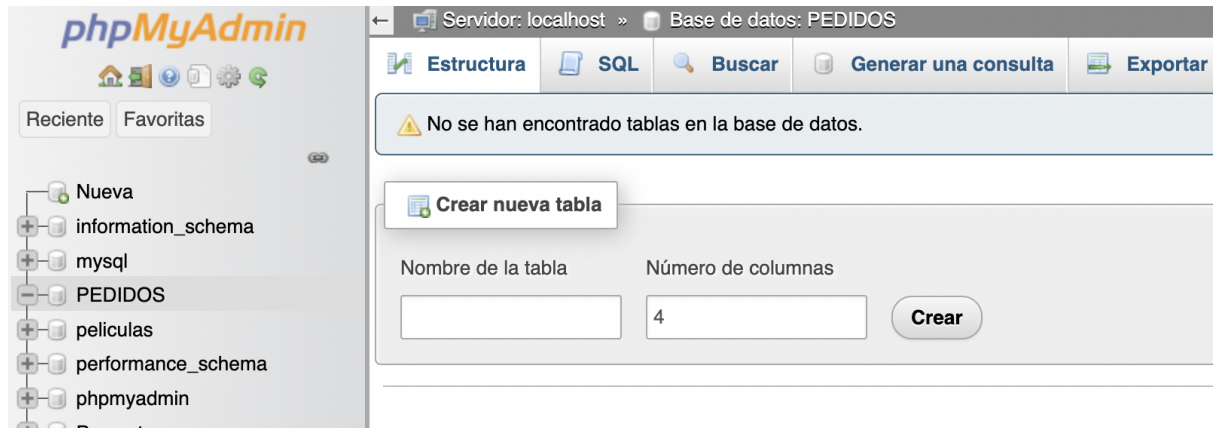


Práctico Base de Datos II

Pedro Fernandez Marquez
2009636
Ing Sistemas.

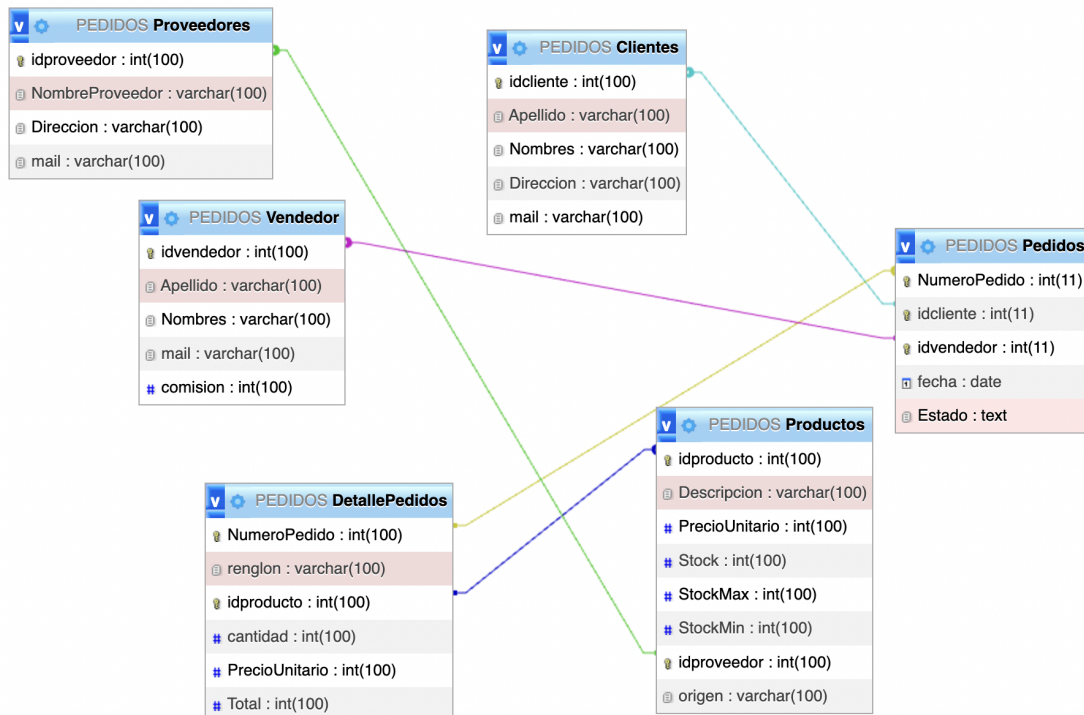
Consigna 1

1)



Se crea la base de datos PEDIDOS.

2)



Se crean las tablas de la base de datos PEDIDOS con sus respectivas relaciones y teniendo en cuenta las restricciones.

3)

Se poblan las tablas de la base de datos PEDIDOS ingresando un conjunto de datos.

```
INSERT INTO `Clientes` (`idcliente`, `Apellido`, `Nombres`, `Direccion`, `mail`) VALUES ('4', 'Verstappen', 'Max', 'Salsipuedes 456', 'vm@gmail.com'), ('5', 'Alonso', 'Fernando', 'Madrid 12', 'fa@gmail.com');
```

```
INSERT INTO `Proveedores` (`idproveedor`, `NombreProveedor`, `Direccion`, `mail`) VALUES ('1', 'Samsung', 'Corea 88', 'samsung@gmail.com'), ('2', 'Apple', 'California 890', 'apple@gmail.com'), ('3', 'Xiaomi', 'China 56', 'xiaomi@gmail.com');
```

```
INSERT INTO `Vendedor` (`idvendedor`, `Apellido`, `Nombres`, `mail`, `comision`) VALUES ('1', 'Hamilton', 'Lewis', 'lh@gmail.com', '10'), ('2', 'Vettel', 'Sebastian', 'sb@gmail.com', '15'), ('3', 'Perez', 'Checo', 'cp@gmail.com', '20');
```

```
INSERT INTO `Productos` (`idproducto`, `Descripcion`, `PrecioUnitario`, `Stock`, `StockMax`, `StockMin`, `idproveedor`, `origen`) VALUES ('1', 'Celular', '200', '321', '400', '10', '2', 'Importado'), ('2', 'Computadora', '500', '428', '500', '10', '1', 'Nacional'), ('3', 'Monitor', '271', '100', '200', '10', '3', 'Importado'), ('4', 'Gabinete', '100', '100', '239', '10', '2', 'Nacional'), ('5', 'Mouse', '20', '200', '390', '10', '1', 'Importado'), ('6', 'Teclado', '20', '219', '300', '10', '3', 'Nacional'), ('7', 'Pendrive', '10', '200', '300', '10', '2', 'Importado'), ('8', 'Proyector', '100', '398', '400', '10', '1', 'Importado'), ('9', 'Parlante', '35', '354', '400', '10', '3', 'Nacional'), ('10', 'Auriculares', '20', '488', '500', '10', '2', 'Importado');
```

```
INSERT INTO `Pedidos` (`NumeroPedido`, `idcliente`, `idvendedor`, `fecha`, `Estado`) VALUES ('1', '5', '1', '2022-09-17', 'Listo'), ('2', '1', '3', '2022-09-18', 'Anulado'), ('3', '3', '2', '2022-09-19', 'Listo'), ('4', '2', '2', '2022-09-20', 'Espera'), ('5', '4', '3', '2022-10-17', 'Espera'), ('6', '5', '2', '2022-10-11', 'Listo'), ('7', '1', '1', '2022-10-12', 'Listo'), ('8', '3', '1', '2022-10-13', 'Anulado'), ('9', '2', '3', '2022-11-17', 'Espera'), ('10', '2', '2', '2022-11-22', 'Listo');
```

```
INSERT INTO `DetallePedidos` (`NumeroPedido`, ` renglon`, `idproducto`, `cantidad`, `PrecioUnitario`, `Total`) VALUES ('2', '1', '10', '2', '20', '40'), ('8', '2', '1', '3', '200', '600'), ('4', '3', '2', '4', '500', '2000'), ('5', '3', '4', '5', '100', '500'), ('9', '2', '3', '6', '271', '1626'), ('1', '1', '5', '7', '20', '140'), ('3', '1', '9', '8', '35', '280'), ('6', '2', '7', '9', '10', '90'), ('7', '3', '8', '1', '100', '100'), ('10', '3', '6', '2', '20', '40');
```

4)

Se crean diversas vistas en la base de datos PEDIDOS:

```
CREATE VIEW clientesxfecha AS
SELECT Clientes.*
FROM Clientes, Pedidos
WHERE Clientes.idcliente = Pedidos.idcliente
AND Pedidos.fecha between '2022/09/17' and '2022/10/17'
GROUP BY Clientes.idcliente;
```

Detalle de Clientes que realizaron pedidos entre fechas (Apellido, Nombres, idcliente, correo electrónico)

```
CREATE VIEW vendedoresxpedidos AS
SELECT Vendedor.Apellido, Vendedor.Nombres, Vendedor.mail, Pedidos.idcliente,
Pedidos.idvendedor, COUNT(Pedidos.idvendedor) AS Pedidos_realizados
FROM Vendedor, Pedidos
WHERE Vendedor.idvendedor = Pedidos.idvendedor
GROUP BY Vendedor.idvendedor;
```

[Detalle de Vendedores con la cantidad de pedidos realizados \(Apellido, Nombres, idcliente, correo electrónico, CantidadPedidos\).](#)

```
CREATE VIEW pedidosxtotal AS
SELECT DetallePedidos.NumeroPedido, pedidos.fecha, SUM(DetallePedidos.Total) as total
FROM pedidos, DetallePedidos
WHERE Pedidos.NumeroPedido = DetallePedidos.NumeroPedido
GROUP BY DetallePedidos.NumeroPedido
ORDER BY total;
```

[Detalle de pedidos con un total mayor a un determinado valor umbral \(NumeroPedido, Fecha, TotalPedido\).](#)

```
CREATE VIEW productosxfecha AS
SELECT Productos.Descripcion, SUM(DetallePedidos.cantidad) AS CantidadTotal
FROM Productos, DetallePedidos, Pedidos
WHERE DetallePedidos.idproducto = Productos.idproducto
AND Pedidos.NumeroPedido = DetallePedidos.NumeroPedido
AND Pedidos.fecha between '2022/09/17' and '2022/10/17'
GROUP BY Productos.idproducto;
```

[Lista de productos vendidos entre fechas. \(Descripción, CantidadTotal\)](#)
[CantidadTotal se calcula sumando todas las cantidades vendidas del producto.](#)

```
CREATE VIEW origen AS
SELECT Productos.origen, SUM(DetallePedidos.cantidad) AS Cantidad
FROM Productos, DetallePedidos
WHERE DetallePedidos.idproducto = Productos.idproducto
GROUP BY Productos.origen;
```

[Cantidad Total vendida por Origen de producto, nacional o importado.](#)

```
CREATE VIEW masventas AS
SELECT Productos.idproveedor, Proveedores.NombreProveedor,
SUM(DetallePedidos.cantidad) AS Cantidad
FROM Productos, DetallePedidos, Proveedores
WHERE DetallePedidos.idproducto = Productos.idproducto AND Productos.idproveedor =
Proveedores.idproveedor
GROUP BY Productos.idproveedor
ORDER BY Cantidad DESC
LIMIT 1;
```

[Cuál es el proveedor que realizó más ventas.](#)

```
CREATE VIEW menosde2 AS
SELECT Clientes.*
FROM Clientes, Pedidos
WHERE Pedidos.idcliente = Clientes.idcliente
GROUP BY Clientes.idcliente
HAVING COUNT(Pedidos.idcliente) < 2
```

[Detalle de CLientes que realizaron menos de dos pedidos. \(Apellido, Nombres, mail\).](#)

```
CREATE VIEW ningunpedido AS
SELECT *
FROM clientes
WHERE idcliente NOT IN (
    SELECT idcliente FROM pedidos);
```

[Detalle de Clientes registrados que nunca realizaron un pedido. \(Apellido, Nombres, mail\).](#)

Consigna 2

```
1)
DELIMITER //
CREATE PROCEDURE pa_registro_pedido
(IN _idcliente INT, IN _idvendedor INT, IN _Estado VARCHAR(100), IN _renglon INT, IN
_idproducto INT, IN _cantidad INT)
if (_idcliente in (SELECT idcliente FROM Clientes)) then
BEGIN
    insert into Pedidos(NumeroPedido, idcliente, idvendedor, fecha, Estado) values (NULL,
_idcliente, _idvendedor, CURDATE(), _Estado);
    insert into DetallePedidos(NumeroPedido, renglon, idproducto, cantidad, PrecioUnitario,
Total) values (NULL, _renglon, _idproducto, _cantidad, (SELECT Productos.PrecioUnitario
FROM Productos WHERE Productos.idproducto = _idproducto), (SELECT
Productos.PrecioUnitario*_cantidad FROM Productos WHERE Productos.idproducto =
_idproducto));
UPDATE `Productos` SET `Stock` = `Stock` - _cantidad WHERE `Productos`.`idproducto` =
_idproducto;
END;
END IF;
//
DELIMITER ;
```

[Procedimiento para registrar un pedido \(deberia ser una transacción\), se tienen en cuenta las actualizaciones de Stock y las diversas restricciones/reglas.](#)

```
2)
DELIMITER //
CREATE PROCEDURE pa_anular_pedido
(IN _NumeroPedido INT)
BEGIN
UPDATE `Productos` SET `Stock` = `Stock` + (SELECT DetallePedidos.cantidad
```

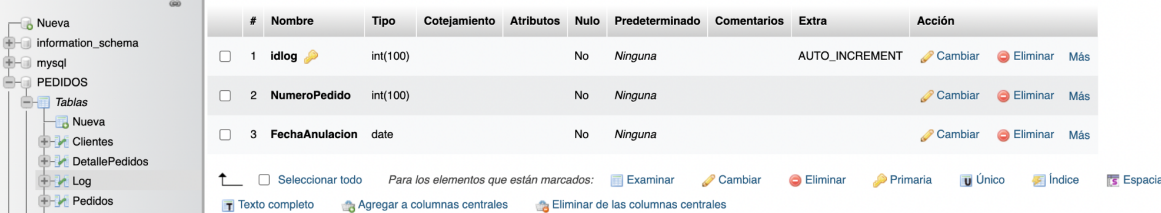
```

FROM DetallePedidos
WHERE DetallePedidos.NumeroPedido = _NumeroPedido) WHERE
`Productos`.`idproducto` = (SELECT DetallePedidos.idproducto
FROM DetallePedidos
WHERE DetallePedidos.NumeroPedido = _NumeroPedido);
UPDATE `Pedidos` SET `Estado` = 'Anulado' WHERE `Pedidos`.`NumeroPedido` =
_NumeroPedido;
//
END
DELIMITER ;

```

Procedimiento almacenado que permite anular un pedido confirmado. El proceso de anulación actualiza los stocks de los artículos del pedido.

3)



#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
1	idlog	int(100)			No	Ninguna		AUTO_INCREMENT	Cambiar Eliminar Más
2	NumeroPedido	int(100)			No	Ninguna			Cambiar Eliminar Más
3	FechaAnulacion	date			No	Ninguna			Cambiar Eliminar Más

☐ Seleccionar todo Para los elementos que están marcados: Examinar Cambiar Eliminar Primaria Único Índice Espacial
 Texto completo Agregar a columnas centrales Eliminar de las columnas centrales

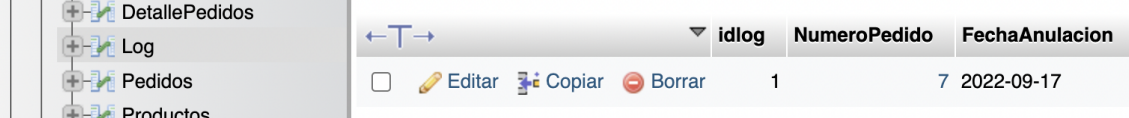
Creación de tabla Log! Contiene idlog, numeroPedido, FechaAnulacion.

4)

```

CREATE TRIGGER tr_anular_pedido
AFTER UPDATE ON Pedidos
FOR EACH ROW
BEGIN
insert into Log(NumeroPedido, FechaAnulacion) values (NEW.NumeroPedido, CURDATE());
END;

```



	idlog	NumeroPedido	FechaAnulacion
<input type="checkbox"/> Editar Copiar Borrar	1	7	2022-09-17

Ver triggers → SHOW TRIGGERS IN PEDIDOS

Trigger que se ejecuta, al momento de anularse un pedido, y registra en la tabla Log, el número de pedido anulado y la fecha de anulacion (Current Date).

5)

```

DELIMITER //
CREATE PROCEDURE pa_actualizar_precio
(IN _origen VARCHAR(100), IN _porcentaje INT)
BEGIN
UPDATE `Productos` SET `PrecioUnitario` = `PrecioUnitario` + (`PrecioUnitario` * 0.23)
WHERE `Productos`.`origen` = 'Importado';
END

```

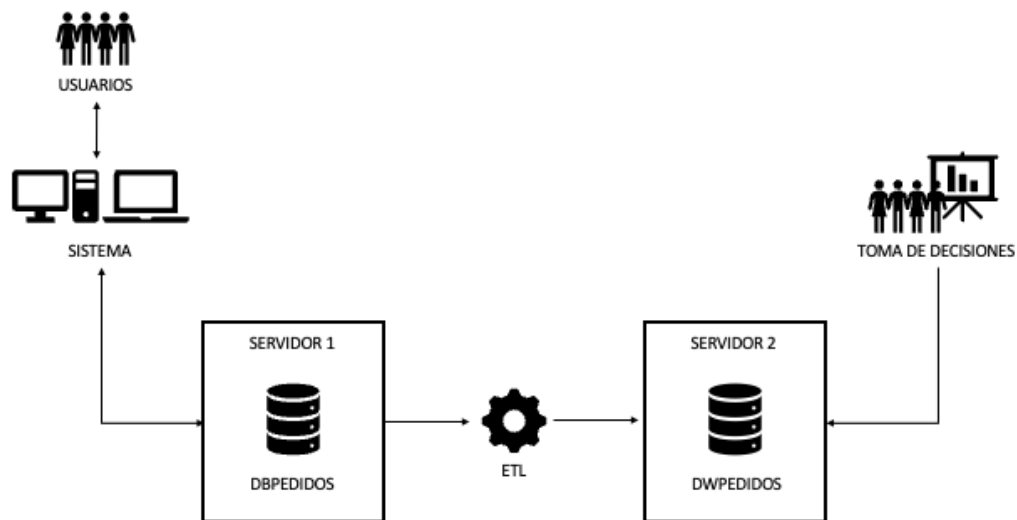
//
DELIMITER ;

Llamado → `call `pa_actualizar_precio`('Importado', 0.15);`

Procedimiento almacenado que permite actualizar el precio de los artículos de un determinado origen (nacional o importado) en un determinado porcentaje.

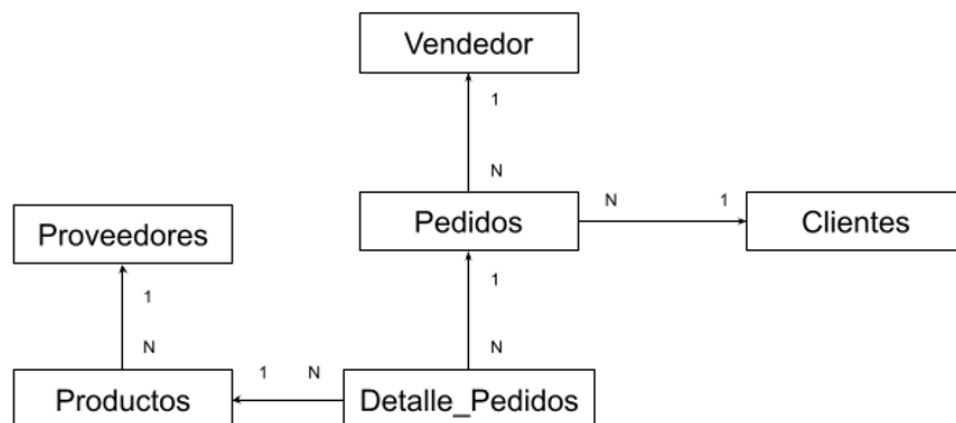
Consigna 3

1)



2)

A la base de datos “pedidos” se pueda acceder por computadoras con capacidad para hacerlo, en esta se encuentra el sistema gestor de base de datos que posee tablas que contienen información relacionadas entre si por claves primarias y otros datos



Estas tablas que almacena información permiten que usuarios con cierto acceso puedan acceder a las diversas vistas mencionadas anteriormente, y otros usuarios pueden usar los procedimientos almacenados para realizar acciones en la base de datos como por ejemplo realizar un pedido, anular pedidos, actualizar precios según el origen de los productos, etc.

También hay triggers que se disparan al momento de ejecutar alguna acción como es el caso de al momento de anular un pedido, registrar en la tabla Log, el número de pedido anulado y la fecha de anulación.

El sistema actúa directamente con la base de datos pero a la vez hay una réplica de la misma que funciona como un Data Warehouse en donde se almacena la misma información para mantener la misma ordenada y por largos periodos mediante procesos ETL (Extract, transform and load) la cual los usuarios de un nivel gerencial pueden acceder para la toma de decisiones.

3) El motor de base de Datos que se aplica es MySQL y los servidores de datos utilizan Linux/Unix.

Consigna 4

1) CREATE DATABASE DWPedidos;

Creación de base de datos DWPedidos.

2)

Nueva

DWPedidos

Nueva

DIMFechas

information_schema

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/>	1	id_Fec	int(100)			No	Ninguna		AUTO_INCREMENT	<a>Cambiar <a>Eliminar <a>Más
<input type="checkbox"/>	2	fecha	date			No	Ninguna			<a>Cambiar <a>Eliminar <a>Más

Nueva

DWPedidos

Nueva

DIMFechas

DIMProductos

FACTPedidos

information_schema

mysql

PEDIDOS

películas

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/>	1	id_pro	int(100)			No	Ninguna		AUTO_INCREMENT	<a>Cambiar <a>Eliminar <a>Más
<input type="checkbox"/>	2	idproducto	int(100)			No	Ninguna			<a>Cambiar <a>Eliminar <a>Más
<input type="checkbox"/>	3	descripcion	varchar(100)	utf8mb4_general_ci		No	Ninguna			<a>Cambiar <a>Eliminar <a>Más
<input type="checkbox"/>	4	nombreproveedor	varchar(100)	utf8mb4_general_ci		No	Ninguna			<a>Cambiar <a>Eliminar <a>Más

Nueva

DWPedidos

Nueva

DIMFechas

DIMProductos

FACTPedidos

information_schema

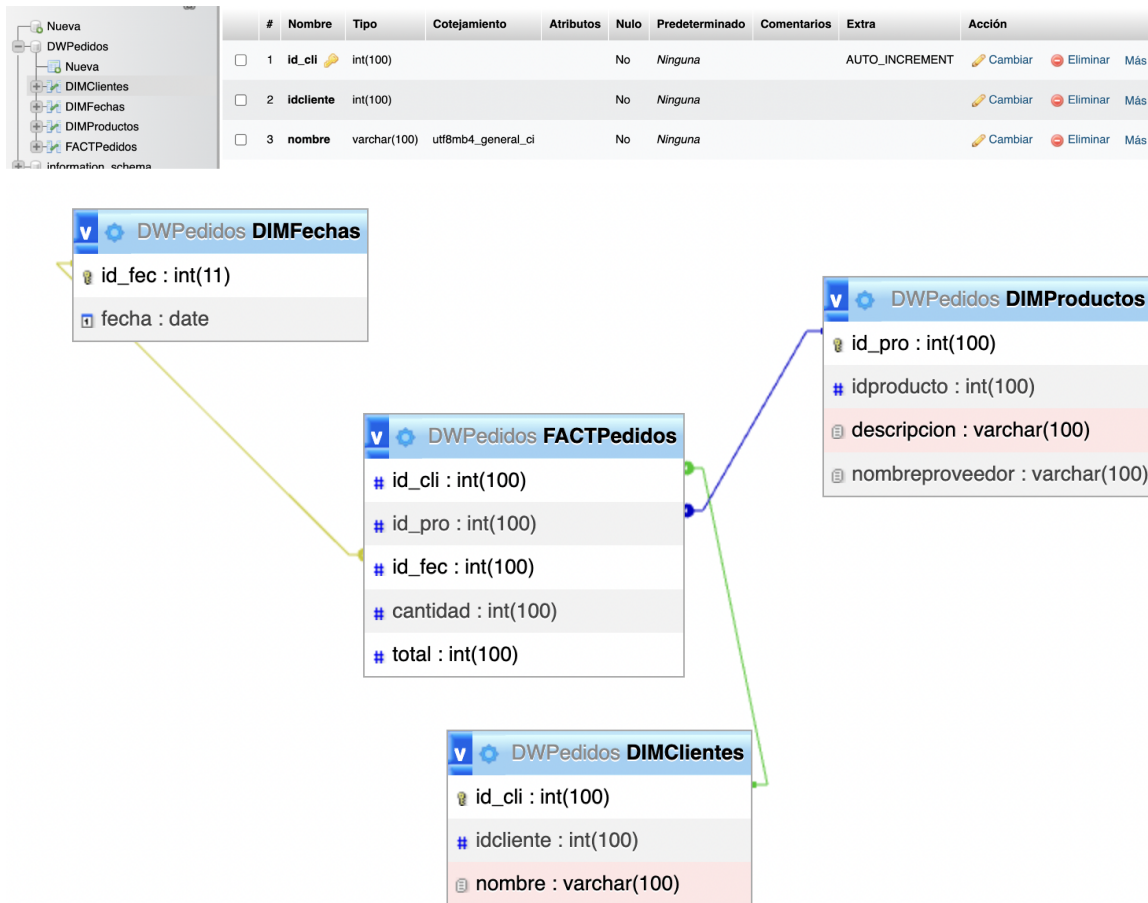
mysql

PEDIDOS

películas

performance_schema

	#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado	Comentarios	Extra	Acción
<input type="checkbox"/>	1	id_cli	int(100)			No	Ninguna			<a>Cambiar <a>Eliminar <a>Más
<input type="checkbox"/>	2	id_pro	int(100)			No	Ninguna			<a>Cambiar <a>Eliminar <a>Más
<input type="checkbox"/>	3	id_fec	int(100)			No	Ninguna			<a>Cambiar <a>Eliminar <a>Más
<input type="checkbox"/>	4	cantidad	int(100)			No	Ninguna			<a>Cambiar <a>Eliminar <a>Más
<input type="checkbox"/>	5	total	int(100)			No	Ninguna			<a>Cambiar <a>Eliminar <a>Más



Se crean las tablas de la base de datos DWPedidos con sus respectivas relaciones y teniendo en cuenta las instrucciones.

3)

```
DELIMITER //
```

```
CREATE PROCEDURE pa_replica_base_pedidos()
```

```
BEGIN
```

```
DELETE FROM DWPedidos.DIMProductos;
```

```
ALTER TABLE DWPedidos.DIMProductos AUTO_INCREMENT = 1;
```

```
DELETE FROM DWPedidos.DIMFechas;
```

```
ALTER TABLE DWPedidos.DIMFechas AUTO_INCREMENT = 1;
```

```
DELETE FROM DWPedidos.DIMClientes;
```

```
ALTER TABLE DWPedidos.DIMClientes AUTO_INCREMENT = 1;
```

```
DELETE FROM DWPedidos.FACTPedidos;
```

```
ALTER TABLE DWPedidos.FACTPedidos AUTO_INCREMENT = 1;
```

```
insert into DWPedidos.DIMProductos(idproducto, descripcion, nombreproveedor) SELECT
idproducto as idproducto, Descripcion as descripcion, idproveedor as nombreproveedor
FROM PEDIDOS.Productos;
```

```
insert into DWPedidos.DIMFechas(fecha) SELECT fecha as fecha FROM
PEDIDOS.Pedidos;
```

```
insert into DWPedidos.DIMClientes(idcliente, nombre) SELECT idcliente as idcliente,
Nombres as nombre FROM PEDIDOS.Clientes;
```



```
insert into DWPedidos.FACTPedidos(id_cli, id_pro, cantidad, total) SELECT idcliente as  
id_cli, idproducto as id_pro, cantidad as cantidad, Total as total  
FROM PEDIDOS.DetallePedidos  
INNER JOIN PEDIDOS.Pedidos on DetallePedidos.NumeroPedido = Pedidos.idcliente;  
END  
//  
DELIMITER ;
```

Se cargan las tablas de la base de datos DWPedidos con los datos de la base de datos de PEDIDOS, mediante procedimiento almacenado en SQL la implementación no es la adecuada ya que borra todos los datos del DWPedidos y introduce todos los valores de nuevo por lo que siempre va a tener los mismos datos que la base de datos PEDIDOS por lo que no cumpliría a la perfección la función de un DataWarehouse y la transformación de datos.