

**RETO FINAL BASE DE DATOS**

**SOFKA U**

**RETO C**

**(TIENDA DON PEPE)**

**JUAN DIEGO SALAZAR GIRALDO**

**1010105080**

**MEDELLIN, ANTIOQUIA**

**17 DE FEBRERO DEL 2023**

## **DESARROLLO DEL TRABAJO**

### **DISEÑO DEL MODELO ENTIDAD RELACION (MER):**

Para iniciar, es necesario identificar las entidades de acuerdo a los requerimientos en el enunciado: las entidades con sus atributos correspondientes para desarrollar el modelo son las siguientes:

Cliente: ID (clave primaria), cédula, nombre, dirección, teléfono, correo electrónico, contraseña.

Producto: ID (clave primaria), nombre, precio, marca, origen, dimensiones (volumen y peso), foto, unidades disponibles (como atributo derivado).

Categoría: ID (clave primaria), nombre, condiciones de almacenamiento (frio, seco y congelado), observaciones.

Pedido: ID del pedido (clave primaria), fecha del pedido, dirección de entrega, importe total del pedido, número de tarjeta, fecha de caducidad.

Zona: Código postal (clave primaria).

Domiciliario: ID (clave primaria), nombre, número de matrícula de la furgoneta.

Proveedor: ID (clave primaria), nombre, teléfono.

Teniendo en cuenta lo anterior, se procede a crear las entidades y sus atributos para iniciar el diseño del MER:

Posteriormente, se iniciará a relacionar las entidades como las veremos a continuación:

Cliente realiza un Pedido donde la relación es 1:N, debido a que un cliente puede realizar muchos pedidos, pero un pedido solo lo realiza un cliente.

Pedido incluye un Producto donde la relación es N:M, debido a que un pedido puede incluir muchos productos, y un producto puede ser incluido en muchos pedidos.

Producto pertenece a una Categoría donde la relación es N:1, debido a que un producto solo pertenece a una categoría, pero una categoría puede tener muchos productos.

Proveedor provee un Producto donde la relación es 1: N, debido a que un proveedor puede proveer muchos productos, pero un producto solo puede ser proveído por un proveedor.

Cliente pertenece a una Zona donde la relación es N:1, debido a que un cliente solo puede pertenecer a una zona, pero en una zona puede haber muchos clientes.

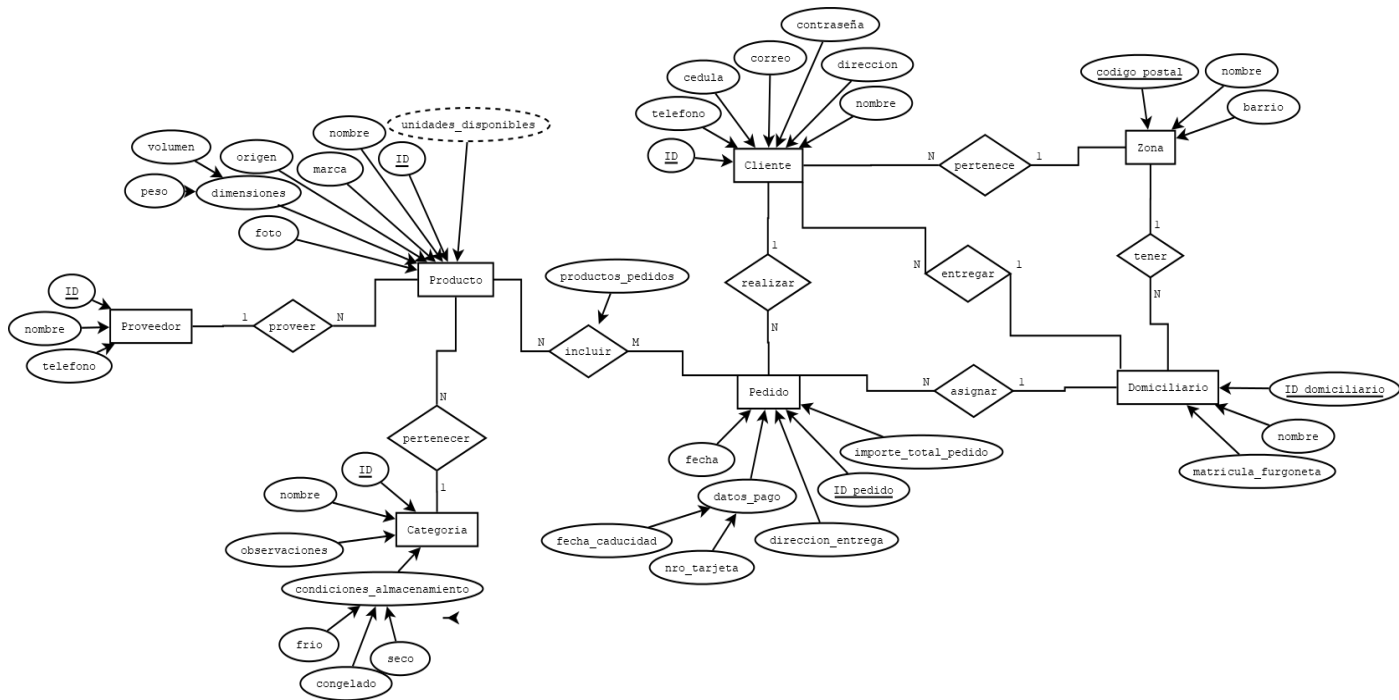
Zona tiene un Domiciliario donde la relación es 1:N, debido a que una zona puede tener muchos domiciliarios, pero en este caso, un domiciliario solo puede estar en una zona.

Domiciliario entrega a un cliente donde la relación es 1:N, debido a que un domiciliario le puede entregar a muchos clientes, pero a un cliente solo le entrega un domiciliario.

Pedido asigna un Domiciliario donde la relación es N:1, debido a que un pedido solo puede ser asignado a un domiciliario, pero un domiciliario se le pueden asignar muchos pedidos.

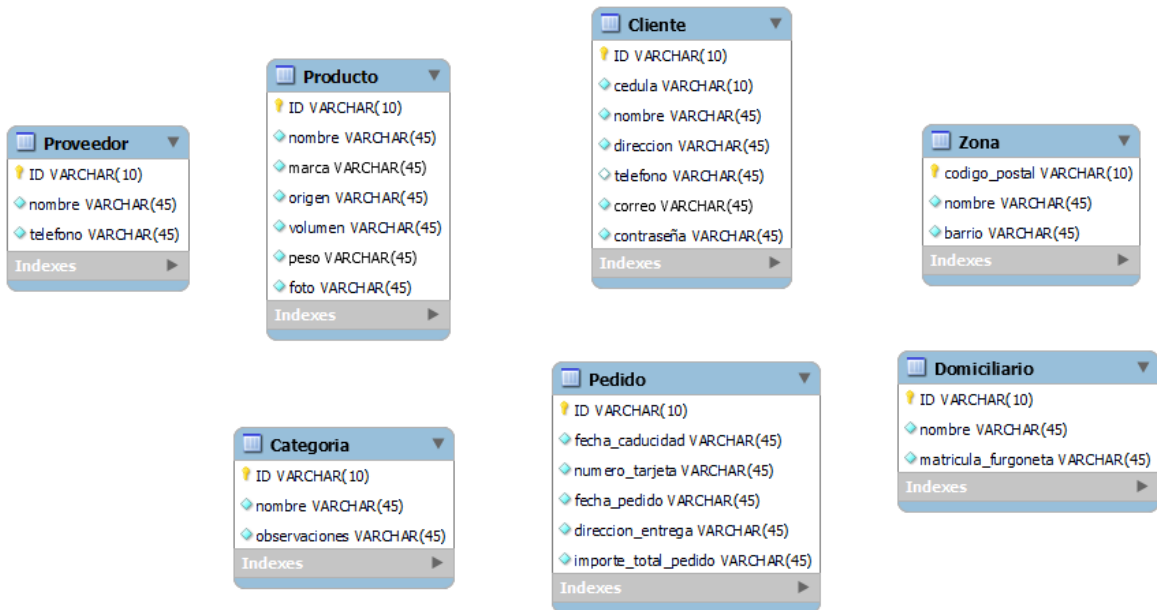
Adicional a lo anterior, en la relación entre Producto y Pedido, se agregó un atributo, el cual corresponde a productos pedidos.

Y luego de implementar todo lo anterior, el diagrama entidad relación queda como lo muestra la siguiente imagen:



Luego de realizar el modelo entidad relación, se realiza el modelo relacional en MySQL WorkBench.

Para iniciar la construcción del modelo, se inicia creando las tablas de cada entidad, como lo vemos en la siguiente imagen:

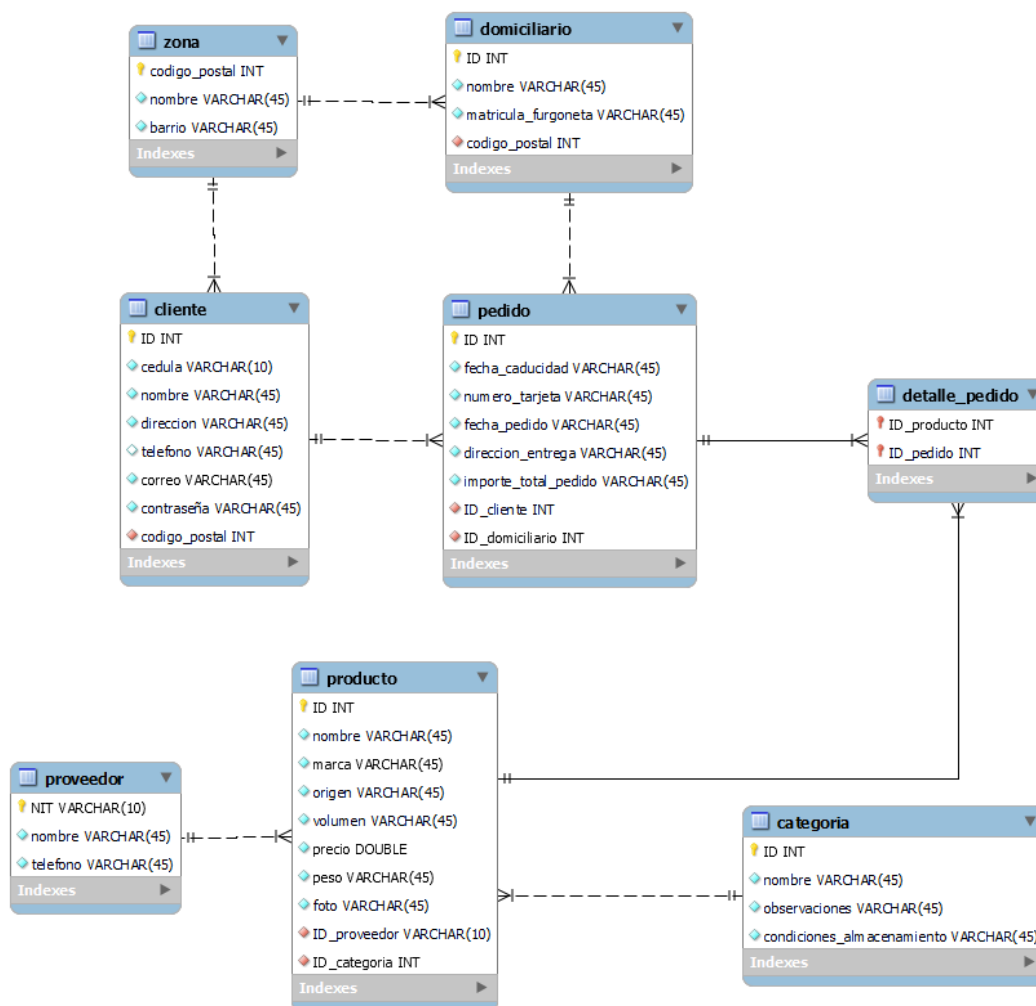


Teniendo en cuenta lo anterior, se procede por medio de las llaves foráneas a relacionar cada tabla de acuerdo al tipo de relación que tengamos en cada caso.

En el caso de la relación entre Pedido y Producto, que es N:M, vamos a crear una nueva tabla, la cual va tener en ella las claves primarias de las dos entidades, pero como llaves foráneas, adicional a el atributo que tenía la misma.

El resto de relaciones, se tratan de 1: N que, en estos casos, en la entidad que le corresponde la N, se llevará como llave foránea la llave primaria de la entidad donde la cardinalidad es 1.

Implementando lo anterior, el modelo relacional queda de la siguiente manera:



Luego de terminar con el diseño del modelo relacional, se verificará si el modelo está normalizado, para esto se tendrá en cuenta lo siguiente:

Es necesario evaluar si cumple con las reglas de normalización, que se dividen en diferentes formas normales.

**A continuación, se describen brevemente las tres formas normales que cumple el diagrama.**

Primera Forma Normal (1NF): El modelo tiene los atributos de cada tabla, las cuales contienen un solo valor. Es decir, no tiene valores múltiples o repetidos en una misma celda, por lo que cumple la primera forma normal.

Segunda Forma Normal (2NF): El modelo está en 1NF y cada atributo no clave depende completamente de la clave primaria, por lo que cumple la segunda forma normal.

Tercera Forma Normal (3NF): El modelo está en 2NF y no hay dependencias transitivas entre los atributos no clave.

Para identificar lo anterior, se tuvo en cuenta que:

Se identificó la clave primaria de cada tabla.

Se verificó si cada atributo contenía un solo valor.

Se verificó si cada atributo no clave dependía completamente de la clave primaria.

Se verificó si no había dependencias transitivas entre los atributos no clave.

Para continuar, se crea la base de datos con sentencias SQL, la cual nos dará como resultado las tablas relacionada con las llaves foráneas.

A continuación, visualizamos los comandos utilizados:

```
1 • CREATE SCHEMA IF NOT EXISTS `DonPepe` DEFAULT CHARACTER SET utf8 ;
2 • USE `DonPepe` ;
3
4 /*TABLA ZONA*/
5 • CREATE TABLE IF NOT EXISTS `DonPepe`.`Zona` (
6     `codigo_postal` INT auto_increment NOT NULL,
7     `nombre` VARCHAR(45) NOT NULL,
8     `barrio` VARCHAR(45) NOT NULL,
9     PRIMARY KEY (`codigo_postal`))
10
11 /*TABLA CLIENTE*/
12 ✖ CREATE TABLE IF NOT EXISTS `DonPepe`.`Cliente` (
13     `ID` INT auto_increment NOT NULL,
14     `cedula` VARCHAR(10) NOT NULL,
15     `nombre` VARCHAR(45) NOT NULL,
16     `direccion` VARCHAR(45) NOT NULL,
17     `telefono` VARCHAR(45) NULL,
18     `correo` VARCHAR(45) NOT NULL,
19     `contraseña` VARCHAR(45) NOT NULL,
20     `codigo_postal` INT NOT NULL,
21     PRIMARY KEY (`ID`),
22     FOREIGN KEY (`codigo_postal`)
23     REFERENCES `Zona` (`codigo_postal`)
24 );
25
26 /*TABLA DOMICILIARIO*/
27 • CREATE TABLE IF NOT EXISTS `DonPepe`.`Domiciliario` (
28     `ID` INT AUTO_INCREMENT NOT NULL,
29     `nombre` VARCHAR(45) NOT NULL,
30     `matricula_furgoneta` VARCHAR(45) NOT NULL,
31     `codigo_postal` INT NOT NULL,
32     PRIMARY KEY (`ID`),
33     FOREIGN KEY (`codigo_postal`)
34     REFERENCES `DonPepe`.`Zona` (`codigo_postal`)
35 );
36
37
38 /*TABLA PEDIDO*/
39 • CREATE TABLE IF NOT EXISTS `DonPepe`.`Pedido` (
40     `ID` INT AUTO_INCREMENT NOT NULL,
41     `fecha_caducidad` VARCHAR(45) NOT NULL,
42     `numero_tarjeta` VARCHAR(45) NOT NULL,
43     `fecha_pedido` VARCHAR(45) NOT NULL,
44     `direccion_entrega` VARCHAR(45) NOT NULL,
45     `importe_total_pedido` VARCHAR(45) NOT NULL,
46     `ID_cliente` INT NOT NULL,
47     `ID_domiciliario` INT NOT NULL,
48     PRIMARY KEY (`ID`),
49     FOREIGN KEY (`ID_cliente`)
50     REFERENCES `DonPepe`.`Cliente` (`ID`),
51     FOREIGN KEY (`ID_domiciliario`)
52     REFERENCES `DonPepe`.`Domiciliario` (`ID`));
```

```

53
54      /*TABLA PROVEEDOR*/
55  ● ○ CREATE TABLE IF NOT EXISTS `DonPepe`.`Proveedor` (
56      `NIT` VARCHAR(10) NOT NULL,
57      `nombre` VARCHAR(45) NOT NULL,
58      `telefono` VARCHAR(45) NOT NULL,
59      PRIMARY KEY (`NIT`));
60
61      /*TABLA CATEGORIA*/
62  ● ○ CREATE TABLE IF NOT EXISTS `DonPepe`.`Categoria` (
63      `ID` INT AUTO_INCREMENT NOT NULL,
64      `nombre` VARCHAR(45) NOT NULL,
65      `observaciones` VARCHAR(45) NOT NULL,
66      `condiciones_almacenamiento` VARCHAR(45) NOT NULL,
67      PRIMARY KEY (`ID`));
68
69      /*TABLA PRODUCTO*/
70  ● ○ CREATE TABLE IF NOT EXISTS `DonPepe`.`Producto` (
71      `ID` INT AUTO_INCREMENT NOT NULL,
72      `nombre` VARCHAR(45) NOT NULL,
73      `marca` VARCHAR(45) NOT NULL,
74      `origen` VARCHAR(45) NOT NULL,
75      `volumen` VARCHAR(45) NOT NULL,
76      `precio` double NOT NULL,
77      `peso` VARCHAR(45) NOT NULL,
78      `foto` VARCHAR(45) NOT NULL,
79      `ID_proveedor` VARCHAR(10) NOT NULL,
80      `ID_categoria` INT NOT NULL,
81      PRIMARY KEY (`ID`),
82      FOREIGN KEY (`ID_proveedor`)
83      REFERENCES `DonPepe`.`Proveedor` (`NIT`),
84      FOREIGN KEY (`ID_categoria`)
85      REFERENCES `DonPepe`.`Categoria` (`ID`)
86      );
87
88      /*TABLA DETALLE_PEDIDO*/
89  ● ○ CREATE TABLE IF NOT EXISTS `DonPepe`.`Detalle_pedido` (
90      `ID_producto` INT NOT NULL,
91      `ID_pedido` INT NOT NULL,
92      PRIMARY KEY (`ID_producto`, `ID_pedido`),
93      FOREIGN KEY (`ID_producto`)
94      REFERENCES `DonPepe`.`Producto` (`ID`),
95      FOREIGN KEY (`ID_pedido`)
96      REFERENCES `DonPepe`.`Pedido` (`ID`)
97      )

```



Después de definir la base de datos con las sentencias SQL, se procede a realizar de a un registro por cada tabla para poder crear las consultas como lo vemos a continuación:

Se realizarán diez consultas, las cual nos servirán para darnos cuenta de información valiosa en el negocio, y así llevar una buena gestión del mismo.

En cada consulta se evidencia la sentencia con su correspondiente resultado.

## CONSULTA 1

```
1 • USE donpepe;
2
3 /*CONSULTA 1
4  VER EL NOMBRE DEL DOMICILIARIO, LA ZONA Y LA FECHA QUE REALIZO EL PEDIDO*/
5
6 • SELECT DOMICILIARIO.NOMBRE, ZONA.NOMBRE AS ZONA, PEDIDO.FECHA_PEDIDO
7 FROM DOMICILIARIO
8 LEFT JOIN ZONA ON DOMICILIARIO.CODIGO_POSTAL = ZONA.CODIGO_POSTAL
9 LEFT JOIN PEDIDO ON DOMICILIARIO.ID = PEDIDO.ID_DOMICILIARIO;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	NOMBRE	ZONA	FECHA_PEDIDO
▶	Pedro	poblado	8/6/23

## CONSULTA 2

```
13 /*CONSULTA 2
14  VER CANTIDAD DE PRODUCTOS HA SUMINISTRADO EL PROVEEDOR*/
15
16 • SELECT PROVE.NOMBRE AS NOMBRE_PROVEEDOR, COUNT(PROD.ID) AS CATIDAD_PRODUCTOS_SUMINISTRADOS
17 FROM PROVEEDOR PROVE JOIN PRODUCTO PROD ON PROVE.NIT = PROD.ID_proveedor
18 GROUP BY PROVE.NOMBRE;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content: [IA](#)

	NOMBRE_PROVEEDOR	CATIDAD_PRODUCTOS_SUMINISTRADOS
▶	colombina	2

### CONSULTA 3

```
22  /* CONSULTA 3
23  VER EL NOMBRE DE CADA PRODUCTO Y LAS VECES QUE HA SIDO PEDIDO*/
24
25  •  SELECT PRODUCTO.NOMBRE AS PRODUCTO, COUNT(*) AS CANTIDAD_PRODUCTOS_PEDIDOS
26      FROM PEDIDO JOIN PRODUCTO ON PEDIDO.ID = PRODUCTO.ID
27      GROUP BY PEDIDO.ID;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
PRODUCTO	CANTIDAD_PRODUCTOS_PEDIDOS			
manzana	1			

### CONSULTA 4

```
28
29  /*CONSULTA 4
30  VER EL NOMBRE DEL PRODUCTO CON SU PRECIO Y A QUE CATEGORIA PERTENECE*/
31
32  •  SELECT PR.NOMBRE AS NOMBRE_PRODUCTO, PR.PRECIO AS PRECIO_PRODUCTO, CTG.NOMBRE AS NOMBRE_CATEGORIA
33      FROM PRODUCTO PR JOIN CATEGORIA CTG ON PR.ID_CATEGORIA = CTG.ID;
```

Result Grid			Filter Rows:	Export:	Wrap Cell Content:
NOMBRE_PRODUCTO	PRECIO_PRODUCTO	NOMBRE_CATEGORIA			
manzana	1	FRUTAS			
POLLO	200	FRUTAS			

(Hasta esta consulta solo se había creado una categoría)

## CONSULTA 5

```
36  /*CONSULTA 5
37  VER EL NOMBRE DE CADA AREA Y LA CANTIDAD DE PEDIDOS QUE HA RECIBIDO*/
38
39  •  SELECT ZONA.NOMBRE AS ZONA, COUNT(*) AS CANTIDAD_PEDIDOS
40     FROM PEDIDO JOIN DOMICILIARIO ON PEDIDO.ID_DOMICILIARIO = DOMICILIARIO.ID
41     JOIN ZONA ON DOMICILIARIO.CODIGO_POSTAL = ZONA.CODIGO_POSTAL
42     GROUP BY ZONA.NOMBRE;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
ZONA	CANTIDAD_PEDIDOS			
poblado	1			

## CONSULTA 6

```
44  /*CONSULTA 6
45  //VER EL CLIENTE Y LA CANTIDAD DE PEDIDOS QUE HA REALIZADO//*/
46
47  •  SELECT CLIENTE.NOMBRE AS CLIENTE, COUNT(*) AS CANTIDAD_PEDIDOS
48     FROM PEDIDO JOIN CLIENTE ON PEDIDO.ID_CLIENTE = CLIENTE.ID
49     GROUP BY CLIENTE.NOMBRE;
```

Result Grid		Filter Rows:	Export:	Wrap Cell Content:
CLIENTE	CANTIDAD_PEDIDOS			
Diego	1			



## CONSULTA 7

```
51  /*CONSULTA 7
52  //VER EL NOMBRE DEL PRODUCTO CON SU PRECIO Y CATEGORIA, Y LA CANTIDAD DE VECES QUE SE HA PEDIDO*/
53
54  •  SELECT PR.NOMBRE AS NOMBRE_PRODUCTO, PR.PRECIO AS PRECIO_PRODUCTO, COUNT(*)
55     AS CATIDAD_PEDIDOS, CTG.NOMBRE AS NOMBRE_CATEGORIA
56     FROM DETALLE_PEDIDO DP JOIN PRODUCTO PR ON PR.ID =DP.ID_PRODUCTO
57     JOIN CATEGORIA CTG ON CTG.ID= PR.ID_CATEGORIA
58     GROUP BY PR.ID,CTG.NOMBRE;
```

Result Grid				Filter Rows:	Export:	Wrap Cell Content:
NOMBRE_PRODUCTO	PRECIO_PRODUCTO	CATIDAD_PEDIDOS	NOMBRE_CATEGORIA			
manzana	1	1	FRUTAS			



## CONSULTA 8

```
60  /*CONSULTA 8
61  //NOMBRE DE REPARTIDOR Y SU AREA, CON LA FECHA QUE REALIZÓ EL PEDIDO//*/
62
63  •  SELECT DOMICILIARIO.NOMBRE, ZONA.NOMBRE AS ZONA, PEDIDO.FECHA_PEDIDO
64      FROM DOMICILIARIO
65      LEFT JOIN ZONA ON DOMICILIARIO.CODIGO_POSTAL = ZONA.CODIGO_POSTAL
66      LEFT JOIN PEDIDO ON DOMICILIARIO.ID = PEDIDO.ID_DOMICILIARIO;
67
```

Result Grid			
Filter Rows: <input type="text"/>			
Export:  Wrap Cell Content: 			
	NOMBRE	ZONA	FECHA_PEDIDO
▶	Pedro	poblado	8/6/23



## CONSULTA 9

```
68  /*CONSULTA 9
69  //VER EL PRODUCTO DE ACUERDO A SU CATEGORIA Y CUAL FUE SU PROVEEDOR*/
70
71  •  SELECT PRODUCTO.NOMBRE AS PRODUCTO, CATEGORIA.NOMBRE AS CATEGORIA,
72      PROVEEDOR.NOMBRE AS PROVEEDOR, PRODUCTO.PRECIO
73      FROM PRODUCTO INNER JOIN CATEGORIA ON PRODUCTO.ID_CATEGORIA = CATEGORIA.ID
74      INNER JOIN PROVEEDOR ON PRODUCTO.ID_PROVEEDOR = PROVEEDOR.NIT;
```

Result Grid				
Filter Rows: <input type="text"/>				
Export:  Wrap Cell Content: 				
	PRODUCTO	CATEGORIA	PROVEEDOR	PRECIO
▶	manzana	FRUTAS	colombina	1
	POLLO	FRUTAS	colombina	200

## CONSULTA 10

```
77  /*CONSULTA 10
78  VER EL ID DEL PEDIDO CON LA FECHA EN LA QUE SE PIDIO Y EL DOMICILIARIO ENCARGADO CON EL NUMERO DE SU FURGONETA,
79  */
80  •  SELECT PEDIDO.ID, PEDIDO.FECHA_PEDIDO,
81      DOMICILIARIO.NOMBRE AS DOMICILIARIO, DOMICILIARIO.MATRICULA_FURGONETA
82      FROM PEDIDO INNER JOIN DOMICILIARIO ON PEDIDO.ID_DOMICILIARIO = DOMICILIARIO.ID;
```

Result Grid			
Filter Rows: <input type="text"/>			
Export:  Wrap Cell Content: 			
	ID	FECHA_PEDIDO	DOMICILIARIO
▶	1	8/6/23	Pedro
			MATRICULA_FURGONETA
			poi876

Teniendo en cuenta las consultas anteriores, podemos crear 4 vistas, de las cuales se diseñaron de acuerdo al enunciado tratado en el taller.

## VISTA 1

```
1  USE DONPEPE;
2
3  /*VISTA 1*/
4
5  • CREATE VIEW PRODUCTOS_DISPONIBLES AS
6  SELECT PRODUCTO.ID, PRODUCTO.NOMBRE, PRODUCTO.MARCA, PRODUCTO.ORIGEN,
7  PRODUCTO.VOLUMEN, PRODUCTO.PRECIO, PRODUCTO.PESO, PRODUCTO.FOTO, CATEGORIA.NOMBRE AS CATEGORIA
8  FROM PRODUCTO
9  INNER JOIN CATEGORIA ON PRODUCTO.ID_CATEGORIA = CATEGORIA.ID;
10
11 • SELECT*FROM PRODUCTOS_DISPONIBLES;
```

ID	NOMBRE	MARCA	ORIGEN	VOLUMEN	PRECIO	PESO	FOTO	CATEGORIA
1	manzana	colombina	colombia	2	1	20g	1	FRUTAS
2	POLLO	COLOMBIA	COL	3	200	300G	POLLO	FRUTAS

## VISTA 2

```
13  /*VISTA 2*/
14
15  • CREATE VIEW CATEGORIAS AS
16  SELECT CATEGORIA.NOMBRE, CATEGORIA.CONDICIONES_ALMACENAMIENTO, CATEGORIA.OBSERVACIONES
17  FROM CATEGORIA;
18
19  • SELECT*FROM CATEGORIAS;
```

NOMBRE	CONDICIONES_ALMACENAMIENTO	OBSERVACIONES
FRUTAS	FRIJO	FRESCAS

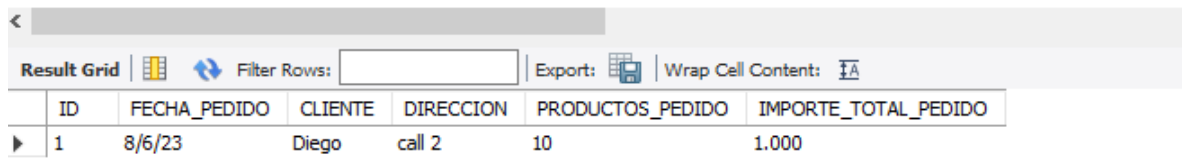
## VISTA 3

```
22  /*VISTA 3*/
23
24  • CREATE VIEW DOMICILIARIO_ZONAS_REPARTO AS
25  SELECT DOMICILIARIO.NOMBRE, ZONA.NOMBRE AS ZONAS
26  FROM DOMICILIARIO
27  INNER JOIN ZONA ON DOMICILIARIO.CODIGO_POSTAL = ZONA.CODIGO_POSTAL;
28
29  • SELECT*FROM DOMICILIARIO_ZONAS_REPARTO;
```

NOMBRE	ZONAS
Pedro	poblado

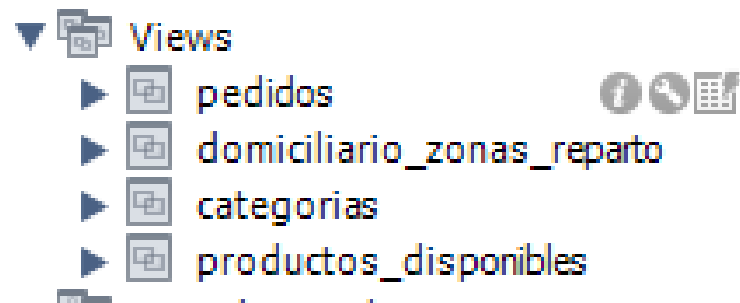
## VISTA 4

```
32      /*VISTA 4*/
33
34  •   CREATE VIEW PEDIDOS AS
35      SELECT PEDIDO.ID, PEDIDO.FECHA_PEDIDO, CLIENTE.NOMBRE AS CLIENTE, CLIENTE.DIRECCION,
36      DETALLE_PEDIDO.PRODUCTOS_PEDIDO, PEDIDO.IMPORTE_TOTAL_PEDIDO
37      FROM CLIENTE
38      JOIN PEDIDO ON PEDIDO.ID_CLIENTE = CLIENTE.ID
39      JOIN DETALLE_PEDIDO ON DETALLE_PEDIDO.ID_PEDIDO = PEDIDO.ID;
40
41  •   SELECT*FROM PEDIDOS;
```



	ID	FECHA_PEDIDO	CLIENTE	DIRECCION	PRODUCTOS_PEDIDO	IMPORTE_TOTAL_PEDIDO
▶	1	8/6/23	Diego	call 2	10	1.000

Como resultado final, en la siguiente imagen se muestra como quedaron agregadas las cuatro vistas:



También se crearon algunos procedimientos, para realizar consultas de una forma más fácil, para tener información valiosa de forma rápida.

## PROCEDIMIENTO 1

En este procedimiento, tendremos la posibilidad de agregar un nuevo cliente cuando se desee.

```
3      /* PROCEDIMIENTO 1 PARA REGISTRAR UN CLIENTE */
4
5      DELIMITER //
6 • CREATE PROCEDURE registrarCliente(
7      IN cedula VARCHAR(45),
8      IN nombre VARCHAR(45),
9      IN direccion VARCHAR(45),
10     IN telefono VARCHAR(45),
11     IN correo VARCHAR(45),
12     IN contraseña VARCHAR(45),
13     IN codigo_postal INT
14 )
15 BEGIN
16     INSERT INTO cliente (cedula, nombre, direccion, telefono, correo, contraseña, codigo_postal)
17     VALUES (cedula, nombre, direccion, telefono, correo, contraseña, codigo_postal);
18 END //
19 DELIMITER ;
20
21 • CALL registrarCliente();
```



Call stored procedure donpepe.registrarCliente



Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

<b>cedula</b>	<input type="text"/>	[IN] VARCHAR(45)
<b>nombre</b>	<input type="text"/>	[IN] VARCHAR(45)
<b>direccion</b>	<input type="text"/>	[IN] VARCHAR(45)
<b>telefono</b>	<input type="text"/>	[IN] VARCHAR(45)
<b>correo</b>	<input type="text"/>	[IN] VARCHAR(45)
<b>contraseña</b>	<input type="text"/>	[IN] VARCHAR(45)
<b>codigo_postal</b>	<input type="text"/>	[IN] INT

Execute


Cancel

## PROCEDIMIENTO 2

```
23 DELIMITER //
```

```
24 • CREATE PROCEDURE agregarProducto(  
25     IN nombre VARCHAR(45),  
26     IN marca VARCHAR(45),  
27     IN origen VARCHAR(45),  
28     IN volumen VARCHAR(45),  
29     IN precio INT,  
30     IN peso VARCHAR(45),  
31     IN foto VARCHAR(45),  
32     IN categoria VARCHAR(45)  
33 )  
34 BEGIN  
35     INSERT INTO productos (nombre, marca, origen, volumen, precio, peso, foto, categoria)  
36     VALUES (nombre, marca, origen, volumen, precio, peso, foto, categoria);  
37 END //
```

```
38 DELIMITER ;  
39  
40 • CALL agregarProducto();
```

 Call stored procedure donpepe.agregarProducto

Enter values for parameters of your procedure and click <Execute> to create an SQL editor and run the call:

nombre	<input type="text"/>	[IN] VARCHAR(45)
marca	<input type="text"/>	[IN] VARCHAR(45)
origen	<input type="text"/>	[IN] VARCHAR(45)
volumen	<input type="text"/>	[IN] VARCHAR(45)
precio	<input type="text"/>	[IN] INT
peso	<input type="text"/>	[IN] VARCHAR(45)
foto	<input type="text"/>	[IN] VARCHAR(45)
categoria	<input type="text"/>	[IN] VARCHAR(45)



### PROCEDIMIENTO 3

```
42      /*PROCEDIMIENTO PARA SABER TODA LA INFORMACION DE LA BASE DE DATOS*/
43
44      DELIMITER //
45 • CREATE PROCEDURE VerTodaLaInformacion()
46 BEGIN
47     SELECT * FROM cliente, domiciliario, zona, pedido, producto, categoria, proveedor;
48 END //
49 DELIMITER ;
50
51 • call VerTodaLaInformacion();
```

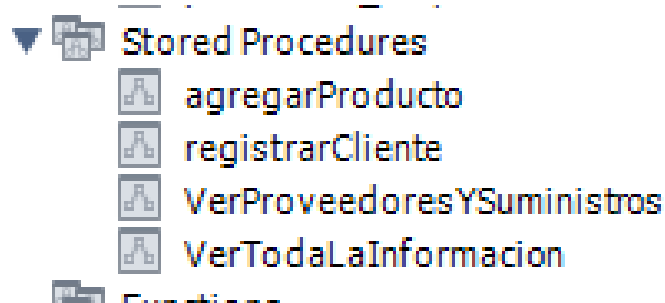
Result Grid   Filter Rows:   Export:   Wrap Cell Content:											
	ID	cedula	nombre	direccion	telefono	correo	contraseña	codigo_postal	ID	nombre	matricula_furgoneta
▶	1	1010105080	Diego	call 2	4443232	diego@gmail.com	1234	1	1	Pedro	poi876
	1	1010105080	Diego	call 2	4443232	diego@gmail.com	1234	1	1	Pedro	poi876

En el resultado, se muestra la información inicial del cliente solamente, debido a que el resto de información está en la tabla hacia la derecha. Además, se muestra dos veces el mismo cliente debido que, hasta hora, este mismo ha realizado dos pedidos.

### PROCEDIMIENTO 4

```
53      /*PROCEDIMIENTO QUE SERVIRA PARA VER LOS PRODUCTOS SUMINISTRADOS POR EL PROVEEDOR*/
54
55      DELIMITER //
56 • CREATE PROCEDURE VerProveedoresYSuministros()
57 BEGIN
58     SELECT pr.nombre AS nombre_proveedor, p.nombre AS nombre_producto,
59           c.nombre AS nombre_categoria, COUNT(*) AS cantidad_productos_suministrados
60     FROM proveedor pr
61     JOIN producto p ON pr.id = p.id_proveedor
62     JOIN categoria c ON p.id_categoria = c.id
63     GROUP BY pr.nombre, p.nombre, c.nombre;
64 END //
65 DELIMITER ;
66
67 • call VerProveedoresYSuministros();
```

Se puede evidenciar en la siguiente imagen, como quedaron registrados los procedimientos en la base de datos.



Para finalizar, es importante crear triggers para verificar o estar notificados de los cambios que se realicen en la base de datos.

Por lo que se crearon 4 triggers o disparadores que actúen cuando se inserte, consulte, elimine o actualice información en la base de datos.

#### TRIGGER 1:

```
3      /*TRIGGER 1*/
4
5      DELIMITER //
6      • CREATE TRIGGER validaProductoInsert
7      BEFORE INSERT
8      ON producto
9      FOR EACH ROW
10     BEGIN
11         IF EXISTS (SELECT * FROM producto WHERE id_producto = NEW.id) THEN
12             SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'El número de identificación ya está en uso.';
13         END IF;
14     END//
15     DELIMITER ;
```

## TRIGGER 2

```
17      /*TRIGGER 2*/
18
19      DELIMITER //
20 •    CREATE TRIGGER eliminaPedidoDelete
21      AFTER DELETE
22      ON cliente
23      FOR EACH ROW
24      BEGIN
25          DELETE FROM pedido WHERE id_cliente = OLD.id;
26      END //
27      DELIMITER ;
```

## TRIGGER 3

```
29      /*TRIGGER 3*/
30
31      DELIMITER //
32 •    CREATE TRIGGER agregaProductosInsert
33      AFTER INSERT
34      ON proveedor
35      FOR EACH ROW
36      BEGIN
37          INSERT INTO producto (ID, nombre, marca, origen, volumen, precio, peso, foto, id_categoria, id_proveedor)
38          SELECT ID, nombre, marca, origen, volumen, precio, peso, foto, id_categoria, NEW.ID
39          FROM producto
40          WHERE id_proveedor IS NULL;
41      END //
42      DELIMITER ;
```

## TRIGGER 4

```
44      /*TRIGGER 4*/
45
46      DELIMITER //
47 •    CREATE TRIGGER actualiza_precios_pedidos
48      AFTER UPDATE
49      ON producto
50      FOR EACH ROW
51      BEGIN
52          UPDATE pedido SET precio_pedido = NEW.precio WHERE id_producto = OLD.id;
53      END //
54      DELIMITER ;
```

Para finalizar, se responde a la pregunta solicitada en el ejercicio:

**¿Está conforme con el resultado obtenido según el contexto o cree que hubiera obtenido un mejor resultado con una base de datos no relacional?**

Teniendo en cuenta toda la información encontrada, al relacionar todas las entidades, se logró el objetivo que fue dar solución al requerimiento del cliente, el cual quería una gestión de su negocio de una forma más óptima, y de la forma que se implementó con la base de datos relacional, fue posible incluso hallar información de valor para realizar un buen desarrollo en las actividades que realice el cliente.