max_2_ProvPieza_v3.sql

```
/* @max_2_ProvPieza_v3.sql */
SET ECHO ON
SET SERVEROUTPUT ON
SET linesize 132
SET pagesize 300
column color   format A32
column numProv format 999

/* Añadir a la tabla Pieza un atributo para el total de proveedores de la pieza
   con la restricción de que dicho valor sea >=0 y por defecto tome el valor 0.
*/
ALTER TABLE Pieza ADD(numProv number DEFAULT 0 CONSTRAINT numProv_NoNeg CHECK(numProv>=0));

/* Crear un procedimiento para calcular el total de proveedores de cada pieza
*/
CREATE OR REPLACE PROCEDURE set_numProvPieza_PR
IS
BEGIN
  UPDATE Pieza P SET numProv=(SELECT count(*) FROM Suministrar WHERE clvPieza = P.clvPieza);
END set_numProvPieza_PR;
/
SHOW ERRORS PROCEDURE set_numProv_Pieza_PR

column OBJECT_NAME       heading "nombre"    format a20;
column DATA_OBJECT_ID    heading "id"        format a2;
column SUBOBJECT_NAME    heading "subobjeto" format a9;

/* mostrar todos los procedures del usuario */
SELECT * FROM user_objects WHERE object_type='PROCEDURE';

/* asignar valor al total de proveedores de cada pieza
*/
EXECUTE set_numProvPieza_PR

/* Crear un disparador para actualizar el nº de Proveedores por pieza
*/
CREATE OR REPLACE TRIGGER updateNumProvPieza_TR
AFTER INSERT OR DELETE OR UPDATE ON Suministrar
FOR EACH ROW
BEGIN
  IF INSERTING THEN
     UPDATE Pieza SET numProv=numProv+1 WHERE clvPieza = :NEW.clvPieza;
   ELSIF DELETING THEN
     UPDATE Pieza SET numProv=numProv-1 WHERE clvPieza = :OLD.clvPieza;
   ELSIF UPDATING THEN
     UPDATE Pieza SET numProv=numProv+1 WHERE clvPieza = :NEW.clvPieza;
     UPDATE Pieza SET numProv=numProv-1 WHERE clvPieza = :OLD.clvPieza;
  END IF;
END updateNumProvPieza_TR;
/
SHOW ERRORS TRIGGER updateNumProvPieza_TR

/* añadir la restricción de que el número de proveedores de una pieza debe ser <= 2
*/
ALTER TABLE Pieza ADD CONSTRAINT chkMaxProv CHECK(numProv<=2);

SELECT * FROM Pieza;
SELECT * FROM Suministrar ORDER BY clvProv, clvPieza;

-- Mostrar todos los disparadores y su estado
SELECT TRIGGER_NAME, STATUS FROM USER_TRIGGERS;

/* añadir un suministro para comprobar que funciona correctamente el disparador
*/
INSERT INTO suministrar VALUES (3, 92);
```

```
/* modificar la definición del disparador para gestionar las excepciones
*/
CREATE OR REPLACE TRIGGER updateNumProvPieza_TR
AFTER INSERT OR DELETE OR UPDATE ON Suministrar
FOR EACH ROW
BEGIN
  IF INSERTING THEN
     UPDATE Pieza SET numProv=numProv+1 WHERE clvPieza = :NEW.clvPieza;
   ELSIF DELETING THEN
     UPDATE Pieza SET numProv=numProv-1 WHERE clvPieza = :OLD.clvPieza;
   ELSIF UPDATING THEN
     UPDATE Pieza SET numProv=numProv+1 WHERE clvPieza = :NEW.clvPieza;
     UPDATE Pieza SET numProv=numProv-1 WHERE clvPieza = :OLD.clvPieza;
  END IF;
EXCEPTION
  WHEN OTHERS THEN
   IF SQLCODE = -2290 THEN DBMS_OUTPUT.PUT_LINE('ERROR: max. 2 Prov/Pieza');
   END IF;
   RAISE; -- propagar la excepción
--   DBMS_OUTPUT.PUT_LINE('ERROR '||SQLCODE||' '||SQLERRM);
END updateNumProvPieza_TR;
/
SHOW ERRORS TRIGGER updateNumProvPieza_TR

-- comprobar el funcionamiento del disparador
INSERT INTO suministrar VALUES (3, 92);

/* probar que funciona el disparador que limita el número de proveedores a 2
   gestionando la excepción generada al sobrepasar el número de proveedores
*/
BEGIN
  INSERT INTO suministrar VALUES (3, 92);
EXCEPTION
  WHEN OTHERS THEN
   IF SQLCODE = -2290 THEN DBMS_OUTPUT.PUT_LINE('operación no permitida');
   ELSE DBMS_OUTPUT.PUT_LINE('ERROR '||SQLCODE||' '||SQLERRM);
   END IF;
   ROLLBACK;
END;
/
/* se pueden definir nuevas excepciones para evitar los códigos numéricos */
DECLARE
  check_violated  EXCEPTION;                      -- declaración de la excepción
  PRAGMA EXCEPTION_INIT (check_violated, -2290);  -- asignar código de error a la excepción
BEGIN
  INSERT INTO suministrar SELECT 3, clvPieza FROM Suministrar WHERE clvProv=2;
EXCEPTION
  WHEN check_violated THEN
    DBMS_OUTPUT.PUT_LINE('operación no permitida');
    ROLLBACK;
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('ERROR '||SQLCODE||' '||SQLERRM);
    ROLLBACK;
END;
/
-- ejemplo de definición de una secuencia y de un procedimiento para añadir piezas
--------------------------------------------------------------------------------
DROP SEQUENCE seq_clvPieza;
CREATE SEQUENCE seq_clvPieza MINVALUE 100 INCREMENT BY 2 START WITH 100;

/* mostrar todas las secuencias definidas por el usuario */
SELECT * FROM USER_SEQUENCES;
CREATE OR REPLACE PROCEDURE ponerPieza (
  elTipo  IN Pieza.nombPieza%TYPE,
  elColor IN Pieza.color%TYPE)
AS
BEGIN
  INSERT INTO Pieza VALUES(seq_clvPieza.NEXTVAL, UPPER(elTipo), UPPER(elColor),0);
END;
/
SHOW ERRORS PROCEDURE ponerPieza
```

```
/* añadir una nueva pieza con el procedimiento creado */
EXECUTE ponerPieza('alicates','rosa')
SELECT * FROM Pieza;

COMMIT;

/* al intentar añadir en una única sentencia más suministros de esta pieza
   que los permitidos se producirá error y no se inserta ningún suministro
*/
DECLARE
  check_violated  EXCEPTION;                       -- declaración de la excepción
  PRAGMA EXCEPTION_INIT (check_violated, -2290);  -- asignar código de error a la excepción
BEGIN
  INSERT INTO suministrar
    SELECT clvProv, seq_clvPieza.CURRVAL
    FROM proveedor;
EXCEPTION
  WHEN check_violated THEN
    DBMS_OUTPUT.PUT_LINE('operación no permitida');
    ROLLBACK;
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('ERROR '||SQLCODE||' '||SQLERRM);
    ROLLBACK;
END;
/
SELECT * FROM Suministrar;

/* si sólo se añaden 2 suministros de esta pieza, ningún problema */
INSERT INTO suministrar
  SELECT clvProv, seq_clvPieza.CURRVAL
  FROM proveedor
  WHERE clvProv < 3;

/* al intentar añadir uno más se vuelve a producir error */
INSERT INTO suministrar VALUES (3, seq_clvPieza.CURRVAL);
SELECT * FROM Suministrar;

CREATE OR REPLACE PROCEDURE ponerSuministro (
  elProv  IN suministrar.clvProv%TYPE,
  laPieza IN suministrar.clvPieza%TYPE)
IS
  check_violated  EXCEPTION;                       -- declaración de la excepción
  PRAGMA EXCEPTION_INIT (check_violated, -2290);  -- asignar código de error a la excepción
BEGIN
  INSERT INTO suministrar (clvProv, clvPieza) VALUES(elProv, laPieza);
EXCEPTION
  WHEN check_violated THEN
    DBMS_OUTPUT.PUT_LINE('operación no permitida');
    ROLLBACK;
  WHEN OTHERS THEN
    DBMS_OUTPUT.PUT_LINE('ERROR '||SQLCODE||' '||SQLERRM);
    ROLLBACK;
END;
/
SHOW ERRORS PROCEDURE ponerSuministro

EXECUTE ponerSuministro(3,92);
SELECT * FROM Suministrar;

SELECT * FROM Pieza P
WHERE numProv <> (SELECT count(*) FROM suministrar WHERE clvPieza = P.clvPieza);
-------------------------------------------------------
-- FINAL DEL TEST.  Eliminar de la BD los elementos añadidos.

-- eliminar los disparadores y procedures creados en este ejemplo
DROP TRIGGER updateNumProvPieza_TR;
DROP PROCEDURE set_numProvPieza_PR;
DROP PROCEDURE ponerPieza;

-- eliminar las tuplas añadidas
DELETE FROM suministrar WHERE clvProv = 3;
DELETE FROM suministrar WHERE clvPieza >= 100;
DELETE FROM pieza WHERE clvPieza >= 100;

-- eliminar el atributo numProv añadido a la tabla Pieza
ALTER TABLE Pieza DROP COLUMN numProv;
```