Programació PL/SQL

Gestió i Administració de Bases de Dades.
Grau en Enginyeria Informàtica

Oriol Ramos Terrades

Carles Sánchez Ramos

Departament de Ciències de la Computació

Continguts

- Introducció.
- Sinònims, seqüencies i índexs.
- Procediments.
- Funcions.
- Disparadors (triggers).

Introducció

Ampliació de comandes SQL més enllà de consultes, actualitzacions, etc.

Sintaxi estàndard SQL.

Bases de Dades Actives: Comandes que activen sentències SQL (processos) quan succeeixen certs esdeveniments en el contingut de la BD.

Sinònim

Objectes del sistema que apunten a altres objectes (taules, vistes, programes, etc.).

Sinònims poden ser públics o privats:

- **Públic:** Són vistos per a tots els usuaris de la BD.
- **Privats:** Només pot ser vist dins l'esquema d'un usuari i sols serà visible a qui el vulgui.

```
CREATE [PUBLIC] SYNONIM nom_S FOR [esquema.]nom_objecte;
```

Seqüència

- Objecte per generar valors que s'incrementen automàticament.
- Es pot definir un valor màxim, poden ser seqüencies circulars i estar ordenades.

```
CREATE SEQUENCE SEQUENCE1
INCREMENT BY 1
START WITH 1
MAXVALUE 100
CYCLE
ORDER;
```

Índexs

- Estructura d'accés a fitxers per a un accés més ràpid.
- Tipus d'índexs:
- Les extensions es van definint, tal com s'hagi especificat en la **clàusula STORAGE**.

```
CREATE   INDEX   nom_index   ON   nom_taula   (atribut_1
  [ASC|DESC],...)
[LOGGING | NOLOGGING]
TABLESPACE nom_tablespace
STORAGE
(
   INITIAL mida_inicial_en_bytes
   NEXT nombre_bytes_extensio
   MINEXTENTS valor_en_bytes
   MAXEXTENTS [ valor_maxi_en_bytes | UNLIMITED ]
   BUFFER_POOL DEFAULT
);
```

Procediments

Paràmetres

- Hi ha 3 tipus:

  IN - Entrada. Es pot referenciar dins el procediment però el procediment no el pot modificar.

  OUT - Sortida: El procediment no el pot referenciar però el pot modificar.

  IN OUT - Entrada/Sortida. Es pot referenciar dins el procediment i el pot modificar.

Sintaxis

```
CREATE [OR REPLACE]
PROCEDURE
nom_procediment [
(parametre [,parametre])
]
IS
   [secció_declaracio]
BEGIN
    secció_executable
[EXCEPTION
   secció_excepció]
END [nom_procediment];
```

Funcions

Paràmetres

- Hi ha 3 tipus:

  IN - Entrada. Es pot referenciar dins el procediment però el procediment no el pot modificar.

  OUT - Sortida: El procediment no el pot referenciar però el pot modificar.

  IN OUT - Entrada/Sortida. Es pot referenciar dins el procediment i el pot modificar.

- Retorn:

  Es defineix el tipus de retorn a la capçalera

Sintaxis

```
CREATE [OR REPLACE]
FUNCTION nom_funcio [
(parametre [,parametre]) ]
RETURN TYPE
IS
   valor_retorn TYPE;
   [secció_declaracio]
BEGIN
    secció_executable
   RETURN valor_retorn;
[EXCEPTION
   secció_excepció]
END [nom_funció];
```

Disparadors (*trigger*)

Sentència que el sistema executa automàticament degut a la modificació de la BD.

Model Esdeveniment-Condició-Acció (ECA):

- **Event (Esdeveniment):** L'operació que causa l'activació del disparador.
- **Condició:** La que cal que es compleixi sobre l'esdeveniment per a que s'activi el disparador.
- **Acció:** Sentències SQL que s'han d'executar si la condició sobre l'esdeveniment és certa.

Disparadors (*trigger*)

Dades persistents i accessibles per a totes les operacions de la BD.

Regles actives: Nom que dona Microsoft als disparadors.

BD actives: BD que suporten disparadors i altres accions quan succeeixen els esdeveniments.

Disparadors (*trigger*)

Aspectes a tenir en compte:

1. Necessitat dels *triggers*
2. *Triggers* en SQL
3. Quan no s'han d'utilitzar els *triggers*
4. Exemples de *triggers*
5. Taules mutants

Necessitat dels *triggers*

Mecanisme per a alertar als usuaris de certs canvis en la BD.

Fer determinades tasques quan es compleixen certes condicions.

S'utilitzen *triggers* per a:

- Forçar regles d'integritat complexes que no es poden definir declarant claus externes.
- Fer canvis de la BD de forma transparent a l'usuari.
- Generar automàticament valors d'atributs derivats d'un altra atribut, el valor del qual s'ha definit amb INSERT o modificat amb UPDATE.
- Sincronitzar el manteniment de taules duplicades localitzades en nodes remots d'una BD distribuïda.

Necessitat dels *triggers*

- **Exemple 1:** Entitat bancària que quan té un compte amb saldo negatiu obre un préstec al client per a la quantitat del deute i posa el saldo del compte a 0.
- Suposem que el client Pepe treu diners i deixa el compte al descobert. Sigui t la tupla del compte amb saldo negatiu. Les accions a fer són:
  1. Inserir una nova tupla `s` a la relació `Prèstec`:

     ```
     s[nom_sucursal] = t[nom_sucursal]
     s[num_compta] = t[num_compta]
     s[saldo] = -t[saldo]
     ```

  2. Inserir una nova tupla `u` a la relació `Prestatari`

     ```
     u[nom_client] = 'Pepe'
     u[num_prestec] = t[num_compta]
     ```

  3. Fer que el `t[saldo]=0`

Necessitat dels *triggers*

Exemple 2: Per a mantenir un mínim d'inventari en un magatzem, si hi ha un producte que està per sota del mínim d'unitats establert es fa un avís per a fer una comanda.

- **Event:** Quan es modifica el nombre d'unitats (nombre, volum, pes) d'un producte.
- **Condició:** Si el nombre d'unitats actualitzat és inferior al mínim establert per al producte.
- **Acció:** Afegir una nova entrada a la taula Comandes.

Forma d'avisar és inserint tuples en una taula, per a després tenir un procés que miri la taula Comandes i si hi ha tuples crear un procés de comanda de producte.

Triggers en SQL

Gestió de *triggers*:

1. Crear *trigger*
2. *Activar trigger*
3. Modificar *trigger*
4. Esborrar *trigger*

Crear *trigger*

- Sintaxi:

<trigger> ::= **CREATE TRIGGER** <nombre del trigger>
(**AFTER** | **BEFORE**) <eventos de activación> **ON** <nombre de la tabla>
[**FOR EACH ROW**]
[**WHEN** <condición>]
<acciones del trigger> ;

<eventos de activación> ::= <evento trigger> {**OR** <evento trigger> }

<evento trigger> ::= **INSERT** | **DELETE** | **UPDATE** [**OF** <nombre columna> { ,
<nombre columna> }]

<acción trigger> ::= <bloque PL/SQL>

Crear *trigger*

Sintaxi (2):

```
CREATE [OR REPLACE] TRIGGER <nom> [AFTER|BEFORE] <event> ON
  <taula>
[REFERENCING [OLD ROW AS <nomO>] [NEW ROW AS <nomN>] ]
FOR EACH ROW
[DECLARE <nom_varible> <tipus_variable>;]
[WHEN <condició>]
BEGIN
[<sentències SQL>]
[IF {DELETING} {OR UPDATING} {OR INSERTING} {OR <condicio>
  THEN} BEGIN <sentències SQL> ENDIF;]
END;]
```

Crear *trigger*

Trigger exemple 1:

```
1: CREATE TRIGGER desc AFTER UPDATE ON Compta
2: REFERENCING NEW ROW AS nFila
3: FOR EACH ROW
4: WHEN nFila.Saldo<0
5: BEGIN
6:   INSERT INTO Prestatari
7:    (SELECT Num_Client, Num_Compta
8:     FROM Impositor
9:     WHERE nFila.Num_Compta=Impositor.Compta);
10:  INSERT INTO Prestec VALUES
11:            (nFila.Num_Compta,nFila.Nom_Sucursal,-
 nFila.Saldo);
12:  UPDATE Compta SET Saldo = 0
13:    WHERE Compta.Num_Compta=nFila.Num_Compta;
14: END;
```

Crear *trigger*: Observacions

`AFTER|BEFORE`: Temporalitat a l'hora de definir l'esdevenimet. Si és abans (`BEFORE`) o després (`AFTER`) de fer l'operador.

`BEFORE`: Evita que la BD perdi integritat.

- Exemple: Si no es permeten descoberts, `BEFORE` pot evitar que el saldo sigui negatiu quan es produeix la condició de l'esdeveniment.
- Exemple: Abans de posar a buit el camp d'un número de telèfon, s'actualitza a valor NULL.

Crear *trigger*: Observacions

`AFTER UPDATE ON COMPTA`: Definició de l'esdeveniment. Després d'actualitzar la taula Compta.

Tres operadors per l'esdeveniment: `INSERT` | `DELETE` | `UPDATE`

- Exemple `DELETE`: Si s'esborra un client, es mira si té entrades comprades i s'esborren també.

Es pot especificar atributs d'actualització de l'esdeveniment de forma explícita:

```
CREATE TRIGGER descobert
AFTER UPDATE OF Saldo ON Compta
```

Crear *trigger*: Observacions

`REFERENCING NEW ROW AS nFila`: Crea una variable de transició nFila que emmagatzema la tupla **JA** actualitzada. Per defecte: `:NEW.<atribut>`

`REFERENCING OLD ROW AS nFila`: Crea una variable de transició nFila que emmagatzema la tupla **ABANS DE SER** actualitzada. Per defecte: `:OLD.<atribut>`

Crear *trigger*

Trigger exemple 1 sense REFERENCING:

```
1: CREATE TRIGGER desc AFTER UPDATE ON Compta
2:
3: FOR EACH ROW
4: WHEN NEW.Saldo<0
5: BEGIN
6:   INSERT INTO Prestatari
7:    (SELECT Num_Client, Num_Compta
8:     FROM Impositor
9:     WHERE :NEW.Num_Compta=Impositor.Compta);
10:  INSERT INTO Prestec VALUES
11:             (:NEW.Num_Compta,nFila.Nom_Sucursal,-
  :NEW.Saldo);
12:  UPDATE Compta SET Saldo = 0
13:    WHERE Compta.Num_Compta=:NEW.Num_Compta;
14: END;
```

Crear *trigger*: Observacions

`3: FOR EACH ROW`: S'especifica que l'acció s'aplicarà per a cada tupla de la taula.

`FOR EACH STATEMENT`: S'especifica una acció única.

- `REFERENCING OLD TABLE, REFERENCING NEW TABLE AS`: Utilitzades per a fer referència a les taules temporals que contenen les tuples afectades (taules de transició).
- Taules de transició sols utilitzades per a disparadors `AFTER`, no `BEFORE`.

Crear *trigger:* Observacions

`4: WHEN <condició>:` S'especifica la condició.

`5-14: BEGIN ... END`: S'especifiquen les instruccions SQL que corresponen a les accions a realitzar si es compleixen les condicions de l'esdeveniment.

Crear *trigger*

Exemple 2. Taules:

- **Inventari (Producte, Q1):** Quantitats actuals d'un producte.
- **Nivell_Minim (Producte, Q2):** Llindars mínims a mantenir de cada producte.
- **Nova_Comanda (Producte, Q3):** Quantitat de producte a demanar quan el nivell cau a mínim.
- **Comanda (Producte, Q4):** Quantitat de producte a demanar.

Crear *trigger*

Trigger exemple 2:

```
1:  CREATE  TRIGGER  n_c  AFTER  UPDATE  OF  Quant  ON
 Inventari
2: REFERENCING OLD ROW As oFila, NEW ROW AS nFila
3: FOR EACH ROW
4: WHEN nFila.Q1<=(SELECT Nivell
5:                    FROM Nivell_Minim NM
6:                  WHERE NM.Producte=nFila.Producte)
7:       AND
8:       oFila.Q1>(SELECT Nivell
9:                   FROM Nivell_Minim NM
10:                  WHERE NM.Producte=oFila.Producte)
11: BEGIN
12:   INSERT INTO Comandes
13:    (SELECT Producte, Q3
14:     FROM Nova_Comanda NC
15:     WHERE NC.Producte=oFila.Producte);
```

Crear *trigger*

Crear variables per emmagatzemar dades SQL (clàusula `INTO`, no `AS`).

Sintaxi:

```
DECLARE <nom_variable> <tipus_variable>
```

Exemple:

```
DECLARE Num_Clients INTEGER;
(SELECT COUNT(*) INTO Num_Clients
 FROM Clients
 WHERE Ciutat='Barcelona');
```

Crear *trigger*

Missatge d'error en pantalla.

Sintaxi:

```
RAISE_APPLICATION_ERROR(<codi_error>,<nom_atribut>      ||
  'text');
```

Exemple:

```
CREATE OR REPLACE TRIGGER T4 BEFORE UPDATE ON Reserves_Hotels
FOR EACH ROW
DECLARE NumHabsTot INTEGER;
BEGIN
   SELECT SUM(Num_Habs_Ind) INTO NumHabsTot
   FROM Hotels
   WHERE Codi_Hotel=:NEW.Codi_Hotel;
   IF NumHabsTot + :NEW.Num_Habs_Ind > 50 THEN
              RAISE_APPLICATION_ERROR(-       20600,    'Hotel'    ||
  :NEW.Codi_Hotel || 'amb  habitacions individuals majors que 50');
   ENDIF;
END;
```

Activar *trigger*

Un trigger pot estar activat (*enabled*) o desactivat (*disabled*). Quan està desactivat no actua.

Sintaxi:

```
ALTER TRIGGER <nom_trigger> ENABLE;

ALTER TRIGGER <nom_taula> ENABLE ALL TRIGGERS;

ALTER TRIGGER <nom_trigger> DISABLE;

ALTER TRIGGER <nom_taula> DISABLE ALL TRIGGERS;
```

Modificar *trigger*

Sintaxi:

```
ALTER TRIGGER <nom> [AFTER|BEFORE] <event> ON
 <taula>
[REFERENCING {OLD ROW AS <nomO>} {NEW ROW AS <nomN>} ]
FOR EACH ROW
[DECLARE <nom_varible> <tipus_variable>;]
[WHEN <condició>]
BEGIN
[<sentències SQL>]
[IF {DELETING} {OR UPDATING} {OR INSERTING} {OR
 <condició> THEN} BEGIN <sentències SQL> ENDIF;]
END;]
```

Esborrar *trigger*

Sintaxi:

```
DROP TRIGGER <nom>;
```

Quan no utilitzar triggers

- **Per obtenir valors globals actualitzats.** Exemples:
  - Sou dels empleats d'un departament.
  - Número de treballadors d'una empresa

  Vistes materialitzades resolen el problema.
- **Replicar la BD:** Actualment les BD tenen eines per a replicar la BD.
- **Compte amb els triggers:** han d'estar ben escrits per evitar accions no desitjades.

Quan no utilitzar *triggers*

- **Cadena infinita de dispars:** Quan l'acció del disparador és un esdeveniment d'un altre disparador:

$$D1 \rightarrow D2 \rightarrow D3 \rightarrow .. \rightarrow Dn \rightarrow D1$$

Limitació de cadena de disparadors fins a 16 o 32.

Exemples de *triggers*

BD empresa:

EMPLEADO

Nombre	Dni	Sueldo	Dno	SuperDni

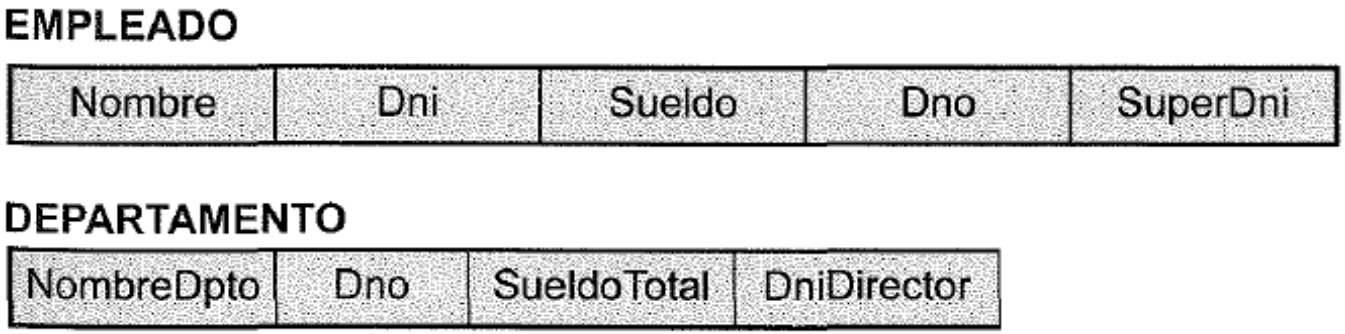
DEPARTAMENTO

NombreDpto	Dno	SueldoTotal	DniDirector

Exemple 1: ATRIBUT *SueldoTotal* **DERIVAT** de la suma de tots els sous dels empleats assignats al departament.

Exemple 2: Comprovar si el sou d'un empleat és major que el sou del seu supervisor → **RESTRICCIÓ.**

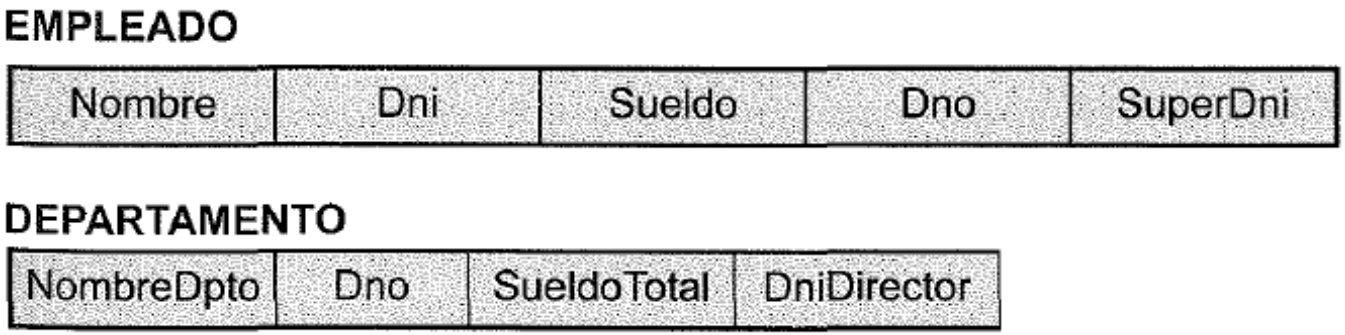
Exemple d'atribut derivat



Manteniment de l'atribut derivat mitjançant regla activa. Cal determinar els **esdeveniments** que poden causar el canvi de l'atribut *SueldoTotal*:

1. Inserció (d'un o més) empleats.
2. Modificació del sou d'un o més empleats existents.
3. Trasllat d'un empleat d'un departament a un altre.
4. Donar de baixa un empleat.

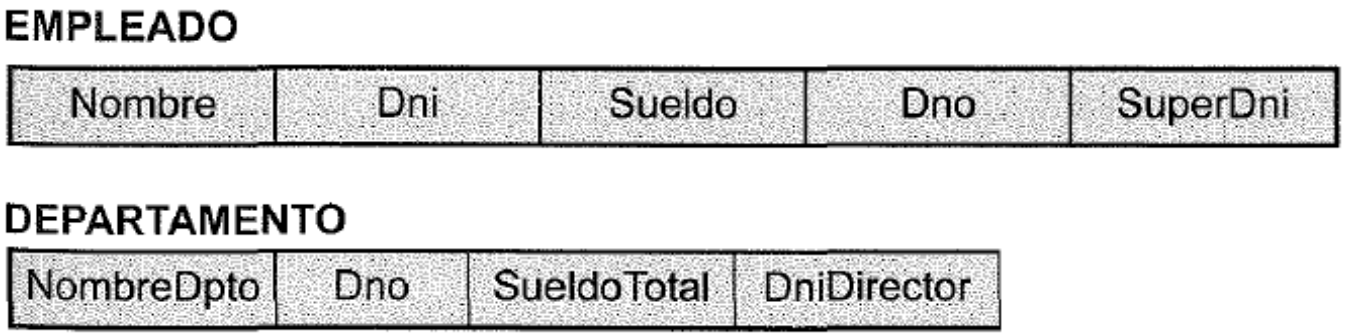
Exemple d'atribut derivat



Condicions dels esdeveniments:

1. Tornar a calcular *SueldoTotal* si l'empleat nou és assignat a un departament (`Empleado.Dno<>NULL`).
2. Condició (`Empleado.Dno<>NULL`) per a l'empleat que li hagin canviat el sou.
3. Sense condició.
4. Condició (`Empleado.Dno<>NULL`) per a l'empleat que es doni de baixa.

Exemple d'atribut derivat



- **Accions** segons els esdeveniments:
  - 1,2,4. Actualitzar *SueldoTotal* del departament de l'empleat actualitzat.
  - 3. Dues accions:
    1. Actualitzar *SueldoTotal* del departament antic de l'empleat.
    2. Actualitzar *SueldoTotal* del departament nou de l'empleat.

Exemple d'atribut derivat

Implementació: 4 regles actives (R1,R2,R3,R4) amb Oracle:

```
R1:   CREATE TRIGGER SouT1 AFTER INSERT ON Empleado
  FOR EACH ROW
  WHEN (NEW.Dno IS NOT NULL)
  UPDATE Departamento
  SET SueldoTotal = SueldoTotal + :NEW.Sueldo
  WHERE Dno = :NEW.Dno;
```

```
R2:   CREATE TRIGGER SouT2 AFTER UPDATE OF Sueldo ON Empleado
  FOR EACH ROW
  WHEN (NEW.Dno IS NOT NULL)
  UPDATE Departamento
  SET SueldoTotal = SueldoTotal + :NEW.Sueldo - :OLD.Sueldo
  WHERE Dno = :NEW.Dno;
```

Exemple d'atribut derivat

Implementació: 4 regles actives (R1,R2,R3,R4) amb Oracle:

```
R3:   CREATE TRIGGER SouT3 AFTER UPDATE OF Dno ON Empleado
  FOR EACH ROW
  BEGIN
    UPDATE Departamento
    SET SueldoTotal = SueldoTotal + NEW.Sueldo
    WHERE Dno = NEW.Dno;
    UPDATE Departamento
      SET SueldoTotal = SueldoTotal - OLD.Sueldo
      WHERE Dno=OLD.Dno;
  END;
```

Exemple d'atribut derivat

Implementació: 4 regles actives (R1,R2,R3,R4) amb Oracle:

```
R4:   CREATE TRIGGER SouT4 AFTER DELETE ON Empleado
  FOR EACH ROW
  WHEN (OLD.Dno IS NOT NULL)
      UPDATE Departamento
      SET SueldoTotal = SueldoTotal - :OLD.Sueldo
      WHERE Dno = :OLD.Dno;
  END;
```

Exemple de restricció

EMPLEADO

Nombre	Dni	Sueldo	Dno	SuperDni

DEPARTAMENTO

NombreDpto	Dno	SueldoTotal	DniDirector

- Comprovar que el sou d'un empleat és major que el del seu supervisor. Cal determinar els **esdeveniments** que poden causar l'activació de la regla:
  1. Inserció (d'un o més) empleats.
  2. Modificació del sou d'un o més empleats existents.
  3. Modificació del supervisor d'un empleat.

Exemple de restricció

Implementació: Una regla activa (R5) amb Oracle:

```
R5:   CREATE TRIGGER Inform_Superv
  BEFORE INSERT OR UPDATE OF Sueldo,SuperDni ON Empleado
  FOR EACH ROW
  WHEN (NEW.Sueldo > ( SELECT Sueldo
                       FROM Empleado
                       WHERE Dni = NEW.SuperDni)
       )
  inform_supervisor (:NEW.SuperDni,:NEW.Dni);
```

Taules mutants

Quan es defineix un trigger a nivell de fila (`FOR EACH ROW`) sobre una taula o atribut i dins el cos del trigger es modifica la mateixa taula o atribut.

Exemple *Reserves_Viatges(Codi_Viatge,NIF_Client,NumPlaces)→Clients(NIF,Nom)*:

```
CREATE OR REPLACE TRIGGER T BEFORE UPDATE ON Clients
FOR EACH ROW
BEGIN
 INSERT INTO Clients
         (SELECT :NEW.NIF,... FROM ... WHERE NIF=:OLD.NIF)
 UPDATE Reserves_Viatges
  SET NIF_Client=:NEW.NIF
  WHERE NIF=:OLD.NIF;
 DELETE
  FROM Clients
  WHERE NIF=:OLD.NIF;
END;
```

Trigger sobre la taula **Clients** i dins el seu cos inserta o elimina tuples a la mateixa taula **Clients**.

Taules mutants: solució

Exemple *Reserves_Viatges(Codi_Viatge,NIF_Client,NumPlaces)→Clients(NIF,Nom)*:

```
CREATE OR REPLACE TRIGGER T BEFORE UPDATE ON Clients
FOR EACH ROW
PRAGMA AUTONOMOUS_TRANSACTION;
BEGIN
 INSERT INTO Clients
         (SELECT :NEW.NIF,... FROM ... WHERE NIF=:OLD.NIF)
 UPDATE Reserves_Viatges
  SET NIF_Client=:NEW.NIF
  WHERE NIF=:OLD.NIF;
 DELETE
  FROM Clients
  WHERE NIF=:OLD.NIF;
END;
```

Trigger sobre la taula **Clients** i dins el seu cos inserta o elimina tuples a la mateixa taula **Clients**.

Dubtes i preguntes?