



PRESENTADO POR:

NEVARDO ANTONIO OSPINA Z.

COACH:

JUAN ESTEBAN PINEDA ANGEL

19/02/2023

Barbería (Ejercicio A)

Una barbería desea llevar el control de sus empleados y de sus clientes, así como de los servicios que se prestan.

Se desea almacenar la siguiente información:

- Empleados: ID, cedula, Nombre, Especialidad (Masaje, Corte, Cejas, etc.)
- Clientes: Datos personales (ID, cedula, Nombre, Profesión, Teléfono, correo, edad y Dirección).
- Historial de Servicios prestados por la barbería: Un registro para saber información del servicio prestado por un empleado a un cliente, productos consumidos, duración del procedimiento y fecha.
- Citas: Fecha y Hora en la que se cita al cliente y el barbero que realizará el servicio.
- Productos vendidos por la barbería: REF, Nombre, Cantidad y Precio.
- Proveedor: los productos vendidos deben tener una fuente.
- Registro de Ventas: Si un barbero vende un producto a un cliente, termina obteniendo una “liga” ganancia ocasional.

Reto a Equipo 3

Entidades:

Proveedor.

Producto

Factura

Empleado

Insumo

Historial_servicio

Servicio

Citas

Reserva

Cliente

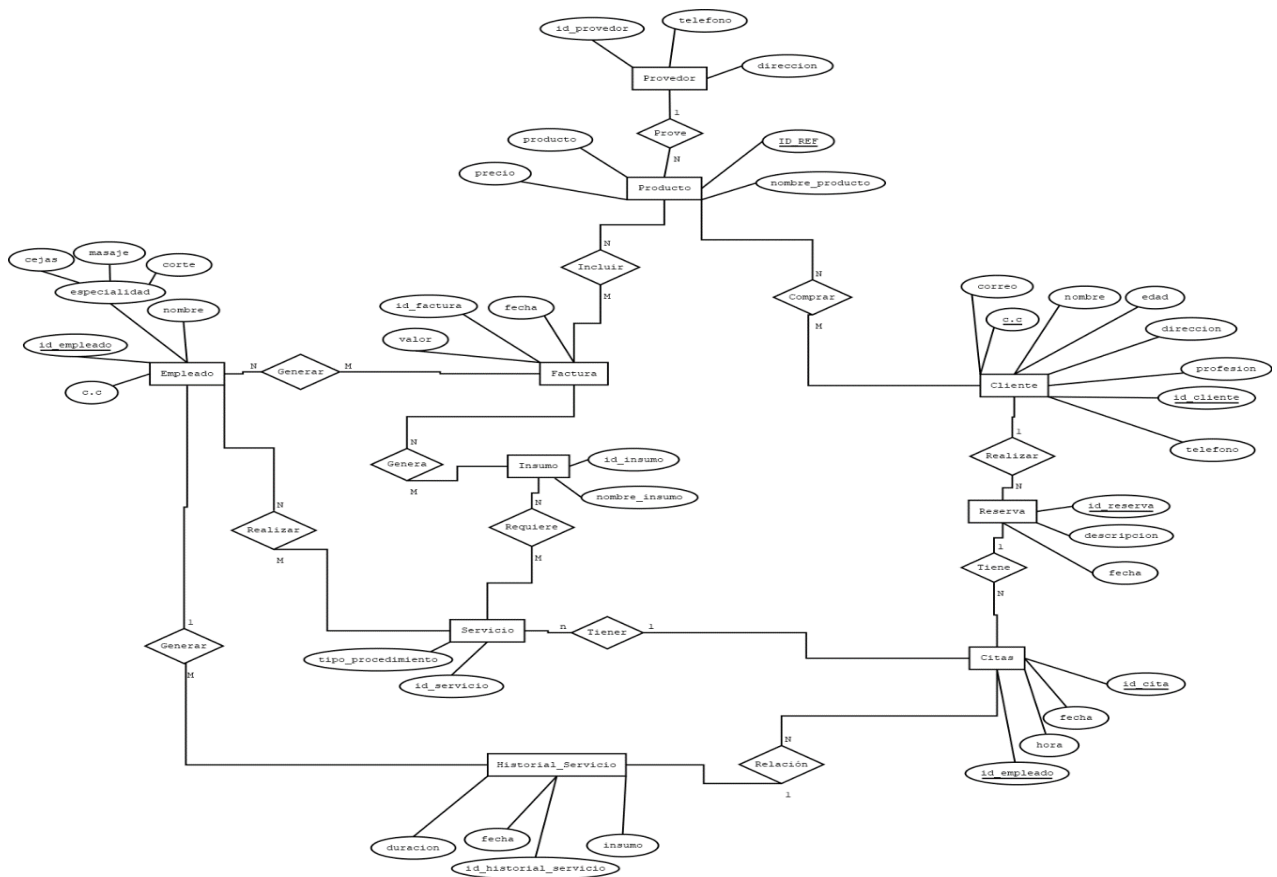
Relaciones:

- Provee
- Comprar
- Realizar
- Obtener
- Requerir
- Realizar
- Tener
- Generar
- Incluir
-

Cardinalidad

- **Generar:** Un empleado puede generar una o varias facturas y una factura puede ser generado por uno o varios empleados
- **Incluir:** un producto puede ser incluido en uno o varias facturas y una factura puede incluir uno o varios productos
- **Generar:** Un empleado le generan una o muchas historias de servicio una historia de servicio es generada para un empleado
- **Tener:** Tener Un servicio tiene una cita y una cita tiene una o muchos servicios
- **Realizar:** Un empleado puede realizar uno o muchos servicios y un servicio es realizado por un empleado
- **Requerir:** Un servicio requiere de uno o muchos insumos y un insumo es requerido por uno o muchos servicios.
- **Obtener:** Una cita es obtenida con una reserva y una reserva podría obtener una o varias citas
- **Realizar:** una reserva es realizada a un cliente y un cliente puede realizar una o varias reservas.
- **Comprar:** Un cliente puede comprar uno o varios productos y un producto es comprado por uno o varios clientes
- **Provee:** un proveedor provee varios productos y un producto se provee de un proveedor

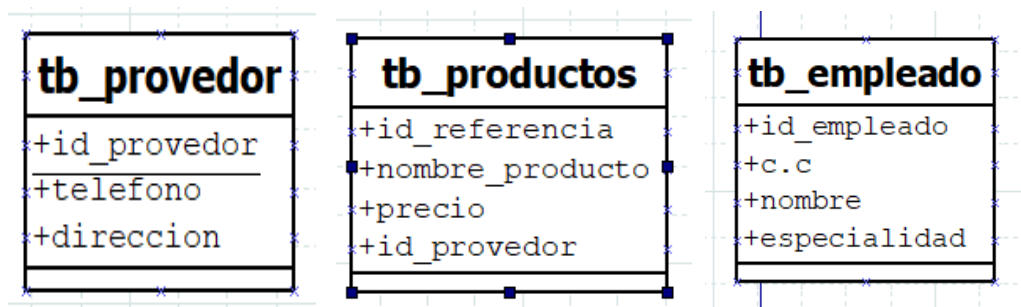
Modelo entidad relación E-R



Modelo entidad relación E-R

Modelo relacional.

Se transforman las entidades del modelo relacional en tablas con sus respectivos atributos y se realiza la primera transformación de relaciones (1:N o N:1).



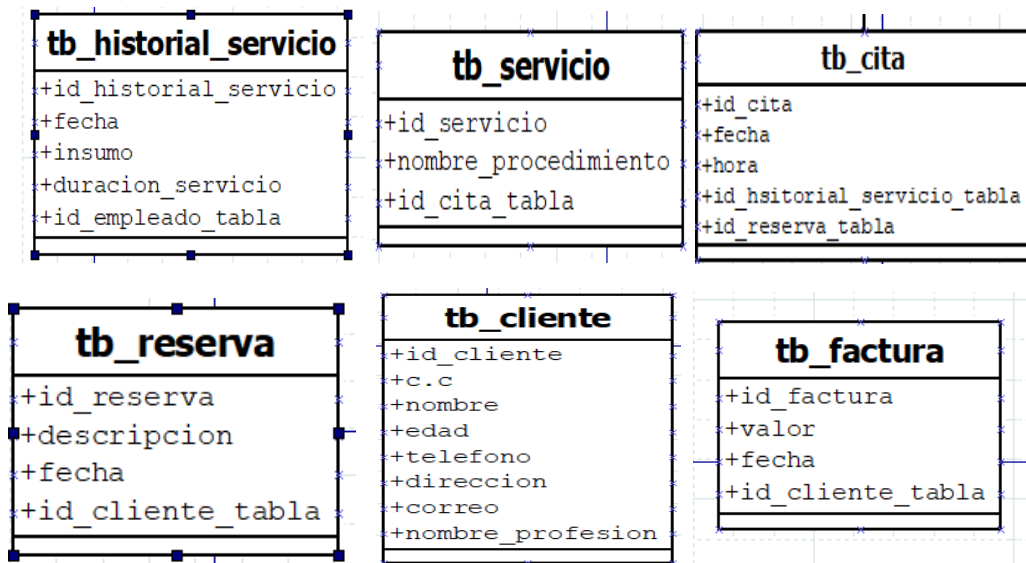
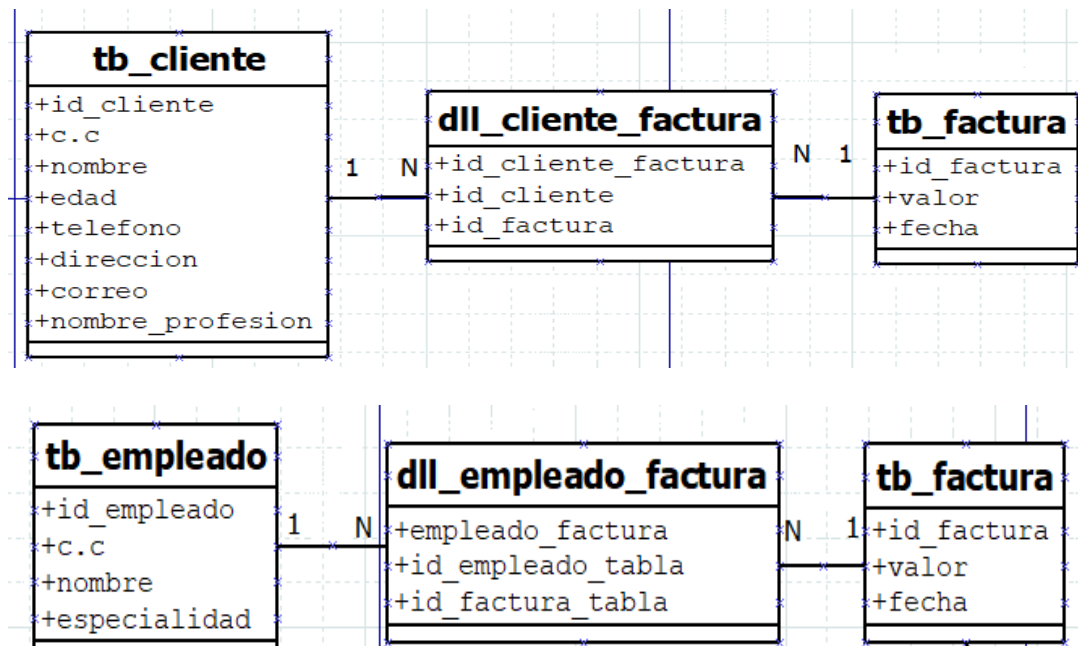
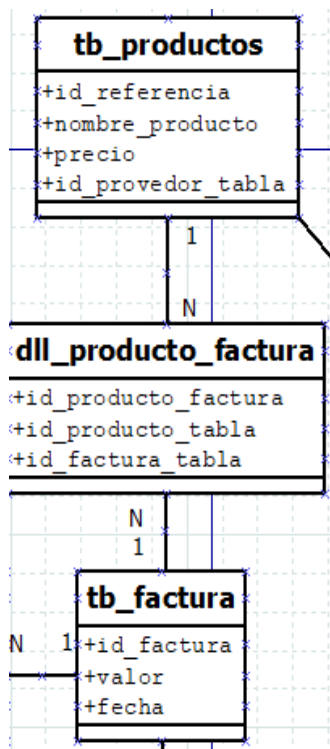
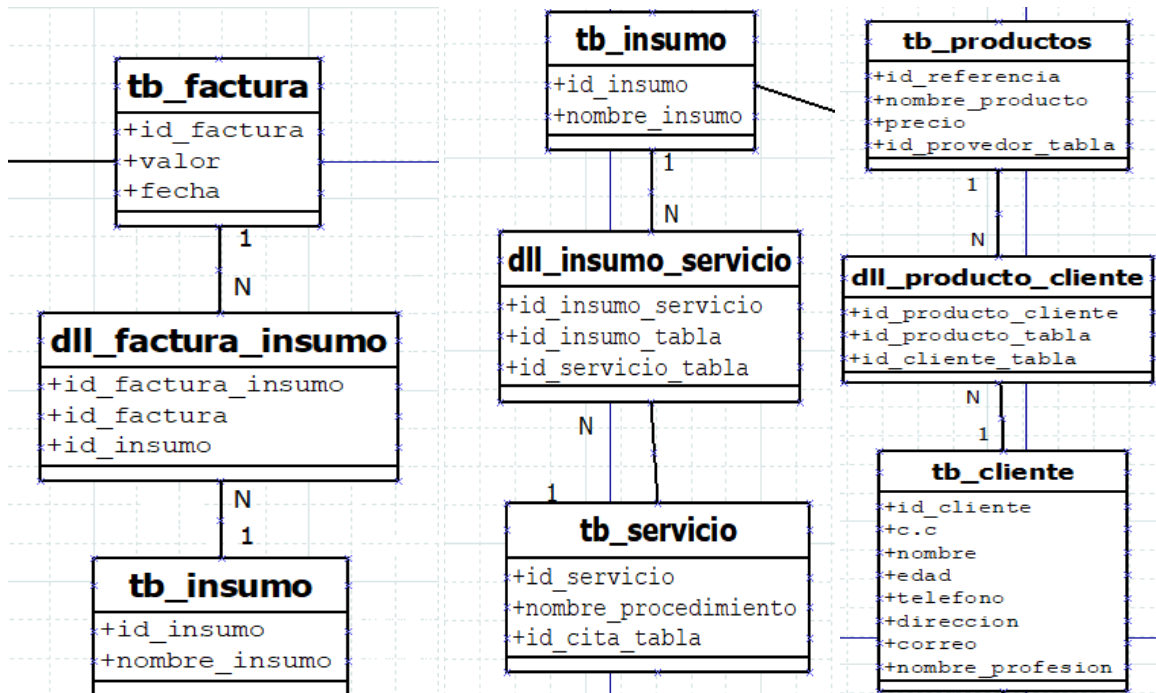


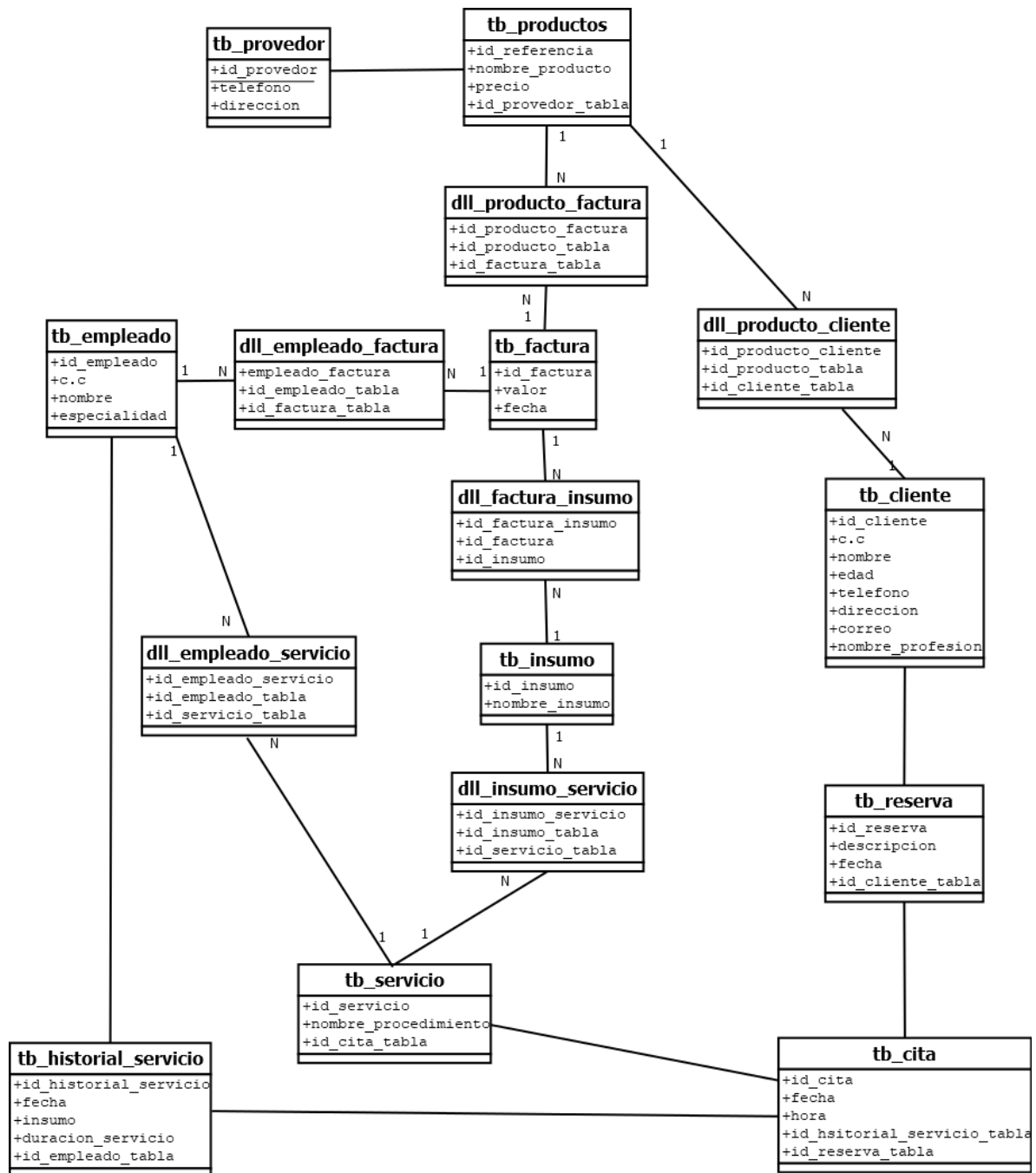
Figura 2 primera transformación

2. Se transforman las relaciones M:N y los atributos multivaluados en tablas





Resultado de la tabla M-R



Normalización

- 1FN

Se cumple con la separación en atributos atómicos, lo cual se visualiza en la imagen anterior, además los atributos dependen únicamente de la clave primaria de cada tabla. Todo esto con el fin de eliminar los valores repetidos en la BD.

2FN

Luego de cumplir con la primera forma normal, se crea la relación entre tablas con sus respectivas claves foráneas, es decir, clave ajena.

3FN

Se crean tablas de detalle a causa de la relación muchos a muchos:

- Tabla detalle entre producto y factura.
- Tabla detalle entre factura y empleado.
- Tabla detalle entre empleado y servicio
- Tabla detalle entre insumo y servicio
- Tabla detalle entre factura y insumo
- Tabla detalle entre producto y cliente

Creación de las tablas en la base de datos MySQL.

Se crea la base de datos barberías

- `CREATE DATABASE barberia CHARACTER set utf8 collate utf8_spanish_ci;`

Se crea la tabla con el nombre contenido en la siguiente imagen

```
use barberia;  
CREATE TABLE IF NOT EXISTS `tb_proveedor` (  
  `id_proveedor` INT NOT NULL,  
  `telefono` VARCHAR(45) NULL,  
  `direccion` VARCHAR(45) NULL,  
  PRIMARY KEY (`id_proveedor`)  
) ENGINE=InnoDB;
```

Se crea la tabla con el nombre contenido en la siguiente imagen


```

CREATE TABLE IF NOT EXISTS `tb_productos` (
  `id_referencia` INT NOT NULL,
  `nombre_producto` VARCHAR(100) NULL,
  `precio` VARCHAR(45) NULL,
  `id_proveedor_tabla` INT NOT NULL,
  PRIMARY KEY (`id_referencia`),
  INDEX `id_proveedor_productos` (`id_proveedor_tabla` ASC) VISIBLE,
  CONSTRAINT `FKidproducto` FOREIGN KEY (`id_proveedor_tabla`) REFERENCES
  `tb_proveedor` (`id_proveedor`)
  ON DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB;

```

Se crea la tabla con el nombre contenido en la siguiente imagen

```

CREATE TABLE IF NOT EXISTS `tb_factura` (
  `id_factura` INT NOT NULL,
  `valor` VARCHAR(45) NULL,
  `fecha` VARCHAR(45) NULL,
  PRIMARY KEY (`id_factura`)
) ENGINE=InnoDB;

```

Se crea la tabla con el nombre contenido en la siguiente imagen

```

CREATE TABLE dll_producto_factura(
  `id_producto_factura` INT NOT NULL,
  `id_producto_tabla` INT NOT NULL,
  `id_factura_tabla` INT NOT NULL,
  PRIMARY KEY (`id_producto_factura`, `id_producto_tabla`, `id_factura_tabla`),
  CONSTRAINT `id_producto_tabla` FOREIGN KEY (`id_producto_tabla`)
  REFERENCES `barberia`.`tb_productos` (`id_referencia`) ON DELETE NO
  ACTION ON UPDATE NO ACTION,
  CONSTRAINT `id_factura_tabla` FOREIGN KEY (`id_factura_tabla`)
  REFERENCES `barberia`.`tb_factura` (`id_factura`) ON DELETE NO
  ACTION ON UPDATE NO ACTION);

```

Se crea la tabla con el nombre contenido en la siguiente imagen

```

CREATE TABLE IF NOT EXISTS `tb_empleado` (
  `id_empleado` INT NOT NULL,
  `c.c` VARCHAR(45) NULL,
  `nombre` VARCHAR(45) NULL,
  `nombre_especialidad` VARCHAR(45) NULL,
  PRIMARY KEY (`id_empleado`)
) ENGINE=InnoDB;

```

Se crea la tabla con el nombre contenido en la siguiente imagen

```
CREATE TABLE dll_empleado_factura(  
  `id_empleado_factura` INT NOT NULL,  
  `id_empleado_tab` INT NOT NULL,  
  `id_factura_tab` INT NOT NULL,  
  PRIMARY KEY (`id_empleado_factura`, `id_empleado_tab`, `id_factura_tab`),  
  CONSTRAINT `id_empleado_tab` FOREIGN KEY (`id_empleado_tab`)  
  REFERENCES `barberia`.`tb_empleado` (`id_empleado`) ON DELETE  
  NO ACTION ON UPDATE NO ACTION,  
  CONSTRAINT `id_factura_tab` FOREIGN KEY (`id_factura_tab`)  
  REFERENCES `barberia`.`tb_factura` (`id_factura`) ON DELETE NO  
  ACTION ON UPDATE NO ACTION);
```

Se crea la tabla con el nombre contenido en la siguiente imagen

```
> CREATE TABLE IF NOT EXISTS `tb_historial_servicio` (  
  `id_historial_servicio` INT NOT NULL,  
  `fecha` VARCHAR(20) NULL,  
  `insumo` VARCHAR(45) NULL,  
  `duracion_servicio` VARCHAR(45) NULL,  
  `id_empleado_tabl` INT NOT NULL,  
  PRIMARY KEY (`id_historial_servicio`),  
  INDEX `id_empleado_tabl_historial_servicio` (`id_empleado_tabl` ASC) VISIBLE,  
  CONSTRAINT `FKid_historial_servic` FOREIGN KEY (`id_empleado_tabl`) REFERENCES  
  `tb_empleado` (`id_empleado`)  
  ON DELETE NO ACTION ON UPDATE NO ACTION  
- ) ENGINE=InnoDB;
```

Se crea la tabla con el nombre contenido en la siguiente imagen

```
⊖ CREATE TABLE IF NOT EXISTS `tb_cliente` (  
  `id_cliente` INT NOT NULL,  
  `c.c` VARCHAR(20) NULL,  
  `nombre` VARCHAR(100) NULL,  
  `edad` VARCHAR(10) NULL,  
  `telefono` VARCHAR(20) NULL,  
  `direccion` VARCHAR(45) NULL,  
  `correo` VARCHAR(70) NULL,  
  `profesion` VARCHAR(45) NULL,  
  PRIMARY KEY (`id_cliente`)  
- ) ENGINE=InnoDB;
```

Se crea la tabla con el nombre contenido en la siguiente imagen

```
> CREATE TABLE dll_producto_cliente(  
  `id_producto_cliente` INT NOT NULL,  
  `id_producto_tablaa` INT NOT NULL,  
  `id_cliente_tablaa` INT NOT NULL,  
  PRIMARY KEY (`id_producto_cliente`, `id_producto_tablaa`, `id_cliente_tablaa`),  
  CONSTRAINT `id_producto_tablaa` FOREIGN KEY (`id_producto_tablaa`)  
  REFERENCES `barberia`.`tb_productos` (`id_referencia`) ON DELETE NO ACTION ON UPDATE NO ACTION,  
  CONSTRAINT `id_cliente_tablaa` FOREIGN KEY (`id_cliente_tablaa`)  
  REFERENCES `barberia`.`tb_cliente` (`id_cliente`) ON DELETE NO ACTION ON UPDATE NO ACTION);
```

Se crea la tabla con el nombre contenido en la siguiente imagen

```
CREATE TABLE IF NOT EXISTS `tb_reserva` (  
  `id_reserva` INT NOT NULL,  
  `descripcion` VARCHAR(200) NULL,  
  `fecha` VARCHAR(40) NULL,  
  `id_cliente_reserva` INT NOT NULL,  
  PRIMARY KEY (`id_reserva`),  
  INDEX `id_cliente_reserva` (`id_cliente_reserva` ASC) VISIBLE,  
  CONSTRAINT `FKid_cliente` FOREIGN KEY (`id_cliente_reserva`) REFERENCES `tb_cliente` (`id_cliente`)  
  ON DELETE NO ACTION ON UPDATE NO ACTION  
) ENGINE=InnoDB;
```

En la siguiente imagen se muestran las tablas creadas en la base de dato llamada barberia

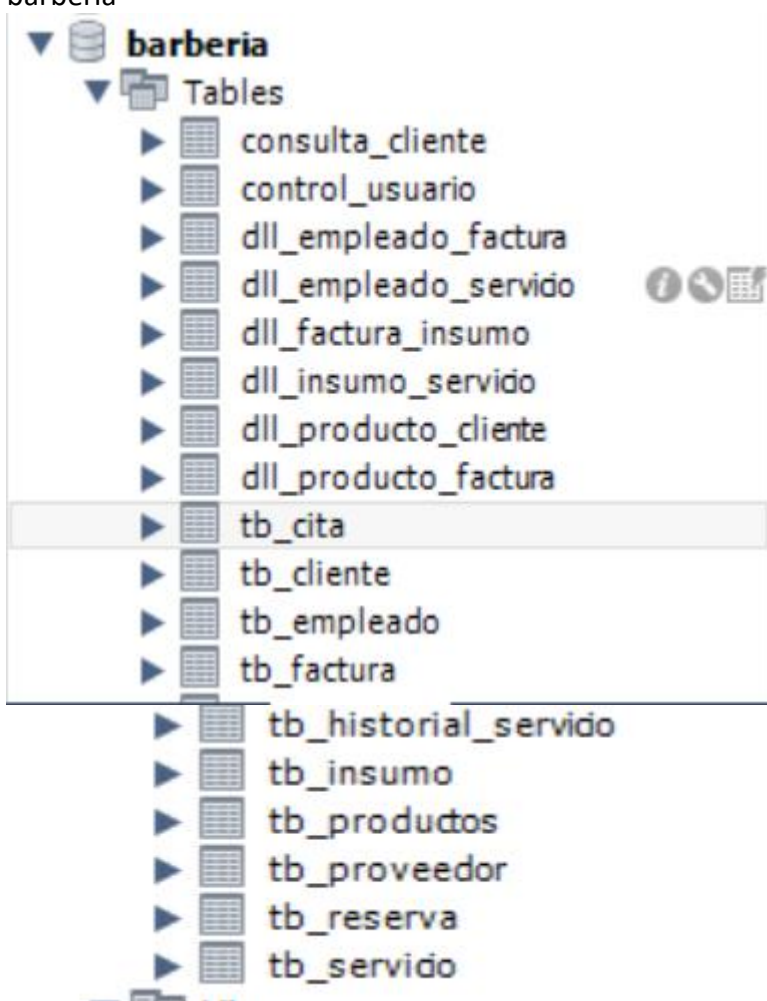
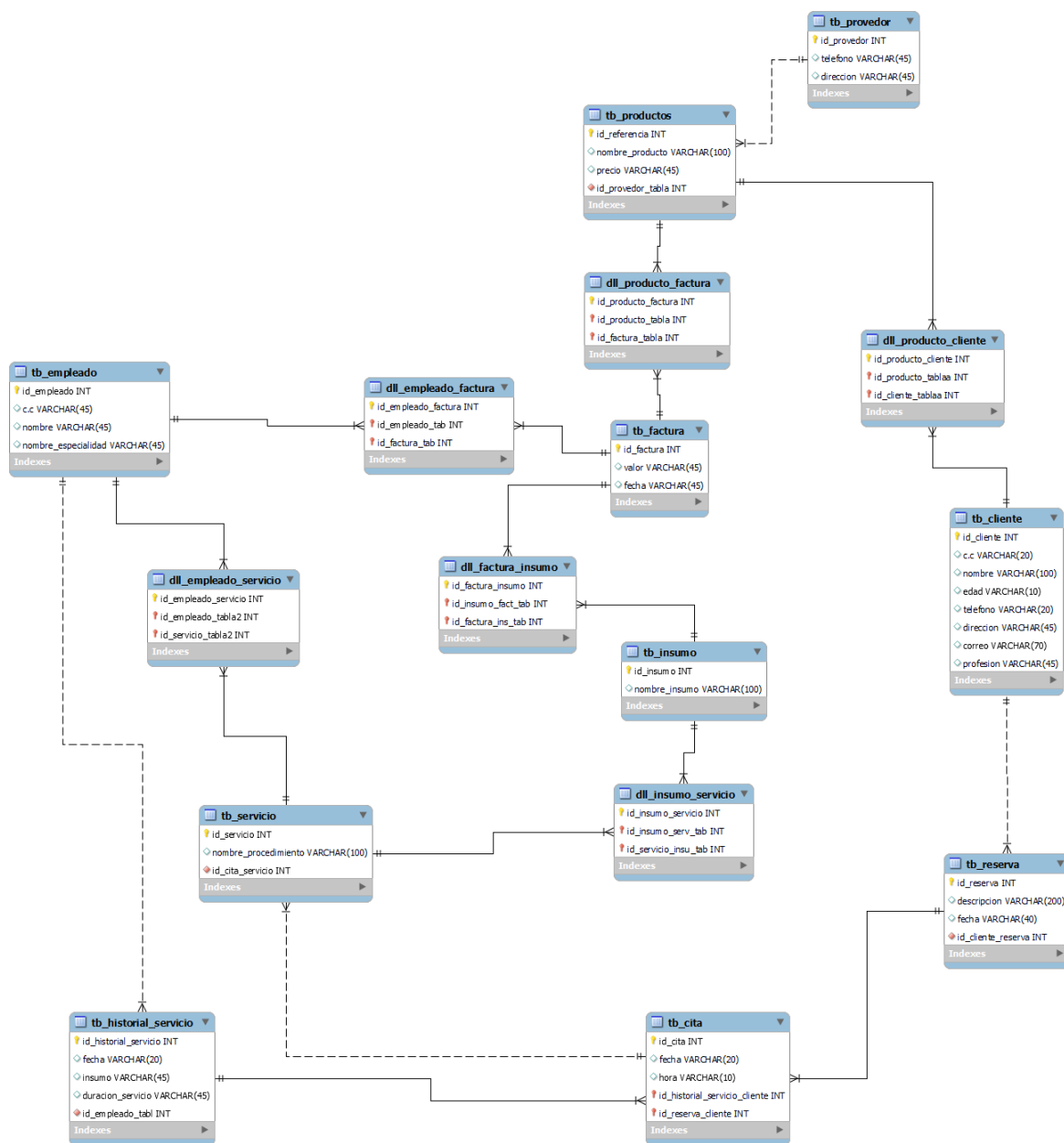


Diagrama del M-R creado en MySql



Escribir consultas que me permitan ver la información de cada tabla o de varias tablas.

1. Consulta de datos personales del cliente

```
SELECT nombre,telefono, correo FROM tb_cliente;
```

nombre	telefono	correo
Antonio Ospina	34475756	ospina_88@hotmail.co
Luis Marin	344757	ona_88@hotmail.co
Antonio Lopez	3447575677	osoa_88@hotmail.co
Liusita Morales	344757565	ospi_88@hotmail.co

2. Consulta para ordenar los datos de una persona en forma ascendente

```
Select * from tb_cliente ORDER BY nombre ASC;
```

id_cliente	c.c	nombre	edad	telefono	direccion	correo	profesion
102	10932003	Antonio Lopez	32	3447575677	Cil 2 bis # 162-35	osoa_88@hotmail.co	A.Q
100	10932665433	Antonio Ospina	34	34475756	Cil 2 bis # 12-35	ospina_88@hotmail.co	Q.A
103	10935433	Liusita Morales	45	344757565	Cil 2 bis # 152-35	ospi_88@hotmail.co	L.Q
101	1093277777	Luis Marin	12	344757	Cil 2 bis # 162-35	ona_88@hotmail.co	Q.A

3. Consulta en una tabla la información empezando por las letras Al

```
SELECT * FROM tb_productos where nombre_producto LIKE 'A1%';
```

id_referencia	nombre_producto	precio	id_proveedor_tabla
400	Alcohol	10.000	300
401	Algodon	200.00	301

4. Consulta de dos tablas para obtener la información del cliente que realizo la reserva

```
select r.descripcion,r.fecha, c.nombre, c.telefono
from tb_reserva as r INNER JOIN
tb_cliente as c on r.id_cliente_reserva=c.id_cliente;
```

descripcion	fecha	nombre	telefono
Depilacion	12-12-...	Antonio Ospina	34475756
Corte de cejas	12-10-...	Luis Marin	344757
Corte de cabello	12-11-...	Antonio Lopez	3447575677
Manicure	12-12-...	Liusita Morales	344757565

5. Consulta el nombre de la especialidad de un empleado

```
SELECT nombre , nombre_especialidad FROM tb_empleado;
```

nombre	nombre_especialidad
Luis Morales	Manicurista
Morado luis	Peluquero
Jeiso Man...	Cejero
Jesus david	Peluera

6. Consulta de insumo que se tiene en el negocio

```
SELECT nombre_insumo FROM tb_insumo;
```

nombre_insumo
Alcohol
Tinter
Gel
Papel

7. Consulta de las fechas que se tiene una reserva y que se va hacer al cliente

```
SELECT fecha, descripcion FROM tb_reserva;
```

Resultado de la consulta

fecha	descripcion
12-12-2022	Depilacion
12-10-2022	Corte de cejas
12-11-2022	Corte de cabello
12-12-2022	Manicure

8. Consulta por rango de hora en la tabla cita

```
SELECT * FROM tb_cita where hora BETWEEN 15 AND 18;
```

id_cita	fecha	hora	id_historial_servicio_cliente	id_reserva_cliente
900	12-23-2022	15:00	500	800
902	12-23-2022	16:00	502	802
903	12-23-2022	17:00	503	803
NULL	NULL	NULL	NULL	NULL

9. Consulta por id de un producto

```
SELECT * FROM tb_productos where id_proveedor_tabla=301;
```

Resultado de lá tabla

id_referencia	nombre_producto	precio	id_proveedor_tabla
401	Algodon	200.00	301
NULL	NULL	NULL	NULL

10. Consulta en la tabla empleado por filtro

```
SELECT * FROM tb_empleado where nombre LIKE 'je%';
```

Result Grid				
Filter Rows:		Edit: Export/Imp		
	id_empleado	c.c	nombre	nombre_especialidad
▶	202	10999955662	Jeiso Man...	Cejero
	203	1093245	Jesus david	Peluera
*	NULL	NULL	NULL	NULL

Generar de 4 a 6 vistas donde se evidencie lo más importante de cada ejercicio (haga una selección muy responsable de la información realmente importante según el contexto).

1. Se crea vista donde se tiene la información del procedimiento fecha y nombre del cliente

```
CREATE VIEW info_procedim_al_cliente as
select r.descripcion,r.fecha, c.nombre, c.telefono
from tb_reserva as r INNER JOIN
tb_cliente as c on r.id_cliente_reserva=c.id_cliente;
```

Resultado de vista

Result Grid				
Filter Rows:		Export: Wrap Cell Content:		
	descripcion	fecha	nombre	telefono
▶	Depilacion	12-12-2022	Antonio Ospina	34475756
	Corte de cejas	12-10-2022	Luis Marin	344757
	Corte de cabello	12-11-2022	Antonio Lopez	3447575677
	Manicure	12-12-2022	Liusita Morales	344757565

2. Se crea vista para filtrar un cliente con los datos personales y servicio realizado

```
CREATE VIEW servicio_cliente AS SELECT tb_servicio.id_servicio, tb_servicio.nombre_procedimiento,
tb_cita.fecha, tb_cliente.nombre, tb_cliente.telefono
FROM tb_servicio
inner JOIN tb_cita On tb_servicio.id_cita_servicio = tb_cita.id_cita
inner join tb_cliente on tb_cita.id_reserva_cliente;
```

Resultado de la vista

nombre_procedimiento	fecha	nombre	telefono
1 Depilacion	12-23-2022	Antonio Ospina	34475756
1 Corte de cabello	12-23-2022	Antonio Ospina	34475756
1 Depilacion	12-23-2022	Antonio Ospina	34475756
1 Corte de cejas	12-23-2022	Antonio Ospina	34475756
1 Depilacion	12-23-2022	Luis Marin	344757
1 Corte de cabello	12-23-2022	Luis Marin	Luis Marin 57
1 Depilacion	12-23-2022	Luis Marin	344757
1 Corte de cejas	12-23-2022	Luis Marin	344757
1 Depilacion	12-23-2022	Antonio Lopez	3447575677
1 Corte de cabello	12-23-2022	Antonio Lopez	3447575677
1 Depilacion	12-23-2022	Antonio Lopez	3447575677
1 Corte de cejas	12-23-2022	Antonio Lopez	3447575677
1 Depilacion	12-23-2022	Liusita Morales	344757565
1 Corte de cabello	12-23-2022	Liusita Morales	344757565

3. Vista que filtra por horario de la cita

```
SELECT * FROM filtrar_cliente_por_hora;
```

```
CREATE VIEW filtrar_cliente_por_hora as
SELECT * FROM tb_cita where hora BETWEEN 15 AND 18;
SELECT * FROM filtrar_cliente_por_hora;
```

Resultado de la vista

fecha	hora	id_historial_serv	id_reserva_cli
12-23-2022	15:00	500	800
12-23-2022	16:00	502	802
12-23-2022	17:00	503	803

Las 4 visitas



Generar al menos 4 procedimientos almacenados.

1. Procedimiento de tipo consulta tabla cliente

```
DELIMITER //
CREATE PROCEDURE consulta_cliente(in id_cliente INT)
BEGIN
    Select * from tb_cliente where id_cliente = id_cliente;
END//
DELIMITER ;
```

Resultado del procedimiento

```
8
9 • CALL consulta_cliente(101);
```

id_cliente	c.c	nombre	edad	telefono	direccion	correo	profesion
101	10932777777	Luis Marin	12	344757	Cll 2 bis # 162-35	ona_88@hotmail.co	Q.A

2. Procedimiento para actualizar

```
delimiter //
create procedure actualizar(in id_proveedor1 int,in telefononuev varchar(45),
in direc nuev varchar(45), in
nombrenuev varchar(100))
begin
update tb_proveedor
set telefono=telefononuev_nue, direccion = direc nuev, nombre=nombrenuev
where id_proveedor=proveedor1;
end
//
delimiter ;
CALL actualizar(303, "6666666", "sin direccion", "Juan Gabriel");
```

Actualización del id 303

id_proveedor	telefono	direccion	nombre
300	3155433	Cla27c # 12-21	Miguel Martinez
301	666566	Cla27c # 12-21	Daniel mejia
302	8777	Cla27c # 152...	Maria Ospina
303	6666666	sin direccion	Juan Gabriel

3. Procedimiento para Eliminar Registro

DELIMITER //

- **CREATE PROCEDURE** eliminar_datos(**IN** id **INT**)
BEGIN
 DELETE FROM tb_proveedor **WHERE** id_proveedor = id;
END //

DELIMITER ;

- **CALL** eliminar_datos(304);

Registro para eliminar con el id_304

id_proveedor	telefono	direccion	nombre
302	8777	Cla27c # 152...	Maria Ospina
303	6666666	sin direccion	Juan Gabriel
304	7777	888ca	Juan Jose

id_proveedor	telefono	direccion	nombre
301	666566	Cla27c # 12-21	Daniel mejia
302	8777	Cla27c # 152...	Maria Ospina
303	6666666	sin direccion	Juan Gabriel
NULL	NULL	NULL	NULL

Registro para eliminar con el id_304

4. Procedimiento para Insertar un nuevo registro a la base de datos

```

14 DELIMITER //
15
16 • CREATE PROCEDURE insertar_registro()
17 BEGIN
18     INSERT INTO barberia.tb_cliente (id_cliente, cedula, nombre, edad,
19                                     telefono, direccion, correo, profesion)
20     VALUES (105, '1093244', 'Emilia Maria Ospina',
21             '3447676', '18', 'Cile 28b # 78-21', 'naoz@gmail.com',
22             'Abogada');
23 END //
24
25 DELIMITER ;
26
27 • CALL insertar_registro;

```

Insertar un nuevo registro a la base de datos

id_proveedor	telefono	direccion	nombre
301	666566	Cla27c # 12-21	Daniel mejia
302	8777	Cla27c # 152...	Maria Ospina
303	6666666	sin direccion	Juan Gabriel
NULL	NULL	NULL	NULL

id_cliente	cedula	nombre	edad	telefono	direccion	correo	profesion
100	10932665433	Antonio Ospina	34	34475756	Cl 2 bis # 12-35	ospina_88@hotmail.co	Q.A
101	10932777777	Luis Marin	12	344757	Cl 2 bis # 162-35	ona_88@hotmail.co	Q.A
102	10932003	Antonio Lopez	32	3447575677	Cl 2 bis # 162-35	osoa_88@hotmail.co	A.Q
103	10935433	Liusita Morales	45	344757565	Cl 2 bis # 152-35	ospi_88@hotmail.co	L.Q
105	1093244	Emilia Maria Ospina	3447676	18	Cle 28b # 78-21	naoz@gmail.com	Abogada

Los cuatros procedimientos

- Stored Procedures
 - actualizar
 - actualizar_cliente
 - consulta_cliente
 - eliminar_datos
 - insertar_registro

Generar al menos 4 triggers

1. Trigger: La funcionalidad de este trigger es para cada vez que ingresen la información completa de la tabla cliente, me guarde la información del nombre y teléfono en otra tabla

```
3
4 DELIMITER |
5 • CREATE TRIGGER infor_cliente BEFORE INSERT ON tb_cliente
6   FOR EACH ROW BEGIN
7     INSERT INTO consulta_cliente(nombre_cliente,telefono) VALUE (new.nombre, new.telefono);
8   END |
~
```

Aquí guarda los datos completos

id_cliente	cedula	nombre	edad	telefono	direccion	correo	profesion
105	1093244	Emilia Maria Ospina	34	18112223	Cle 28b # 78-21	naoz@gmail.com	Abogada
106	109326435	Yarledis Zuñiga	45	54455666	Cl 2 bis # 152-35	yarle@hlo	Ingeniera
107	109326435	Nevardo Zuñiga	54	1088216364	Mz B # 12-35	Zunle@hlo yarle@hlo	Ingeniera

Pero en la tabla consulta cliente me guarda los datos más necesario para mí.

id_respaldo	nombre_cliente	telefono
1	Yarledis Zuñiga	54455666
2	Nevardo Zuñiga	1088216364

Se crea una tabla con los campos acción y fecha donde me guardara la información de los siguientes movimientos realizadas a las siguientes tablas y diferentes métodos creados

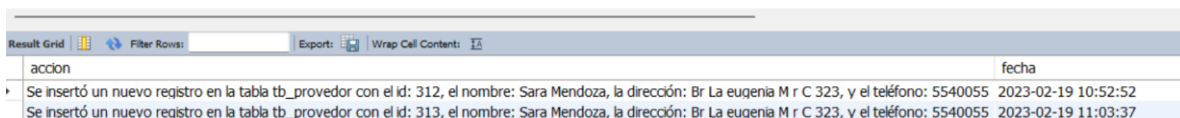
accion	fecha
NULL	NULL

2. Trigger: El siguiente trigger nos respalda la información de la fecha y acción cuando se ingresa nueva información

```
DELIMITER //
```

- **CREATE TRIGGER** insert_tb_proveedor_trigger
AFTER INSERT ON tb_proveedor
FOR EACH ROW
BEGIN
 INSERT INTO control_usuario (accion) **VALUES**
 (CONCAT('Se insertó un nuevo registro en la tabla tb_proveedor con el id: ', NEW.id_proveedor,
 ', el nombre: ', NEW.nombre, ', la dirección: ', NEW.direccion, ', y el teléfono: ', NEW.telefono));
END//
DELIMITER ;

Como se puede observar en la siguiente imagen



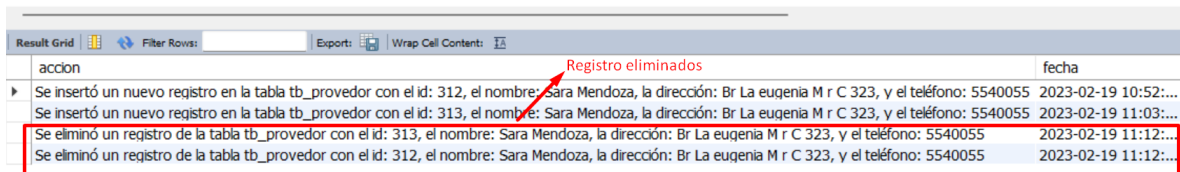
accion	fecha
Se insertó un nuevo registro en la tabla tb_proveedor con el id: 312, el nombre: Sara Mendoza, la dirección: Br La eugenia M r C 323, y el teléfono: 5540055	2023-02-19 10:52:52
Se insertó un nuevo registro en la tabla tb_proveedor con el id: 313, el nombre: Sara Mendoza, la dirección: Br La eugenia M r C 323, y el teléfono: 5540055	2023-02-19 11:03:37

3. Trigger: para cuando se elimine un registro y este registre que información eliminada

```
DELIMITER //
```

- **CREATE TRIGGER** Eliminar_tb_proveedor_trigger
BEFORE DELETE ON tb_proveedor
FOR EACH ROW
BEGIN
 INSERT INTO control_usuario (accion) **VALUES**
 (CONCAT('Se eliminó un registro de la tabla tb_proveedor con el id: ', OLD.id_proveedor,
 ', el nombre: ', OLD.nombre, ', la dirección: ', OLD.direccion, ', y el teléfono: ', OLD.telefono));
END//
DELIMITER ;

Como se puede observar en la siguiente imagen



accion	fecha
Se insertó un nuevo registro en la tabla tb_proveedor con el id: 312, el nombre: Sara Mendoza, la dirección: Br La eugenia M r C 323, y el teléfono: 5540055	2023-02-19 10:52:...
Se insertó un nuevo registro en la tabla tb_proveedor con el id: 313, el nombre: Sara Mendoza, la dirección: Br La eugenia M r C 323, y el teléfono: 5540055	2023-02-19 11:03:...
Se eliminó un registro de la tabla tb_proveedor con el id: 313, el nombre: Sara Mendoza, la dirección: Br La eugenia M r C 323, y el teléfono: 5540055	2023-02-19 11:12:...
Se eliminó un registro de la tabla tb_proveedor con el id: 312, el nombre: Sara Mendoza, la dirección: Br La eugenia M r C 323, y el teléfono: 5540055	2023-02-19 11:12:...

4. Trigger para cuando se actualiza un registro y este registre la información anterior y la nueva

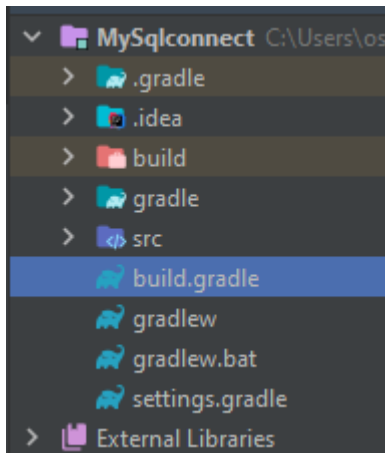
```
UPDATE tb_proveedor  
SET telefono = '31465535555', direccion = 'Calle 27#12-43', nombre = 'Juan Pérez'  
WHERE id_proveedor = 311;
```

Como se puede observar en la siguiente imagen

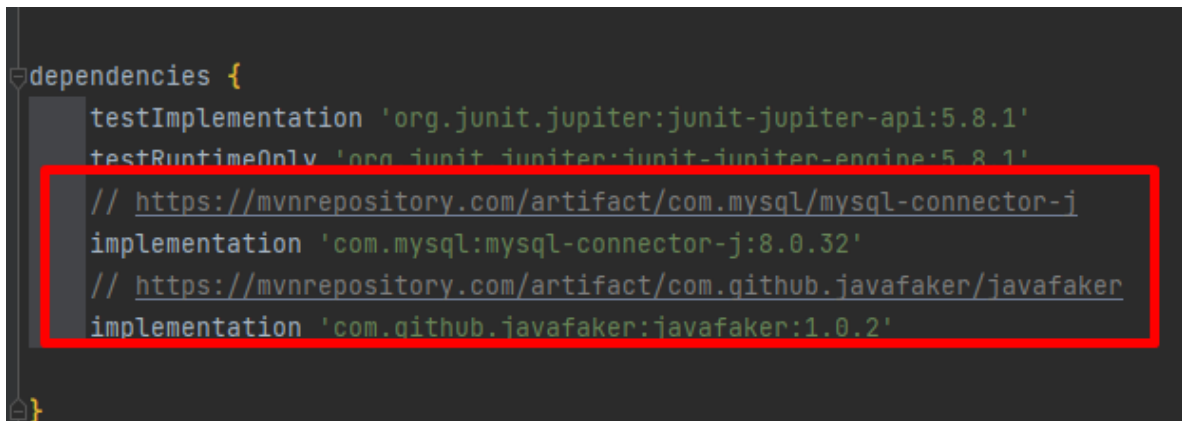
accion	fecha
Se insertó un nuevo registro en la tabla tb_proveedor con el id: 312, el nombre: Sara Mendoza, la dirección: Br La eugenia M r C 323, y el teléfono: 5540055	2023-02-19 10:52:...
Se insertó un nuevo registro en la tabla tb_proveedor con el id: 313, el nombre: Sara Mendoza, la dirección: Br La eugenia M r C 323, y el teléfono: 5540055	2023-02-19 11:03:...
Se eliminó un registro de la tabla tb_proveedor con el id: 313, el nombre: Sara Mendoza, la dirección: Br La eugenia M r C 323, y el teléfono: 5540055	2023-02-19 11:12:...
Se eliminó un registro de la tabla tb_proveedor con el id: 312, el nombre: Sara Mendoza, la dirección: Br La eugenia M r C 323, y el teléfono: 5540055	2023-02-19 11:12:...
Se actualizó un registro en la tabla tb_proveedor con el id: 311, el nombre: Juan Pérez, la dirección: Calle 27#12-43, y el teléfono: 3146553555	2023-02-19 11:20:...

Se crea la conexión java y MySQL workbench con los siguientes pasos.

1. Se crea el proyecto,



2. Se implementa las dependencia MySQL



3. La clase MySQLConstants

```

public class MySqlConstants {

    2 usages
    public static final String MY_SQL_JDBC_DRIVER="com.mysql.cj.jdbc.Driver";

    2 usages
    public static final String CONNECTION_STRING="jdbc:mysql://%s/%s?user=%s&password=%s";

}

```

4. Creación de la clase MySqlOperation e implementación de la clase DataBase

```

3 usages
public class MySqlOperation implements DataBase { Complexity is 9 It's time to do something...
    4 usages
    private Connection connection = null;
    5 usages
    private Statement statement = null;
    6 usages
    private ResultSet resultSet = null;
    4 usages
    private String sqlStatement;
    3 usages
    private String server;
    3 usages
    private String dataBaseName;
    3 usages
    private String user;
    3 usages
    private String password;

    public String getSqlStatement() { return sqlStatement; }
    10 usages
    public void setSqlStatement(String sqlStatement) { this.sqlStatement = sqlStatement; }
    public String getServer() { return server; }

```

```

    public String getDataBaseName() { return dataBaseName; }
    1 usage
    public void setDataBaseName(String dataBaseName) { this.dataBaseName = dataBaseName; }
    public String getUser() { return user; }
    1 usage
    public void setUser(String user) { this.user = user; }
    public String getPassword() { return password; }
    1 usage
    public void setPassword(String password) { this.password = password; }
    2 usages
    @Override
    public void configurateDataBaseConnection() {
        try {
            Class.forName(MY_SQL_JDBC_DRIVER);
            connection= DriverManager.getConnection(
                String.format(CONNECTION_STRING,
                    this.server,
                    this.dataBaseName,
                    this.user,
                    this.password)
            );
            statement=connection.createStatement();

        }catch (Exception e){
            close();
            System.out.println(e.getMessage());
        }
    }

```



```

public void executeSqlStatement() {
    try {
        configurateDataBaseConnection();
        resultSet = statement.executeQuery(sqlStatement);
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}

@Override
public ResultSet getResultSet() { return null; }

2 usages
@Override
public void close() { Complexity is 8 It's time to do something...
    try {
        if (resultSet != null) {
            resultSet.close();
        }
        if (statement != null) {
            statement.close();
        }
        if (connection != null) {
            connection.close();
        }
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}
}

```

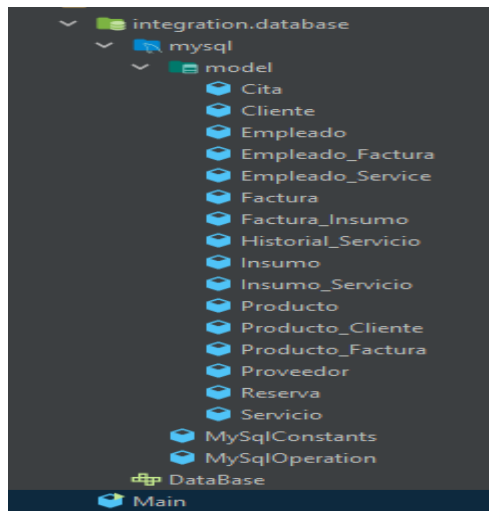
```

@Override
public void printResultSet() throws SQLException { Complexity is 6 It's time to do something...
    ResultSetMetaData resultSetMetaData = resultSet.getMetaData();
    int totalColumnNumber = resultSetMetaData.getColumnCount();
    while (resultSet.next()) {
        for (int columnNumber = 1; columnNumber <= totalColumnNumber; columnNumber++) {
            if (columnNumber > 1) {
                System.out.print(",");
            }
            String columnValue = resultSet.getString(columnNumber);
            System.out.print(resultSetMetaData.getColumnName(columnNumber) + ": " + columnValue);
        }
        System.out.print("\n");
    }
}

10 usages
@Override
public void executeSqlStatementVoid() {
    try {
        configurateDataBaseConnection();
        statement.execute(sqlStatement);
    } catch (Exception e) {
        System.out.println(e.getMessage());
    }
}
}

```

5. Se crea una clase para cada entidad de la tabla con sus atributos



6. Se crea la class Main

```
public static void main(String[] args) throws SQLException {  
    openConnection();  
    //List<Integer> empleados = agregarEmpleado();  
    //System.out.println(empleados);  
    List<Integer> clientes = agregarCliente();  
    //List<Integer> proveedores = agregarProveedor();  
    //List<Integer> reservas = agregarReserva();  
    //List<Integer> servicio = agregarServicio();  
    //List<Integer> insumos = agregarInsumo();  
    //List<Integer> producto = agregarProducto();  
    //List<Integer> HistorialServicio = agregarHistorialServicio();  
    //List<Integer> facturas = agregarFacturas();  
    //List<Integer> citas = agregarCita();  
    closeConnection();  
}
```

```

public class Main { Complexity is 6 It's time to do something...

    1 usage
    private static final String SERVER = "localhost";
    1 usage
    private static final String DATA_BASE_NAME = "barberia";
    1 usage
    private static final String USER = "root";
    1 usage
    private static final String PASSWORD = "admin";
    25 usages
    private static final MySQLOperation mySqlOperation = new MySQLOperation();
    2 usages
    public static List<Integer> id_empleado = new ArrayList<>();
    public static List<Integer> id_cliente = new ArrayList<>();
    public static List<Integer> id_proveedor = new ArrayList<>();
    public static List<Integer> id_reserva = new ArrayList<>();
    public static List<Integer> id_servicio = new ArrayList<>();
    public static List<Integer> id_insumo = new ArrayList<>();
    2 usages
    public static List<Integer> id_referencia = new ArrayList<>();
    2 usages
    public static List<Integer> id_historial_servicio = new ArrayList<>();
    2 usages
    public static List<Integer> id_factura = new ArrayList<>();
    2 usages
    public static List<Integer> id_cita = new ArrayList<>();
    private static Faker faker = new Faker();

```

8.

```

private static List<Integer> agregarCliente() { Complexity is 5 Everything is cool!
    String insertCliente = "";
    List<Integer> id_cliente = new ArrayList<>();
    for (int i = 1; i <= 50; i++) {
        Cliente cliente = new Cliente();
        Faker faker = new Faker();
        cliente.setIdCliente(i);
        cliente.setCedula(faker.number().digits(10));
        cliente.setNombre(faker.name().fullName());
        cliente.setEdad(String.valueOf(faker.number().numberBetween(18, 65)));
        cliente.setTelefono(faker.phoneNumber().cellPhone());
        cliente.setDireccion(faker.address().streetAddress());
        cliente.setCorreo(faker.internet().emailAddress());
        cliente.setProfesion(faker.job().title());
        insertCliente = "INSERT INTO tb_cliente(id_cliente, cedula, nombre, edad, telefono, direccion, correo, profesion)" +
            " VALUES(%s,%s,%s,%s,%s,%s,%s,%s)";
        insertCliente = String.format(insertCliente,
            cliente.idCliente(),
            cliente.cedula(),
            cliente.nombre(),
            cliente.edad(),
            cliente.telefono(),
            cliente.direccion(),
            cliente.correo(),
            cliente.profesion());
        insertIntoCliente(insertCliente);
        id_cliente.add(cliente.idCliente());
    }
    return id_cliente;
}

```

```

1 usage
private static void insertIntoCliente(String sentencia) {
    mySqlOperation.setSqlStatement(sentencia);
    mySqlOperation.executeSqlStatementVoid();
}

```

Poblar la base de datos (50 registros por tabla) utilizando una conexión desde Java.

Registros en la tabla cliente

id_cliente	cedula	nombre	edad	telefono	direccion	correo	profesion
1	6175441080	Kyle Borer	42	1-592-061-3516	457 Kiera Street	eve.ankunding@yahoo.com	Direct Government Supervisor
2	2769298556	Ted Baumbach DVM	33	773-038-3990	6920 Eboni Brooks	troy.beatty@hotmail.com	Lead Design Analyst
3	7148342443	Blaine Bradtke	54	072.623.9384	83002 Noe Port	gordon.smitham@yahoo.com	Chief Representative
4	9828727601	Stan Haag	41	(017) 365-8498	16029 Martin Plaza	beulah.harris@hotmail.com	Education Liaison
5	0078148631	Mrs. America Nolan	57	239-127-7173	69184 Lloyd Port	maricela.lubowitz@gmail.com	International Assistant
6	3407233313	Cristy Becker	50	902-984-0002	7176 Nubia Green	celestine.runolfsdottir@hotmail.com	Senior Director
7	2837324002	Harris Murphy DDS	39	572-442-8542	542 Abbott Way	melvin.cummings@yahoo.com	District Administrator
8	8614830288	Fatimah Nitzsche	41	1-623-309-0078	5586 Bechtelar Tunnel	vennie.pacocha@gmail.com	Legacy Engineer
9	0597058554	Shane Dooley	42	1-347-093-0199	9282 Tressa View	frankie.rohan@hotmail.com	Farming Officer
10	6485790974	Jayson Kilback	33	045-499-9858	2114 Barrows Mission	valeria.kohler@hotmail.com	IT Designer
11	5260271298	Eldridge Hickle	31	182-863-1068	0835 Omer Isle	hung.sawayn@hotmail.com	Principal Sales Developer
12	6333474557	Florence Sanford	20	(073) 868-4786	987 Norman Knolls	brock.runolfsdottir@yahoo.com	Marketing Executive
13	6459449826	Terry Hodkiewicz V	21	016-286-7914	3958 Weldon Mountain	enrique.zboncak@yahoo.com	Customer Design Technician
14	0910973661	Dr. Anton Bruen	18	082.654.4309	09220 Lamar Views	dominica.glover@hotmail.com	Hospitality Representative

id_cliente	cedula	nombre	edad	telefono	direccion	correo	profesion
37	3088114440	Mrs. Delbert Hettinger	21	746.114.9401	2942 Corkery Ports	leoma.howell@gmail.com	Internal Community-Services ...
38	4678707236	Mauricio Kuphal	48	499-332-0233	2867 Jacobs Mountains	angelita.guskowski@gmail.com	Farming Administrator
39	6465314685	Giovanna Rippin	58	1-873-197-7909	15830 Jacobson Stre...	emery.kuhic@hotmail.com	Forward Strategist
40	5948857024	Mr. Olivia Hermiston	48	826.436.7290	9361 Florida Mount	jeff.zulauf@gmail.com	Sales Analyst
41	0498007186	Felipe Steuber Sr.	44	1-729-301-8197	7595 Ezra Estate	fanny.larkin@hotmail.com	Advertising Producer
42	1155428318	Preston Breitenberg	19	129.523.3897	03327 Lakin Views	marylin.botsford@gmail.com	Forward Manufacturing Facilit...
43	2717025515	Mrs. Zora Jacobi	55	1-062-275-7595	698 Grant Springs	lulu.rath@yahoo.com	Hospitality Technician
44	3177985170	Ange Coler	40	069.759.5951	01022 Octavio Road	chuck.torp@gmail.com	Legal Developer
45	2359902476	Stacey Satterfield	44	476.026.5595	299 Maurita Plaza	nela.tremblay@hotmail.com	District Consulting Specialist
46	1728565961	Bess Harber	22	(430) 029-6871	426 Waelchi Tunnel	veronica.rowe@yahoo.com	Investor Advertising Director
47	6412417462	Jena Abernathy	35	(306) 027-7753	99798 Pearlne Dam	charisse.heathcote@gmail.com	Legacy Liaison
48	5330523103	Mr. Ceola Abbott	28	(873) 147-2400	64566 Okuneva Knoll	denny.schmidt@yahoo.com	Customer Liaison
49	1736762178	Nidia Rodriguez	37	140-653-3785	98134 Spencer Locks	nora.olson@yahoo.com	Hospitality Director
50	4119982247	Teddy Kerluke	19	1-833-382-5045	535 Koch Key	martine.funk@yahoo.com	Human Liaison

Tabla empleada

id_empleado	cedula	nombre	nombre_especialidad
37	3020946937	Marcelina ...	Ms. Shanae Renner
38	1715939714	Zelma Lo...	Clark Bahringer
39	5124186542	Mrs. Thi K...	Alisia Runolfsdottir
40	8060138133	Mignon S...	Donnie Herman
41	7481985417	Tyson Mc...	Isiah Wisozk II
42	1703462764	Kenneth ...	Coy Nienow
43	8068247553	Dorathy ...	Allan Volkman
44	1780008573	Raul Marq...	Cherise Smitham
45	0106210465	Vanesa Ni...	Ms. Marty Kunze
46	1947372764	Ezequiel E...	Marina Beahan
47	2866902089	Brittney P...	Karly Rolfson
48	4180258683	Dellah E...	Anibal Kunde
49	6109902058	Jimmie M...	Noemi Olson V
50	7089391038	Linh Kessler	Michel Monahan

Y de esa forma se realizaron los registros a las demás tablas en el proyecto se anexará el código completo de la actividad.

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

Estoy muy satisfecho con el resultado del proyecto porque que cumple con todo, e incluso una base relacional es suficiente para almacenar información requerida en el proyecto. Entre sus diferencias principales, las bases de datos SQL combinan de forma eficiente diferentes tablas para extraer información relacionadas y las NoSQL no lo permiten o son muy limitadas. y normalmente la elección de una depende mucho del contexto y las necesidades del cliente.