Trabajo Final MySQL

Coderhouse

Alumno: Cristian Baron

Fecha 9/6/2025

Índice

Introducción:

El trabajo está enfocado a una Clínica con la finalidad de poder asignar turnos eficientemente buscando evitar problemas comunes como sobreturnos, falta de datos y eficiencia, y dejando un registro claro que a su vez permite llevar una historia clínica de los pacientes atendidos.

Objetivo

Desarrollar una base de datos relacional para gestionar de forma eficiente los turnos, pacientes, doctores y la información clínica asociada a una clínica médica. El sistema permitirá organizar los recursos disponibles, optimizar la asignación de turnos y mantener un historial clínico detallado y actualizado por paciente, mejorando así la calidad del servicio y facilitando el acceso a la información médica.

Situación Problemática

Actualmente, muchas clínicas enfrentan dificultades para gestionar de forma ordenada y centralizada la información relativa a turnos, pacientes, profesionales y diagnósticos médicos. Estas dificultades suelen incluir:

- Registro manual o descentralizado de turnos.
- Sobreturnos o superposición de citas por falta de control sobre la disponibilidad.
- Historial clínico incompleto o disperso.
- Recetas, diagnósticos y tratamientos mal organizados o poco accesibles.
- Falta de trazabilidad de los datos ya que no hay un control de cuándo se hacen cambios ni quién los hizo, lo que puede generar errores o confusiones.
- Procesos ineficientes en la atención y en la administración general de la clínica.

Estas limitaciones no solo afectan la productividad del personal, sino también la calidad de atención brindada al paciente. La implementación de una base de datos relacional permite resolver estas brechas, ofreciendo una estructura organizada, consistente y escalable para gestionar de manera integral la información médica y administrativa de la clínica. Además, permite generar reportes, controlar el estado de los turnos, verificar la agenda médica y acceder rápidamente al historial clínico de cada paciente.

Modelo de Negocio

El modelo se basa en una clínica médica que brinda atención personalizada a pacientes a través de distintos profesionales de la salud. Esta clínica puede contar con varios consultorios, especialidades y profesionales, así como con más de una sede (establecimientos).

La organización requiere un sistema que permita:

Registrar pacientes y su historial clínico.

Administrar los turnos de forma eficiente según la disponibilidad de cada médico.

Asociar a cada turno información clínica relevante (diagnóstico, tratamiento, recetas).

Tener visibilidad sobre la agenda médica y evitar conflictos de horarios.

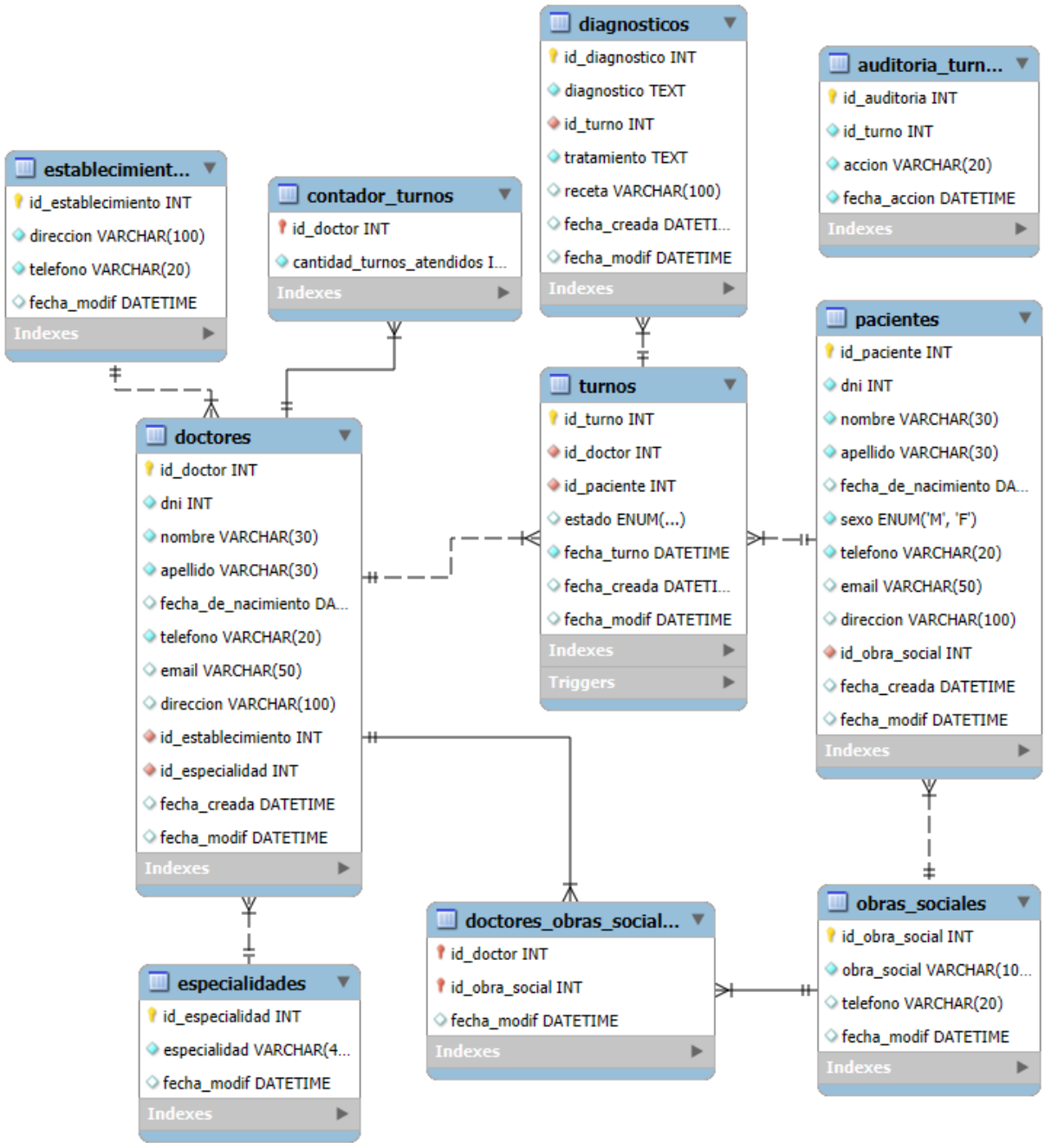
Asignar médicos a consultorios y establecer qué obras sociales atienden.

Llevar registro de auditoría sobre cuándo y cómo se modifica cada dato (fechas de creación y actualización).

Brindar informes eficientes y explicativos.

Este modelo busca digitalizar el flujo de trabajo tradicional, integrando todas las áreas funcionales (administrativa, médica, de atención al cliente) en una única base de datos relacional que facilite la operación diaria y el análisis posterior.

Diagrama E-R (Entidad-Relación)


establecimient...
id_establecimiento INT
direccion VARCHAR(100)
telefono VARCHAR(20)
fecha_modif DATETIME
Indexes
contador_turnos
id_doctor INT
cantidad_turnos_atendidos I...
Indexes
diagnosticos
id_diagnostico INT
diagnostico TEXT
id_turno INT
tratamiento TEXT
receta VARCHAR(100)
fecha_creada DATETI...
fecha_modif DATETIME
Indexes
auditoria_turn...
id_auditoria INT
id_turno INT
accion VARCHAR(20)
fecha_accion DATETIME
Indexes
doctores
id_doctor INT
dni INT
nombre VARCHAR(30)
apellido VARCHAR(30)
fecha_de_nacimiento DA...
telefono VARCHAR(20)
email VARCHAR(50)
direccion VARCHAR(100)
id_establecimiento INT
id_especialidad INT
fecha_creada DATETIME
fecha_modif DATETIME
Indexes
turnos
id_turno INT
id_doctor INT
id_paciente INT
estado ENUM(...)
fecha_turno DATETIME
fecha_creada DATETI...
fecha_modif DATETIME
Indexes
Triggers
pacientes
id_paciente INT
dni INT
nombre VARCHAR(30)
apellido VARCHAR(30)
fecha_de_nacimiento DA...
sexo ENUM('M', 'F')
telefono VARCHAR(20)
email VARCHAR(50)
direccion VARCHAR(100)
id_obra_social INT
fecha_creada DATETIME
fecha_modif DATETIME
Indexes
especialidades
id_especialidad INT
especialidad VARCHAR(4...
fecha_modif DATETIME
Indexes
doctores_obras_social...
id_doctor INT
id_obra_social INT
fecha_modif DATETIME
Indexes
obras_sociales
id_obra_social INT
obra_social VARCHAR(10...
telefono VARCHAR(20)
fecha_modif DATETIME
Indexes

Listado de Tablas

Tabla turnos:

id_turno: tipo Integer. Es la **Primary Key** de la entidad. Es el identificador del turno.

id_doctor: tipo Integer. Es **FK** de id_doctor de la tabla Doctores. Es el identificador del doctor con el que se atiende el paciente.

id_paciente: tipo Integer. Es **FK** de id_paciente de la tabla Pacientes. Es el identificador del paciente que tomo el turno.

estado: tipo ENUM. Permite ver el estado del turno, por Default es "confirmado" (el turno está confirmado). Sino puede ser "ausente" (si el paciente no se presenta) y "atendido" una vez que el paciente se atendió.

fecha_turno: tipo DATETIME. Indica fecha del turno.

fecha_creada: tipo DATETIME. Se guarda la fecha y hora de cuando se creó el registro.

fecha_modif: tipo DATETIME. Se updatea la fecha y hora de cuando se modifica el registro.

unq_tuno_unico: es una restricción UNIQUE para que no haya dos turnos con igual id_doctor, fecha y hora para evitar sobreturnos.

Tabla doctores:

id_doctor: tipo Integer. Es la **Primary Key** de la entidad.

dni: tipo INT (8). El número de documento del doctor.

nombre: tipo VARCHAR (30). Nombre de pila del doctor

apellido: tipo VARCHAR (30). Apellido del doctor

teléfono: tipo VARCHAR. Teléfono del doctor

email: tipo VARCHAR. Dirección de email del doctor

dirección: tipo VARCHAR.

id_establecimiento: tipo Integer. Es **FK** de id_establecimiento de la tabla Establecimientos. Es el identificador del establecimiento donde atiende el doctor.

id_especialidad: tipo Integer. Es **FK** de id_especialidad de la tabla Especialidades. Es el identificador de la especialidad que trata el doctor.

fecha_creada: tipo DATETIME. Se guarda la fecha y hora de cuando se creó el registro.

fecha_modif: tipo DATETIME. Se updatea la fecha y hora de cuando se modifica el registro.

Tabla especialidades:

id_especialidad: tipo Integer. Es la **Primary Key** de la entidad. Es el identificador de especialidades.

especialidad: tipo VARCHAR. Son las especialidades existentes.

fecha_modif: tipo DATETIME. Se updatea la fecha y hora de cuando se modifica el registro.

Tabla pacientes:

id_paciente: tipo Integer. Es la **Primary Key** de la entidad. Es el identificador de pacientes.

dni: tipo Integer (8). El número de documento del paciente.

nombre: tipo VARCHAR. Nombre de pila del paciente.

apellido: tipo VARCHAR. Apellido del paciente.

fecha_de_nacimiento: DATE**.**

sexo: ENUM (F o M).

teléfono: tipo VARCHAR. Teléfono del paciente.

email: tipo VARCHAR. Dirección de email del paciente.

dirección: tipo VARCHAR.

id_obra_social: tipo Integer. Es **FK** de id_obra_social de la tabla Obra Social. Es el identificador de la obra social del paciente.

fecha_creada: tipo DATETIME. Se guarda la fecha y hora de cuando se creó el registro.

fecha_modif: tipo DATETIME. Se updatea la fecha y hora de cuando se modifica el registro.

Tabla establecimientos:

id_establecimiento: tipo Integer. Es la **Primary Key** de la entidad. Es el identificador de establecimientos existentes.

dirección: tipo VARCHAR. Es el lugar donde de encuentran los distintos establecimientos de la clínica.

teléfono: tipo VARCHAR. Teléfono del establecimiento.

fecha_modif: tipo DATETIME. Se updatea la fecha y hora de cuando se modifica el registro.

Tabla obras_sociales:

id_obra_social: tipo Integer. Es la **Primary Key** de la entidad. Es el identificador de obras sociales existentes.

obra_social: tipo VARCHAR. Son las distintas obras sociales existentes.

teléfono: tipo VARCHAR. Teléfono de la obra social.

fecha_modif: tipo DATETIME. Se updatea la fecha y hora de cuando se modifica el registro.

Tabla diagnosticos:

id_diagnostico: tipo Integer. Es la **Primary Key** de la entidad. Es el identificador de diagnósticos.

diagnóstico: tipo TEXT. El doctor guarda el diagnóstico del paciente.

id_turno: tipo Integer. Es **FK** de id_turno de la tabla turnos. Cada turno va a tener un diagnóstico.

tratamiento: tipo TEXT. El doctor guarda el tratamiento a llevar a cabo por paciente.

receta: tipo VARCHAR. Se guarda una copia online de la receta.

fecha_creada: tipo DATETIME. Se guarda la fecha y hora de cuando se creó el registro.

fecha_modif: tipo DATETIME. Se updatea la fecha y hora de cuando se modifica el registro.

Tabla doctores_obras_sociales (tabla intermedia):

id_doctor & **id_obras_sociales**: tipo Integer: son las Primary Key (compuesta) ya que un doctor puede cubrir mas de una obra social. Son FK de id_doctor (doctores) y id_obra_social (obras_sociales).

fecha_modif: tipo DATETIME. Se updatea la fecha y hora de cuando se modifica el registro.

Tabla auditoria_turnos (tabla log):

id_auditoria: tipo Integer. Es la **Primary Key** de la entidad.

id_turno: INT. Muestra el ID del turno que sufrió un cambio.

Acción: VARCHAR: Muestra el tipo de cambio que sufrió.

fecha_accion: DATETIME de cuando se realizo la acción.

Tabla contador_turnos (tabla count):

Id_doctor: tipo Integer. Es la **Primary Key** de la entidad.

Cantidad_turnos_atendidos: INT. Cuenta los turnos que pasan a estado “atendido”.

Listado de Vistas

v_turno: para ver el listado completo de turnos

vista_doctores_convenios: para ver las obras sociales que atende cada doctor

v_info_pacientes: vista para ver toda la información de los pacientes

v_doctores: Vista para ver toda la información de los Doctores

vista_especialidades_doctores: Vista para ver todas las Especialidades y que doctores las cubren

vista_turnos_confirmados_por_doctor: vista para ver cantidad de turnos en estado “confirmado” (los que aun no sucedieron) por doctor

v_trigger_contador_turnos: vista para ver la tabla donde la tabla que guarda la cantidad de turnos atendidos por doctor, que es accionada por un trigger.

Listado de Funciones

proxima_fecha_turno: Función para ver la próxima fecha de turno de un paciente por DNI

edad_promedio_doctores: función para ver edad promedio de doctores

edad_promedio_pacientes: función para edad pacientes promedio

Listado de Stored Procedures

crear_turno: Procedimiento para crear Turnos

actualizar_estado_turno: Procedure para actualizar el estado del Turno. Es decir, una vez que paso la fecha, se debe actualizar a si el paciente fue atendido o si se ausento (el estado es confirmado por default)

turnos_confirmados_doctor: Procedure para ver los Turnos asignados por Doctor entre dos fechas (seria la agenda del doctor para ese periodo).

registrar_diagnostico: Procedure para registrar un nuevo Diagnostico

p_historia_clinica_paciente: Procedure para ver el historial clínico de un paciente por DNI

Listado de Triggers

after_insert_turno: Trigger para registar inserciones en la tabla Turnos. En tabla **auditoria_turnos.**

before_delete_turno: Trigger para registrar eliminaciones en la tabla Turnos. En tabla **auditoria_turnos.**

after_update_turno: Trigger para registrar updates en la tabla Turnos. En tabla **auditoria_turnos.**

before_insert_turno: Trigger para enviar mensaje de error al querer asignar un turno ya ocupado.

sumar_turno_atendido: trigger para contar turnos atendidos por doctor

Informes generados en base a la información almacenada en las tablas

Se generaron informes en formato CSV (exportados con Export Wizard).

Para realizar un estudio más en profundidad, también se conectaron las tablas a Tableau para poder realizar visualizaciones de la base de datos y obtener informes de mayor calidad. Este mismo se subió a Tableau Public: https://public.tableau.com/shared/Q37K8B6FY?:display_count=n&:origin=viz_share_link

Además, se realizó un backup de la bbdd desde cmd.

- **Exportar:** mysqldump -u root -p codersql > backup.sql
- **Importar:** mysqldump -u root -p codersql < backup.sql

También se garantizo acceso a la BBDD:

- **CREATE USER 'profesor'@'localhost' IDENTIFIED BY 'password1';**
- **GRANT ALL PRIVILEGES ON codersql.* TO 'profesor'@'localhost';**

Herramientas y tecnologías que utilizaste

Se utilizo MySQL para gestionar la base de datos.

Se utilizo Tableau para generar visualizaciones de la base de datos.

Se utilizo GitHub como repositorio.