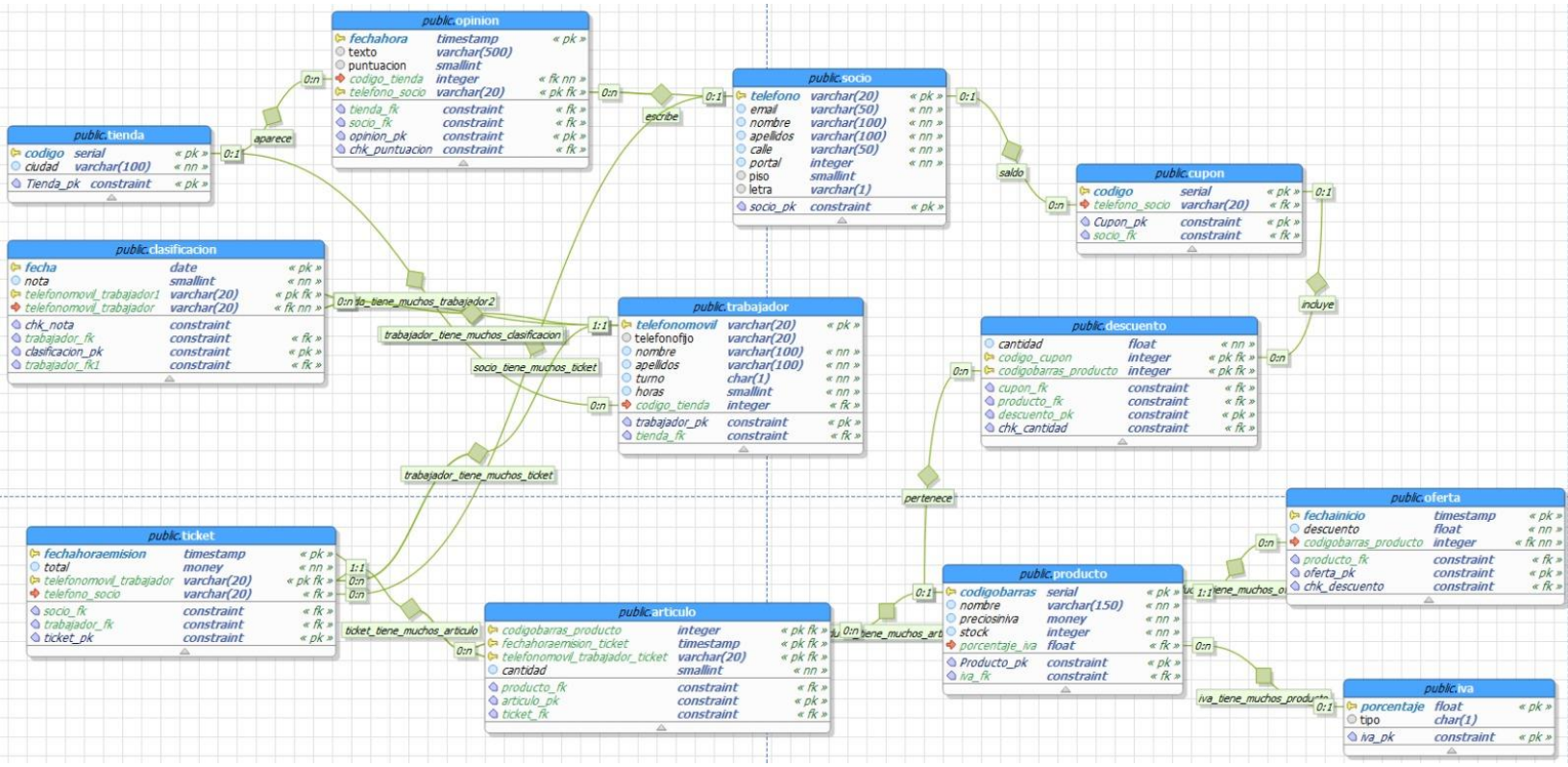
Bases de Datos
Grado II

PECL3 2019-20

Grupo: Ñ
Componentes del grupo:
Marcel Andrei Voicu
Carlos Javier Hellín Asensio


public.opinion
fechahora timestamp « pk »
texto varchar(500)
puntuacion smallint
codigo_tienda integer « fk nn »
telefono_socio varchar(20) « pk fk »
tienda_fk constraint « fk »
socio_fk constraint « fk »
opinion_pk constraint « pk »
chk_puntuacion constraint « fk »
public.tienda
codigo serial « pk »
ciudad varchar(100) « nn »
Tienda_pk constraint « pk »
aparece
escribe
public.socio
telefono varchar(20) « pk »
email varchar(50) « nn »
nombre varchar(100) « nn »
apellidos varchar(100) « nn »
calle varchar(50) « nn »
portal integer « nn »
piso smallint
letra varchar(1)
socio_pk constraint « pk »
saldo
public.cupon
codigo serial « pk »
telefono_socio varchar(20) « fk »
Cupon_pk constraint « pk »
socio_fk constraint « fk »
incluye
public.clasificacion
fecha date « pk »
nota smallint « nn »
telefonomovil_trabajador1 varchar(20) « pk fk »
telefonomovil_trabajador varchar(20) « fk nn »
chk_nota constraint
trabajador_fk constraint « fk »
clasificacion_pk constraint « pk »
trabajador_fk1 constraint « fk »
la_tiene_muchos_trabajador2
trabajador_tiene_muchos_clasificacion
socio_tiene_muchos_ticket
public.trabajador
telefonomovil varchar(20) « pk »
telefonofijo varchar(20)
nombre varchar(100) « nn »
apellidos varchar(100) « nn »
turno char(1) « nn »
horas smallint « nn »
codigo_tienda integer « fk »
trabajador_pk constraint « pk »
tienda_fk constraint « fk »
public.descuento
cantidad float « nn »
codigo_cupon integer « pk fk »
codigobarras_producto integer « pk fk »
cupon_fk constraint « fk »
producto_fk constraint « fk »
descuento_pk constraint « pk »
chk_cantidad constraint « fk »
trabajador_tiene_muchos_ticket
pertenece
public.oferta
fechainicio timestamp « pk »
descuento float « nn »
codigobarras_producto integer « fk nn »
producto_fk constraint « fk »
oferta_pk constraint « pk »
chk_descuento constraint « fk »
public.ticket
fechahoraemision timestamp « pk »
total money « nn »
telefonomovil_trabajador varchar(20) « pk fk »
telefono_socio varchar(20) « fk »
socio_fk constraint « fk »
trabajador_fk constraint « fk »
ticket_pk constraint « pk »
ticket_tiene_muchos_articulo
public.articulo
codigobarras_producto integer « pk fk »
fechahoraemision_ticket timestamp « pk fk »
telefonomovil_trabajador_ticket varchar(20) « pk fk »
cantidad smallint « nn »
producto_fk constraint « fk »
articulo_pk constraint « pk »
ticket_fk constraint « fk »
public.producto
codigobarras serial « pk »
nombre varchar(150) « nn »
preciosiniva money « nn »
stock integer « nn »
porcentaje_iva float « fk »
Producto_pk constraint « pk »
iva_fk constraint « fk »
iva_tiene_muchos_producto
public.iva
porcentaje float « pk »
tipo char(1)
iva_pk constraint « pk »

1.

Normalización de tablas

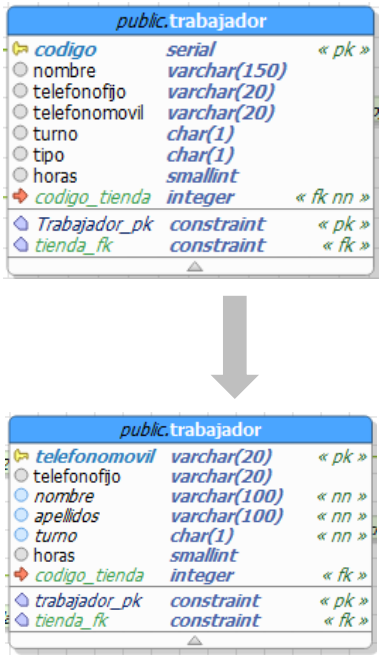
Las tablas de la base de datos deben cumplir las tres formas normales:

- **1FN**: Los atributos son atómicos y la tabla contiene PK
- **2FN**: Los atributos dependen totalmente de la PK
- **3FN**: Ningún atributo depende transitivamente de la PK

En las tablas normalizadas que explicaremos a continuación, si una forma normal se cumple, lo indicaremos con la palabra “Cumple”.

Tabla Trabajador

código → todos los atributos
telefonomovil → todos los atributos
horas → tipo



Consideramos que telefonofijo no determina nada porque puede haber dos trabajadores con mismo teléfono fijo.

- **1FN**: *nombre* es un atributo compuesto. Debemos transformarlo en dos atributos *nombre* y *apellidos*.
- **2FN**: Cumple
- **3FN**: *código* determina a todos. El *telefonomovil*, a su vez, determina al resto de atributos. Además, *horas* determina a *tipo*. Por lo tanto, podemos eliminar a *código* de la tabla, ya que con *telefonomovil* podemos identificar a cualquier trabajador. Después, tenemos que eliminar la dependencia transitiva que hay entre *telefonomovil* y *tipo*, ya que *telefonomovil* → *horas* → *tipo*, y lo hacemos quitando a *tipo* de la tupla, creando otra con *horas* y *tipo*.

El resultado sería:
telefonomovil → nombre, apellidos, telefonofijo, turno, horas, código_tienda
horas → tipo

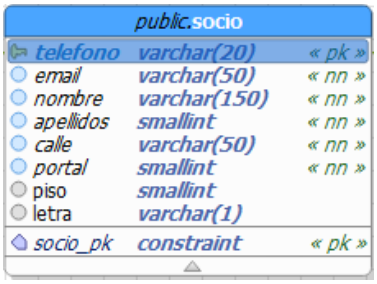
Pero consideramos ineficiente crear una tabla con atributos (*horas, tipo*) ya que tendría que existir una entrada (8, ‘R’) y otras siete para cada hora menor que 8, que correspondan al tipo “Cajero”. Por lo que, optamos por eliminar el atributo *tipo* y volverlo derivado. Mediante consultas SQL sabremos que un trabajador es un reponedor, si sus horas de trabajo son ocho, o cajero, si son menores.

Resultado final: PK(telefonomovil)
telefonomovil → telefonofijo, nombre, apellidos, turno, horas

Tabla Socio

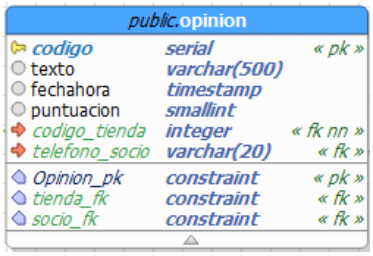
numero → todos los atributos
teléfono → todos los atributos
email → todos los atributos

- **1FN**: *nombres* y *dirección* son compuestos. Transformamos en *nombre, apellidos, calle, portal, piso* y *letra*.
- **2FN**: Cumple
- **3FN**: Cumple



La PK puede ser tanto *numero*, como *teléfono*, como *email*. Eliminamos *numero* ya que no nos aporta ningún dato informativo sobre el socio. Escogemos como PK el *teléfono*.

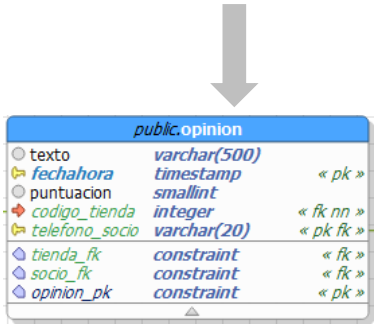
Tabla Opinión



código → todos los atributos
fechahora, teléfono_socio → todos los atributos

Consideramos que con *fechahora* y *teléfono_socio* determinamos el *texto,* la *puntuación* y el *código_tienda* suponiendo que es imposible que un socio haga dos comentarios distintos en la misma fecha y hora.

- **1FN**: Cumple
- **2FN**: Cumple
- **3FN**: Si la PK es (*fechahora*, *teléfono_socio*), encontramos una dependencia transitiva en PK → *código* → *texto, puntuación, código_tienda*. En circunstancias normales, eliminaríamos la dependencia PK →*texto, puntuación, código_tienda* y crearíamos otra tabla con esos tres atributos y PK(*código)* pero, puesto que *código* no es un atributo que ofrezca información valiosa, terminamos eliminando a *código* de la tabla.



Resultado: PK(fechahora, teléfono_socio)
fechahora, teléfono_socio → todos los atributos

Resto de tablas

En el resto de tablas realizamos cambios menores o, simplemente, ninguno. En las tablas *IVA*, *Oferta*, *Descuento* y *Ticket* el único cambio necesario ha sido el de eliminar sus respectivos

atributos *código* que actuaban como PK de cada tabla ya que, con otros atributos como la *fechahoraemision* y el *telefonomovil_trabajador* en el caso de la tabla *Ticket*, podíamos determinar al resto de atributos, sin necesidad de uno extra que, además, no aporta ningún dato real a la tabla. En la tabla *Oferta*, además, hemos eliminado *fechafin* ya que se puede hallar a partir de la *fechainicio* sumándole siete días.
Las demás tablas ya estaban en tercera forma normal. Solo se actualizaron para que las claves foráneas fueran las nuevas claves primarias de las tablas cambiadas anteriormente.

2.

```
-- /// TRIGGERS \\\

-- calculaCoste

create or replace function calculaCoste() returns Trigger

as

$$

declare

        desccupon float8 := (select descuento.cantidad

                                        from articulo join ticket on ticket.fechahoraemision =
new.fechahoraemision_ticket and ticket.telefonomovil_trabajador =
new.telefonomovil_trabajador_ticket

                                                join cupon on
ticket.telefono_socio = cupon.telefono_socio

                                                join descuento on cupon.codigo =
descuento.codigo_cupon and articulo.codigobarras_producto =
descuento.codigobarras_producto);

        descoferta float8 := (select descuento

                                        from articulo join ticket on
ticket.fechahoraemision = new.fechahoraemision_ticket and ticket.telefonomovil_trabajador =
new.telefonomovil_trabajador_ticket

                                                join producto on
new.codigobarras_producto = producto.codigobarras

                                                join oferta on
producto.codigobarras = oferta.codigobarras_producto and fechahoraemision between
fechainicio and fechainicio + interval '7 days');
```

```
precioconiva float8 := (select (preciosiniva::numeric::float8 * (porcentaje_iva + 1) * 0.01)

                                                    from articulo join producto on
new.codigobarras_producto = producto.codigobarras);

begin


update ticket

set total = (new.cantidad * (precioconiva - coalesce(descoferta, 0) - coalesce(desccupon,
0)))::numeric::money

from ticket AS ti

where ti.fechahoraemision = new.fechahoraemision_ticket and ti.telefonomovil_trabajador =
new.telefonomovil_trabajador_ticket;



return new;

End

$$

Language plpgsql;



create trigger Ticket_Update after insert on articulo

for each row

execute procedure calculaCoste();

-- drop trigger Ticket_Update on articulo



-- calculaStock

create or replace function calculaStock() returns Trigger

as

$$
```

```
begin


update producto

set stock = pr.stock - new.cantidad

from producto AS pr

where pr.codigobarras = new.codigobarras_producto;


return new;

end

$$

Language plpgsql;


create trigger Stock_update after insert on articulo

for each row

execute procedure calculaStock();

-- drop trigger Stock_Update on articulo
```

3.

Usando el JDBC Driver en un proyecto de Netbeans se realiza la conexión con la base de datos usando el método getConnection de la clase DriverManager y se ejecuta las consultas primero creando una clase Statement con el método createStatement() y luego con el método executeQuery de la clase ResultSet. Todo ello se encuentra en la clase PECL2 con sus métodos Conectar, Consultar y Mostrar, además de los distintos métodos para cada ejercicio de las consultas solicitadas en PECL2.

```
package pecl3;


import java.sql.*;


public class PECL2 {
```

```
private Statement s;
private Connection c;

public void Conectar() {
    try {
        Class.forName("org.postgresql.Driver");
    } catch(ClassNotFoundException cnfe) {
        cnfe.printStackTrace();
        System.exit(1);
    }

    try {
        c = DriverManager.getConnection("jdbc:postgresql://127.0.0.1:5432/supermercado2",
                "postgres", "postgres");
    } catch(SQLException se) {
        se.printStackTrace();
        System.exit(1);
    }

    try {
        s = c.createStatement();
    } catch(SQLException se) {
        se.printStackTrace();
        System.exit(1);
    }
}

public ResultSet Consultar(String sql) {
    ResultSet rs = null;
    try {
```

```
        rs = s.executeQuery(sql);
    } catch(SQLException se) {
        se.printStackTrace();
        System.exit(1);
    }

    return rs;
}

public void Mostrar(ResultSet rs) {
    int index = 0;
    try {
        while (rs.next()) {
            ResultSetMetaData rsmd = rs.getMetaData();
            int totalColumnas = rsmd.getColumnCount();
            for (int i = 1; i <= totalColumnas; i++) {
                if (i > 1) System.out.print(",  ");
                System.out.print(rsmd.getColumnName(i) + ": " + rs.getString(i));
            }
            System.out.println("");
        }
    } catch(SQLException se) {
        System.out.println("Error grave al mostrar datos");
        se.printStackTrace();
        System.exit(1);
    }
}

public void Ejercicio(String numero, String sql) {
    System.out.println("Ejercicio " + numero + ":");
```

```
        ResultSet rs = Consultar(sql);
        Mostrar(rs);
    }

    public void Ejercicio1() {
        Ejercicio("1", "SELECT codigoBarras, precioSinIva FROM producto");
    }

    public void Ejercicio2() {
        Ejercicio("2", "select nombre, case when horas < 8 then 'Cajero' else 'Repartidor' end as tipo
from trabajador");
    }

    public void Ejercicio3() {
        // Considerando que los días laborales son de Lunes a Viernes y que 4 horas diarias * 5 dias
hacen 20 semanales
        Ejercicio("3", "SELECT nombre FROM trabajador WHERE horas > 4 AND horas < 8");
    }

    public void Ejercicio4() {
        Ejercicio("4", "select sum(total) as total_facturado from ticket");
    }

    public void Ejercicio5() {
        Ejercicio("5", "select sum(total) as total_facturado from ticket");
    }

    public void Ejercicio6() {
        Ejercicio("6", "select nombre from producto join descuento on producto.codigobarras =
descuento.codigobarras_producto limit 5");
    }
```

```
    public void Ejercicio7() {

        Ejercicio("7", "SELECT SUM(puntuacion)/COUNT(puntuacion) AS notaMedia FROM
opinion");

    }


    public void Ejercicio8() {

        Ejercicio("8", "select count(ticket.fechahoraemision) as numero_tickets, nombre, ciudad from
ticket join trabajador on ticket.telefonomovil_trabajador = trabajador.telefonomovil join tienda on
trabajador.codigo_tienda = tienda.codigo group by telefonomovil_trabajador, nombre, ciudad order
by numero_tickets desc");

    }


    public void Ejercicio9() {

        Ejercicio("9", "SELECT tienda.codigo AS tienda, COUNT(trabajador) AS nTrabajadores
FROM tienda INNER JOIN trabajador ON tienda.codigo = codigo_tienda GROUP BY
tienda.codigo ORDER BY nTrabajadores ASC");

    }


    public void Ejercicio10() {

        Ejercicio("10", "select nombre, telefonofijo, telefonomovil, (sum(nota) /
count(telefonomovil_trabajador)) as media from clasificacion join trabajador on
trabajador.telefonomovil = clasificacion.telefonomovil_trabajador group by nombre, telefonofijo,
telefonomovil_trabajador, telefonomovil having (sum(nota) / count(telefonomovil_trabajador)) >=
10");

    }


    public void Ejercicio11() {

        Ejercicio("11", "SELECT codigoBarras_producto, descuento FROM oferta WHERE
fechaInicio BETWEEN '2019/05/01 00:00:00' AND '2019/05/01 23:59:59'");

    }


    public void Ejercicio12() {
```

```
    Ejercicio("12", "select distinct email from socio join ticket on telefono = telefono_socio join
articulo on fechahoraemision = fechahoraemision_ticket join oferta on
articulo.codigobarras_producto = oferta.codigobarras_producto and fechainicio >= (DATE('2019-
05-31 12:00:00') - 7) and fechainicio <= '2019-05-31 23:59:59'");

  }


  public void Ejercicio13() {

    Ejercicio("13", "SELECT nombre FROM (SELECT codigo FROM tienda WHERE ciudad
LIKE 'M%') AS temp1 INNER JOIN trabajador ON codigo_tienda = temp1.codigo ORDER BY
nombre ASC");

  }


  public void Ejercicio14() {

    Ejercicio("14", "select email from socio join cupon on telefono = telefono_socio join
descuento on cupon.codigo = codigo_cupon group by email order by sum(cantidad) desc limit 1");

  }


  public void Ejercicio15() {

    Ejercicio("15", "SELECT nombre FROM (SELECT nombre, COUNT(*) AS cantidad FROM
(SELECT fechahoraemision FROM ticket WHERE total < '€0.00') AS temp1 INNER JOIN articulo
ON temp1.fechahoraemision = fechahoraemision INNER JOIN producto ON
articulo.codigoBarras_producto = codigoBarras GROUP BY nombre ORDER BY cantidad DESC
LIMIT 1) AS temp2");

  }


  public void Ejercicio16() {

    Ejercicio("16", "select nombre from ticket join trabajador on ticket.telefonomovil_trabajador =
trabajador.telefonomovil join tienda on trabajador.codigo_tienda = tienda.codigo group by
telefonomovil_trabajador, nombre, ciudad order by count(telefonomovil_trabajador) desc limit 1");

  }


  public void Ejercicio17() {

    Ejercicio("17", "SELECT nombre FROM (SELECT nombre FROM (SELECT telefono_socio,
MAX(puntuacion) AS maxima FROM opinion GROUP BY telefono_socio ORDER BY maxima
DESC LIMIT 1) AS temp1 INNER JOIN socio ON temp1.telefono_socio = telefono) AS temp2");
```

```
    }


    public void Ejercicio18() {

        Ejercicio("18", "select fechahoraemision, total from ticket join trabajador on
ticket.telefonomovil_trabajador = trabajador.telefonomovil join tienda on trabajador.codigo_tienda
= tienda.codigo where nombre like 'A%' and ciudad like 'M%'");

    }


    public void Ejercicio19() {

        Ejercicio("19", "SELECT nombre, fechahoraemision AS ticket FROM (SELECT nombre,
trabajador.telefonomovil FROM (SELECT codigo FROM tienda WHERE ciudad = 'Alcala de
Henares') AS temp1 INNER JOIN trabajador ON temp1.codigo = codigo_tienda) AS temp2 INNER
JOIN ticket ON temp2.telefonomovil = telefonomovil_trabajador");

    }


    public void Ejercicio20() {

        Ejercicio("20", "select fechahoraemision, nombre from ticket join trabajador on
ticket.telefonomovil_trabajador = trabajador.telefonomovil join tienda on trabajador.codigo_tienda
= tienda.codigo join articulo on fechahoraemision = fechahoraemision_ticket left join oferta on
articulo.codigobarras_producto = oferta.codigobarras_producto where oferta.fechainicio is null and
telefono_socio is null and ciudad = 'Alcala de Henares'");

    }


}
```

4.

Se han creado los siguientes grupos:

- Administrador que no tiene ningún tipo de restricción
- Gestor que solo inserta, actualiza, elimina y consulta
- Cajero que inserta, actualiza y consulta la tabla ticket, artículo y producto
- Reponedor que solo inserta, actualiza, elimina y consulta la tabla producto
- Socio que solo inserta, actualiza, elimina y consulta la tabla opinión

Se han creado los siguientes usuarios:

- Juan con contraseña juan que pertenece al grupo de Administrador
- Pedro con contraseña pedro que pertenece al grupo de Gestor
- Pablo con contraseña pablo que pertenece al grupo de Cajero
- Alberto con contraseña Alberto que pertenece al grupo de Reponedor
- Santiago con contraseña santiago que pertenece al grupo de Socio

```
CREATE GROUP administrador;
GRANT ALL PRIVILEGES ON DATABASE supermercado TO administrador;
GRANT ALL PRIVILEGES ON SCHEMA public TO administrador;
GRANT ALL PRIVILEGES ON ALL tables IN schema public TO administrador;

CREATE GROUP gestor;
REVOKE CREATE ON SCHEMA public FROM PUBLIC;
GRANT USAGE ON SCHEMA public TO gestor;
GRANT INSERT, UPDATE, DELETE, SELECT ON ALL tables IN schema public TO gestor;

CREATE GROUP cajero;
REVOKE CREATE ON SCHEMA public FROM PUBLIC;
GRANT USAGE ON SCHEMA public TO cajero;
GRANT INSERT, UPDATE, SELECT ON ticket, producto, articulo TO cajero;

CREATE GROUP reponedor;
REVOKE CREATE ON SCHEMA public FROM PUBLIC;
GRANT USAGE ON SCHEMA public TO reponedor;
GRANT INSERT, UPDATE, SELECT, DELETE ON producto TO reponedor;

CREATE GROUP socio;
REVOKE CREATE ON SCHEMA public FROM PUBLIC;
GRANT USAGE ON SCHEMA public TO socio;
GRANT INSERT, UPDATE, SELECT, DELETE ON opinion TO socio;

CREATE USER juan WITH PASSWORD 'juan' IN GROUP administrador;
CREATE USER pedro WITH PASSWORD 'pedro' IN GROUP gestor;
CREATE USER pablo WITH PASSWORD 'pablo' IN GROUP cajero;
CREATE USER alberto WITH PASSWORD 'alberto' IN GROUP reponedor;
CREATE USER santiago WITH PASSWORD 'santiago' IN GROUP socio;
```

Se han realizado los siguientes códigos SQL para probar el funcionamiento de los permisos de cada grupo, conectando previamente a la base de datos con cada uno de los usuarios:

```
INSERT INTO tienda VALUES (11, 'Torrejón');
UPDATE tienda SET ciudad = 'Alcalá' WHERE codigo = 11;
SELECT * FROM tienda;
DELETE FROM tienda WHERE codigo = 11;
CREATE TABLE prueba (codigo serial PRIMARY KEY);
ALTER TABLE prueba ADD COLUMN columna varchar;
DROP TABLE prueba;
```

Además, se han realizado las siguientes pruebas específicas para cada uno de los grupos que requería hacer más comprobaciones y saber si los permisos son correctos.

Conectando a la base de datos con Pablo se han comprobado que los permisos del grupo Cajero son correctos con lo siguiente:

```
INSERT INTO ticket(fechaHoraEmision,total,telefonomovil_trabajador,telefono_socio)
VALUES ('2019-11-15 12:36:07','€105,00','655 514 8373','276 501 6392');
UPDATE ticket SET total = '€150,00' WHERE identificador = 27;
SELECT * FROM ticket;
DELETE FROM ticket WHERE identificador = 27;

INSERT INTO producto(codigobarras,nombre,preciosiniva,stock,porcentaje_iva) VALUES
(51,'Seedlings - Mix, Organic','€9,24',564,0.04);
UPDATE producto SET preciosiniva = '€10,00' WHERE codigobarras = 51;
SELECT * FROM producto;
DELETE FROM producto WHERE codigobarras = 51;

INSERT INTO
articulo(cantidad,fechahoraemision_ticket,codigobarras_producto,telefonomovil_trabajador_tic
ket) VALUES (13,'2019-03-15 21:24:03',22, '956 724 8638');
UPDATE articulo SET cantidad = 14 WHERE fechahoraemision_ticket = '2019-03-15
21:24:03' AND codigobarras_producto = 22;
SELECT * FROM articulo;
DELETE FROM articulo WHERE fechahoraemision_ticket = '2019-03-15 21:24:03' AND
codigobarras_producto = 22;
```

Por ejemplo, como Pablo pertenece al grupo Cajero puede consultar la tabla ticket, pero no crear una tabla nueva:

```
1  SELECT * FROM ticket;
```

Data Output | Explain | Messages | Notifications

	identificador [PK] integer	fechahoraemision timestamp without time zone	total money	codigo_trabajador integer	numero_socio integer
1	1	2019-11-15 06:53:21	132,18 €	27	26
2	2	2018-12-20 01:02:56	-84,84 €	13	2
3	3	2019-06-28 05:43:56	54,23 €	12	12
4	4	2019-08-22 10:30:59	109,05 €	24	20
5	5	2019-07-16 19:11:32	-16,77 €	5	12

```
1  CREATE TABLE prueba (codigo serial PRIMARY KEY);
```

Data Output | Explain | Messages | Notifications

```
ERROR:  permiso denegado al esquema public
LÍNEA 1: CREATE TABLE prueba (codigo serial PRIMARY KEY);
                      ^
SQL state: 42501
Character: 14
```

Conectando a la base de datos con Alberto se han comprobado que los permisos del grupo Reponedor son correctos con lo siguiente:

INSERT INTO producto(codigobarras,nombre,preciosiniva,stock,porcentaje_iva) VALUES (51,'Seedlings - Mix, Organic','€9,24',564,0.04);
UPDATE producto SET stock = 565 WHERE codigobarras = 51;
SELECT * FROM producto;
DELETE FROM producto WHERE codigobarras = 51;

Por ejemplo, como Alberto pertenece al grupo Reponedor puede consultar la tabla producto, pero no otra tabla distinta:

```
1  SELECT * FROM producto;
```

Data Output | Explain | Messages | Notifications

	codigobarras [PK] integer	nombre character varying (150)	preciosiniva money	stock integer	codigo_iva integer
1	1	Mushroom - Chantrelle, Fresh	82,01 €	150	3
2	2	Energy Drink - Franks Original	8,21 €	370	1
3	3	Apple - Delicious, Red	63,95 €	98	3
4	4	Bagels Poppyseed	80,06 €	882	3

```
1  SELECT * FROM tienda;
```

Data Output | Explain | Messages | Notifications

```
ERROR:  permiso denegado a la relación tienda
SQL state: 42501
```

Conectando a la base de datos con Santiago se han comprobado que los permisos del grupo Socio son correctos con lo siguiente:

INSERT INTO opinion(texto,fechahora,puntuacion,codigo_tienda,telefono_socio) VALUES ('Me gusta que vendan comida','2016-03-15 18:27:43',10,10,'105 825 4258');UPDATE opinion SET texto = 'Esta bien' WHERE codigo = 21;
UPDATE opinion SET texto = 'Esta bien' WHERE codigo = 21;
SELECT * FROM opinion;
DELETE FROM opinion WHERE fechahora = '2016-03-15 18:27:43';

Por ejemplo, como Santiago pertenece al grupo Socio puede insertar a la tabla opinión, pero no otra tabla distinta:

```
1  INSERT INTO opinion(codigo,texto,fechahora,puntuacion,codigo_tienda,numero_socio)
2  VALUES (21,'Me gusta que vendan comida','2016-03-15 18:27:43',10,10,20);
```

Data Output | Explain | Messages | Notifications

```
INSERT 0 1

Query returned successfully in 69 msec.
```

```
1  INSERT INTO tienda VALUES (11, 'Torrejón');
```

Data Output | Explain | Messages | Notifications

```
ERROR:  permiso denegado a la relación tienda
SQL state: 42501
```

5.

Usando parte del código Java del ejercicio anterior, se muestra un menú en la pantalla de la consola de Java para seleccionar los distintos usuarios disponibles de cada grupo creado. Una vez seleccionado uno de los usuarios, se ejecutan las consultas en los métodos Probar*NombreGrupo*() para comprobar que dicho grupo tiene los permisos correctos.

```
public class Main {

    public static void main(String[] args) throws SQLException {
        PECL2 pecl2 = new PECL2();

        Scanner scan = new Scanner(System.in);
        int i;
        boolean bucle = true;

        while (bucle) {
            System.out.println("--------------------------------");
            System.out.println("1: Juan del grupo adminitrador.");
            System.out.println("2: Pedro del grupo gestor.");
            System.out.println("3: Pablo del grupo cajero.");
            System.out.println("4: Alberto del grupo reponedor.");
            System.out.println("5: Santiago del grupo socio.");
            System.out.println("--------------------------------");
            System.out.println("Selecciona un número de usuario o escribe 0 para terminar el
programa:\n");
```

```
            i = scan.nextInt();

            pecl2.Desconectar();

            switch (i) {
                case 0:
                    bucle = false;
                    break;
                case 1:
                    pecl2.Conectar("juan", "juan");
                    pecl2.ProbarAdministrador();
                    break;
                case 2:
                    pecl2.Conectar("pedro", "pedro");
                    pecl2.ProbarGestor();
                    break;
                case 3:
                    pecl2.Conectar("pablo", "pablo");
                    pecl2.ProbarCajero();
                    break;
                case 4:
                    pecl2.Conectar("alberto", "alberto");
                    pecl2.ProbarReponedor();
                    break;
                case 5:
                    pecl2.Conectar("santiago", "santiago");
                    pecl2.ProbarSocio();
                    break;
            }
        }
    }

public class PECL2 {
    private Statement s;
    private Connection c;

    public void Conectar(String usuario, String contrasena) {
        try {
            Class.forName("org.postgresql.Driver");
        } catch(ClassNotFoundException cnfe) {
            cnfe.printStackTrace();
            System.exit(1);
        }

        try {
            c = DriverManager.getConnection("jdbc:postgresql://127.0.0.1:5432/supermercado",
                usuario, contrasena);
        } catch(SQLException se) {
            se.printStackTrace();
```

```
            System.exit(1);
        }

        try {
            s = c.createStatement();
        } catch(SQLException se) {
            se.printStackTrace();
            System.exit(1);
        }
    }

    public void Desconectar() throws SQLException {
        if (s != null) {
            s.close();
        }

        if (c != null) {
            s.close();
        }
    }

    public ResultSet Consultar(String sql) {
        ResultSet rs = null;
        try {
            rs = s.executeQuery(sql);
        } catch(SQLException se) {
            se.printStackTrace();
            System.exit(1);
        }

        return rs;
    }

    public void Actualizar(String sql) {
        try {
            s.executeUpdate(sql);
        } catch (SQLException se) {
            se.printStackTrace();
            System.exit(1);
        }
    }

    public void Mostrar(ResultSet rs) {
        int index = 0;
        try {
            while (rs.next()) {
                ResultSetMetaData rsmd = rs.getMetaData();
                int totalColumnas = rsmd.getColumnCount();
                for (int i = 1; i <= totalColumnas; i++) {
```

```
                if (i > 1) System.out.print(",  ");
                System.out.print(rsmd.getColumnName(i) + ": " + rs.getString(i));
            }
            System.out.println("");
        }
    } catch(SQLException se) {
        System.out.println("Error grave al mostrar datos");
        se.printStackTrace();
        System.exit(1);
    }
}

public void Probar(String sql) {
    ResultSet rs = Consultar(sql);
    if (rs != null) {
        Mostrar(rs);
    }
}

void ProbarAdministrador() {
    Actualizar("INSERT INTO tienda VALUES (11, 'Torrejón')");
    Actualizar("UPDATE tienda SET ciudad = 'Alcalá' WHERE codigo = 11");
    Probar("SELECT * FROM tienda");
    Actualizar("DELETE FROM tienda WHERE codigo = 11");
    Actualizar("CREATE TABLE prueba (codigo serial PRIMARY KEY)");
    Actualizar("ALTER TABLE prueba ADD COLUMN columna varchar");
    Actualizar("DROP TABLE prueba");
}

void ProbarGestor() {
    Actualizar("INSERT INTO tienda VALUES (11, 'Torrejón')");
    Actualizar("UPDATE tienda SET ciudad = 'Alcalá' WHERE codigo = 11");
    Probar("SELECT * FROM tienda");
    Actualizar("DELETE FROM tienda WHERE codigo = 11");
}

void ProbarCajero() {
    Actualizar("INSERT INTO
ticket(fechaHoraEmision,total,telefonomovil_trabajador,telefono_socio) VALUES ('2019-11-15
12:36:07','€105,00','655 514 8373','276 501 6392')");
    Actualizar("UPDATE ticket SET total = '€150,00' WHERE fechaHoraEmision = '2019-
11-15 12:36:07'");
    Probar("SELECT * FROM ticket");

    Actualizar("INSERT INTO
producto(codigobarras,nombre,preciosiniva,stock,porcentaje_iva) VALUES (51,'Seedlings -
Mix, Organic','€9,24',564,0.04)");
    Actualizar("UPDATE producto SET preciosiniva = '€10,00' WHERE codigobarras =
51");
```

```
        Probar("SELECT * FROM producto");

        Actualizar("INSERT INTO
articulo(cantidad,fechahoraemision_ticket,codigobarras_producto,telefonomovil_trabajador_ti
cket) VALUES (13,'2019-03-15 21:24:03',22, '956 724 8638')");
        Actualizar("UPDATE articulo SET cantidad = 14 WHERE fechahoraemision_ticket =
'2019-03-15 21:24:03' AND codigobarras_producto = 22");
        Probar("SELECT * FROM articulo");
    }

    void ProbarReponedor() {
        Actualizar("INSERT INTO
producto(codigobarras,nombre,preciosiniva,stock,porcentaje_iva) VALUES (51,'Seedlings -
Mix, Organic','€9,24',564,0.04)");
        Actualizar("UPDATE producto SET stock = 565 WHERE codigobarras = 52");
        Probar("SELECT * FROM producto");
        Actualizar("DELETE FROM producto WHERE codigobarras = 52");
    }

    void ProbarSocio() {
        Actualizar("INSERT INTO
opinion(texto,fechahora,puntuacion,codigo_tienda,telefono_socio) VALUES ('Me gusta que
vendan comida','2016-03-15 18:27:43',10,10,'105 825 4258')");
        Actualizar("UPDATE opinion SET texto = 'Esta bien' WHERE fechahora = '2016-03-15
18:27:43'");
        Probar("SELECT * FROM opinion");
        Actualizar("DELETE FROM opinion WHERE fechahora = '2016-03-15 18:27:43'");
    }

}
```