XUNTA DE GALICIA | CONSELLERÍA DE CULTURA, EDUCACIÓN, FORMACIÓN PROFESIONAL E UNIVERSIDADES
UNIÓN EUROPEA
IES SAN MAMEDE
Rúa do Castelo Nº 3. 32700. Maceda (Ourense)
Tlf: 988 788 561 ies.san.mamede@edu.xunta.gal
http://www.iessanmamede.com

DISTRIBUIDORA DE PRODUCTOS QUÍMICOS

https://github.com/uxiarddmg/ProyectoBBDD.git

Uxía Rodríguez Domínguez

Curso: ***1º DAM*** **Materia:** ***Bases de Datos – Proyecto Final 24/25***

Contido

1. Introducción

En este proyecto final de la materia de Bases de Datos de 1º de DAM se tratará con un supuesto cliente generado por la aplicación PartyRock el cual se dedica a la distribución de productos químicos. Se realizará todo el proceso de creación de la base de datos necesarios para esta empresa.

2. Descripción del Problema / Requisitos

El problema propuesto por el cliente es el siguiente:

"Soy propietario de una distribuidora de productos químicos que abastece a industrias y laboratorios en la región. Nos especializamos en la comercialización de productos químicos industriales, solventes, reactivos y materias primas para diversos sectores productivos.

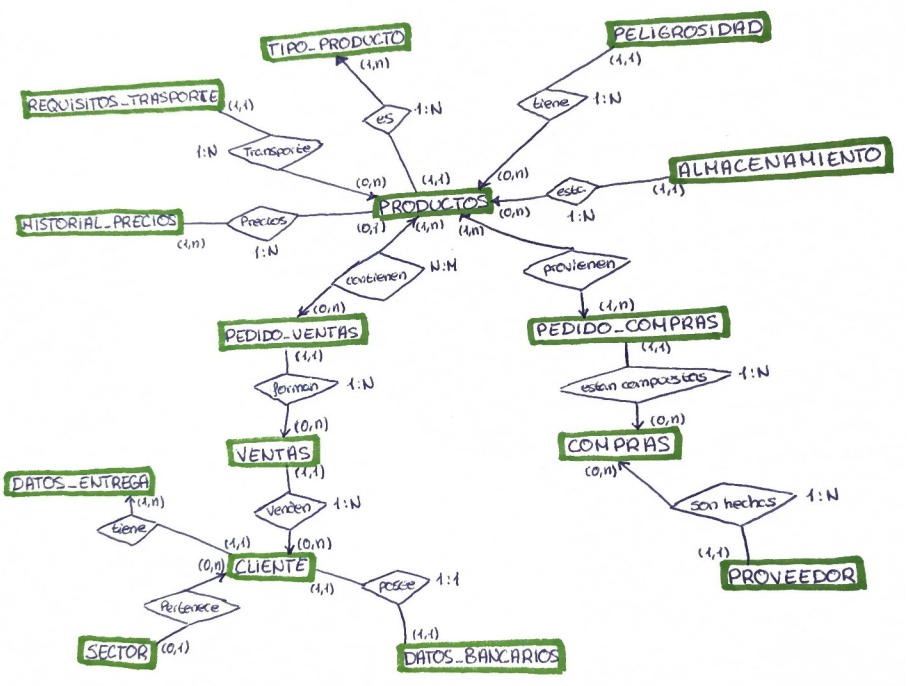
Nuestra operación se centra en mantener un inventario óptimo de productos almacenados bajo estrictas normas de seguridad en nuestras instalaciones certificadas. Contamos con una flota de transporte especializado para garantizar entregas seguras y puntuales.

La clave de nuestro negocio está en las relaciones sólidas tanto con proveedores internacionales como con clientes locales, además del cumplimiento riguroso de todas las regulaciones y certificaciones necesarias para el manejo de productos químicos. Nuestro personal está altamente capacitado en el manejo de materiales peligrosos y protocolos de seguridad.

Los márgenes del negocio varían según el tipo de producto, pero nos enfocamos en mantener precios competitivos mientras aseguramos la calidad y autenticidad de todos nuestros productos."

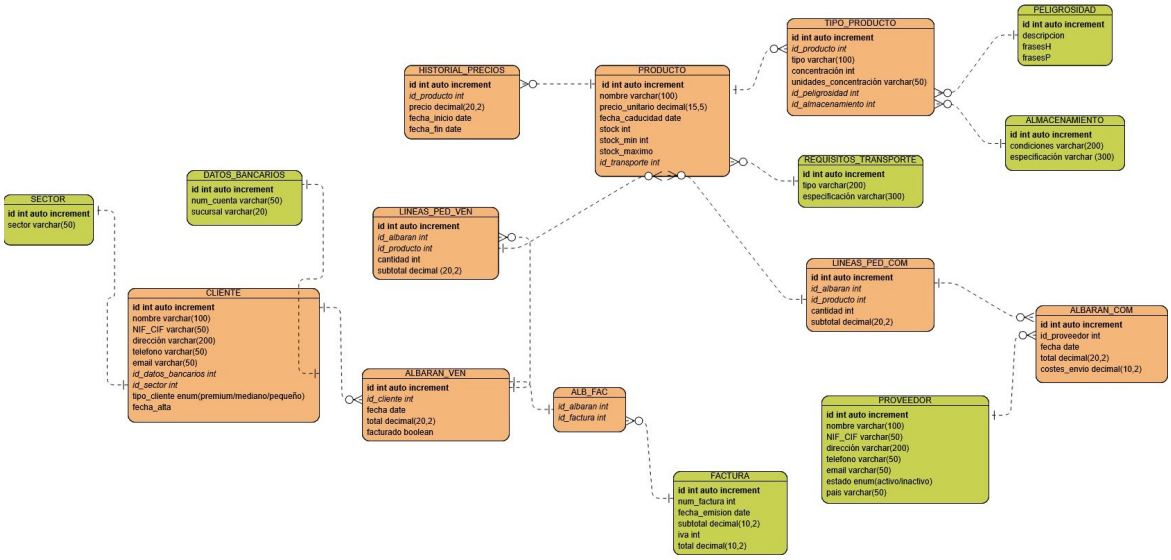
Se realizan las preguntas pertinentes para poder sacar las entidades y sus atributos para poder realizar la primera aproximación del modelo conceptual.

3. Modelo Conceptual



4. Modelo Relacional

Se realizan varios cambios al realizar el modelo relacionar con respecto al modelo conceptual. Viendo la necesidad de la creación de nuevas tablas y eliminación de otras. Ya que se observa la necesidad se separar las líneas de cada pedido, los albaranes y las facturas.



Tablas en verde → no tienen claves foráneas

Visualización de las tablas en texto plano:

PRODUCTOS: id(PK), nombre, precio_unitario, fecha_caducidad, stock, stock_minimo, stock_maximo, id_requisitos_transporte(FK)

TIPO_PRODUCTO: id(PK), id_producto(FK), tipo, concentración, unidades_concentración, peligrosidad(FK), almacenamiento(FK)

PELIGROSIDAD: id(PK), descripcion, frasesH, frasesP

ALMACENAMIENTO: id(PK), condiciones, especificación (varchar 300)

REQUISITOS_TRANSPORTE: id(PK), tipo, restricciones (varchar 100)

HISTORIAL_PRECIOS: id (PK), id_producto (FK), precio, fecha_inicio, fecha_fin

LINEAS_PED_VENT: id(PK), id_albarán(FK), id_producto(FK), cantidad, subtotal

ALB_FAC: id_albaran(FK), id_factura(FK)

ALBARAN_VEN: id(PK), id_cliente(FK), fecha, total, facturado(si/no)

FACTURA: id(PK) , num_factura, fecha_emision, subtotal, iva, total

CLIENTES: id(PK), nombre, NIF_CIF, direccion_fiscal, telefono, email, datos_bancarios(FK), sector(FK), tipo_cliente (premium/mediano/pequeño), fecha_alta

DATOS_BANCARIOS: id(PK), num_cuenta, sucursal

SECTOR: id(PK), sector.

ALBARAN_COM: id(PK), id_proveedor (FK), fecha, total, costesEnvio

LINEAS_PEDIDOS_COMPRAS: id(PK), id_albaran(FK), id_producto(FK), cantidad, subtotal

PROVEEDORES: id(PK), nombre, NIF_CIF, dirección_fiscal, telefono, email, estado (activo/inactivo), país

5. Proceso de Normalización

Primera Forma Normal (1FN)

- Valores atómicos (indivisibles)
- Sin grupos repetitivos
- Clave primaria definid

Todas las tablas propuestas la cumplen

Segunda Forma Normal (2FN)

- Cumple 1FN
- Sin dependencias parciales

Creación de una tabla independiente de medidas de concentración

UNIDADES_CONCENTRACION: id(PK), unidad, descripción

El subtotal en la tabla LINEAS_PED_VEN debe ser calculado

Tercera Forma Normal (3FN)

- Cumple 2FN
- Sin dependencias transitivas

Se identifican dependencias transitivas:

Mover stock_min y stock_max de PRODUCTO a TIPO_PRODUCTO:

PRODUCTOS: id(PK), nombre, precio_unitario, fecha_caducidad, stock, id_requisitos_transporte(FK)

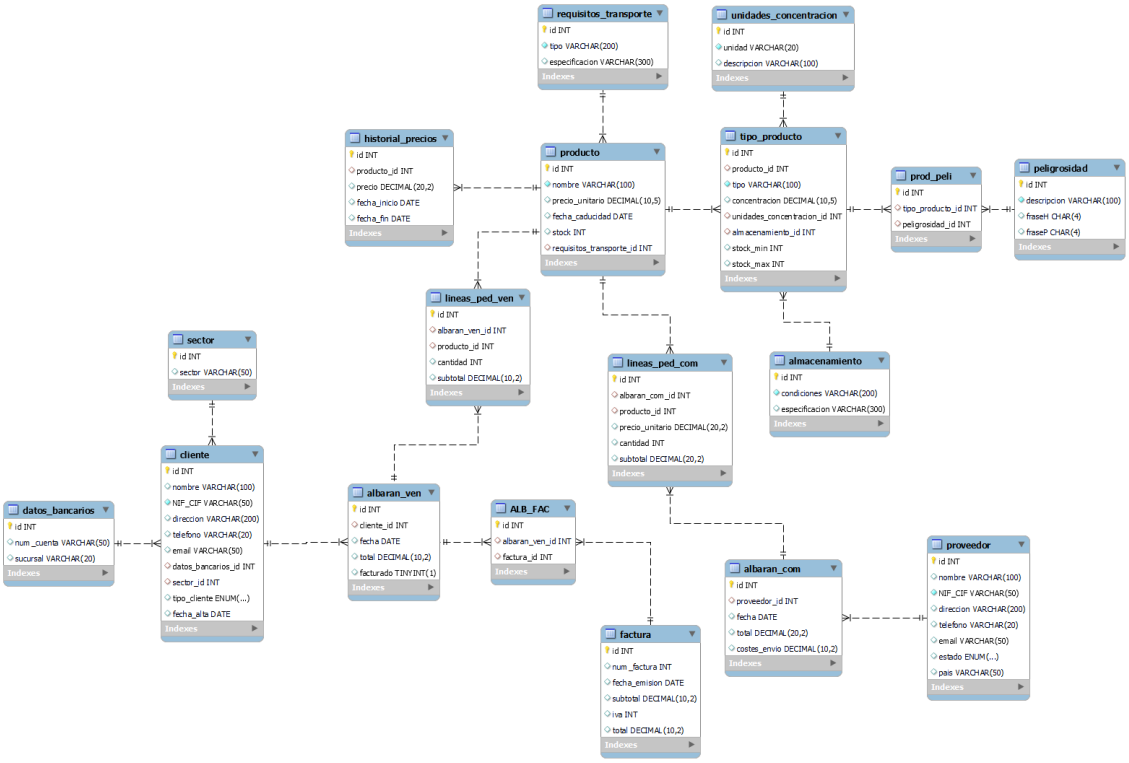
TIPO_PRODUCTO: id(PK), id_producto(FK), tipo, concentración, unidades_concentración, peligrosidad(FK), almacenamiento(FK), stock_minimo, stock_maximo

El total de los albaranes debe ser un campo calculado

Entre tipo_producto y peligrosidad debe existir una nueva tabla que relacione ambos, ya que un tipo de producto puede tener varias categorías de peligrosidad.

PRO_PEL: id(PK), id_tipo_producto(FK), id_peligrosidad(FK)

6. Script de Creación de la Base de Datos




```
drop database if exists quimicos;
create database quimicos;

use quimicos;
create table peligrosidad(
id int auto_increment primary key,
descripcion varchar(100) unique not null,
fraseH char(4),
fraseP char(4));

create table almacenamiento(
id int auto_increment primary key,
condiciones varchar(200) not null,
especificacion varchar(300));

create table requisitos_transporte(
id int auto_increment primary key,
tipo varchar(200) unique not null,
especificacion varchar(300));

create table producto(
id int auto_increment primary key,
nombre varchar(100) not null, -- No puede ser unico ya que puede existir de
varias concentraciones
precio_unitario decimal(10,2),
fecha_caducidad date,
stock int,
requisitos_transporte_id int, -- cambio el orden del nombre para que sea más
facil de leer
foreign key (requisitos_transporte_id) references requisitos_transporte (id)
on update cascade);
```

```
create table unidades_concentracion(
id int auto_increment primary key,
unidad varchar(20) unique not null,
descripcion varchar(100));

create table tipo_producto(
id int auto_increment primary key,
producto_id int,
tipo varchar(100) not null,
concentracion decimal(20,5),
unidades_concentracion_id int,
almacenamiento_id int,
stock_min int,
stock_max int,
foreign key (producto_id) references producto (id),
foreign key (unidades_concentracion_id) references unidades_concentracion
(id),
foreign key (almacenamiento_id) references almacenamiento (id));

create table prod_peli(
id int auto_increment primary key,
tipo_producto_id int,
peligrosidad_id int,
foreign key (tipo_producto_id) references tipo_producto (id),
foreign key (peligrosidad_id) references peligrosidad (id));

create table historial_precios(
id int auto_increment primary key,
producto_id int,
precio decimal(20,2),
fecha_inicio date,
fecha_fin date,
foreign key (producto_id) references producto (id));

create table proveedor(
id int auto_increment primary key,
nombre varchar(100),
NIF_CIF varchar(50) unique not null,
direccion varchar(200),
telefono varchar(20),
email varchar(50),
estado enum("activo", "inactivo"),
pais varchar(50));

create table albaran_com(
id int auto_increment primary key,
proveedor_id int,
fecha date,
total decimal(20,2),
costes_envio decimal(10,2),
foreign key (proveedor_id) references proveedor (id));

create table lineas_ped_com(
id int auto_increment primary key,
albaran_com_id int,
producto_id int,
precio_unitario decimal(20,2),
cantidad int,
subtotal decimal(20,2) generated always as (precio_unitario * cantidad)
stored,
foreign key (albaran_com_id) references albaran_com (id),
foreign key (producto_id) references producto (id));
```

```
create table sector(
id int auto_increment primary key,
sector varchar(50));

create table datos_bancarios(
id int auto_increment primary key,
num_cuenta varchar(50),
sucursal varchar(100));

create table cliente(
id int auto_increment primary key,
nombre varchar(100),
NIF_CIF varchar(50) unique not null,
direccion varchar(200),
telefono varchar(20),
email varchar(50),
datos_bancarios_id int,
sector_id int,
tipo_cliente enum("premium", "mediano", "pequeño"),
fecha_alta date,
foreign key (sector_id) references sector (id),
foreign key (datos_bancarios_id) references datos_bancarios (id));

create table albaran_ven(
id int auto_increment primary key,
cliente_id int,
fecha date,
total decimal(10,2),
facturado boolean,
foreign key (cliente_id) references cliente (id));

create table lineas_ped_ven(
id int auto_increment primary key,
albaran_ven_id int,
producto_id int,
cantidad int,
subtotal decimal(10,2),
foreign key (albaran_ven_id) references albaran_ven (id),
foreign key (producto_id) references producto (id));

create table factura(
id int auto_increment primary key,
num_factura int,
fecha_emision date,
subtotal decimal(10,2),
iva int,
total decimal(10,2) generated always as (subtotal * iva/100) stored);

create table alb_fac(
id int auto_increment primary key,
albaran_ven_id int,
factura_id int,
foreign key (albaran_ven_id) references albaran_ven (id),
foreign key (factura_id) references factura (id));
```

7. Carga de Datos Inicial

Action Output

#	Time	Action	Message	Duration / Fetch
1	13:20:48	use quimicos	0 row(s) affected	0.000 sec
2	13:20:48	insert into peligrosidad (descripcion, fraseH, fraseP) values ('Inflam...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.031 sec
3	13:20:48	insert into almacenamiento (condiciones, especificacion) values ('A...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.016 sec
4	13:20:48	insert into requisitos_transporte (tipo, especificacion) values ('Susta...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.015 sec
5	13:20:48	insert into unidades_concentracion (unidad, descripcion) values ('...	14 row(s) affected Records: 14 Duplicates: 0 Warnings: 0	0.016 sec
6	13:20:48	insert into producto (nombre, precio_unitario, fecha_caducidad, st...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.016 sec
7	13:20:48	insert into tipo_producto (producto_id, tipo, concentracion, unidad...	11 row(s) affected Records: 11 Duplicates: 0 Warnings: 0	0.015 sec
8	13:20:48	insert into prod_peli (tipo_producto_id, peligrosidad_id) values (1, 2...	23 row(s) affected Records: 23 Duplicates: 0 Warnings: 0	0.016 sec
9	13:20:48	insert into historial_precios (producto_id, precio, fecha_inicio, fech...	25 row(s) affected Records: 25 Duplicates: 0 Warnings: 0	0.016 sec
10	13:20:48	insert into proveedor (nombre, NIF_CIF, direccion, telefono, email, ...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.015 sec
11	13:20:48	insert into albaran_com (proveedor_id, fecha, total, costes_envio) ...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.016 sec
12	13:20:48	insert into lineas_ped_com (albaran_com_id, producto_id, precio_...	5 row(s) affected Records: 5 Duplicates: 0 Warnings: 0	0.015 sec
13	13:20:48	insert into sector (sector) values ('Química'), ('Biotecnología'), ('Agr...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.016 sec
14	13:20:48	insert into datos_bancarios (num_cuenta, sucursal) values ('ES12 ...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.016 sec
15	13:20:48	insert into cliente (nombre, NIF_CIF, direccion, telefono, email, dat...	15 row(s) affected Records: 15 Duplicates: 0 Warnings: 0	0.015 sec
16	13:20:48	insert into albaran_ven (cliente_id, fecha, total, facturado) values (...	10 row(s) affected Records: 10 Duplicates: 0 Warnings: 0	0.016 sec
17	13:20:48	insert into lineas_ped_ven (albaran_ven_id, producto_id, cantidad,...	11 row(s) affected Records: 11 Duplicates: 0 Warnings: 0	0.016 sec
18	13:20:49	insert into factura (num_factura, fecha_emision, subtotal, iva) valu...	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.015 sec
19	13:20:49	insert into alb_fac (albaran_ven_id, factura_id) values (1, 1), (3, 2),...	6 row(s) affected Records: 6 Duplicates: 0 Warnings: 0	0.016 sec

8. Funciones y Procedimientos Almacenados

Procedimientos

A. Actualizar el precio de un producto

```
drop procedure if exists actualizar_precio_producto;


delimiter //
create procedure actualizar_precio_producto(in nombre_producto varchar(100),
precio_nuevo decimal(10,2))
begin
	declare id_producto int;
    set id_producto = (select id from producto
		where nombre = nombre_producto);
    update producto set precio_unitario = precio_nuevo
		where id = id_producto;
    update historial_precios set fecha_fin = curdate()
		where producto_id = id_producto;
    insert into historial_precios (producto_id, precio,    fecha_inicio)
values(id_producto, precio_nuevo, curdate());
end;
//
delimiter ;
```

B. Consultar productos por debajo del stock mínimo

```
drop procedure if exists stock_bajo_minimo;

delimiter //
create procedure stock_bajo_minimo()
begin
	select P.id, P.nombre, P.stock, TP.stock_min, TP.stock_max
		from producto as P
		join tipo_producto as TP on P.id = TP.producto_id
		where P.stock < TP.stock_min;
end;
//
delimiter ;
```

Funciones

C. Obtener el precio actual de un producto

```
drop function if exists precio_actual;


delimiter //
create function precio_actual(nombre_producto varchar(100)) returns
decimal(10,2)
deterministic
begin
	declare precio decimal(10,2);
	set precio = (select HP.precio from historial_precios as HP
		join producto as P on HP.producto_id = P.id
		where nombre_producto like P.nombre and HP.fecha_fin
		is null);
 return precio;
end;
//
delimiter ;
```

D. Cantidad total vendida de un producto en un periodo

```
drop function if exists cantidad_vendida;

delimiter //
create function cantidad_vendida(prod_id int, fecha_inicio date, fecha_fin
date) returns int
deterministic
begin
	declare cantidad int;
	set cantidad = (select sum(LPV.cantidad)
		from lineas_ped_ven as LPV
		join albaran_ven as AV on LPV.albaran_ven_id = AV.id
		where LPV.producto_id = prod_id and AV.fecha between
		fecha_inicio and fecha_fin);

    return cantidad;
end;
//
delimiter ;
```

9. Triggers

E. Antes de realizar una venta comprobar si hay stock suficiente

```
drop trigger if exists comprobar_stock;


delimiter //
create trigger comprobar_stock
before insert on lineas_ped_ven
for each row
begin
	if(new.cantidad < (select stock from producto where id =
		new.producto_id)) then
		update producto set stock = stock - new.cantidad where id =
		new.producto_id;
	else
		signal sqlstate '45000'
	  set message_text = 'No hay stock suficiente';
	end if;
end;
//
delimiter ;
```

F. Despues de realizar una compra aumentar el stock

```
drop trigger if exists actualizar_stock;


delimiter //
create trigger actualizar_stock
before insert on lineas_ped_com
for each row
begin
	if((new.cantidad +
		(select stock from producto where id = new.producto_id)) <
		(select stock_max from tipo_producto where id =
	new.producto_id)) then
	  update producto set stock = stock + new.cantidad where id =
	new.producto_id;
	else
		signal sqlstate '45000'
		set message_text = 'Supera el stock max permitido';
	end if;
end;
//
delimiter ;
```

G. Auditoria de la tabla clientes

```
create table auditoria_Tcliente(
id int auto_increment primary key,
fecha_cambio date,
id_cliente int,
nombre varchar(100),
NIF_CIF varchar(50),
direccion varchar(200),
telefono varchar(20),
email varchar(50),
datos_bancarios_id int,
sector_id int,
tipo_cliente enum("premium", "mediano", "pequeño"),
fecha_alta date
);


drop trigger if exists auditoria_clientes;


delimiter //
create trigger auditoria_clientes
after update on cliente
for each row
begin
	insert into auditoria_Tcliente(fecha_cambio, id_cliente, nombre,
	NIF_CIF, direccion, telefono, email, datos_bancarios_id,
	sector_id, tipo_cliente, fecha_alta)

	values

		(now(), old.id, old.nombre, old.NIF_CIF, old.direccion,
	old.telefono, old.email, old.datos_bancarios_id,
	old.sector_id, old.tipo_cliente, old.fecha_alta);
end;
//
delimiter ;
```

10. Consultas SQL

1. Para cada producto mostrar que tipos de producto tiene, su concentración y su peligrosidad

```
select P.nombre, TP.tipo, TP.concentracion, UC.unidad, PE.descripcion,
PE.fraseH, PE.fraseP
from producto as P
join tipo_producto as TP on P.id = TP.producto_id
join unidades_concentracion as UC on TP.unidades_concentracion_id = UC.id
join prod_peli as PP on TP.id = tipo_producto_id
join peligrosidad as PE on PP.peligrosidad_id = PE.id;
```

2. 5 sectores a los que pertenecen más clientes

```
select S.sector, count(C.id) as 'Número de clientes'
from sector as S
join cliente as C on S.id = C.sector_id
group by C.sector_id
order by count(C.id) desc
limit 5;
```

3. Clientes con mayor volumen de compras

```
select C.nombre, sum(AV.total) as 'Total comprado'
from cliente as C
join albaran_ven as AV on C.id = AV.cliente_id
group by C.id
order by sum(AV.total) desc
limit 5;
```

4. Productos que nunca han sido vendidos

```
select P.nombre
from producto as P
left join lineas_ped_ven as LPV on P.id = LPV.producto_id
where LPV.producto_id is null;
```

5. Clientes inactivos en el último año

```
select C.nombre
from cliente as C
where C.id not in (select distinct AV.cliente_id
                   from albaran_ven as AV where AV.fecha between
                   date_sub(now(),interval 1 year)and now()
                   );
```

6. Producto que más varia su precio

```
select P.nombre, count(HP.id) as 'N° de cambios'
from producto as P
join historial_precios as HP on P.id = HP.producto_id
group by HP.producto_id
order by count(HP.id) desc;
```

7. Proveedor al que mas compras se le han realizado

```
select PR.nombre, count(AC.id) as 'N° de Albaranes'
from proveedor as PR
join albaran_com as AC on PR.id = AC.proveedor_id
group by PR.id
order by count(AC.id) desc
limit 1;
```

8. Datos bancarios de clientes

```
select C.id, C.nombre, C.NIF_CIF, DB.num_cuenta
from cliente as C
join datos_bancarios as DB on C.datos_bancarios_id = DB.id;
```

9. Productos más vendidos

```
select P.nombre, sum(LPV.cantidad) as 'Total vendido'
from producto as P
join lineas_ped_ven as LPV on P.id = LPV.producto_id
group by P.id
order by sum(LPV.cantidad) desc
limit 5;
```

10. Productos con stock mínimo no alcanzado --> se deben comprar

```
select P.nombre, P.stock, TP.stock_min
from producto as P
join tipo_producto as TP on P.id = TP.producto_id
where P.stock < TP.stock_min;
```

11. Casos de Prueba y Simulación

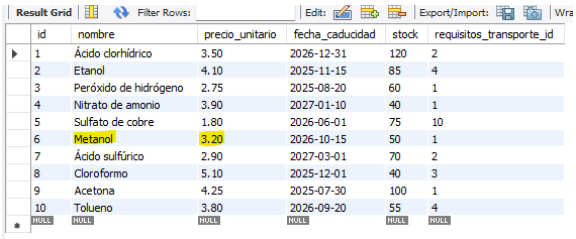
Incluidos en el script *ScriptProcedimientos_Funciones_Triggers.sql*

Pruebas Procedimientos

Prueba procedimiento A

```
call actualizar_precio_producto("Metanol", 3.30);
```

Precio inicial del Metanol en la tabla productos

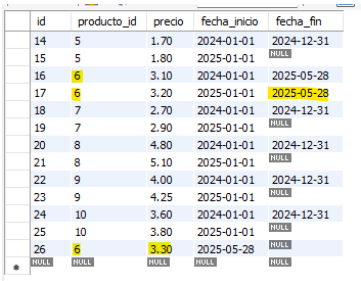


id	nombre	precio_unitario	fecha_caducidad	stock	requisitos_transporte_id
1	Ácido clorhídrico	3.50	2026-12-31	120	2
2	Etanol	4.10	2025-11-15	85	4
3	Peróxido de hidrógeno	2.75	2025-08-20	60	1
4	Nitrato de amonio	3.90	2027-01-10	40	1
5	Sulfato de cobre	1.80	2026-06-01	75	10
6	Metanol	3.20	2026-10-15	50	1
7	Ácido sulfúrico	2.90	2027-03-01	70	2
8	Cloroformo	5.10	2025-12-01	40	3
9	Acetona	4.25	2025-07-30	100	1
10	Tolueno	3.80	2026-09-20	55	4

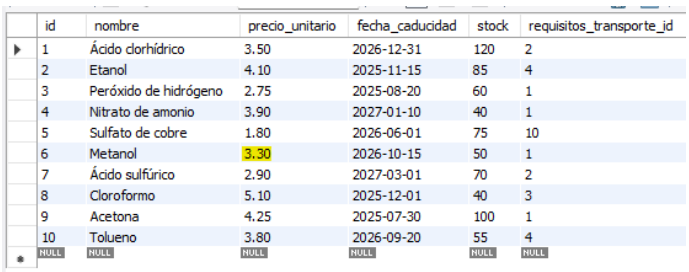
Precios del Metanol producto_id = 6 en la tabla historial_precio

id	producto_id	precio	fecha_inicio	fecha_fin
10	4	3.60	2023-03-01	2024-02-28
11	4	3.80	2024-03-01	2025-02-28
12	4	3.90	2025-03-01	NULL
13	5	1.50	2022-01-01	2023-12-31
14	5	1.70	2024-01-01	2024-12-31
15	5	1.80	2025-01-01	NULL
16	6	3.10	2024-01-01	2024-12-31
17	6	3.20	2025-01-01	NULL
18	7	2.70	2024-01-01	2024-12-31
19	7	2.90	2025-01-01	NULL
20	8	4.80	2024-01-01	2024-12-31
21	8	5.10	2025-01-01	NULL

Cambios en los precios en historial_precios y actualización en productos



id	producto_id	precio	fecha_inicio	fecha_fin
14	5	1.70	2024-01-01	2024-12-31
15	5	1.80	2025-01-01	NULL
16	6	3.10	2024-01-01	2025-05-28
17	6	3.20	2025-01-01	2025-05-28
18	7	2.70	2024-01-01	2024-12-31
19	7	2.90	2025-01-01	NULL
20	8	4.80	2024-01-01	2024-12-31
21	8	5.10	2025-01-01	NULL
22	9	4.00	2024-01-01	2024-12-31
23	9	4.25	2025-01-01	NULL
24	10	3.60	2024-01-01	2024-12-31
25	10	3.80	2025-01-01	NULL
26	6	3.30	2025-05-28	NULL



id	nombre	precio_unitario	fecha_caducidad	stock	requisitos_transporte_id
1	Ácido clorhídrico	3.50	2026-12-31	120	2
2	Etanol	4.10	2025-11-15	85	4
3	Peróxido de hidrógeno	2.75	2025-08-20	60	1
4	Nitrato de amonio	3.90	2027-01-10	40	1
5	Sulfato de cobre	1.80	2026-06-01	75	10
6	Metanol	3.30	2026-10-15	50	1
7	Ácido sulfúrico	2.90	2027-03-01	70	2
8	Cloroformo	5.10	2025-12-01	40	3
9	Acetona	4.25	2025-07-30	100	1
10	Tolueno	3.80	2026-09-20	55	4

Prueba procedimiento B

```
update producto set stock = 14 where id = 2;
call stock_bajo_minimo();
```

Se actualiza el stock del Etanol

id	nombre	precio_unitario	fecha_caducidad	stock	requisitos_transporte_id
1	Ácido clorhídrico	3.50	2026-12-31	120	2
2	Etanol	4.10	2025-11-15	14	4
3	Peróxido de hidrógeno	2.75	2025-08-20	60	1

Se busca si hay algún producto con stock por debajo del mínimo establecido

id	nombre	stock	stock_min	stock_max
2	Etanol	14	15	120

Pruebas Funciones

Prueba Función C

```
select id from producto where nombre like "Peróxido de Hidrogeno";
select precio_actual("Peróxido de Hidrogeno") as 'Precio actual';
```

Se busca el id del “Peroxido de Hidrogeno” en la tabla productos → id = 3

Se busca el precio actual del producto con id 3 en historial_precios

id	producto_id	precio	fecha_inicio	fecha_fin
4	2	3.80	2023-04-01	2024-05-31
5	2	4.00	2024-04-01	2025-05-31
6	2	4.10	2025-04-01	NULL
7	3	2.50	2022-09-01	2023-08-31
8	3	2.65	2023-09-01	2024-08-31
9	3	2.75	2024-09-01	NULL
10	4	3.60	2023-03-01	2024-02-28

Al llamar a la función se puede ver el precio en la actualidad del “Peroxido de Hidrogeno”

Precio actual
2.75

Prueba Función D

```
select cantidad_vendida(1, '2025-05-01', '2025-06-01') as 'Cantidad';
```

En la tabla de hineas_ped_ven se ven los albaranes en las que se vendió el producto 1

id	albaran_ven_id	producto_id	cantidad	subtotal
1	1	1	100	350.00
2	1	1	200	700.00
3	2	2	80	320.00

Se comprueba las fecha en albaran_ven y el resultado de la función

id	cliente_id	fecha	total	facturado
1	1	2025-05-01	1050.00	1
2	2	2025-05-03	320.00	0
3	3	2025-05-05	180.00	1

Result Grid

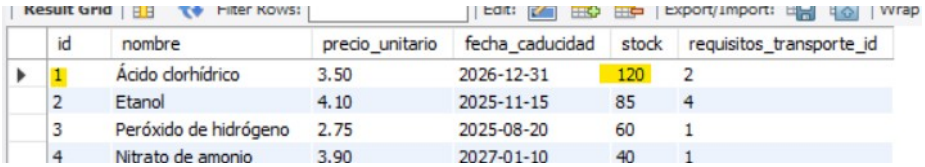
Cantidad
300

Pruebas Triggers

Prueba Trigger E

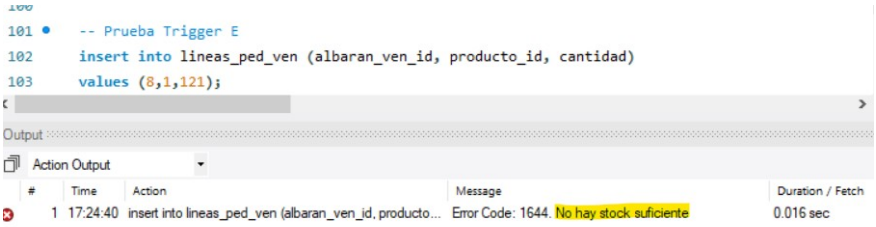
```
insert into lineas_ped_ven (albaran_ven_id, producto_id, cantidad)
values (8,1,121);
```

Se comprueba el stock del "Ácido clorhídrico" id = 1 y es de 120



id	nombre	precio_unitario	fecha_caducidad	stock	requisitos_transporte_id
1	Ácido clorhídrico	3.50	2026-12-31	120	2
2	Etanol	4.10	2025-11-15	85	4
3	Peróxido de hidrógeno	2.75	2025-08-20	60	1
4	Nitrato de amonio	3.90	2027-01-10	40	1

Se intenta hacer una venta de 121 unidades del producto y el trigger funciona correctamente



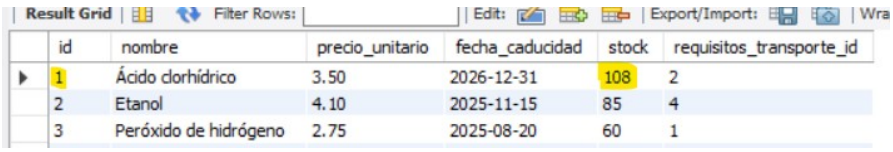
```
101 •  -- Prueba Trigger E
102    insert into lineas_ped_ven (albaran_ven_id, producto_id, cantidad)
103    values (8,1,121);
```

Output

Action Output

#	Time	Action	Message	Duration / Fetch
1	17:24:40	insert into lineas_ped_ven (albaran_ven_id, producto...	Error Code: 1644. No hay stock suficiente	0.016 sec

Sin embargo, si se intenta vender 12 unidades lo permite hacer y actualiza la tabla producto con la disminución de stock correspondiente



id	nombre	precio_unitario	fecha_caducidad	stock	requisitos_transporte_id
1	Ácido clorhídrico	3.50	2026-12-31	108	2
2	Etanol	4.10	2025-11-15	85	4
3	Peróxido de hidrógeno	2.75	2025-08-20	60	1

Prueba Trigger F

```
insert into lineas_ped_com (albaran_com_id, producto_id,
precio_unitario, cantidad) values (4,4,3.90,30);
```

Si se intenta hacer una compra de una cantidad superior al stock máximo permitido impide hacerla y alerta del problema.

```
126 •  -- Prueba Trigger F
127    insert into lineas_ped_com (albaran_com_id, producto_id, precio_unitario, cantidad)
128    values (4,4,3.90,30);
```

Output

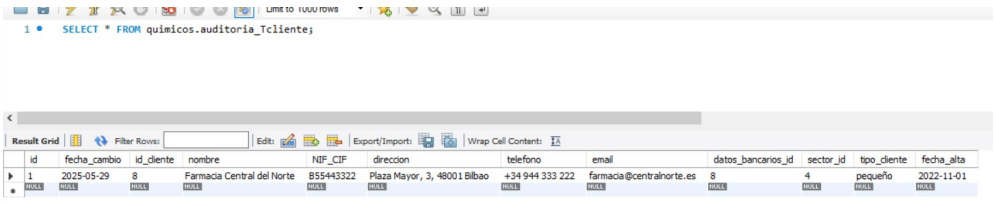
Action Output

#	Time	Action	Message	Duration / Fet
1	18:30:40	insert into lineas_ped_com (albaran_com_id, product...	Error Code: 1644. Supera el stock max permitido	0.000 sec

Prueba Trigger G

```
update cliente set nombre = 'Farmacia Central' where id = 8;
```

Al realizar algún cambio en la tabla clientes, se guardan los datos anteriores en la tabla de auditoria.



```
1 •  SELECT * FROM quimicos.auditoria_Tcliente;
```

Result Grid

id	fecha_cambio	id_cliente	nombre	NIF_CIF	direccion	telefono	email	datos_bancarios_id	sector_id	tipo_cliente	fecha_alta
1	2025-05-29	8	Farmacia Central del Norte	B55443322	Plaza Mayor, 3, 48001 Bilbao	+34 944 333 222	farmacia@centralnorte.es	8	4	pequeño	2022-11-01

12. Conclusiones y Mejoras Futuras

El principal reto enfrentado fue la cantidad de tablas y atributos que se desplegaban con respecto al tema escogido. Por ello, las mejoras se hacen más evidentes al no poder abarcar todo el problema en el tiempo destinado para ello.

La primera mejora es la creación de una tabla facturas_compras para visualizar los albaranes facturados. A continuación se podría entrar en detalle con los documentos de certificación de los productos así como las leyes establecidas para su manipulación. Otro aspecto a ampliar es el del trasporte, incluyendo empresas de transporte habituales, rutas de entrega, horarios de recogida…

Con respecto a los triggers para que la base de datos fuese completamente operativa se necesitan bastantes más, para la actualización de campos automáticamente y auditorias de compras y ventas como mínimo.

En este trabajo se pudo visualizar la necesidad de una estructura inicial antes de empezar el desarrollo y la ventaja de un diseño preparado para su ampliación ya que desde la idea inicial a la final existieron muchos cambios.

13. Enlace al Repositorio en GitHub

https://github.com/uxiarddmg/ProyectoBBDD.git

- README.md
- **Imagenes**: ModeloConceptual.jpg, ModeloRelacional.jpg, ModeloRelacionalFinal.jpg
- **Scripts**: ScriptCreacionBDquimicos.sql, ScriptDatosBDquimicos.sql, ScriptProcedimientos_Funciones_Triggers.sql, ConsultasSQL.sql
- **PDFs**: Conversaciones con el cliente.pdf, Memoria Proyecto BBDD UxíaRD.pdf