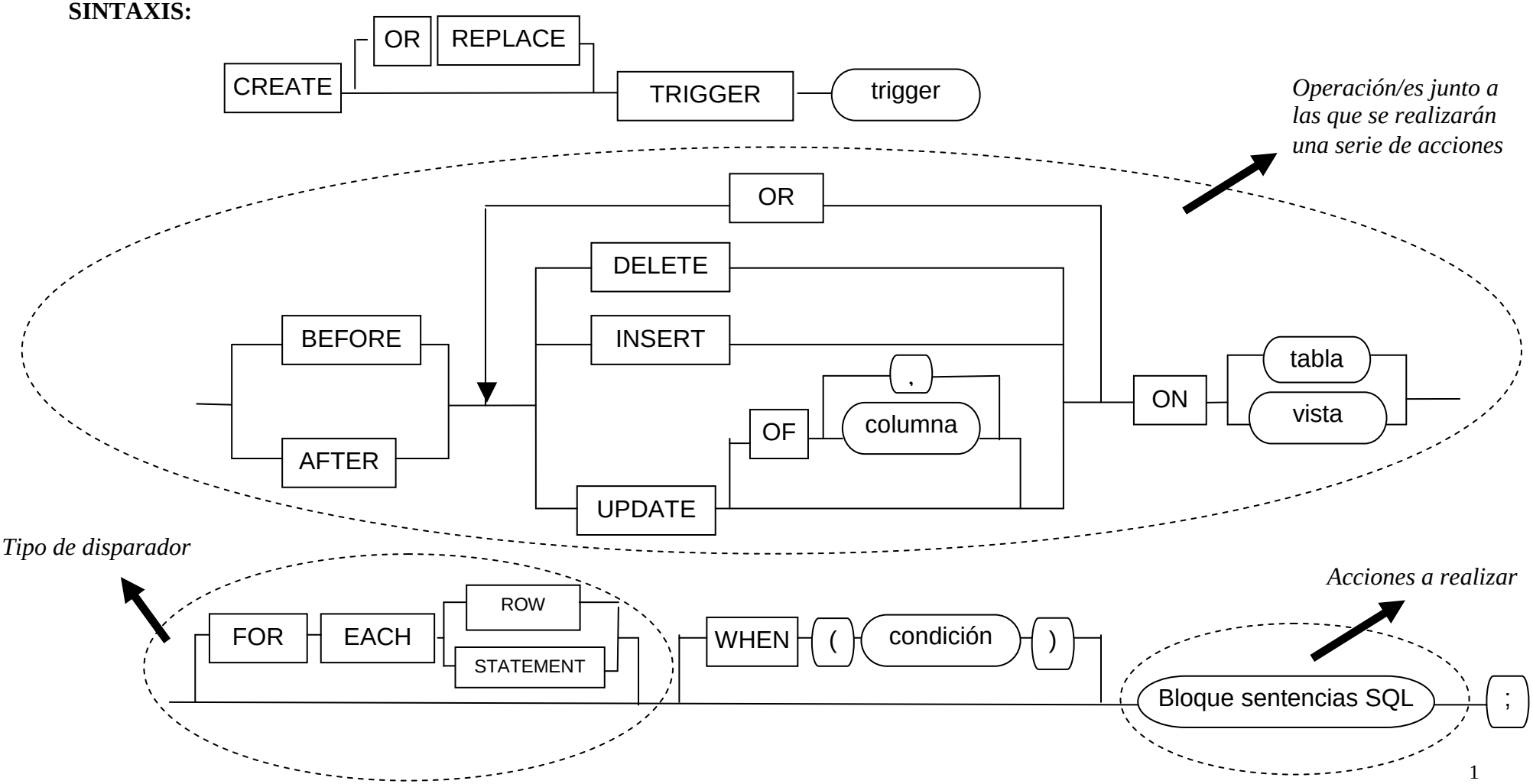
DEFINICIÓN DE DISPARADORES EN PL/SQL : CREATE TRIGGER (I)

Crear un disparador (trigger) 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 esa operación 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).

SINTAXIS:



¿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 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 votantes y queremos referirnos al dni de cada fila insertada, dentro de FOR EACH ROW nos referiremos al dni con :new.dni .

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

Ejemplo 1

Trigger para controlar que la generalización de EMPLEADO sea DISJUNTA.
Se muestra como ejemplo el trigger que controla que los empleados que insertemos en ***emprecepcion*** *no estén ya en las tablas* ***empanimacion***, ***emplimpieza***, ***emprestaurante*** *y* ***empservicios****. Para reflejar que esa generalización es disjunta habría que completar con triggers similares en las tablas* ***empanimacion***, ***emplimpieza***, ***emprestaurante*** *y* ***empservicios.***

```
create or replace trigger disj_emprecepcion
before insert or update of nif
on emprecepcion
for each row
declare cuantos number(1):=0;
begin
cuantos:=0;
select count(*) into cuantos from empanimacion where nif=:new.nif;
if (cuantos=1)
  then raise_application_error(-20601,'El nif '||:new.nif || ' ya es empleado de animación');
end if;
select count(*) into cuantos from emplimpieza where nif=:new.nif;
if (cuantos=1)
  then raise_application_error(-20601,'El nif '||:new.nif || ' ya es empleado de limpieza');
end if;
select count(*)  into cuantos from emprestaurante where nif=:new.nif;
if (cuantos=1)
  then raise_application_error(-20601,'El nif '||:new.nif || ' ya es empleado de restaurante');
end if;
select count(*)  into cuantos from empservicios where nif=:new.nif;
if (cuantos=1)
then raise_application_error(-20601,'El nif '||:new.nif || ' ya es empleado de servicios');
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 uitilicemos 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