Examen de Bases de datos
1 de febrero de 2021

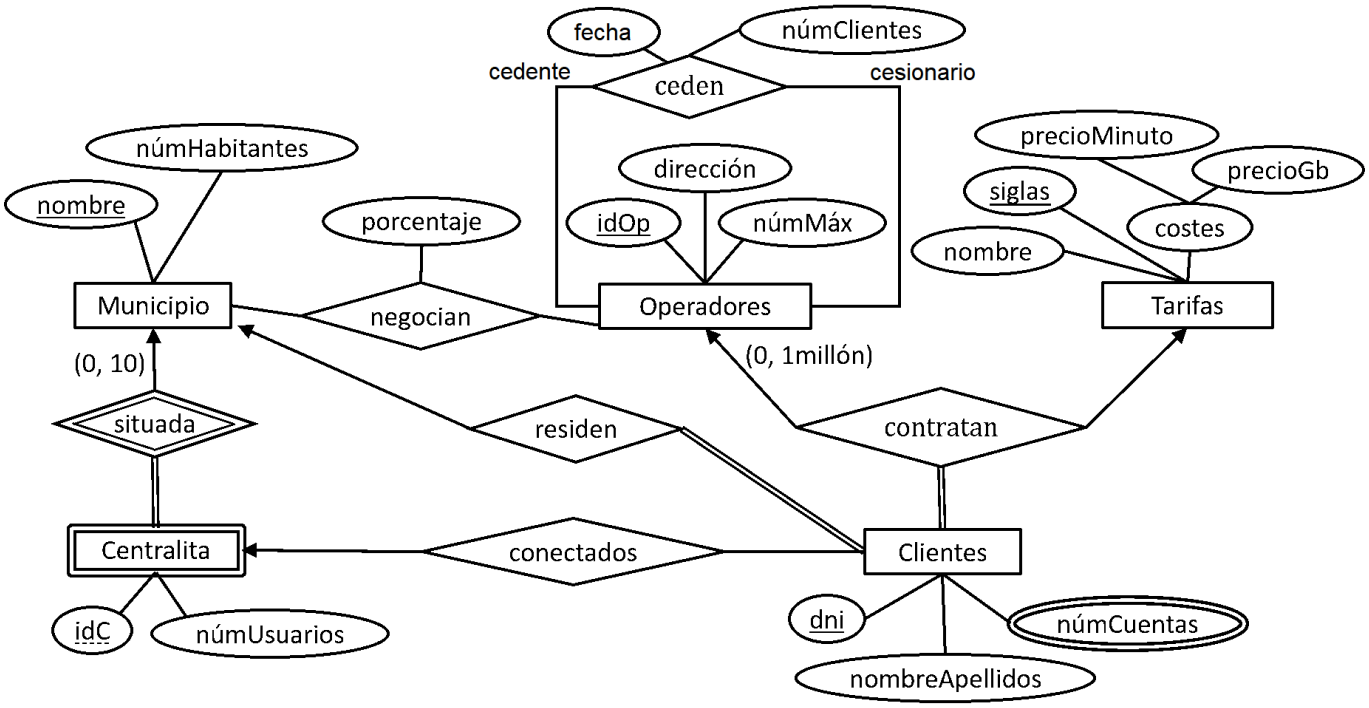
Ejercicio 1. (2 puntos)
Queremos desarrollar una base de datos para la gestión de las Telecomunicaciones de un país que va a liberalizar el sector pasando de un único proveedor a varios. Partimos del siguiente texto que nos describe los datos que queremos almacenar en la base de datos:

- En esta primera etapa de liberación del sector se ha decidido publicar una lista de tarifas común a todo el sector de telecomunicaciones. De las tarifas conoceremos su nombre que es único, así como sus siglas que también son únicas y los costes: el precio por minuto de conversación y el precio por GB consumido.
- Los operadores tienen un código de identificación único, una dirección, así como la cantidad máxima de clientes con los que pueden realizar los contratos. Para que no se monopolice el mercado cada operador a lo sumo tendrá 1 millón de clientes.
- De los clientes conocemos su identificador único, su nombre completo, sus números de cuentas y el municipio en el que viven. Los clientes podrán contratar una tarifa de cualquiera de las tarifas ofrecidas, con cualquier operador. En esta base de datos no almacenaremos clientes que no tengan ningún contrato asociado. Al principio puede haber tarifas y operadores sin clientes que las hayan contratado.
- Debemos almacenar también las centralitas que posee el país. Cada centralita está situada en un único municipio, pudiendo albergar un municipio hasta 10 centralitas. No todos los municipios tienen centralitas. De cada centralita tenemos un identificador que la identifica dentro del municipio, pero que puede haber sido utilizado en otro municipio. Además, también sabemos el número de usuarios a los que puede dar servicio.
- Los municipios tienen un nombre propio que los diferencia entre ellos. También conocemos el número de habitantes que residen en dicho municipio.
- Los clientes estarán conectados a una única centralita, pudiendo en una etapa inicial de contratación no tener asignada la centralita a la que estarán conectados. Al comienzo puede haber centralitas sin clientes conectados.
- Por otro lado, y también con el fin de evitar el monopolio, se ha establecido la siguiente restricción: los operadores negocian con los municipios una cantidad máxima de clientes que nunca superará el 50% de los que pueden albergar todas las instalaciones del municipio. Esta negociación será libre y abierta de tal forma que cada operador conseguirá un porcentaje diferente para el mismo municipio. Ni todos los municipios ni todos los operadores negocian.
- Con el fin de permitir algo más de flexibilidad se permitirá llegar a acuerdos entre operadores. De esta manera un operador podrá ceder una cantidad de clientes a otro. Se debe almacenar el número de clientes que los operadores se ceden entre sí en una fecha determinada.

Diseña un diagrama entidad-relación que incluya restricciones de cardinalidad y participación, indicando las restricciones que no se pueden representar en el modelo entidad-relación.

Solución:

Diagrama E/R



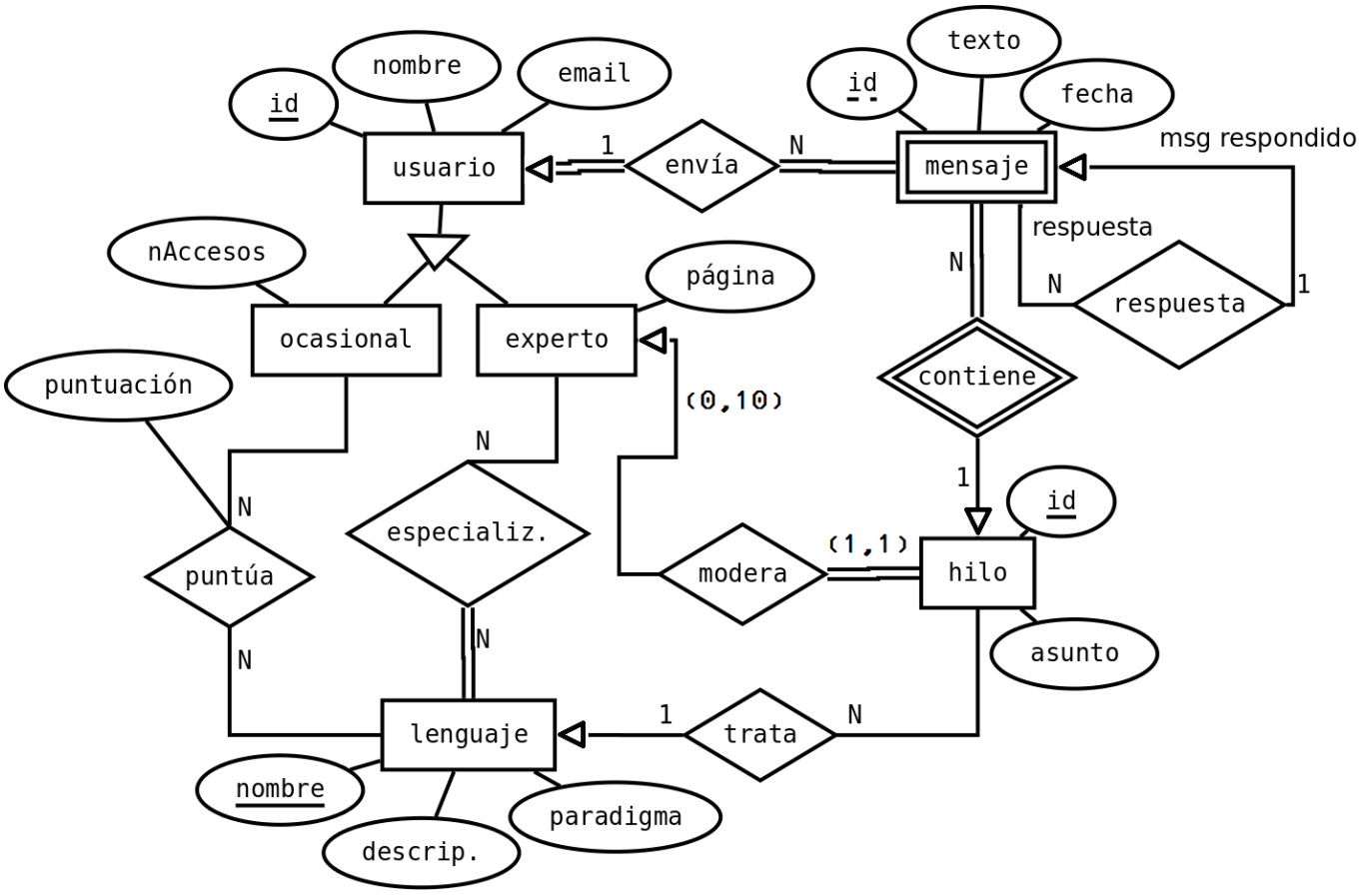
Como alternativa a este diagrama se puede añadir una relación "ofertan" entre "Operadores" y "Tarifas", agregando estos tres elementos y conectándolos con "Clientes" con una relación binaria "contratan". Como ventaja, se pueden conocer las tarifas que oferta cada operador antes de que se haga ningún contrato.

Restricciones no representables en el diagrama

- Dominio del atributo "porcentaje" en "negocian": (0,50].
- La suma de "porcentaje" en "negocian" por cada municipio debe ser menor o igual que 100.
- La cardinalidad del conjunto "conectados" no puede ser mayor que "númUsuarios".
- La cardinalidad del conjunto "residen" no puede ser mayor que "númHabitantes".
- La cardinalidad del conjunto "contratan" no puede ser mayor que "númMáx".
- El atributo "nombre" de "Tarifas" es clave candidata.

Ejercicio 2 (1,5 puntos) Se dispone del siguiente diagrama Entidad-Relación sobre un foro de consultas y respuestas sobre lenguajes de programación con las siguientes especificaciones:

- Para poder participar, los usuarios se registran en el sistema con un identificador único, su nombre y correo electrónico.
- Los mensajes se agrupan en "hilos" (las conocidas conversaciones). Un hilo tiene un identificador y se debe almacenar el asunto del hilo.
- Los usuarios son de dos tipos: expertos y ocasionales. De los primeros se debe saber su página personal, mientras que es necesario almacenar el número de accesos de los usuarios ocasionales.
- Los usuarios envían mensajes al sistema para plantear preguntas o contestar a otros mensajes. Cada mensaje tiene un identificador que es único dentro de cada hilo. Además, si un mensaje es respuesta de otro, debe almacenarse cuál es el mensaje que se está respondiendo.
- Cuando se crea un hilo se debe asignar un moderador, que es un usuario experto seleccionado por sus conocimientos sobre el hilo. Un experto puede moderar hasta 10 hilos.
- Cada hilo puede tratar sobre lenguajes de programación, pero hay muchos hilos que no tratan de ningún lenguaje en particular.
- Los usuarios expertos están especializados en algunos de los lenguajes tratados en el foro. Todo lenguaje tiene al menos un usuario experto especializado.
- Por último, los usuarios ocasionales pueden puntuar la utilidad de los lenguajes tratados en el foro. Se debe registrar la puntuación que cada usuario ocasional da a cada lenguaje.



Traduce el diagrama Entidad-Relación a un esquema relacional, indicando las restricciones de integridad referencial resultantes y las restricciones de integridad que se han perdido en la traducción.

Solución:

```
usuario (id, nombre, email)

experto (id, página)
  experto.id → usuario.id

ocasional (id,  naccesos)
  ocasional.id → usuario.id

mensaje (idmensaje,idhilo, texto, fecha, msjresp*, hiloresp*, idusuario)
  mensaje.idusuario → usuario.id
  mensaje.{msjresp, hiloresp} → mensaje.{idmensaje, idhilo}
  mensaje.idhilo → hilo.id

hilo (id, asunto, idusuario)
  hilo.idmoderador → experto.id

lenguaje (nombre, paradigma, descrip)

trata (idhilo, lenguaje)
  trata.lenguaje → lenguaje.nombre
  trata.idhilo → hilo.id
```

(`trata` utiliza un esquema separado para evitar valores nulos. Otra alternativa válida sería simplificarla en `lenguaje`: `lenguaje (nombre, paradigma, descrip, lenguaje*)`)

```
especializado (lenguaje, idexperto)
  especializado.lenguaje → lenguaje.nombre
  especializado.idexperto → experto.id

puntua (lenguaje, idusuario, puntuacion)
  puntua.lenguaje → lenguaje.nombre
  puntua.idusuario → ocasional.id
```

Restricciones de integridad que se han perdido en la transformación:

- Participación total de `usuario` en `envía`
- Participación total de `lenguaje` en `especializado`
- Participación mínima y máxima de `hilo` en `modera`
- Todo usuario debe ser ocasional o experto (no debe haber más entradas en `usuario` que la unión -sin repetidos- de `experto` y `ocasional`)

Ejercicio 3 (6 puntos) El siguiente modelo relacional de una base de datos almacena la información de gestión de telecomunicaciones. El esquema de las tablas (no necesariamente resultado del ejercicio anterior de diseño) es:

```
Clientes(DNI, nombre, apellido, municipio)
Operadores(código, nombreOperador)
Contratan(operador, DNICliente, tarifaMes, GBMes)
  Contratan.operador -> Operadores.código
  Contratan.DNICliente -> Clientes.DNI
```

Si no se especifica, no es necesario ordenar. Se pide:

a. (0,5 puntos) Una consulta SQL que muestre los operadores con más de 1000 Clientes. Esquema: (nombreOperador).

Solución:

Las siguientes consultas DDL y DML son simplemente para comprobar las consultas DQL a continuación (no se pide en el enunciado):

```
DROP TABLE Contratan;
DROP TABLE Clientes;
DROP TABLE Operadores;
DROP TABLE ResumenOperadores;

CREATE TABLE Operadores(
  código         VARCHAR2(9) PRIMARY KEY,
  nombreOperador VARCHAR2(100));

INSERT INTO Operadores VALUES
  ('AI', 'Airtalk');
INSERT INTO Operadores VALUES
  ('TM', 'T-Movil');
INSERT INTO Operadores VALUES
  ('VZ', 'Verizona');

CREATE TABLE Clientes(
  DNI       VARCHAR2(9) PRIMARY KEY,
  nombre    VARCHAR2(100),
  apellido  VARCHAR2(100),
  municipio VARCHAR2(200));

INSERT INTO Clientes VALUES
  ('0000','Alicia','Guarda','Castro-Urdiales');
INSERT INTO Clientes VALUES
  ('0001','Norberto','Valera','Valladolid');
INSERT INTO Clientes VALUES
  ('0002','Jaime','Armenteros','Castro-Urdiales');
INSERT INTO Clientes VALUES
  ('0003','Nicanor','Cienfuegos','Almendralejo');
INSERT INTO Clientes VALUES
  ('0004','Alejo','Cantarranas','Utrera');

CREATE TABLE Contratan(
  operador VARCHAR2(20) REFERENCES Operadores(código),
  DNICliente VARCHAR2(9) REFERENCES Clientes(DNI),
```

```
  tarifaMes NUMBER(5,2),
  GBMes NUMBER(5,2),
  PRIMARY KEY (operador, DNICliente));

INSERT INTO Contratan VALUES
  ('AI','0000',10,10);
INSERT INTO Contratan VALUES
  ('TM','0000',5,5);
INSERT INTO Contratan VALUES
  ('AI','0001',15,15);
INSERT INTO Contratan VALUES
  ('VZ','0001',20,20);
INSERT INTO Contratan VALUES
  ('TM','0002',20,20);
INSERT INTO Contratan VALUES
  ('AI','0003',30,30);

SELECT op.nombreOperador
FROM Contratan co JOIN Operadores op ON co.operador = op.código
GROUP BY op.nombreOperador
HAVING COUNT(*) > 1000;
```

b. (0,5 puntos) Una consulta SQL que muestre DNI, nombre y apellido de los clientes, y cuánto gastan al mes. Deben aparecer primero quienes más gastan. Esquema: (DNI, nombre, apellido, gastoMensual).

Solución:

```
SELECT DNI, nombre, apellido, SUM(tarifaMes) AS gastoMensual
FROM Clientes JOIN Contratan ON DNI = DNICliente
GROUP BY DNI, nombre, apellido
ORDER BY gastoMensual DESC;
```

c. (0,5 puntos) Una consulta SQL que muestre DNI, nombre y apellido de los clientes que tienen contrato con más de un operador. Esquema: (DNI, nombre, apellido).

Solución:

```
SELECT DNI, nombre, apellido
FROM Clientes
WHERE DNI IN (SELECT DNICliente
            FROM Contratan
            GROUP BY DNICliente
            HAVING COUNT(*) > 1);

SELECT Cl.DNI, Cl.nombre, Cl.apellido
FROM Clientes Cl JOIN Contratan Co ON Cl.DNI = Co.DNICliente
GROUP BY Cl.DNI, Cl.nombre, Cl.apellido
HAVING COUNT(*) > 1;
```

d. (0,75 puntos) Una consulta SQL que muestre DNI, nombre y apellido de todos los clientes junto con el número de operadores contratados (que podría ser cero). Deben aparecer primero los clientes con más contratos. Esquema: (DNI, nombre, apellido, operadoresContratados).

Solución:

```
SELECT * FROM
```

```
(SELECT DNI, nombre, apellido, COUNT(*) operadoresContratados
FROM Clientes JOIN Contratan ON DNI = DNICliente
GROUP BY DNI, nombre, apellido
UNION ALL
SELECT DNI, nombre, apellido, 0 operadoresContratados
FROM Clientes LEFT OUTER JOIN Contratan ON DNI = DNICliente
WHERE operador IS NULL
)
ORDER BY 4 DESC;

SELECT DNI, nombre, apellido, COUNT(operador) operadoresContratados
FROM Clientes LEFT OUTER JOIN  Contratan ON DNI = DNICliente
GROUP BY DNI, nombre, apellido
ORDER BY operadoresContratados DESC;
```

e. (0,75 puntos) Una consulta SQL que muestre el nombre y apellido del cliente (o clientes) con más GB contratados en total. Esquema: (nombre, apellido).

Solución:

```
SELECT nombre, apellido
FROM Clientes JOIN Contratan ON DNI = DNICliente
GROUP BY DNI, nombre, apellido
HAVING SUM(GBMes) >= ALL (SELECT SUM(GBMes)
                          FROM Contratan
                          GROUP BY DNICliente);
```

f. (2 puntos) Un procedimiento que reciba un municipio y produzca un listado que muestre los operadores y los clientes de ese municipio (deben aparecer solo los operadores que tengan clientes en ese municipio, ordenados por nombre). Por cada cliente, se mostrará el DNI, nombre, apellido y los datos de su contrato con el operador (deben aparecer ordenados por DNI). Si no hay ningún cliente en el municipio, debe emitirse una excepción definida por el usuario que escriba un mensaje indicándolo. Un ejemplo de ejecución del procedimiento es:

```
OPERADORES Y CLIENTES DEL MUNICIPIO: Castro-Urdiales
------------------------------------------------------------
Operador: Airtalk
    0000: Alicia Guarda: 10 Eur, 10GB
Operador: T-Movil
    0000: Alicia Guarda: 5 Eur, 5GB
    0002: Jaime Armenteros: 20 Eur, 20GB
```

Solución:

```
CREATE OR REPLACE PROCEDURE pr_listado(p_muni IN Clientes.municipio%TYPE) AS

  CURSOR c_ope IS
    SELECT DISTINCT op.código, op.nombreOperador
    FROM Operadores op JOIN Contratan co ON op.código = co.operador
    JOIN Clientes cl ON co.DNICliente = cl.DNI
    WHERE cl.municipio = p_muni
    ORDER BY op.nombreOperador;

  CURSOR c_clientes(p_operador Operadores.código%TYPE, p_muni
Clientes.municipio%TYPE) IS
    SELECT DNI, nombre, apellido, tarifaMes, GBMes
```

```
    FROM Clientes JOIN Contratan ON DNI=DNICliente
    WHERE operador = p_operador AND municipio = p_muni
    ORDER BY DNI;
  v_clientes_muni INTEGER;
  e_no_existe EXCEPTION;

BEGIN
  SELECT COUNT(*)
  INTO v_clientes_muni
  FROM Contratan co JOIN Clientes cl ON co.DNICliente = cl.DNI
  WHERE municipio = p_muni;

  IF (v_clientes_muni = 0) THEN
    RAISE e_no_existe;
  END IF;

  DBMS_OUTPUT.PUT_LINE('OPERADORES Y CLIENTES DEL MUNICIPIO: ' || p_muni);
  FOR r_ope IN c_ope LOOP
    DBMS_OUTPUT.PUT_LINE('Operador: ' || r_ope.nombreOperador);
    FOR r_cli IN c_clientes(r_ope.código, p_muni) LOOP
      DBMS_OUTPUT.PUT_LINE('    ' || r_cli.DNI || ': ' || r_cli.nombre || ' '
        || r_cli.apellido || ': ' || r_cli.tarifaMes
        || ' Eur, ' || r_cli.GBMes || 'GB');
    END LOOP;
  END LOOP;
EXCEPTION
  WHEN e_no_existe THEN
    DBMS_OUTPUT.PUT_LINE('Municipio sin clientes: ' || p_muni);
END;
/
set serveroutput on;
exec pr_listado('Castro-Urdiales');
```

g. (1 punto) Un disparador tal que si se inserta, elimina o modifica un contrato, mantenga actualizada la tabla ResumenOperadores(operador, numContratos, totalTarifasMes, totalGBMes). Esta tabla contiene por cada operador el número de contratos, el importe total de sus tarifas mensuales (en definitiva, el dinero que recibe el operador por todos sus contratos) y la cantidad total de GB contratados al mes en dicho operador. El disparador debe programarse teniendo en cuenta que esta tabla está inicialmente vacía.

Solución:

```
CREATE TABLE ResumenOperadores(
  operador VARCHAR2(20) PRIMARY KEY,
  numContratos NUMBER(5,0),
  totalTarifasMes NUMBER(10,2),
  totalGBMes NUMBER(10,2));

CREATE OR REPLACE TRIGGER tr_actualiza
    AFTER INSERT OR DELETE OR UPDATE ON Contratan
    FOR EACH ROW
DECLARE
    v_existe INT;
BEGIN
    IF INSERTING OR UPDATING THEN
        SELECT COUNT(*)
        INTO v_existe
```

```
        FROM ResumenOperadores
        WHERE operador = :NEW.operador;
        IF v_existe = 0 THEN
          INSERT INTO ResumenOperadores VALUES
          (:NEW.operador, 0, 0, 0);
        END IF;
    END IF;

  IF INSERTING THEN
    UPDATE ResumenOperadores SET
    numContratos = numContratos + 1,
    totalTarifasMes = totalTarifasMes + :NEW.tarifaMes,
    totalGBMes = totalGBMes + :NEW.GBMes
    WHERE operador = :NEW.operador;

  ELSIF DELETING THEN
    UPDATE ResumenOperadores SET
    numContratos = numContratos - 1,
    totalTarifasMes = totalTarifasMes - :OLD.tarifaMes,
    totalGBMes = totalGBMes - :OLD.GBMes
    WHERE operador = :OLD.operador;

  ELSIF UPDATING THEN
      UPDATE ResumenOperadores SET
      numContratos = numContratos - 1,
      totalTarifasMes = totalTarifasMes - :OLD.tarifaMes,
      totalGBMes = totalGBMes - :OLD.GBMes
      WHERE operador = :OLD.operador;

      UPDATE ResumenOperadores SET
      numContratos = numContratos + 1,
      totalTarifasMes = totalTarifasMes + :NEW.tarifaMes,
      totalGBMes = totalGBMes + :NEW.GBMes
      WHERE operador = :NEW.operador;
  END IF;

END;
```

Pruebas:
Los INSERT del apartado anterior y:

```
DELETE FROM Contratan WHERE
  operador='AIRTEL' AND DNICliente='0001' and fecha='1-1-2019';

UPDATE Contratan SET operador='PEPEPHONE' WHERE operador='AIRTEL' AND
DNICliente='0001' and fecha='1-1-2019';
```

Ejercicio 4. (0,5 puntos) Examina el siguiente script SQL de Oracle, que se ejecuta nada más iniciar sesión (y autocompromiso desactivado por defecto).

```
 1: CREATE TABLE Dulces(nombre VARCHAR2(100) PRIMARY KEY,  dulzor NUMBER);
 2: SAVEPOINT paso_uno;
 3: INSERT INTO Dulces VALUES ('brownie', 100);
 4: COMMIT;
 5: UPDATE Dulces SET dulzor = dulzor - 1;
 6: ROLLBACK TO SAVEPOINT paso_uno;
 7: SAVEPOINT paso_dos;
 8: INSERT INTO Dulces VALUES ('brownie', 10);
 9: UPDATE Dulces SET dulzor = dulzor - 1;
10: ROLLBACK TO SAVEPOINT paso_dos;
11: INSERT INTO Dulces VALUES ('cookie', 6);
12: CREATE TABLE Goloso(DNI CHAR(9));
13: INSERT INTO Goloso VALUES ('11111111X');
14: ROLLBACK;
```

¿Cuántas transacciones se han ejecutado? Indica el inicio y el fin de cada una de ellas usando los números de línea.
¿Qué datos y que tablas quedan al final de la ejecución del script?

Solución:

T1: 1-1
T2: 2-4
T3: 5-11
T4: 12-12
T5: 13-14

La tabla Dulces queda con las filas ('brownie', 99) y ('cookie', 6). La tabla Goloso queda vacía.