max_2_ProvPieza_v1.sql

```
/* @max_2_provPieza_v1.sql */
SET ECHO ON
SET SERVEROUTPUT ON

/* Crear un disparador para impedir que el nº de Proveedores por pieza sea > 2
*/
CREATE OR REPLACE TRIGGER tst_maxProv_TR
BEFORE INSERT ON Suministrar
FOR EACH ROW
DECLARE
  totProv number;
BEGIN
  SELECT count(*) INTO totProv
  FROM Suministrar
  WHERE clvPieza = :NEW.clvPieza;
  DBMS_OUTPUT.PUT_LINE('hay '||totProv||' proveedores de la pieza '||:NEW.clvPieza);
  IF totProv > 1 THEN
    raise_application_error( -20501, 'la pieza tiene el máximo de proveedores');
  END IF;
END tst_maxProv_TR;
/
SHOW ERRORS TRIGGER tst_maxProv_TR
SELECT * FROM Suministrar ORDER BY clvProv, clvPieza;

-- no se producirá error, pues suministrar no es mutante ¡en este caso particular! (una tupla)
INSERT INTO suministrar VALUES (2, 95);

-- el disparador aborta la transacción normalmente (suministrar no es mutante en este caso)
INSERT INTO suministrar VALUES (3, 95);

-- SÍ se producirá error, pues suministrar es mutante (se intenta insertar más de una tupla)
INSERT INTO suministrar SELECT 3, clvPieza FROM Suministrar WHERE clvProv=2;

-- Si el disparador se define como AFTER, la tabla siempre será mutante
/* Crear un disparador para impedir que el nº de Proveedores por pieza sea > 2
*/
CREATE OR REPLACE TRIGGER tst_maxProv_TR
AFTER INSERT ON Suministrar
FOR EACH ROW
DECLARE
  totProv number;
BEGIN
  SELECT count(*) INTO totProv
  FROM Suministrar
  WHERE :NEW.clvPieza = clvPieza;
  DBMS_OUTPUT.PUT_LINE('hay '||totProv||' proveedores de la pieza '||:NEW.clvPieza);
  IF totProv > 1 THEN
    raise_application_error( -20501, 'la pieza tiene el máximo de proveedores');
  END IF;
END tst_maxProv_TR;
/
SHOW ERRORS TRIGGER tst_maxProv_TR

DELETE FROM suministrar WHERE (clvProv, clvPieza) IN ((2,95));

-- se producirá error, pues suministrar es mutante dentro del disparador (se ha cambiado)
INSERT INTO suministrar VALUES (2, 95);

DROP TRIGGER tst_maxProv_TR;

/* Un modo bastante sencillo y general de resolver el problema de las tablas mutantes consiste en
implementar el tratamiento en base a dos disparadores: uno a nivel de tupla en el que simplemente
se guardan las tuplas modificadas por el evento en una estructura auxiliar (por ejemplo una tabla),
y otro a nivel de sentencia en el que se tratan las tuplas guardadas en la estructura auxiliar.
Como este disparador funciona a nivel de sentencia, la tabla ya no es mutante y, por eso, en
el tratamiento se pueden realizar consultas sobre la tabla modificada por el evento.
Para recorrer las tuplas guardadas se puede utilizar un cursor.
*/
-- como ejemplo, se va a resolver el problema utilizando esta técnica:
```

```
/* Crear una tabla auxiliar para guardar las tuplas modificadas por el evento
*/
CREATE TABLE SumPend (
  clvProv  number(9),
  clvPieza number(9)
);

/* Crear un disparador para guardar en la tabla auxiliar los cambios en Suministrar
*/
CREATE OR REPLACE TRIGGER tst_maxProv_TR
BEFORE INSERT OR UPDATE ON Suministrar
FOR EACH ROW
BEGIN
  INSERT INTO SumPend VALUES(:NEW.clvProv, :NEW.clvPieza);
  DBMS_OUTPUT.PUT_LINE('intento de añadir suministro ' || :NEW.clvProv || ' ' || :NEW.clvPieza);
END tst_maxProv_TR;
/
SHOW ERRORS TRIGGER tst_maxProv_TR

/* Crear un disparador a nivel de sentencia para procesar los cambios guardados en tabla SumPend
*/
CREATE OR REPLACE TRIGGER nuevosSuministros_TR
AFTER INSERT OR UPDATE ON Suministrar
DECLARE
  laPieza     Suministrar.clvPieza%TYPE;
  elProveedor Suministrar.clvProv%TYPE;
  totalProv   number;
  CURSOR selCambio IS SELECT clvPieza, clvProv FROM SumPend;
BEGIN
  OPEN selCambio;
  LOOP
    FETCH selCambio INTO laPieza, elProveedor;
    EXIT WHEN selCambio%NOTFOUND;
    SELECT count(*) INTO totalProv
      FROM Suministrar S
      WHERE S.clvPieza = laPieza;
    IF totalProv > 2 THEN
      DBMS_OUTPUT.PUT_LINE('la pieza '|| laPieza || ' ya tiene '|| (totalProv-1) || '
proveedores');
      raise_application_error( -20501, 'número máximo de proveedores alcanzado');
    END IF;
  END LOOP;
  DELETE FROM SumPend;
END tst_maxProv_TR;
/
SHOW ERRORS TRIGGER nuevosSuministros_TR

-- no se producirá error)
INSERT INTO suministrar VALUES (2, 95);

-- el disparador aborta la transacción normalmente
INSERT INTO suministrar VALUES (3, 95);
INSERT INTO suministrar SELECT 3, clvPieza FROM Suministrar WHERE clvProv=2;

-- dejar la tabla suministrar como estaba antes de ejecutar el ejemplo
DELETE FROM suministrar WHERE (clvProv, clvPieza) IN ((2,95));
DELETE FROM suministrar WHERE clvProv = 3;

SELECT * FROM Suministrar ORDER BY clvProv, clvPieza;

-- eliminar la tabla auxiliar
DROP TABLE SumPend;

-- eliminar los disparadores creados
DROP TRIGGER nuevosSuministros_TR;
DROP TRIGGER tst_maxProv_TR;
```