Reto BD

Contents

Reto asignado: Reto 1

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 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.

Realizar el modelo E-R

- Se identificaron las siguientes entidades
    - Empleados: ID, cedula, Nombre, Especialidad.
    - Cliente: ID, cedula, Nombre, Profesión, Teléfono, correo, edad, Dirección.
    - Servicio: ID, fecha, hora inicio, hora final, tiempo duración, costo.
    - Insumo: ID, nombre
    - Factura: ID
    - Producto REF, nombre, cantidad precio
    - Proveedor: NIT, nombre, dirección
    - Cita: ID, fecha asignada.
- Relaciones entre las entidades:
- Cabe aclarar que en la Barberia se asigna un servicio por cita solicitada, en caso de requerir más servicios es importante solicitar una o más citas.

Entidades	Descripción	Cardinalidad
Cliente - cita	un cliente puede tener varias citas con la barbería en diferentes momentos, pero cada cita solo puede estar asociada a un cliente en particular.	1:N
Cita- servicio	una cita puede incluir la realización de varios servicios por parte del barbero, y un servicio puede ser realizado en varias citas diferentes.	1:N
Servicio-insumo	un servicio puede requerir la utilización de varios insumos y un insumo puede ser utilizado en la prestación de varios servicios diferentes.	N:N
Servicio - empleado	un servicio puede ser asistido por un solo empleado de la barbería, pero un empleado puede asistir varios servicios diferentes.	1:N
Empleado – venta	un empleado puede registrar muchas ventas en un período de tiempo determinado, pero una venta	1:N

	específica solo puede ser registrada por un empleado.	
Venta – cliente	un cliente puede realizar muchas compras en la barbería, pero cada venta registrada en la tabla de registro de Ventas la cual solo puede estar asociada con un cliente en particular	1:N
Venta - producto	un cliente puede realizar muchas compras en la barbería, pero cada venta registrada en la tabla de registro de ventas solo puede estar asociada con un cliente en particular	M:N
Producto - proveedor	un proveedor puede suministrar muchos productos diferentes, pero cada producto solo puede ser suministrado por un proveedor	1:N

De acuerdo con las anteriores descripciones se generó el siguiente diagrama ER

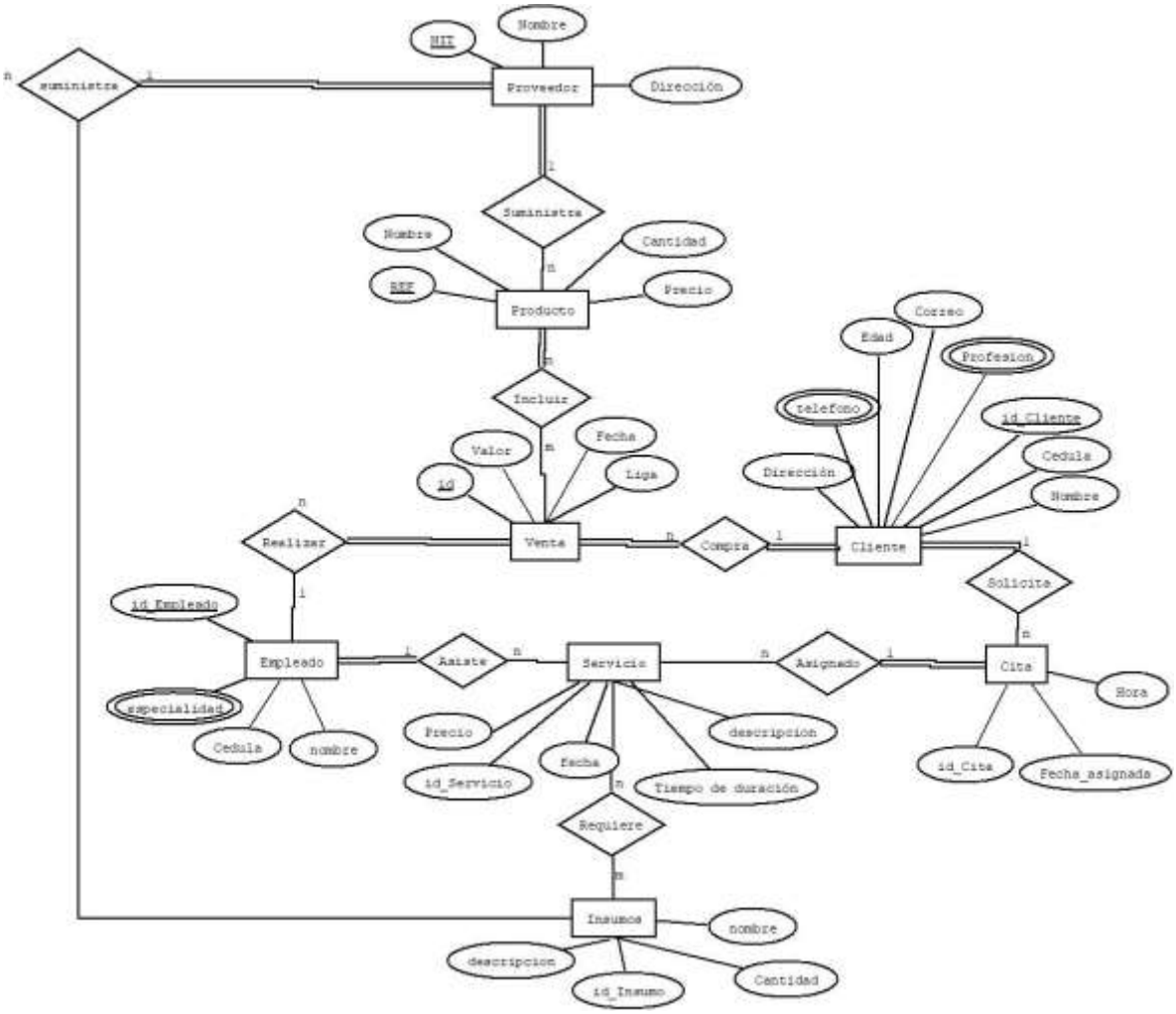

Figura 1: Modelo ER

Realizar el modelo relacional

Transformación del modelo ER al modelo M-R

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

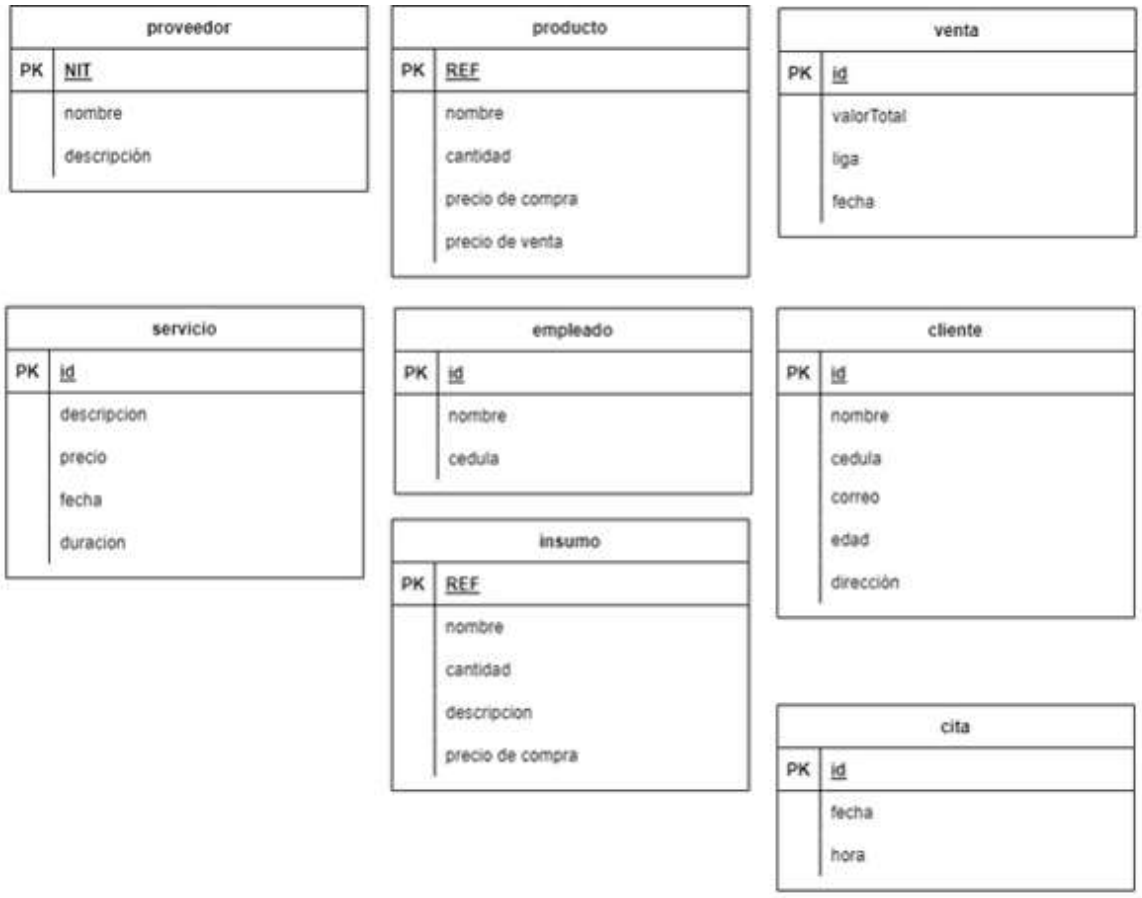

Figura 2: Transformación de entidades a tablas

2. Se transforman atributos multivaluados en tablas.

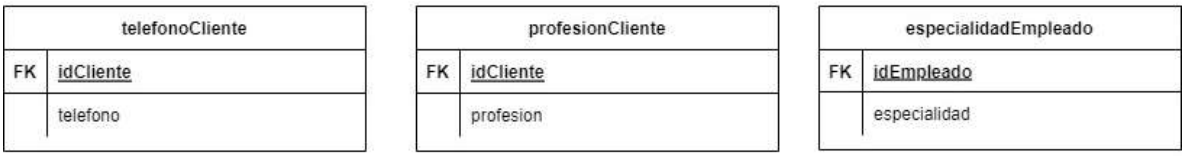

Figura 3: Tablas generadas a partir de atributos multivaluados

3. Las relaciones muchos a muchos se definen tablas intermedias

ventaProducto	
FK	idProducto
FK	idVenta
	cantidadVendida
	precioUnitario

servicioEmpleado	
FK	idServicio
FK	idEmpleado

servicioInsumo	
FK	idServicio
FK	idInsumo
	cantidadUsada

Figura 4: Definición de tablas intermedias.

4. Se establecen relaciones

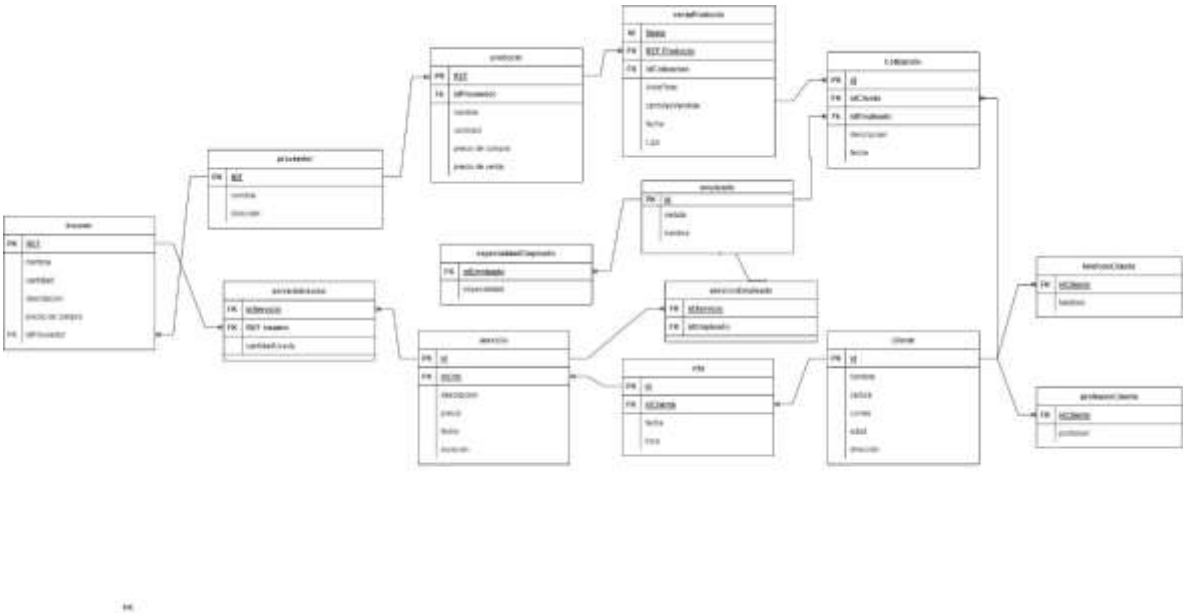

Figura 5: modero MR

Normalizar correctamente

1FN	Se definieron las tablas con atributos atómicos y sin atributos multivaluados para evitar registros duplicados. Las claves primarias se establecieron según se muestra en la figura 2. Asimismo, se crearon tablas para los atributos multivaluados, tal como se presenta en la figura 3.
2FN	Cumple con la primera forma normal, se crea la relación entre tablas con sus respectivas llaves foráneas, es decir con la clave ajena, lo cual garantiza que que cada atributo esté relacionado con la clave primaria completa de su tabla correspondiente para evitar la redundancia de datos
3FN	Para manejar las relaciones de muchos a muchos, se crearon tablas intermedias según se muestra en la figura 4:

- Tabla detalle entre venta, producto
- Tabla detalle entre servicio y empleado.
- Tabla detalle entre servicio e insumo.

Escribir con sentencias SQL toda la definición de la base de datos.

```
CREATE SCHEMA IF NOT EXISTS barberia DEFAULT CHARACTER SET utf8 ;

USE barberia ;

#  Tabla proveedor

CREATE TABLE IF NOT EXISTS proveedor (
  NIT INT NOT NULL,
  nombre VARCHAR(25) NOT NULL,
  direccion VARCHAR(30) NOT NULL,
  PRIMARY KEY (NIT));


# Tabla producto

CREATE TABLE IF NOT EXISTS producto (
  REF INT NOT NULL,
  nombre VARCHAR(45) NOT NULL,
  cantidad INT NOT NULL,
  precioCompra DOUBLE NOT NULL,
  precioVenta DOUBLE NOT NULL,
  NITProveedor INT NOT NULL,
  PRIMARY KEY (REF),
    FOREIGN KEY (NITProveedor) REFERENCES proveedor (NIT));


#Tabla insumo

CREATE TABLE IF NOT EXISTS insumo (
  REF INT NOT NULL,
  nombre VARCHAR(45) NOT NULL,
  cantidad INT NOT NULL,
  precioCompra DOUBLE NOT NULL,
  NITProveedor INT NOT NULL,
  PRIMARY KEY (REF),
    FOREIGN KEY (NITProveedor) REFERENCES proveedor (NIT));
```

```sql
# Tabla cliente

CREATE TABLE IF NOT EXISTS cliente (
  id INT NOT NULL,
  nombre VARCHAR(30) NOT NULL,
  cedula VARCHAR(15) NOT NULL,
  correo VARCHAR(30) NOT NULL,
  edad VARCHAR(3) NOT NULL,
  direccion VARCHAR(50) NOT NULL,
  PRIMARY KEY (id));


# Tabla empleado

CREATE TABLE IF NOT EXISTS empleado (
  id INT NOT NULL,
  cedula VARCHAR(15) NOT NULL,
  nombre VARCHAR(25) NOT NULL,
  PRIMARY KEY (id));


# Tabla cotización

CREATE TABLE IF NOT EXISTS cotizacion (
  id INT NOT NULL,
  descripcion VARCHAR(50) NOT NULL,
  idCliente INT NOT NULL,
  idEmpleado INT NOT NULL,
  PRIMARY KEY (id),
    FOREIGN KEY (idCliente) REFERENCES cliente (id),
    FOREIGN KEY (idEmpleado) REFERENCES empleado (id));

# Tabla venta

CREATE TABLE IF NOT EXISTS venta (
  id INT NOT NULL,
  valorTotal DOUBLE NOT NULL,
  liga DOUBLE NOT NULL,
  fecha DATE NOT NULL,
  idProducto INT NOT NULL,
  idCotizacion INT NOT NULL,
  cantidad INT NOT NULL,
  PRIMARY KEY (id),
    FOREIGN KEY (idProducto) REFERENCES producto (REF),
    FOREIGN KEY (idCotizacion) REFERENCES cotizacion(id));
```

```
# Tabla telefono cliente

CREATE TABLE IF NOT EXISTS telefonoCliente (
  idCliente INT NOT NULL,
  telefono VARCHAR(20) NOT NULL,
    FOREIGN KEY (idCliente) REFERENCES cliente (id));

# Tabla correo cliente

CREATE TABLE IF NOT EXISTS correoCliente (
  idCliente INT NOT NULL,
  correo VARCHAR(30) NOT NULL,
    FOREIGN KEY (idCliente) REFERENCES cliente (id));

# Tabla profesion cliente

CREATE TABLE IF NOT EXISTS profesionCliente(
  idCliente INT NOT NULL,
  profesion VARCHAR(30) NOT NULL,
    FOREIGN KEY (idCliente) REFERENCES cliente (id));

# Tabla especialidad empleado

CREATE TABLE IF NOT EXISTS especialidadEmpleado (
  idEmpleado INT NOT NULL,
  especialidad VARCHAR(30) NOT NULL,
    FOREIGN KEY (idEmpleado) REFERENCES empleado (id));


# Tabla cita

CREATE TABLE IF NOT EXISTS cita (
  id INT NOT NULL,
  idCliente INT NOT NULL,
  fechaSolicitda DATE NOT NULL,
  horaSolicitada VARCHAR(10) NOT NULL,
  PRIMARY KEY (id, idCliente),
    FOREIGN KEY (idCliente) REFERENCES cliente (id));


# Tabla servicio

CREATE TABLE IF NOT EXISTS servicio (
  id INT NOT NULL,
```

```
  idCita INT NOT NULL,
  precio DOUBLE NOT NULL,
  descripción VARCHAR(50) NOT NULL,
  fecha DATE NOT NULL,
  duracion DOUBLE NOT NULL,
  PRIMARY KEY (id),
    FOREIGN KEY (idCita) REFERENCES cita (id));


#Tabla servicio empleado

CREATE TABLE IF NOT EXISTS servicioEmpleado (
  idServicio INT NOT NULL,
  idEmpleado INT NOT NULL,
    FOREIGN KEY (idEmpleado) REFERENCES empleado (id),
    FOREIGN KEY (idServicio) REFERENCES servicio (id));


# Tabla servicio insumo
CREATE TABLE IF NOT EXISTS servicioInsumo (
  idServicio INT NOT NULL,
  REFinsumo INT NOT NULL,
    FOREIGN KEY (idServicio) REFERENCES servicio (id),
    FOREIGN KEY (REFinsumo) REFERENCES insumo (REF));
```

Consultas

CONSULTA 1: obtener el nombre del producto, precio de compra y precio de venta para calcular la ganancia neta por producto

```
SELECT REF, nombre, cantidad, precioCompra, precioVenta, NITProveedor, (precioVenta - precioCompra) AS ganacia_por_producto
FROM producto
ORDER BY precioCompra ASC;
```

REF	nombre	cantidad	precioCompra	precioVenta	NITProveedor	ganacia_por_producto
111	Producto1	5	30000	350000	111	320000
112	Cosmetico2	5	40000	450000	112	410000

CONSULTA 2: Obtener la descripción de la cotización, la fecha de venta y el nombre del producto vendido:

```
SELECT cotizacion.descripcion as descripción_Cotizacion, venta.fecha, producto.nombre
FROM venta
```

```
INNER JOIN producto ON venta.idProducto = producto.REF
INNER JOIN cotizacion ON venta.idCotizacion = cotizacion.id;
```

descripción_Cotizacion	fecha	nombre
descripcion1	2023-02-02	Producto1

```
#CONSULTA 3: la cantidad de ligas realizadas por todos los empleados en determinadas fechas.
SELECT empleado.nombre AS nombre_Empleado, COUNT(venta.liga) as total_ligas
FROM venta
JOIN cotizacion ON venta.idCotizacion = cotizacion.id
JOIN empleado ON cotizacion.idEmpleado = empleado.id
WHERE venta.fecha BETWEEN '2022-01-01' AND '2022-12-31'
GROUP BY empleado.nombre;
```

nombre_Empleado	total_ligas
Diana	1

```
#CONSULTA 4: Obtener los empleados que tienen asignadas citas en un rango de fechas:
SELECT empleado.nombre as nombreEmpleado, COUNT(cita.id) as num_citas
FROM empleado
INNER JOIN servicioEmpleado ON servicioEmpleado.idEmpleado = empleado.id
INNER JOIN servicio ON servicio.id = servicioEmpleado.idServicio
INNER JOIN cita ON cita.id = servicio.idCita
WHERE cita.fechaSolicitda BETWEEN '2023-02-01' AND '2023-02-15'
GROUP BY empleado.nombre;
```

nombreEmpleado	num_citas
Diana	1
July	1

```
# CONSULTA 5: obtiene la lista de clientes con los empleados que atendieron durante la
cotización.
SELECT cliente.nombre as nombre_cliente, empleado.nombre as nombre_empleado,
telefonoCliente.telefono as telefono
FROM cotizacion
JOIN empleado ON cotizacion.idEmpleado = empleado.id
JOIN cliente ON cotizacion.idCliente = cliente.id
JOIN telefonoCliente ON telefonoCliente.idCliente = cliente.id;
```

nombre_cliente	nombre_empleado	telefono
Rosa	Diana	3700000
Ana	July	2340000

```
# CONSULTA 6: SELECT cliente.nombre as nombre_cliente, cita.fechaSolicitda,
servicio.descripción as descripción_Servicio, servicio.fecha as fecha_de_servicio,
insumo.nombre as nombre_insumo, servicioinsumo.REFinsumo
FROM cliente
INNER JOIN cita ON cliente.id = cita.idCliente
INNER JOIN servicio ON cita.id = servicio.idCita
INNER JOIN servicioinsumo ON servicio.id = servicioinsumo.idServicio
INNER JOIN insumo ON servicioinsumo.REFinsumo = insumo.REF;
```

	nombre_cliente	fechaSolicitda	descripción_Servicio	fecha_de_servicio	nombre_insumo	REFinsumo
▶	Rosa	2022-02-02	des1	2022-02-02	Producto3	111
	Ana	2022-02-02	descripcion2	2022-02-02	Cosmetico4	112

#CONSULTA 7: costo generado por el servicio realizado a un cliente

```
SELECT servicio.descripción, servicio.precio, cita.fechaSolicitda, cliente.nombre as cliente,
cliente.correo
FROM servicio
INNER JOIN cita ON servicio.idCita = cita.id
INNER JOIN cliente ON cita.idCliente = cliente.id;
```

	descripción	precio	fechaSolicitda	cliente	correo
▶	des1	640000	2022-02-02	Rosa	r@gmail.com
	descripcion2	64000	2022-02-02	Ana	p@gmail.com

#CONSULTA 8: Obtener la cantidad de citas realizadas por el cliente, incluyendo su profesión ya que la barberia desea premiar al cliente con más citas pero con una temática relacionada a su profesión.

```
SELECT cliente.nombre, profesionCliente.profesion, COUNT(cita.id) AS cantidad_citas
FROM cliente
INNER JOIN profesionCliente ON cliente.id = profesionCliente.idCliente
INNER JOIN cita ON cliente.id = cita.idCliente
WHERE cita.fechaSolicitda = '2022-02-02'
GROUP BY cliente.nombre, profesionCliente.profesion;
```

	nombre	profesion	cantidad_citas
▶	Rosa	Medico	1
	Ana	Artista	1

#CONSULTA 9: obtener productos relacionados con proveedores

```
SELECT proveedor.nombre as nombre_proveedor, producto.nombre as nombre_producto,
producto.precioCompra, producto.precioVenta, producto.cantidad
FROM producto
INNER JOIN proveedor ON producto.NITProveedor = proveedor.NIT;
```

	nombre_proveedor	nombre_producto	precioCompra	precioVenta	cantidad
▶	Productos SA	Producto1	30000	350000	5
	Cosmeticos SA	Cosmetico2	40000	450000	5

#Consulta 10: obtener lista de productos proveídos, con el respectivo precio de compra

```
SELECT proveedor.nombre AS nombre_proveedor, insumo.nombre AS nombre_insumo,
insumo.precioCompra
FROM proveedor
INNER JOIN insumo ON proveedor.NIT = insumo.NITProveedor
GROUP BY proveedor.nombre, insumo.nombre;
```

	nombre_proveedor	nombre_insumo	precioCompra
▶	Productos SA	Producto3	30000
	Cosmeticos SA	Cosmetico4	40000

Vistas

VISTA 1:

```
SELECT cliente.nombre AS nombre_cliente, cita.fechaSolicitda, servicio.descripción AS
descripción_Servicio, servicio.fecha AS fecha_de_servicio, insumo.nombre AS nombre_insumo,
servicioinsumo.REFinsumo
FROM cliente
INNER JOIN cita ON cliente.id = cita.idCliente
INNER JOIN servicio ON cita.id = servicio.idCita
INNER JOIN servicioinsumo ON servicio.id = servicioinsumo.idServicio
INNER JOIN insumo ON servicioinsumo.REFinsumo = insumo.REF;

SELECT * FROM cliente_servicio_insumo;
```

nombre_cliente	fechaSolicitda	descripción_Servicio	fecha_de_servicio	nombre_insumo	REFinsumo
Rosa	2022-02-02	des1	2022-02-02	Producto3	111
Ana	2022-02-02	descripcion2	2022-02-02	Cosmetico4	112

VISTA 2: Cotizaciones que finalizaron con éxito de compra.

```
CREATE VIEW ventaDeProducto AS
SELECT cliente.nombre AS nombre_cliente, cotizacion.id AS id_cotizacion, venta.valorTotal,
    empleado.nombre AS atendido_por
FROM cliente
INNER JOIN cotizacion ON cliente.id = cotizacion.idCliente
INNER JOIN venta ON cotizacion.id = venta.idCotizacion
INNER JOIN empleado ON cotizacion.idEmpleado = empleado.id;

SELECT * FROM ventaDeProducto;
```

nombre_cliente	id_cotizacion	valorTotal	atendido_por
Rosa	10	350000	Diana

VISTA 3: la cantidad de ligas realizadas por los empleados detrmina la catidad de ventas asistidas durante una cotización.

```
CREATE VIEW ventas_realizadas_por_Empleado AS
SELECT empleado.nombre AS nombre_Empleado, COUNT(venta.liga) as total_ligas
FROM venta
JOIN cotizacion ON venta.idCotizacion = cotizacion.id
JOIN empleado ON cotizacion.idEmpleado = empleado.id
WHERE venta.fecha BETWEEN '2022-01-01' AND '2023-12-31'
GROUP BY empleado.nombre;

SELECT * FROM ventas_realizadas_por_Empleado;
```

nombre_Empleado	total_ligas
Diana	1

VISTA 4: Costo generada por un servicio prestado

```
CREATE VIEW costo_servicio_cliente AS
SELECT servicio.id, servicio.descripción, servicio.precio, cita.fechaSolicitda, cliente.nombre as
cliente, cliente.correo
FROM servicio
INNER JOIN cita ON servicio.idCita = cita.id
INNER JOIN cliente ON cita.idCliente = cliente.id;

SELECT * FROM costo_servicio_cliente;
```

id	descripción	precio	fechaSolicitda	cliente	correo
1	des1	640000	2022-02-02	Rosa	r@gmail.com
2	descripcion2	64000	2022-02-02	Ana	p@gmail.com

Generar al menos 4 procedimientos almacenados.

Generar al menos 4 triggers

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

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

documente muy bien su proceso (paso a paso) en un archivo PDF escriba todas las aclaraciónes o especificaciones necesarias para realizar el ejercicio.