



Nombre de la práctica	Creación de las bases de datos en MySQL			No.	
Asignatura:	Taller de bases de datos	Carrera:	Ing. Sistemas computacionales	Duración de la práctica (Hrs)	

I. Competencia(s) específica(s):

II. Lugar de realización de la práctica (laboratorio, taller, aula u otro): Aula

III. Material empleado:

- Laptop
- Navicat

IV. Desarrollo de la práctica:

Problemario DDL y Constraints en MySQL.

Problema 1: Registro de empleados con restricciones salariales

Una empresa quiere guardar los Empleados, no cuentan con un registro como numero de empleados, anteriormente se tenia registro en un Excel con el nombre, email y salario, las reglas de la empresa no permiten un salario mensual menor a \$3000 ni mayor a \$50000. Crea la base de datos llamada Ejercicio_Constraints_1 y la tabla Empleados que cumpla con estos requisitos.

```

1 CREATE TABLE Empleados(
2   id_empleados INT PRIMARY KEY AUTO_INCREMENT,
3   nombre VARCHAR(30) NOT NULL,
4   apellido1 VARCHAR(30) NOT NULL,
5   apellido2 VARCHAR(30) NOT NULL,
6   email VARCHAR(50) NOT NULL CHECK (email LIKE "__@__."),
7   salario_men INT CHECK (salario_men<=3000 & salario_men>=50000)
8 );

```

Problema 2: Relación entre empleados y departamentos

Una empresa necesita organizar a sus empleados según los departamentos a los que pertenecen. Cada departamento tiene un nombre único. La empresa quiere almacenar la información de los empleados junto con el departamento al que están asignados. Actualmente, cada empleado tiene un número de identificación, número de empleado y el nombre del departamento. Se requiere crear una relación entre ambas tablas para que cada empleado esté asignado a un departamento.

Crea la base de datos llamada `Ejercicio_Constraints_2` y las tablas `Empleados` y `Departamentos` con las restricciones necesarias para cumplir con esta relación.



```
1 CREATE TABLE Departamentos (
2     id_departamento INT AUTO_INCREMENT PRIMARY KEY, -- ID único para cada departamento
3     nombre_departamento VARCHAR(100) NOT NULL -- Nombre único del departamento
4 );
5
6 -- Crear la tabla Empleados
7 CREATE TABLE Empleados (
8     numero_empleado INT AUTO_INCREMENT PRIMARY KEY, -- Número único de empleado
9     nombre_empleado VARCHAR(100) NOT NULL, -- Nombre del empleado
10    id_departamento INT, -- ID del departamento al que pertenece el empleado
11    FOREIGN KEY (id_departamento) REFERENCES Departamentos(id_departamento) -- Restricción de clave foránea
12 );
```

Problema 3: Control de inventario de productos

Una empresa quiere llevar el control de los productos que vende. La información almacenada en la tabla de Productos debe incluir el nombre del producto, código de barras y su precio. A su vez, necesitan añadir una columna para el stock de cada producto, la cual no puede ser nula y debe tener un valor por defecto de 100. Además, el precio de los productos debe ser siempre mayor a 0, y el nombre de cada producto debe ser único para evitar duplicados.

Crea la base de datos llamada Ejercicio_Constraints_3 y realiza las modificaciones necesarias en la tabla Productos para cumplir con estos requisitos.

```
Objects | DDL @ejercicio_constraints_1 (mysql80-l... | DDL @ejercicio_constraints_2 (mysql80-l... | DDL @ejercicio_constraints_3 (mysql80-l...
Save | Query Builder | Beautify SQL | Code Snippet
mysql80-localhost | ejercicio_constraints_3 | Run | Stop | Explain
1 CREATE TABLE Productos (
2     id_producto INT AUTO_INCREMENT PRIMARY KEY, -- ID único para cada producto
3     nombre_producto VARCHAR(100) NOT NULL UNIQUE, -- Nombre del producto, obligatorio y único
4     codigo_barras VARCHAR(50) NOT NULL, -- Código de barras del producto
5     precio DECIMAL(10, 2) NOT NULL CHECK (precio > 0), -- Precio del producto, mayor a 0
6     stock INT NOT NULL DEFAULT 100 -- Stock del producto, no nulo, valor por defecto 100
7 );
```

Problema 4: Control de pedidos con validación de montos

Una empresa quiere registrar los pedidos que recibe, pero necesita asegurarse de que el total de cada pedido sea proporcional a la cantidad de productos solicitados. Específicamente, el total de cada pedido debe ser al menos igual a la cantidad de productos multiplicada por 10. Además, cada pedido debe contener al menos 1 producto.

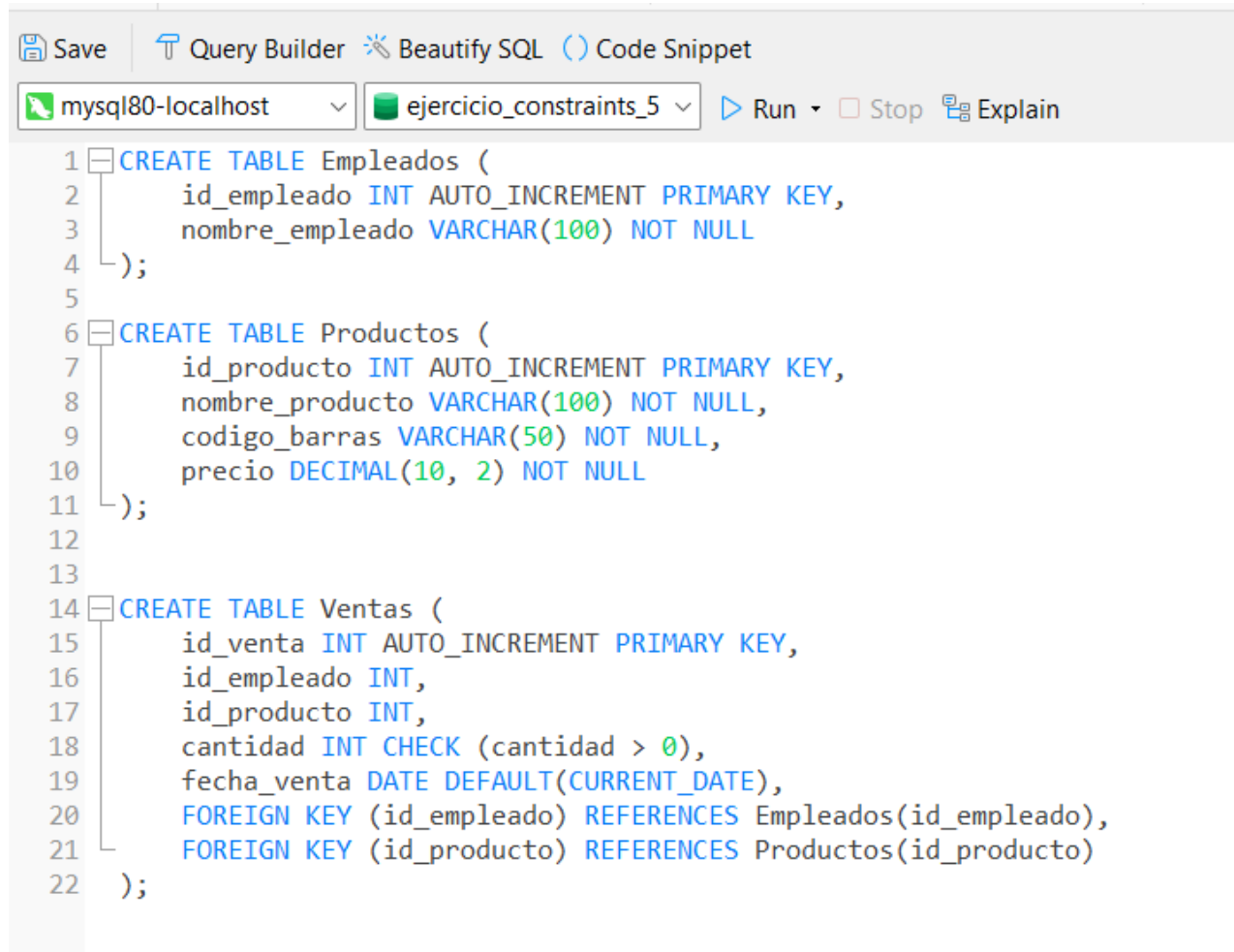
Crea la base de datos llamada Ejercicio_Constraints_4 y la tabla Pedidos que cumpla con estas validaciones usando restricciones CHECK.

```
Save | Query Builder | Beautify SQL | Code Snippet
mysql80-localhost | ejercicio_constraints_4 | Run | Stop | Explain
1 CREATE TABLE Pedidos (
2     id_pedido INT AUTO_INCREMENT PRIMARY KEY,
3     cantidad_productos INT NOT NULL CHECK (cantidad_productos >= 1),
4     total INT NOT NULL,
5     CHECK (total >= cantidad_productos * 10)
6 );
7
```

Problema 5: Control de ventas de productos por empleados

Una empresa de ventas necesita registrar las ventas que realiza. Cada venta está asociada a un empleado y a un producto específico. Para garantizar la integridad de los datos, se requiere que cada venta tenga la referencia tanto del empleado como del producto, y que las ventas sean realizadas en una fecha válida (no futura). Además, la cantidad de productos vendidos debe ser mayor a 0.

Crea la base de datos llamada Ejercicio_Constraints_5 y las tablas necesarias para almacenar esta información, incluyendo las claves foráneas para relacionar las tablas de empleados y productos con las ventas.



```
Save Query Builder Beautify SQL Code Snippet
mysql80-localhost ejercicio_constraints_5 Run Stop Explain

1 CREATE TABLE Empleados (
2     id_empleado INT AUTO_INCREMENT PRIMARY KEY,
3     nombre_empleado VARCHAR(100) NOT NULL
4 );
5
6 CREATE TABLE Productos (
7     id_producto INT AUTO_INCREMENT PRIMARY KEY,
8     nombre_producto VARCHAR(100) NOT NULL,
9     codigo_barras VARCHAR(50) NOT NULL,
10    precio DECIMAL(10, 2) NOT NULL
11 );
12
13
14 CREATE TABLE Ventas (
15     id_venta INT AUTO_INCREMENT PRIMARY KEY,
16     id_empleado INT,
17     id_producto INT,
18     cantidad INT CHECK (cantidad > 0),
19     fecha_venta DATE DEFAULT(CURRENT_DATE),
20     FOREIGN KEY (id_empleado) REFERENCES Empleados(id_empleado),
21     FOREIGN KEY (id_producto) REFERENCES Productos(id_producto)
22 );
```

Modificación del esquema de la base de datos

Realiza los siguientes ejercicios en MySQL

Ejercicio 1

Diseñar y modificar un esquema para una base de datos de biblioteca.

1. Objetivo: Diseñar en un esquema de la base de datos para gestionar la información de una biblioteca.

El sistema debe ser capaz de almacenar datos sobre libros, los clientes y los préstamos realizados.

2. Entidades y Atributos:

Libro:

- id_Libro: Identificador único del libro. título: Título del libro.
 - autor: Autor del libro. año_publicacion:
 - Año de publicación del libro.
 - ISBN: Número estándar internacional del libro
- Cliente:**
- id_Cliente: identificador único del cliente. nombre: Nombre del
 - cliente. direccion:
 - Dirección del cliente.
 - telefono: Número de teléfono del cliente.

Préstamo:

- id_Prestamo: Identificador único del préstamo. id_Libro: Referencia al libro
- prestado. id_Cliente: Referencia al cliente que realiza el préstamo.
- fecha_Prestamo: Fecha en la que se realiza el préstamo.
- fecha_Devolucion: Fecha en la que se espera la devolución del libro.
-

Libro

Prestamo

Cliente

Incluye

Prestamo

CREATE TABLE Libro (

ID_Libro INT PRIMARY KEY,

Titulo VARCHAR(255), Autor

VARCHAR(255),

Año_Publicacion INT,

ISBN VARCHAR(20));

CREATE TABLE Cliente (

ID_Cliente INT PRIMARY KEY,

Nombre VARCHAR(255),

Direccion VARCHAR(255),

Telefono VARCHAR(20));

CREATE TABLE Prestamo (

ID_Prestamo INT PRIMARY KEY,

ID_Libro INT,

ID_Cliente INT,

Fecha_Prestamo DATE,

Fecha_Devolucion DATE,

FOREIGN KEY (ID_Libro) REFERENCES Libro(ID_Libro),

FOREIGN KEY (ID_Cliente) REFERENCES Cliente(ID_Cliente));

3. Normalización y Modificación

-
- Modifica el esquema para cumplir con las 3 formas normales.
Descompón la tabla Libro si el ISBN puede repetirse o si los autores deben estar en una tabla separada.
Normaliza la tabla Cliente si la dirección se repite o en múltiples componentes



Creación de las tablas y definición de los atributos para la base de datos mediante lenguaje SQL.

```
1 CREATE TABLE Libro (  
2   ISBN INT PRIMARY KEY,  
3   Titulo VARCHAR(255),  
4   Año_Publicacion INT  
5 );  
6  
7 CREATE TABLE Autor (  
8   ID_Autor INT PRIMARY KEY,  
9   Nombres VARCHAR(255),  
10  Apellido1 VARCHAR(255),  
11  Apellido2 VARCHAR(255),  
12  Calle VARCHAR(255),  
13  No_Calle VARCHAR(255),  
14  Estado VARCHAR (255)  
15 );  
16  
17 CREATE TABLE Libro_Autor (  
18   ID_LibroAutor INT PRIMARY KEY,  
19   ISBN INT,  
20   ID_Autor INT,  
21   FOREIGN KEY (ISBN) REFERENCES Libro(ISBN),  
22   FOREIGN KEY (ID_Autor) REFERENCES Autor(ID_Autor)  
23 );  
24  
25 CREATE TABLE Cliente (  
26   ID_Cliente INT PRIMARY KEY,  
27   Nombre VARCHAR(255),  
28   Apellido1 VARCHAR(255),  
29   Apellido2 VARCHAR(255),  
30   Telefono VARCHAR(20),  
31   Calle VARCHAR(255),  
32   No_Calle VARCHAR(255),  
33   Estado VARCHAR (255)  
34 );  
35  
36 CREATE TABLE Prestamo (  
37   ID_Prestamo INT PRIMARY KEY,  
38   ID_Cliente INT
```



```
35
36 CREATE TABLE Prestamo (
37     ID_Prestamo INT PRIMARY KEY,
38     ID_Cliente INT,
39     Fecha_Prestamo DATE,
40     Fecha_Devolucion DATE,
41     FOREIGN KEY (ID_Cliente) REFERENCES Cliente(ID_Cliente)
42 );
43
44 CREATE TABLE Libro_Prestamo (
45     ID_LibroPrestamo INT PRIMARY KEY ,
46     ID_Prestamo INT,
47     ISBN INT,
48     FOREIGN KEY (ID_Prestamo) REFERENCES Prestamo(ID_Prestamo),
49     FOREIGN KEY (ISBN) REFERENCES Libro(ISBN)
50 );
51
```

Ejercicio 2

Diseñar y modificar un esquema para un sistema de ventas.

1. Objetivo: Crear y modificar un esquema de base de datos para un sistema de ventas. El sistema debe gestionar la información de productos, clientes, ventas y los detalles de cada venta.

2. Entidades y Atributos:

Producto:

- id_producto: Identificador único del producto. nombre:
- Nombre del producto. precio: Precio del producto.
- categoria: Categoría a la que pertenece el producto.

Cliente:

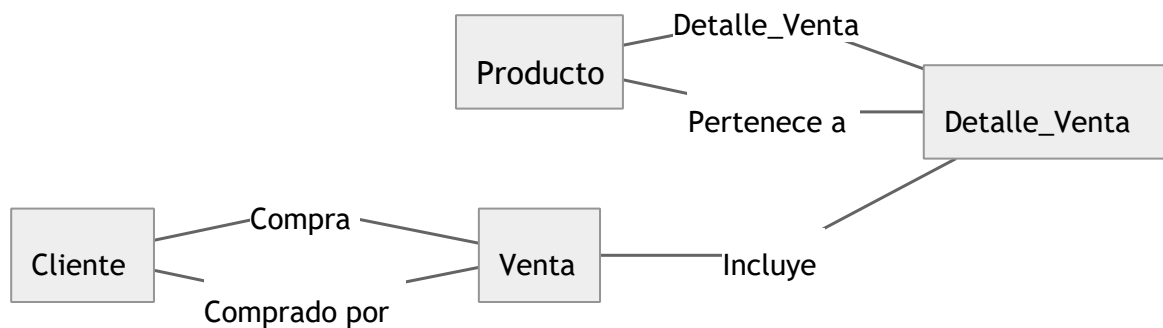
- id_cliente: Identificador único del cliente. nombre:
- Nombre del Cliente.
- email: Correo electrónico del cliente.

Venta:

- id_venta: Identificador único de la venta. id_cliente:
- Referencia al cliente que realiza la venta.
- fecha_venta: Fecha en la que se realizó la venta.

Detalle_venta:

- id_detalle: Identificador único del detalle de la venta. id_venta: Referencia a la
- venta a la que pertenece el detalle. id_producto: Referencia al producto
- vendido. cantidad: Cantidad del producto vendido en esta venta.
-



```
CREATE TABLE Producto (
    ID_Producto INT PRIMARY KEY,
    Nombre VARCHAR(255),
    Precio DECIMAL(10, 2), Categoria VARCHAR(100)
);
```

```
CREATE TABLE Cliente (
    ID_Cliente INT PRIMARY KEY,
    Nombre VARCHAR(255),
    Email VARCHAR(255) );
```

```
CREATE TABLE Venta (
    ID_Venta INT PRIMARY KEY,
    ID_Cliente INT,
    Fecha_Venta DATE,
    FOREIGN KEY (ID_Cliente) REFERENCES Cliente(ID_Cliente) );
```

```
CREATE TABLE Detalle_Venta (
    ID_Detalle INT PRIMARY KEY,
```

ID_Venta INT,
ID_Producto INT,
Cantidad INT,
FOREIGN KEY (ID_Venta) REFERENCES Venta(ID_Venta),
FOREIGN KEY (ID_Producto) REFERENCES Producto(ID_Producto));

3. Normalización y Modificación:

-
- Normaliza las tablas si hay redundancia en la Venta o en Detalle_Venta .
Asegúrate de que la tabla Producto no contenga valores redundantes para la categoría.

Creación de las tablas y definición de los atributos para la base de datos mediante lenguaje SQL.

```
1 CREATE TABLE producto (  
2   id_producto INT PRIMARY KEY AUTO_INCREMENT,  
3   nombre VARCHAR (30) NOT NULL,  
4   precio DECIMAL (10, 2) NOT NULL CHECK (precio>0),  
5   id_categoria INT,  
6   FOREIGN KEY (id_categoria) REFERENCES categoria(id_categoria) ON UPDATE CASCADE ON DELETE RESTRICT  
7 );  
8  
9 CREATE TABLE categoria (  
10  id_categoria INT PRIMARY KEY AUTO_INCREMENT,  
11  nombre VARCHAR (38) NOT NULL  
12 );  
13  
14  
15 CREATE TABLE cliente (  
16  id_cliente INT PRIMARY KEY AUTO_INCREMENT,  
17  nombre VARCHAR(30) NOT NULL,  
18  apellido1 VARCHAR (30) NOT NULL,  
19  apellido2 VARCHAR (38) NOT NULL,  
20  calle VARCHAR(100) NOT NULL,  
21  localidad VARCHAR(100) NOT NULL,  
22  municipio VARCHAR(188) NOT NULL,  
23  codigo_postal VARCHAR (10) NOT NULL,  
24  estado VARCHAR(100) NOT NULL,  
25  pais VARCHAR(108) NOT NULL,  
26  telefono VARCHAR(20) NOT NULL  
27 );  
28  
29  
30 CREATE TABLE venta(  
31  id_venta INT PRIMARY KEY AUTO_INCREMENT,  
32  fecha DATE NOT NULL,  
33  id_cliente INT,  
34  FOREIGN KEY (id_cliente) REFERENCES cliente(id_cliente) ON UPDATE CASCADE ON DELETE RESTRICT  
35 );
```



```
36
37 CREATE TABLE detalle_venta (
38   id_detalle INT PRIMARY KEY AUTO_INCREMENT,
39   cantidad INT NOT NULL CHECK (cantidad>0),
40   id_venta INT NOT NULL,
41   id_producto INT NOT NULL,
42   FOREIGN KEY (id_venta) REFERENCES venta(id_venta),
43   FOREIGN KEY (id_producto) REFERENCES producto (id_producto)
44 );
```

Ejercicio 3

Diseñar y modificar un esquema para un sistema hospitalario.

1. Objetivo: Crear y modificar un esquema de base de datos para un sistema de hospital. El sistema debe gestionar la información de pacientes, médicos y citas.

2. Entidades y Atributos:

Paciente:

id_paciente: Identificador único del paciente.

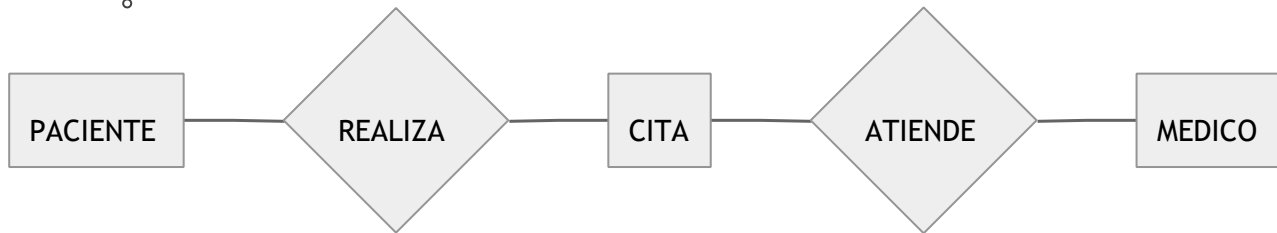
- nombre: Nombre del paciente. fecha_nacimiento: Fecha de
- nacimiento del paciente. sexo: Sexo del paciente.
-

Médico:

- id_medico: Identificador único del médico.
- nombre: Nombre del médico. especialidad:
- Especialidad del médico.

Cita:

- id_cita: Identificador único de la cita. id_paciente:
- Referencia al paciente que tiene la cita. id_medico:
- Referencia al médico que atiende la cita. fecha_cita:
- Fecha de la cita. hora_cita: Hora de la cita.
-



```
CREATE TABLE Paciente (  
  ID_Paciente INT PRIMARY KEY,  
  Nombre VARCHAR(255),  
  Fecha_Nacimiento DATE,  
  Sexo CHAR(1) );
```

```
CREATE TABLE Medico (  
  ID_Medico INT PRIMARY KEY,  
  Nombre VARCHAR(255),  
  Especialidad VARCHAR(100) );
```

```
CREATE TABLE Cita (  
  ID_Cita INT PRIMARY KEY,  
  ID_Paciente INT,  
  ID_Medico INT,  
  Fecha_Cita DATE,  
  Hora_Cita TIME,  
  FOREIGN KEY (ID_Paciente) REFERENCES Paciente(ID_Paciente),  
  FOREIGN KEY (ID_Medico) REFERENCES Medico(ID_Medico) );
```

3. Normalización y Modificación



- Revisa si hay redundancia en la tabla Cita o en la información del Medico . Considera agregar una tabla de Especialidad si hay múltiples médicos con la misma especialidad.

Creación de las tablas y definición de los atributos para la base de datos mediante lenguaje SQL.

```
1 CREATE TABLE paciente (  
2   id_paciente INT PRIMARY KEY AUTO_INCREMENT,  
3   nombre VARCHAR(30) NOT NULL,  
4   apellido1 VARCHAR (30) NOT NULL,  
5   apellido2 VARCHAR (30) NOT NULL,  
6   fecha_nac DATE NOT NULL,  
7   sexo ENUM('Masculino', 'Femenino', 'Otro', 'No especificado') NOT NULL  
8 );  
9  
10 CREATE TABLE Cita (  
11   id_cita INT PRIMARY KEY AUTO_INCREMENT,  
12   id_paciente INT,  
13   id_medico INT,  
14   fecha_cita DATE NOT NULL,  
15   hora TIME NOT NULL,  
16   FOREIGN KEY (id_paciente) REFERENCES paciente(id_paciente),  
17   FOREIGN KEY (id_medico) REFERENCES medico(id_medico)  
18 );  
19  
20 CREATE TABLE medico(  
21   id_medico INT PRIMARY KEY AUTO_INCREMENT,  
22   nombre VARCHAR(38) NOT NULL,  
23   apellido1 VARCHAR (30) NOT NULL,  
24   apellido2 VARCHAR (30) NOT NULL  
25 );  
26  
27 CREATE TABLE especialidad(  
28   id_especialidad INT PRIMARY KEY AUTO_INCREMENT,  
29   nombre VARCHAR(30) NOT NULL  
30 );  
31  
32 CREATE TABLE medico_especialidad (  
33   id_medico INT,  
34   id_especialidad INT,  
35   PRIMARY KEY (id_medico, id_especialidad),  
36   FOREIGN KEY (id_medico) REFERENCES medico(id_medico) ON UPDATE CASCADE ON DELETE RESTRICT,  
37   FOREIGN KEY (id_especialidad) REFERENCES especialidad(id_especialidad) ON UPDATE CASCADE ON DELETE RESTRICT  
38 );
```

Nota: Puedes agregar más tablas si es que al normalizarlas te es necesario, así como agregar atributos que veas necesarios sin afectar las necesidades iniciales.



V. Conclusiones:

El uso de **MySQL** en **Navicat** simplifica notablemente la administración y gestión de bases de datos, ya que esta interfaz gráfica facilita la creación, modificación y consulta de tablas y datos sin necesidad de utilizar exclusivamente comandos SQL. Navicat ofrece una experiencia más visual e intuitiva, lo cual es beneficioso tanto para principiantes como para usuarios avanzados.

La creación de bases de datos y tablas en Navicat es sencilla y permite la inclusión de **constraints** (restricciones) de forma eficiente. Los **constraints** juegan un papel fundamental en la integridad de los datos, garantizando que las reglas predefinidas se respeten, como la unicidad de datos, las claves primarias (PRIMARY KEY), las relaciones entre tablas a través de claves foráneas (FOREIGN KEY), y restricciones de valores con CHECK.

El uso de **restricciones** dentro de las bases de datos es esencial para garantizar la consistencia y la validez de los datos almacenados, asegurando que las entradas de datos sean precisas y confiables. Las restricciones permiten controlar el comportamiento de los datos, evitar duplicados, mantener la integridad referencial entre tablas y asegurar que los valores introducidos cumplan con ciertos criterios lógicos.