Entrega práctica 5: PL/SQL

Grupo: DG04

Autor: Manuel Ortega Salvador.
Autor: Francisco Javier Blázquez Martínez.

Apartado 1.- Generación de los ficheros:

Se muestran los ficheros tal y como fueron creados y con el formato exacto que presentan en la ejecución de los siguientes apartados.

“Códigos postales I.txt”

```
08050;Parets;Barcelona;
14200;Peñarroya;Córdoba;
14900;Lucena;Córdoba;
;Arganda;Sevilla;
08050;Zaragoza;Zaragoza;
28040;Arganda;Madrid;
28040;Madrid;Madrid;
```

“Domicilios I.txt”

```
12345678A;Avda. Complutense;28040;
12345678A;Cántaro;28004;
12345678P;Diamante;14200;
12345678P;Carbón;14901;
```

Apartado 2.- Creación de tablas:

Se muestran exactamente las mismas instrucciones que fueron ejecutadas sobre nuestro usuario en Oracle SQL Developer. Creación de tablas sin restricciones de integridad. Destacamos que tanto en la tabla “Domicilios I” como en la tabla “Códigos postales I” el atributo que guarda el C.P se llama exactamente “Código postal”, el hecho de nombrarlo en posteriores apartados de forma distinta (como cambiar la ‘p’ por ‘P’ en ‘postal’) genera errores.

```
CREATE TABLE "Domicilios I"
(
   DNI                CHAR(9),
   Calle              CHAR(50),
   "Código postal" CHAR(5)
);

CREATE TABLE "Códigos postales I"
(
   "Código postal"  CHAR(5),
   Población        CHAR(50),
   Provincia        CHAR(50)
);
```

Respuesta del servidor:

Table “Domicilios I” created.
Table “Códigos postales I” created.

Apartado 3.- Importar los datos del apartado 1 con Oracle Loader:

3.1.- Carga de datos en la tabla “Domicilios I”:
sqlldr userid=DG04@BDd/DG04PWD control='Domicilios I.ctl' log='Domicilios I informe.txt'

“Domicilios I.ctl”
```
LOAD DATA
INFILE 'Domicilios I.txt'
APPEND
INTO TABLE "Domicilios I"
FIELDS TERMINATED BY ';'
 (DNI, Calle, "Código postal")
```

“Domicilios I informe.txt”
```
…
Tabla "Domicilios I":
  4 Filas se ha cargado correctamente.
  0 Filas no cargada debido a errores de datos.
  0 Filas no cargada porque todas las cláusulas WHEN han fallado.
  0 Filas no cargada porque todos los campos eran nulos.
…
```

El fichero “Domicilios I.bad” no fue creado al no haber tuplas rechazadas.

3.2.- Carga de datos en la tabla “Códigos postales I”:
sqlldr userid=DG04@BDd/DG04PWD control='Códigos postales I.ctl' log='Códigos postales I informe.txt'

“Códigos postales I.ctl”
```
LOAD DATA
INFILE 'Códigos postales I.txt'
APPEND
INTO TABLE "Códigos postales I"
FIELDS TERMINATED BY ';'
   ("Código postal", Población, Provincia)
```

“Códigos postales I informe.txt”
```
…
Tabla "Códigos postales I":
  7 Filas se ha cargado correctamente.
  0 Filas no cargada debido a errores de datos.
  0 Filas no cargada porque todas las cláusulas WHEN han fallado.
  0 Filas no cargada porque todos los campos eran nulos.
…
```

El fichero “Códigos postales I.bad” no fue creado al no haber tuplas rechazadas.

Respuesta del servidor:

```
SQL*Loader: Release 11.2.0.1.0 - Production on Jue Nov 22 18:32:22 2018
Copyright (c) 1982, 2009, Oracle and/or its affiliates.  All rights reserved.
Punto de confirmaci¾n alcanzado - recuento de registros l¾gicos 7
```

Apartado 4.- Procedimiento que dectecte valores nulos en "Códigos postales I":

```
CREATE OR REPLACE PROCEDURE comprobar_NULL IS

   /* We define a cursor to point the tuples with some value NULL */
   CURSOR tuplasConNuloEnCP IS

      SELECT *
      FROM "Códigos postales I"
      WHERE "Código postal" is NULL or
                  Población         is NULL or
                  Provincia          is NULL;

   /* Variables needed for doing FETCH */
   postCode "Códigos postales I"."Código postal"%TYPE;
   homeTown "Códigos postales I".Población%TYPE;
   province "Códigos postales I".Provincia%TYPE;

   /* Counter variable (%ROWCOUNT could be used) */
   counter integer := 0;
   tuplesWithNull exception;

BEGIN

   OPEN tuplasConNuloEnCp;
   FETCH tuplasConNuloEnCP INTO  postCode,homeTown,province;

   WHILE tuplasConNuloEnCp%FOUND LOOP
      DBMS_OUTPUT.put_line('Valor nulo en la tupla: ' || postCode || ' ' || homeTown || ' ' ||
province);
      counter := counter+1;

      FETCH tuplasConNuloEnCP INTO  postCode,homeTown,province;
   END LOOP;

   CLOSE tuplasConNuloEnCp;

   IF counter!=0 THEN
      RAISE tuplesWithNull;
   END IF;

EXCEPTION
   WHEN tuplesWithNull THEN
      DBMS_OUTPUT.put_line('Se han encontrado ' || counter || ' tuplas con algún atributo nulos');
   WHEN OTHERS THEN
      DBMS_OUTPUT.put_line('Oh,Oh; this is unexpected!');
END;

Procedure COMPROBAR_NULL compilado
Valor nulo en la tupla:  Arganda                                Sevilla
Se han encontrado 1 tuplas con algún atributo nulos
Procedimiento PL/SQL terminado correctamente.
```

Apartado 5.- Procedimiento que compruebe violaciones de clave primaria en la tabla “Códigos postales I”:

```
CREATE OR REPLACE PROCEDURE comprobar_PK IS

   /* We define a cursor to point the tuples with some value NULL */
   CURSOR postCode_numApariciones IS

      SELECT "Código postal",count(*)
      FROM "Códigos postales I"
      GROUP BY "Código postal";

   /* Variables needed for doing FETCH */
   postCode "Códigos postales I"."Código postal"%TYPE;
   numApariciones integer;

   /* Exception to be thrown */
   tuplesWithNull exception;
   duplicatedPrimKey exception;

BEGIN

   OPEN postCode_numApariciones;

   FETCH postCode_numApariciones INTO  postCode,numApariciones;

   WHILE postCode_numApariciones%FOUND LOOP

      IF postCode IS NULL THEN
         RAISE tuplesWithNull;
      ELSIF numApariciones>1 THEN
         RAISE duplicatedPrimKey;
      END IF;

      FETCH postCode_numApariciones INTO  postCode,numApariciones;
   END LOOP;

   CLOSE postCode_numApariciones;

EXCEPTION

   WHEN tuplesWithNull THEN
      DBMS_OUTPUT.put_line('Se ha encontrado una tupla con clave primaria nula');
   WHEN duplicatedPrimKey THEN
      DBMS_OUTPUT.put_line('Se ha encontrado una clave primaria repetida');
   WHEN OTHERS THEN
      DBMS_OUTPUT.put_line('Oh,Oh; this is unexpected!');
END;

EXECUTE COMPROBAR_PK;
```

--Salida del servidor:

Se ha encontrado una tupla con clave primaria nula

Procedimiento PL/SQL terminado correctamente.

--Actualizamos los códigos postales nulos y comprobamos que está repetido

UPDATE "Códigos postales I" SET "Código postal"=14900 WHERE "Código postal" IS NULL;

1 fila actualizadas.

EXECUTE COMPROBAR_PK();

--Salida del servidor:

Se ha encontrado una clave primaria repetida

Procedimiento PL/SQL terminado correctamente.

Apartado 6.- Comprobar si el campo “Domicilios I”.”Código postal” es foreign key de “Códigos postales I”.”Código postal”:

```
CREATE OR REPLACE PROCEDURE comprobar_FK(numErrorsLimit IN integer) IS

   /* We define a cursor to point the tuples with some value NULL */
   CURSOR  foreignKeyViolations IS

      SELECT "Domicilios I"."Código postal","Domicilios I".DNI
      FROM   "Domicilios I"
      WHERE  "Domicilios I"."Código postal"
      NOT IN (select "Código postal" from   "Códigos postales I");

   /* Variables needed for doing FETCH */
   postCode "Domicilios I"."Código postal"%TYPE;
   empDni   "Domicilios I".DNI%TYPE;

   /* Counter variable */
   counter integer := 0;

   /* Exception to be thrown */
   FK_VIOLATION Exception;

BEGIN

   OPEN  foreignKeyViolations;

   FETCH foreignKeyViolations INTO  postCode,empDni;

   WHILE foreignKeyViolations%FOUND and counter<numErrorsLimit LOOP
      DBMS_OUTPUT.put_line('El código postal ' || postCode || ' (empleado con DNI ' || empDni ||
```

```
') es una FK_VIOLATION.');
        counter := counter+1;

        FETCH foreignKeyViolations INTO  postCode,empDni;
    END LOOP;

    CLOSE foreignKeyViolations;

    IF counter!=0 THEN
        RAISE FK_VIOLATION;
    END IF;

EXCEPTION

    WHEN FK_VIOLATION THEN
        DBMS_OUTPUT.put_line('Se han encontrado ' || counter || ' violaciones de clave foránea.');
    WHEN OTHERS THEN
        DBMS_OUTPUT.put_line('Oh,Oh, this is unexpected!');
END;

-- Salida del servidor:
Procedure COMPROBAR_FK compilado;

-- Ejemplos de ejecución:
EXECUTE COMPROBAR_FK(5);

El código postal 14901 (empleado con DNI 12345678P) es una FK_VIOLATION.
Se han encontrado 1 violaciones de clave foránea.

Procedimiento PL/SQL terminado correctamente.

-- Introducimos dos tuplas más erróneas y un límite de erroes menor que tres:

INSERT INTO "Domicilios I" VALUES
('47399024A', 'Petrenko', '33333');

INSERT INTO "Domicilios I" VALUES
('47399024B', 'PetrenkoB', '33334');

EXECUTE COMPROBAR_FK(2);

El código postal 14901 (empleado con DNI 12345678P) es una FK_VIOLATION.
El código postal 33334 (empleado con DNI 47399024B) es una FK_VIOLATION.
Se han encontrado 2 violaciones de clave foránea.

Procedimiento PL/SQL terminado correctamente.

-- En efecto vemos que la limitación de errores funciona correctamente.
```