Documentación del Proyecto: Consultorio Medico

INSTITUTO DE ENSEÑANZA SUPERIOR "ALFREDO COVIELLO"

Base de Datos I

Prof. Ing. Carla Aria Acuña

Alumnos: Claudio Ezequiel Gamarro, Diaz Emilse Salome, Montenegro Hector Rober

---

Índice

1. Resumen ejecutivo
2. Alcance y objetivos
3. Modelo conceptual (ER) — descripción general
4. Diccionario de datos (tabla por tabla)
5. Script SQL (estructura y datos de ejemplo)
6. Consultas de ejemplo (reportes y búsquedas comunes)
7. Manual de instalación (MySQL Workbench / phpMyAdmin)
8. Backup, exportación y restauración
9. Seguridad, buenas prácticas y recomendaciones
10. Índices y optimización
11. Mantenimiento y futuras mejoras
12. Anexos: cambios y notas del desarrollo
13. Conclusión

---

1. Resumen ejecutivo

Este documento explica en detalle el diseño, la estructura, la implementación y las instrucciones de uso de la base de datos **consultorioMedico**, pensada para un consultorio de la especialidad **Cardiología**. Incluye el modelo de datos, las tablas, relaciones, ejemplos de datos, consultas útiles, pasos para instalar la base en MySQL y recomendaciones de seguridad y mantenimiento.

---

2. Alcance y objetivos

Alcance:

- Registro y gestión de pacientes, médicos y turnos.
- Historial médico detallado mediante fichas y consultas.
- Gestión de recetas y estudios asociados a cada consulta.
- Control de facturación por turno (particular u obra social).
- Gestión básica de usuarios y roles para el acceso al sistema.

Objetivos:

- Garantizar integridad referencial entre entidades.
- Mantener historial clínico completo (no sobrescribir registros).
- Facilitar consultas y reportes típicos del consultorio.
- Ofrecer una base extensible y escalable.

---

3. Modelo conceptual (ER) — descripción general

Entidades principales y relaciones:

👤 Paciente — 🗓️ Turno

Relación: `paciente solicita turno`
Cardinalidad: 1 paciente ⟶ N turnos
Descripción:
Cada paciente puede solicitar varios turnos para diferentes fechas o doctores.
Cada turno pertenece únicamente a un paciente.

👨‍⚕️ Doctor — 🗓️ Turno

Relación: `doctor atiende turno`
Cardinalidad: 1 doctor ⟶ N turnos
Descripción:
Un doctor puede atender varios turnos a lo largo del tiempo.
Cada turno está asignado a un único doctor.

🗓️ Turno — 💬 Consulta Médica

Relación: `turno genera consulta médica`
Cardinalidad: 1 turno ⟷ 1 consulta médica
Descripción:
Cada turno, al concretarse, da lugar a una consulta médica.
Una consulta solo puede originarse de un turno.

💬 Consulta Médica — 💊 Receta Médica

Relación: `consulta médica produce receta médica`
Cardinalidad: 1 consulta ⟶ N recetas
Descripción:
Durante una consulta, el médico puede emitir una o más recetas.
Cada receta pertenece a una única consulta.

💬 Consulta Médica — 🧪 Estudio Médico

Relación: `consulta médica solicita estudio médico`
Cardinalidad: 1 consulta ⟶ N estudios
Descripción:
El médico puede indicar uno o varios estudios médicos (análisis, electrocardiograma, etc.) derivados de una consulta.

👨‍⚕️ Doctor — 📋 Ficha Médica

Relación: `doctor gestiona ficha médica`
Cardinalidad: 1 doctor ⟶ N fichas médicas
Descripción:
Un doctor puede llevar el seguimiento de varios pacientes a través de sus fichas médicas. Cada ficha médica tiene un único doctor responsable.

👤 Paciente — 📋 Ficha Médica

Relación: `paciente posee ficha médica`
Cardinalidad: 1 paciente ⟷ 1 ficha médica
Descripción:
Cada paciente tiene una ficha médica única donde se registran sus antecedentes, consultas y tratamientos.

📋 Ficha Médica — 💬 Consulta Médica

Relación: `ficha médica registra consulta médica`
Cardinalidad: 1 ficha ⟶ N consultas
Descripción:
La ficha médica de un paciente almacena todas las consultas médicas realizadas.

🗓️ Turno — 💰 Facturación

Relación: `turno genera factura`
Cardinalidad: 1 turno ⟷ 1 factura
Descripción:
Cada atención (turno cumplido) puede generar una factura asociada al pago o cobertura de la consulta.

💰 Facturación — 🏥 Obra Social

Relación: `factura utiliza obra social`
Cardinalidad: N facturas ⟶ 1 obra social
Descripción:
Las facturas pueden estar asociadas a una obra social que cubre los costos del paciente.

👤 Paciente — 🏥 Obra Social

Relación: `paciente está afiliado a obra social`
Cardinalidad: N pacientes ⟶ 1 obra social
Descripción:
Una obra social puede cubrir a varios pacientes.
Cada paciente, en este modelo, puede tener una sola obra social activa.

💬 Consulta Médica — 👨‍⚕️ Doctor

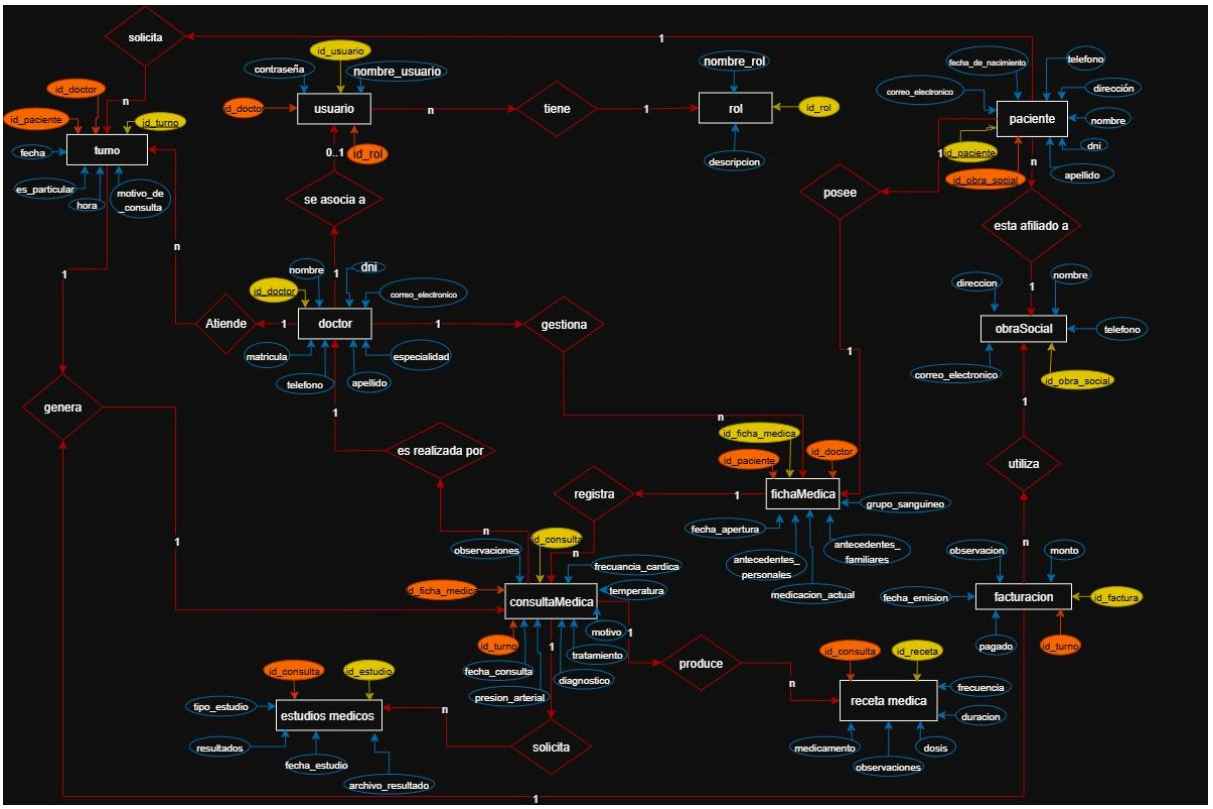
Relación: `consulta médica es realizada por doctor`
Cardinalidad: N consultas ⟶ 1 doctor
Descripción:
Cada consulta médica es atendida por un doctor, pero un doctor puede realizar muchas consultas.

👨‍💻 Usuario — 👩‍💼 Rol

Relación: `usuario tiene rol`
Cardinalidad: N usuarios ⟶ 1 rol
Descripción:
Cada usuario del sistema (secretaria, cardiólogo, administrador) posee un rol que define sus permisos.
Un mismo rol puede estar asignado a muchos usuarios.

👨‍⚕️ Doctor — 👨‍💻 Usuario

Relación: `doctor se asocia a usuario`
Cardinalidad: 1 doctor ⟶ 0..1 usuario
Descripción:
Un doctor puede tener un usuario del sistema para acceder y registrar sus consultas, pero no todos los doctores necesariamente tienen acceso directo al sistema.



---

4. Diccionario de datos

Tabla: Doctor

Propósito: Almacena los datos de los médicos del consultorio.
Dado que el consultorio es de especialidad única (Cardiología), todos los doctores pertenecen a esta especialidad.

Campo	Tipo de dato	Nulo	Clave	Descripción
id_doctor	int	NO	PK	Identificador Único del médico.
apellido	VARCHAR(50)	NO		Apellido del médico.
nombre	VARCHAR(50)	NO		Nombre del médico.
dni	VARCHAR(15)	NO		Documento de identidad del médico.
matricula	VARCHAR(20)	NO	UNIQUE	Matrícula profesional del médico.
teléfono	VARCHAR(20)	SI	UNIQUE	Número de contacto del médico.

Campo	Tipo de dato	Nulo	Clave	Descripción
correo_electronico	VARCHAR(100)	SI		Correo electrónico profesional.
especialidad	VARCHAR(50)	SI		Especialidad médica. (Cardiología)

Tabla: Paciente

Propósito: Contiene los datos personales de los pacientes y su vínculo con una obra social, si corresponde.

Campo	Tipo de dato	Nulo	Clave	Descripción
Id_paciente	INT	NO	PK	Identificador único del paciente.
apellido	VARCHAR(50)	NO		Apellido del paciente.
nombre	VARCHAR(50)	NO		Nombre del paciente.
dni	VARCHAR(15)	NO	UNIQUE	Documento de identidad del paciente.
telefono	VARCHAR(20)	SI		Teléfono de contacto.
correo_electronico	VARCHAR(100)	SI		Correo electrónico del paciente.
fecha_de_nacimiento	DATE	SI		Fecha de nacimiento.
direccion	VARCHAR(100)	SI		Dirección del domicilio.
id_obra_social	INT	SI	FK	Identificador de la obra social.

Tabla: ObraSocial

Propósito: Registra las diferentes obras sociales o coberturas médicas que pueden tener los pacientes.

Campo	Tipo de dato	Nulo	Clave	Descripción
id_obra_social	INT	NO	PK	Identificador único de la obra social.
nombre	VARCHAR(100)	NO		Nombre de la obra social.
direccion	VARCHAR(100)	SI		Dirección de la sede o sucursal.
telefono	VARCHAR(100)	SI		Teléfono de contacto.

correo_electronico	VARCHAR(100)	SI		Correo electrónico institucional.

Tabla: Turno

Propósito: Almacena los turnos médicos entre pacientes y doctores.

Campo	Tipo de dato	Nulo	Clave	Descripción
id_turno	INT	NO	PK	Identificador único del turno.
id_paciente	INT	NO	FK	Relación con el paciente que asiste.
id_doctor	INT	NO	FK	Relación con el medico que atiende.
fecha	DATE	NO		Fecha en la que se realiza el turno.
hora	TIME	NO		Hora del turno.
motivo_de_consulta	TEXT	SI		Motivo por el cual se solicita el turno.
es_particular	BOOLEAN	SI		Indica si el turno es particular(**TRUE**) o por obra social (**FALSE**).

Tabla: Facturacion

Propósito: Registra la facturación o cobro de los turnos.

Campo	Tipo de dato	Nulo	Clave	Descripción
id_factura	INT	NO	PK	Identificador único de la factura
id_turno	INT	NO	FK	Turno al que pertenece la facturación.
fecha_emision	DATE	NO		Fecha en la que se emite la factura.
monto	DECIMAL	NO		Importe total del turno.
observacion	TEXT	SI		Observaciones sobre el pago o cobertura

pagado	BOOLEAN	SI		Indica si el turno fue abonado (**TRUE** o **FALSE**).

Tabla: FichaMedica

Propósito: Guarda la información médica general y permanente del paciente.

Campo	Tipo de dato	Nulo	Clave	Descripción
id_ficha_medica	INT	NO	PK	Identificador único de la ficha medica.
id_paciente	INT	NO	FK	Paciente al que pertenece la ficha.
id_doctor	INT	SI	FK	Médico responsable de la ficha.
fecha_apertura	DATE	NO		Fecha de creación de la ficha
grupo_sanguineo	VARCHAR(50)	SI		Tipo de sangre (por ejemplo,"O+")
alergias	TEXT	SI		Alergias conocidas.
antecedentes_personales	TEXT	SI		Enfermedades previas o crónicas.
antecedentes_familiares	TEXT	SI		Antecedentes hereditarios.
medicacion_actual	TEXT	SI		Medicamentos que el paciente consume actualmente.

Tabla: ConsultaMedica

Propósito: Registra cada atención o consulta realizada, vinculada a una ficha médica y un turno.

Campo	Tipo de dato	Nulo	Clave	Descripción
id_consulta	INT	NO	PK	Identificador único de la consulta.
id_ficha_medica	INT	NO	FK	Ficha médica a la que pertenece.
id_turno	INT	SI	FK	Turno asociado (si corresponde).
fecha_consulta	DATE	NO		Fecha en la que se realiza la atención.

motivo	TEXT	SI		Motivo de la consulta.
diagnostico	TEXT	SI		Diagnóstico realizado.
tratamiento	TEXT	SI		Tratamiento indicado.
presion_arterial	VARCHAR(10)	SI		Valor de presión arterial.
frecuancia_cardiaca	INT	SI		Frecuencia cardiaca en latidos por minuto.
temperatura	DECIMAL	SI		Temperatura corporal del paciente.
observaciones	TEXT	SI		Notas adicionales del médico.

Tabla: RecetaMedica

Propósito: Guarda los medicamentos indicados en cada consulta médica.

Campo	Tipo de dato	Nulo	Clave	Descripción
id_receta	INT	NO	PK	Identificador de la receta.
id_consulta	INT	NO	FK	Consulta medica asociada.
medicamento	VARCHAR(100)	NO		Nombre del medicamento prescrito
dosis	VARCHAR(50)	SI		Cantidad o presentación (ej. “1 comprimido”).
frecuencia	VARCHAR(50)	SI		Frecuencia de consumo (ej. “cada 8 horas”)
duracion	VARCHAR(50)	SI		Tiempo de tratamiento.
observaciones	TEXT	SI		Instrucciones o aclaraciones adicionales.

Tabla: EstudioMedico

Propósito: Contiene los exámenes y análisis médicos realizados al paciente.

Campo	Tipo de dato	Nulo	Clave	Descripción
id_estudio	INT	NO	PK	Identificador único del estudio.
id_consulta	INT	NO	FK	Consulta en la que se ordenó el estudio.

tipo_estudio	VARCHAR(100)	NO		Nombre del estudio(ej. “Electrocardiograma”).
fecha_estudio	DATE	SI		Fecha del estudio.
resultados	TEXT	SI		Resultados del estudio.
archivo_resultado	VARCHAR(255)	SI		Ruta o nombre del archivo del estudio escaneado.

Tabla: Rol

Propósito: Define los diferentes niveles de acceso o funciones del sistema.

Campo	Tipo de dato	Nulo	Clave	Descripción
id_rol	INT	NO	PK	Identificador único del rol.
nombre_rol	VARCHAR(50)	NO		Nombre del rol(Administrador, Secretaria, Cardiólogo).
descripcion	TEXT	SI		Descripción de las funciones o permisos del rol.

Tabla: Usuario

Propósito: Contiene las credenciales y permisos de acceso de los usuarios del sistema.

Campo	Tipo de dato	Nulo	Clave	Descripción
id_usuario	INT	NO	PK	Identificador único del usuario.
nombre_usuario	VARCHAR(50)	NO	UNIQUE	Nombre de usuario para login.
contrasena	VARCHAR(255)	NO		Contraseña.
id_rol	INT	NO	FK	Rol asignado al usuario.
id_doctor	INT	SI	FK	Doctor vinculado.

5. Script SQL

```
CREATE DATABASE consultorioMedico;

USE consultorioMedico;


CREATE TABLE Doctor (
    id_doctor INT PRIMARY KEY AUTO_INCREMENT,
    apellido VARCHAR(50) NOT NULL,
    nombre VARCHAR(50) NOT NULL,
    dni VARCHAR(15) UNIQUE NOT NULL,
    matricula VARCHAR(20) UNIQUE NOT NULL,
    telefono VARCHAR(20),
    correo_electronico VARCHAR(100),
    especialidad VARCHAR(50) DEFAULT 'Cardiología'
);


INSERT INTO Doctor (apellido, nombre, dni, matricula, telefono, correo_electronico)
VALUES ('Montenegro', 'Rober', '38293043', '18231', '3812498238',
'rober.montenegro@gmail.com');


CREATE TABLE ObraSocial (
    id_obra_social INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    direccion VARCHAR(100),
    telefono VARCHAR(20),
    email VARCHAR(100)
);
```

```
INSERT INTO ObraSocial (nombre, direccion, telefono, email)
VALUES ('OSDE', 'Av. Belgrano 1500', '3815556677', 'contacto@osde.com.ar');


CREATE TABLE Paciente (
    id_paciente INT PRIMARY KEY AUTO_INCREMENT,
    apellido VARCHAR(50) NOT NULL,
    nombre VARCHAR(50) NOT NULL,
    dni VARCHAR(15) UNIQUE NOT NULL,
    telefono VARCHAR(20),
    correo_electronico VARCHAR(100),
    fecha_de_nacimiento DATE,
    direccion VARCHAR(100),
    id_obra_social INT NULL,
    CONSTRAINT fk_paciente_obra_social FOREIGN KEY (id_obra_social)
        REFERENCES ObraSocial(id_obra_social)
);


INSERT INTO Paciente (apellido, nombre, dni, telefono, correo_electronico,
fecha_de_nacimiento, direccion, id_obra_social)
VALUES
('Lupi', 'Sebastián', '381293043', '3812311233', 'sebastian.lupi@gmail.com', '1993-04-28',
'Villa Amalia', 1),
('Gómez', 'Ana', '38900123', '3815558877', 'ana.gomez@gmail.com', '1988-12-15', 'San
Martín 245', NULL);


CREATE TABLE Turno (
    id_turno INT PRIMARY KEY AUTO_INCREMENT,
```

```
    id_paciente INT NOT NULL,
    id_doctor INT NOT NULL,
    fecha DATE NOT NULL,
    hora TIME NOT NULL,
    motivo_de_consulta TEXT,
    es_particular BOOLEAN DEFAULT TRUE,
    FOREIGN KEY (id_paciente) REFERENCES Paciente(id_paciente),
    FOREIGN KEY (id_doctor) REFERENCES Doctor(id_doctor)
);

INSERT INTO Turno (id_paciente, id_doctor, fecha, hora, motivo_de_consulta,
es_particular)
VALUES
(1, 1, '2025-10-11', '09:57:00', 'Chequeo de presión arterial', FALSE),
(2, 1, '2025-10-12', '10:15:00', 'Control de rutina', TRUE);

CREATE TABLE Facturacion (
    id_factura INT PRIMARY KEY AUTO_INCREMENT,
    id_turno INT NOT NULL,
    fecha_emision DATE NOT NULL,
    monto DECIMAL(10,2) NOT NULL,
    observacion TEXT,
    pagado BOOLEAN DEFAULT TRUE,
    FOREIGN KEY (id_turno) REFERENCES Turno(id_turno)
);

INSERT INTO Facturacion (id_turno, fecha_emision, monto, observacion, pagado)
```

```
VALUES
(1, '2025-10-11', 0.00, 'Cubierto por OSDE', TRUE),
(2, '2025-10-12', 15000.00, 'Pago particular en efectivo', TRUE);

CREATE TABLE FichaMedica (
    id_ficha_medica INT PRIMARY KEY AUTO_INCREMENT,
    id_paciente INT NOT NULL,
    id_doctor INT,
    fecha_apertura DATE NOT NULL,
    grupo_sanguineo VARCHAR(5),
    alergias TEXT,
    antecedentes_personales TEXT,
    antecedentes_familiares TEXT,
    medicacion_actual TEXT,
    CONSTRAINT fk_ficha_paciente FOREIGN KEY (id_paciente) REFERENCES
Paciente(id_paciente),
    CONSTRAINT fk_ficha_doctor FOREIGN KEY (id_doctor) REFERENCES
Doctor(id_doctor)
);

INSERT INTO FichaMedica (id_paciente, id_doctor, fecha_apertura, grupo_sanguineo,
alergias, antecedentes_personales)
VALUES
(1, 1, '2025-10-01', 'O+', 'Penicilina', 'Hipertensión controlada');

CREATE TABLE ConsultaMedica (
    id_consulta INT PRIMARY KEY AUTO_INCREMENT,
    id_ficha_medica INT NOT NULL,
```

```
    id_turno INT,
    fecha_consulta DATE NOT NULL,
    motivo TEXT,
    diagnostico TEXT,
    tratamiento TEXT,
    presion_arterial VARCHAR(10),
    frecuencia_cardiaca INT,
    temperatura DECIMAL(4,1),
    observaciones TEXT,
    FOREIGN KEY (id_ficha_medica) REFERENCES FichaMedica(id_ficha_medica),
    FOREIGN KEY (id_turno) REFERENCES Turno(id_turno)
);


INSERT INTO ConsultaMedica (id_ficha_medica, id_turno, fecha_consulta, motivo,
diagnostico, tratamiento, presion_arterial, frecuencia_cardiaca)
VALUES
(1, 1, '2025-10-11', 'Chequeo general', 'Presión normal', 'Control anual', '120/80', 75);


CREATE TABLE RecetaMedica (
    id_receta INT PRIMARY KEY AUTO_INCREMENT,
    id_consulta INT NOT NULL,
    medicamento VARCHAR(100) NOT NULL,
    dosis VARCHAR(50),
    frecuencia VARCHAR(50),
    duracion VARCHAR(50),
    observaciones TEXT,
    FOREIGN KEY (id_consulta) REFERENCES ConsultaMedica(id_consulta)
```

```
);


INSERT INTO RecetaMedica (id_consulta, medicamento, dosis, frecuencia, duracion,
observaciones)

VALUES

(1, 'Aspirina 100mg', '1 comprimido', '1 vez al día', '30 días', 'Tomar después de las
comidas');

CREATE TABLE EstudioMedico (

    id_estudio INT PRIMARY KEY AUTO_INCREMENT,

    id_consulta INT NOT NULL,

    tipo_estudio VARCHAR(100) NOT NULL,

    fecha_estudio DATE,

    resultados TEXT,

    archivo_resultado VARCHAR(255),

    FOREIGN KEY (id_consulta) REFERENCES ConsultaMedica(id_consulta)

);


INSERT INTO EstudioMedico (id_consulta, tipo_estudio, fecha_estudio, resultados)

VALUES

(1, 'Electrocardiograma', '2025-10-15', 'Sin alteraciones significativas');


CREATE TABLE Rol (

    id_rol INT PRIMARY KEY AUTO_INCREMENT,

    nombre_rol VARCHAR(50) NOT NULL,

    descripcion TEXT

);
```

```
INSERT INTO Rol (nombre_rol, descripcion)
VALUES
('Administrador', 'Acceso total al sistema'),
('Secretaria', 'Gestión de turnos y pacientes'),
('Cardiólogo', 'Acceso a fichas médicas y consultas');

CREATE TABLE Usuario (
    id_usuario INT PRIMARY KEY AUTO_INCREMENT,
    nombre_usuario VARCHAR(50) UNIQUE NOT NULL,
    contrasena VARCHAR(255) NOT NULL,
    id_rol INT NOT NULL,
    id_doctor INT NULL,
    FOREIGN KEY (id_rol) REFERENCES Rol(id_rol),
    FOREIGN KEY (id_doctor) REFERENCES Doctor(id_doctor)
);

INSERT INTO Usuario (nombre_usuario, contrasena, id_rol, id_doctor)
VALUES
('admin', 'admin123', 1, NULL),
('recepcion', 'recepcion123', 2, NULL),
('dr_montenegro', 'montenegro123', 3, 1);
```

---

6. Consultas de ejemplo

6.1 Historial clínico de un paciente

```
SELECT p.apellido, p.nombre, f.id_ficha_medica, c.fecha_consulta, c.diagnostico,
c.tratamiento
```

```
FROM Paciente p
JOIN FichaMedica f ON f.id_paciente = p.id_paciente
JOIN ConsultaMedica c ON c.id_ficha_medica = f.id_ficha_medica
WHERE p.dni = '381293043'
ORDER BY c.fecha_consulta DESC;
```

6.2 Turnos próximos por médico

```
SELECT t.id_turno, p.apellido, p.nombre, t.fecha, t.hora, t.motivo_de_consulta,
t.es_particular
FROM Turno t
JOIN Paciente p ON p.id_paciente = t.id_paciente
WHERE t.id_doctor = 1 AND t.fecha >= CURDATE()
ORDER BY t.fecha, t.hora;
```

6.3 Facturación por obra social (rango de fechas)

```
SELECT os.nombre AS obra_social, SUM(f.monto) AS total_facturado,
COUNT(f.id_factura) AS cantidad_facturas
FROM Facturacion f
JOIN Turno t ON t.id_turno = f.id_turno
JOIN Paciente p ON p.id_paciente = t.id_paciente
JOIN ObraSocial os ON os.id_obra_social = p.id_obra_social
WHERE f.fecha_emision BETWEEN '2025-01-01' AND '2025-12-31'
GROUP BY os.nombre;
```

6.4 Pacientes sin obra social

```
SELECT id_paciente, apellido, nombre, dni, telefono
FROM Paciente
WHERE id_obra_social IS NULL;
```

6.5 Consultas con diagnósticos de hipertensión

```
SELECT p.apellido, p.nombre, c.fecha_consulta, c.diagnostico, c.tratamiento
FROM ConsultaMedica c
JOIN FichaMedica f ON f.id_ficha_medica = c.id_ficha_medica
JOIN Paciente p ON p.id_paciente = f.id_paciente
WHERE c.diagnostico LIKE '%hipertensión%'
```

6.6 Estudios pendientes de resultados

```
SELECT e.id_estudio, p.apellido, p.nombre, e.tipo_estudio, e.fecha_estudio
FROM EstudioMedico e
JOIN ConsultaMedica c ON c.id_consulta = e.id_consulta
JOIN FichaMedica f ON f.id_ficha_medica = c.id_ficha_medica
```

```
JOIN Paciente p ON p.id_paciente = f.id_paciente
WHERE e.resultados IS NULL OR e.resultados = '';
```

---

7. Manual de Instalación (MySQL Workbench)

El siguiente procedimiento describe paso a paso cómo **instalar, crear y ejecutar** la base de datos del proyecto **Consultorio Médico (Cardiología)** utilizando la herramienta **MySQL Workbench**.

El objetivo es permitir la correcta creación de la estructura de datos, las relaciones entre tablas y la carga de información de ejemplo, garantizando la integridad referencial del sistema.

7.1. Requisitos previos

Antes de iniciar la instalación, se deben cumplir los siguientes requisitos técnicos:

- **MySQL Server** instalado (versión 5.7 o superior, idealmente 8.0).
- **MySQL Workbench** instalado (versión 8.0 o superior).
- Archivo del proyecto:
    - `consultorioMedico.sql` → script SQL con todas las tablas, claves y datos de ejemplo.
- Acceso a un usuario con permisos administrativos (por ejemplo, `root`).

Configuración recomendada:

- Sistema operativo: Windows 10/11, Linux o macOS.
- Memoria RAM: 4 GB o más.
- Espacio en disco: mínimo 50 MB.

7.2. Instalación y carga del script SQL

🔹 Paso 1 — Abrir MySQL Workbench

Abrir **MySQL Workbench** y conectarse al servidor local de MySQL.
Si no hay una conexión creada:

1. Hacer clic en el ícono **“+” (New Connection)**.
2. Escribir un nombre descriptivo, por ejemplo: *Conexión Local Consultorio*.
3. En **Hostname**, dejar `localhost`.
4. En **Port**, usar `3306`.
5. Ingresar usuario (por defecto `root`) y contraseña.
6. Probar la conexión con **Test Connection**.
7. Guardar y abrir la conexión.

🔹 Paso 2 — Crear o limpiar la base de datos

Antes de ejecutar el script, asegurarse de que no exista una base con el mismo nombre.
Si ya existe, eliminarla ejecutando:

```
DROP DATABASE consultorioMedico;
```

Luego crear una nueva base vacía:

```
CREATE DATABASE consultorioMedico;
USE consultorioMedico;
```

🔹 Paso 3 — Cargar el archivo SQL

1. Ir al menú **File → Open SQL Script**.
2. Buscar el archivo `consultorioMedico.sql` en tu carpeta del proyecto.
3. Hacer clic en **Abrir**.

4. El contenido del script aparecerá en una nueva pestaña.

🔹 Paso 4 — Ejecutar el script completo

1. Seleccionar todo el contenido (Ctrl + A).
2. Hacer clic en el botón del rayo ⚡ o en **“Execute All”**.
3. Esperar a que finalice la ejecución.
4. Verificar en la consola inferior que todos los mensajes digan **“Query OK”**.

🔹 Paso 5 — Verificar la creación de las tablas

Ejecutar las siguientes consultas de verificación:

```
SHOW DATABASES;
USE consultorioMedico;
SHOW TABLES;
```

Deberían aparecer las tablas principales del sistema:

```
Doctor
Paciente
ObraSocial
Turno
Facturacion
FichaMedica
ConsultaMedica
RecetaMedica
EstudioMedico
Rol
Usuario
```

7.3. Verificación de datos cargados

Para comprobar que los registros de ejemplo se cargaron correctamente, ejecutar:

```
SELECT * FROM Doctor;
SELECT * FROM Paciente;
SELECT * FROM Turno;
SELECT * FROM Facturacion;
```

Si los resultados muestran información, significa que las inserciones se realizaron correctamente.

También se puede verificar la integridad de relaciones con:

```
SELECT p.apellido, p.nombre, t.fecha, t.hora, d.apellido AS doctor
FROM Turno t
JOIN Paciente p ON p.id_paciente = t.id_paciente
JOIN Doctor d ON d.id_doctor = t.id_doctor;
```

7.4. Recomendaciones de instalación y mantenimiento

- **Ejecutar el script solo una vez.** Si se ejecuta nuevamente, puede generar errores por duplicación de claves.
- **Eliminar la base existente antes de reinstalar**, usando `DROP DATABASE consultorioMedico;`.
- **No modificar los nombres de tablas o campos**, ya que rompería las claves foráneas.
- **Realizar copias de seguridad** del archivo `.sql` antes de cambios o pruebas.
- **Verificar siempre los mensajes de la consola inferior** en MySQL Workbench (cualquier error de sintaxis o relación aparecerá allí).
- **Mantener MySQL actualizado** para evitar incompatibilidades con funciones modernas.

7.5. Confirmación de instalación exitosa

La instalación se considera correcta cuando:

1. La base `consultorioMedico` aparece listada con `SHOW DATABASES;`.

2. Todas las tablas se visualizan con `SHOW TABLES;`.

3. Los `SELECT` sobre las tablas muestran datos válidos.

4. Las relaciones entre `Paciente`, `Doctor`, `Turno`, `FichaMedica` y `Facturacion` funcionan sin errores.

Una vez completado este proceso, la base de datos queda lista para realizar las **consultas de prueba** del documento *"Control y Prueba del Sistema"*.

8. Backup y Restauración

El siguiente apartado explica los procedimientos para **realizar copias de seguridad (backup)** y **restaurar (importar)** la base de datos del proyecto **Consultorio Médico (Cardiología)** utilizando MySQL.

El objetivo es garantizar la integridad y disponibilidad de la información ante posibles fallos, pérdidas de datos o reinstalaciones del sistema.

8.1. Concepto de backup

Un **backup** es una copia exacta del estado actual de la base de datos en un momento determinado.
Permite **restaurar** el sistema en caso de pérdida, daño o error humano.

En MySQL, los backups se pueden generar mediante:

- **MySQL Workbench** (desde la interfaz gráfica).
- **mysqldump** (desde la línea de comandos).

8.2. Realizar un backup desde MySQL Workbench

🔹 Paso 1 — Abrir MySQL Workbench

Abrir la aplicación y conectarse al servidor de MySQL (por ejemplo, con el usuario `root`).

🔹 Paso 2 — Acceder a la opción de exportación

1. En el panel inicial, seleccionar la conexión y abrirla.
2. Ir al menú superior: **Server** → **Data Export**.
3. En la lista de bases, seleccionar **consultorioMedico**.

🔹 Paso 3 — Elegir método de exportación

- Seleccionar la opción **"Dump Structure and Data"** para incluir tanto las tablas como los registros.

Elegir una carpeta de destino para guardar el archivo, por ejemplo:

```
C:\Backups\consultorioMedico_backup.sql
```

-

🔹 Paso 4 — Ejecutar la exportación

- Hacer clic en **Start Export**.
- Esperar a que aparezca el mensaje **"Export completed successfully"**.
- Verificar que se haya creado el archivo `.sql` en la carpeta elegida.

8.3. Realizar un backup desde la consola (mysqldump)

Otra forma de hacer una copia de seguridad es utilizando el comando `mysqldump`, desde la terminal o símbolo del sistema.

Ejemplo:

```
mysqldump -u root -p consultorioMedico >
C:\Backups\consultorioMedico_backup.sql
```

Explicación:

- `-u root`: usuario de MySQL.

- `-p`: pedirá la contraseña al ejecutarlo.
- `consultorioMedico`: nombre de la base de datos.
- `>` redirecciona la salida al archivo indicado.

Resultado: se genera el archivo `consultorioMedico_backup.sql` con toda la estructura y los datos actuales.

8.4. Restaurar (importar) una base de datos

En caso de que la base se haya perdido, dañado o deba reinstalarse, se puede restaurar el backup de dos formas:

Opción 1 — Desde MySQL Workbench

1. Abrir **MySQL Workbench** y conectarse.
2. Ir a **Server** → **Data Import**.
3. Seleccionar la opción **Import from Self-Contained File**.

Buscar el archivo:

```
C:\Backups\consultorioMedico_backup.sql
```

4.
5. En **Default Target Schema**, crear o elegir una base de datos (por ejemplo, `consultorioMedico`).
6. Presionar **Start Import**.

Una vez finalizado, la base quedará restaurada con todas sus tablas y datos.

Opción 2 — Desde la consola

Si preferís usar comandos, podés restaurarla desde la terminal:

```
mysql -u root -p consultorioMedico <
C:\Backups\consultorioMedico_backup.sql
```

Explicación:

- `mysql -u root -p`: inicia MySQL con el usuario root.
- `consultorioMedico`: base de datos destino.
- `<`: indica que se importará desde el archivo SQL especificado.

8.5. Recomendaciones de seguridad

- Guardar las copias en **carpetas protegidas o servicios en la nube** (Google Drive, OneDrive, etc.).
- Realizar **backups periódicos** (semanales o antes de grandes cambios).
- No modificar manualmente el archivo `.sql` generado.

Incluir la fecha en el nombre del archivo de respaldo, por ejemplo:

```
consultorioMedico_backup_2025_10_10.sql
```

- Mantener al menos **dos versiones** de respaldo (actual y anterior).

8.6. Confirmación de restauración exitosa

Una restauración se considera correcta si:

1.La base de datos `consultorioMedico` se visualiza con:

```
SHOW DATABASES;
```

2.Las tablas aparecen con:

```
SHOW TABLES;
```

3.Los datos de ejemplo están disponibles con:

```
SELECT * FROM Paciente;
```

4.No se generan errores de integridad referencial ni duplicados.

9. Seguridad y Buenas Prácticas

El sistema de base de datos del proyecto *Consultorio Médico (Cardiología)* debe garantizar la **confidencialidad, integridad y disponibilidad** de la información médica y personal de los pacientes.
A continuación, se detallan las principales medidas y recomendaciones aplicadas y sugeridas.

9.1. Protección de contraseñas

- Las contraseñas de los usuarios se almacenan en la tabla `Usuario` dentro del campo `contrasena`.
- **Nunca deben guardarse en texto plano.**

Se recomienda aplicar **hashing** con algoritmos seguros como `SHA2()` o `bcrypt`.
Ejemplo de inserción con hash:

```
INSERT INTO Usuario (nombre_usuario, contrasena, id_rol)
VALUES ('admin', SHA2('admin123', 256), 1);
```

- En entornos productivos, el hash debe gestionarse desde la aplicación que accede a la base de datos, no directamente desde SQL.

9.2. Permisos mínimos de usuarios

- Crear usuarios con **privilegios limitados**:
    - Un usuario de lectura/escritura para tareas cotidianas.
    - Un usuario administrativo para creación o mantenimiento.

Ejemplo:

```
CREATE USER 'consultorio_user'@'localhost' IDENTIFIED BY
'seguro123';

GRANT SELECT, INSERT, UPDATE, DELETE ON consultorioMedico.* TO
'consultorio_user'@'localhost';

FLUSH PRIVILEGES;
```

9.3. Copias de seguridad

- Implementar un plan de **backups automáticos** semanales utilizando `mysqldump` o las funciones de exportación de MySQL Workbench.
- Guardar los archivos en dispositivos externos o servicios en la nube (Google Drive, OneDrive).
- Mantener una política de retención de al menos dos versiones anteriores.

9.4. Control de acceso físico y lógico

- Limitar el acceso al equipo donde se aloja la base de datos.
- Proteger el servidor con contraseña y, si es posible, con autenticación de dos factores.
- En entornos web, usar **conexiones seguras (HTTPS)** y restringir el acceso por IP.

9.5. Integridad referencial

- Se aplican **claves primarias y foráneas** en todas las relaciones entre tablas (`Paciente`, `Doctor`, `Turno`, `FichaMedica`, etc.) para evitar datos huérfanos.
- Las restricciones `ON DELETE` y `ON UPDATE` se pueden usar para mantener coherencia en cascada.

---

10. Índices y Optimización

El uso de índices permite mejorar el rendimiento de las consultas más frecuentes dentro del sistema.
A continuación se sugieren los índices más importantes y las estrategias de optimización.

10.1. Índices sugeridos

Tabla	Campo	Tipo de índice	Motivo
Paciente	dni	UNIQUE INDEX	Permite la búsquedas rápidas por DNI.
Doctor	dni	UNIQUE INDEX	Evita duplicados y mejora búsquedas por documento.
Turno	id_paciente, id_dictir,fecha	INDEX compuesto	Mejora el rendimiento de las consultas por paciente, médico o fecha
Facturacion	id_turno	INDEX	Permite acceder rápidamente a los pagos relacionados con cada turno.
ConsultaMedica	id_ficha_medica,fecha_consulta	INDEX	Mejora la búsqueda de consultas por ficha o fecha.

10.2. Optimización adicional

- Evitar el uso de `SELECT *` en consultas complejas; seleccionar solo los campos necesarios.
- Usar **tipos de datos adecuados** (por ejemplo, `BOOLEAN`, `DATE`, `DECIMAL`) para reducir espacio.
- Revisar las **consultas con JOIN** para asegurar que las claves estén indexadas.
- Limpiar datos obsoletos o registros duplicados periódicamente.

---

11. Mantenimiento y Futuras Mejoras

La base de datos del *Consultorio Médico (Cardiología)* fue diseñada con posibilidad de expansión y mantenimiento a largo plazo.
A continuación se presentan sugerencias para su evolución y mejoras futuras.

11.1. Mantenimiento preventivo

Revisar mensualmente la integridad de las tablas con:

```
CHECK TABLE nombre_tabla;
```

-

Optimizar el rendimiento con:

```
OPTIMIZE TABLE nombre_tabla;
```

-
- Realizar un backup completo antes de cada actualización o cambio estructural.

11.2. Mejoras futuras sugeridas

1. **Auditoría de acciones:**
   Crear una tabla `Auditoria` para registrar logins, inserciones o modificaciones de datos sensibles (pacientes, facturación, recetas).

2. **Facturación electrónica:**
   Integrar módulos para generar comprobantes con códigos QR o CAE (según AFIP).

3. **Sistema de roles detallado:**
   Ampliar la tabla `Rol` para definir permisos específicos (lectura, edición, reportes).

4. **Módulo de estadísticas médicas:**
   Permitir obtener informes de cantidad de consultas por mes, patologías frecuentes, etc.

5. **Integración con historia clínica digital:**
   Vincular la base con sistemas externos para compartir información con otros profesionales.

📎 12. Anexos y Cambios Realizados

Esta sección resume las **modificaciones estructurales y mejoras aplicadas** durante el desarrollo del proyecto, en comparación con versiones anteriores.

12.1. Cambios principales aplicados

N°	Modificación	Descripción
1	Eliminación de tabla `Especialidad`	El consultorio se centra únicamente en la especialidad de Cardiología.
2	Agregado de campo `correo_electronico`	Incorporado en tablas `Doctor` y `Paciente` para contacto digital.
3	Creación de tabla `ObraSocial`	Permite vincular pacientes con su cobertura médica
4	Inclusión del campo `es_particular` en `Turno`	Indica si el turno fue abonado de forma particular o mediante obra social.
5	Incorporación de tabla `Facturacion`	Registra los pagos y estado de facturación de cada turno.
6	Ampliación de `FichaMedica` y creación de `ConsultaMedica`	Permite mantener un historial médico detallado por paciente
7	Nuevas tablas `RecetaMedica` y `EstudioMedico`	Guardan la información de medicamentos y estudios realizados.
8	Creación de tablas `Usuario` y `Rol`	Añaden control de acceso y gestión de permisos
9	Implementación de buenas prácticas y claves foráneas	Se reforzó la integridad referencial entre todas las entidades.

12.2. Observaciones finales

El diseño actual ofrece una estructura **escalable, normalizada y segura**, cumpliendo con los objetivos de organización y control de un consultorio médico de especialidad única.
Las futuras versiones podrán ampliar el sistema a múltiples especialidades, incorporar auditoría y facturación electrónica.

Conclusión

El desarrollo del proyecto “Consultorio Médico (Cardiología)” permitió aplicar de forma práctica los conceptos fundamentales de modelado, normalización, integridad referencial y seguridad de datos vistos durante la materia *Base de Datos I*.

La estructura final de la base de datos garantiza un sistema organizado, escalable y confiable, capaz de gestionar pacientes, turnos, médicos, fichas médicas, recetas y facturación de manera eficiente.

Durante el proceso de diseño se aplicaron buenas prácticas de modelado relacional, uso adecuado de claves primarias y foráneas, así como recomendaciones de seguridad y mantenimiento para asegurar la continuidad del sistema en el tiempo.

Como resultado, se logró una solución sólida que puede ser ampliada fácilmente con nuevas funciones, como auditorías, roles más detallados o integración con sistemas externos de facturación y gestión médica.

Este trabajo representa un paso importante en nuestra formación profesional, ya que nos permitió aplicar los conocimientos adquiridos en la materia *Base de Datos I* para el desarrollo completo de un sistema funcional y seguro.
A través del proyecto pudimos analizar, diseñar y documentar una base de datos real, comprendiendo la importancia de la organización de la información, la integridad de los datos y las buenas prácticas en el uso de MySQL.

INSTITUTO DE ENSEÑANZA SUPERIOR “ALFREDO COVIELLO”
Materia: Base de Datos I
Profesora: Prof. Ing. Carla Aria Acuña

Alumnos:
- Claudio Ezequiel Gamarro
- Diaz Emilse Salomé
- Montenegro Héctor Rober

Fecha de presentación: Octubre de 2025