PROYECTO FINAL BASE DE DATOS

Diseño de Base de Datos para el Hospital Buen Día

Optimización y gestión eficiente de la información hospitalaria

Por Kevin Simbaña. Johan Vargas

Objetivo del Proyecto

OBJETIVO:

Diseñar e implementar una base de datos eficiente y segura para la gestión de pacientes, médicos, citas, historiales médicos, facturación y medicamentos en el Hospital Buen Día.

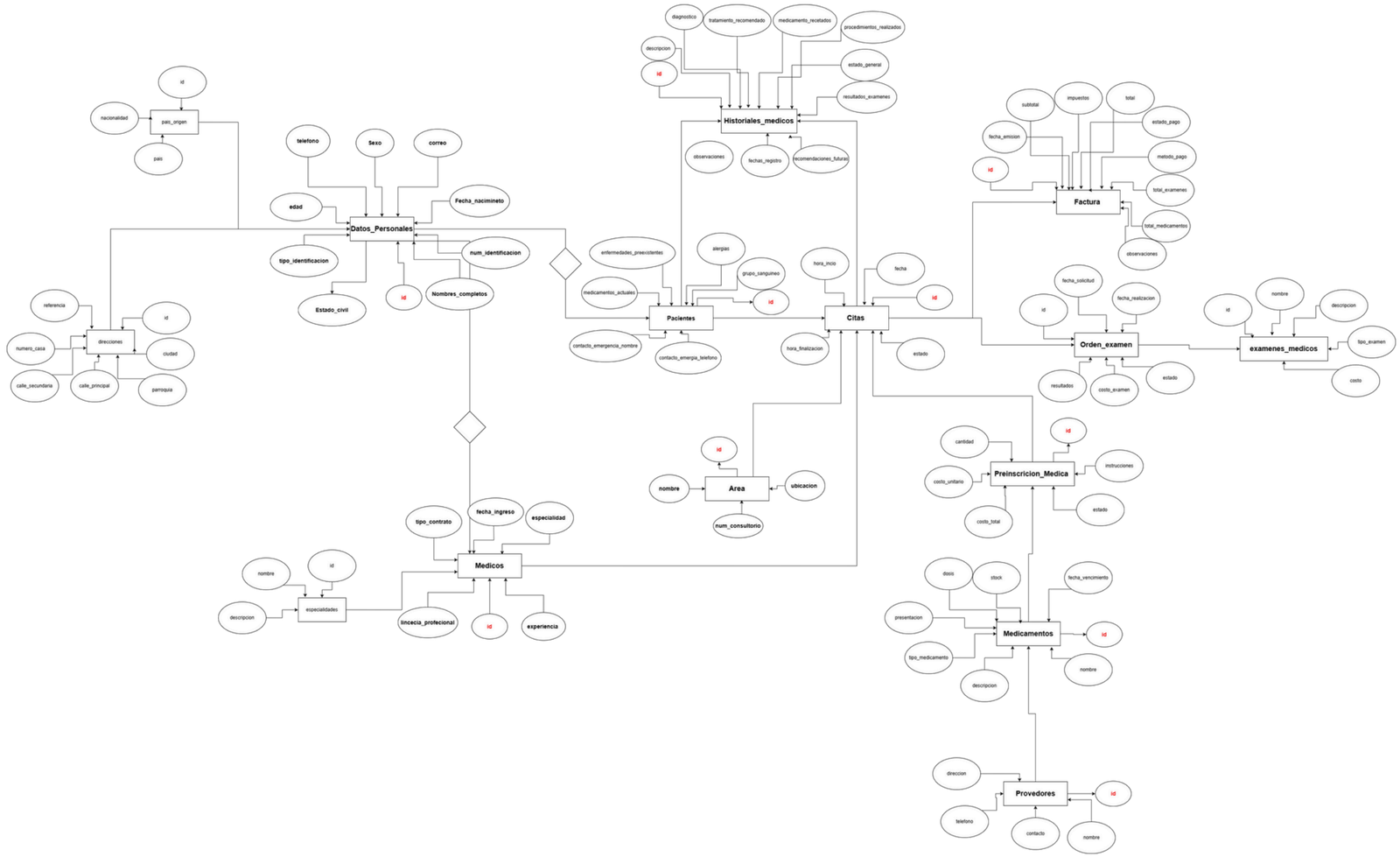
Modelado de la Base de Datos

Modelos desarrollados:

- Modelo Conceptual (Entidades y relaciones principales)
- Modelo Lógico (Definición de atributos y relaciones)
- Modelo Físico (Creación de tablas en MySQL)

Principales entidades:

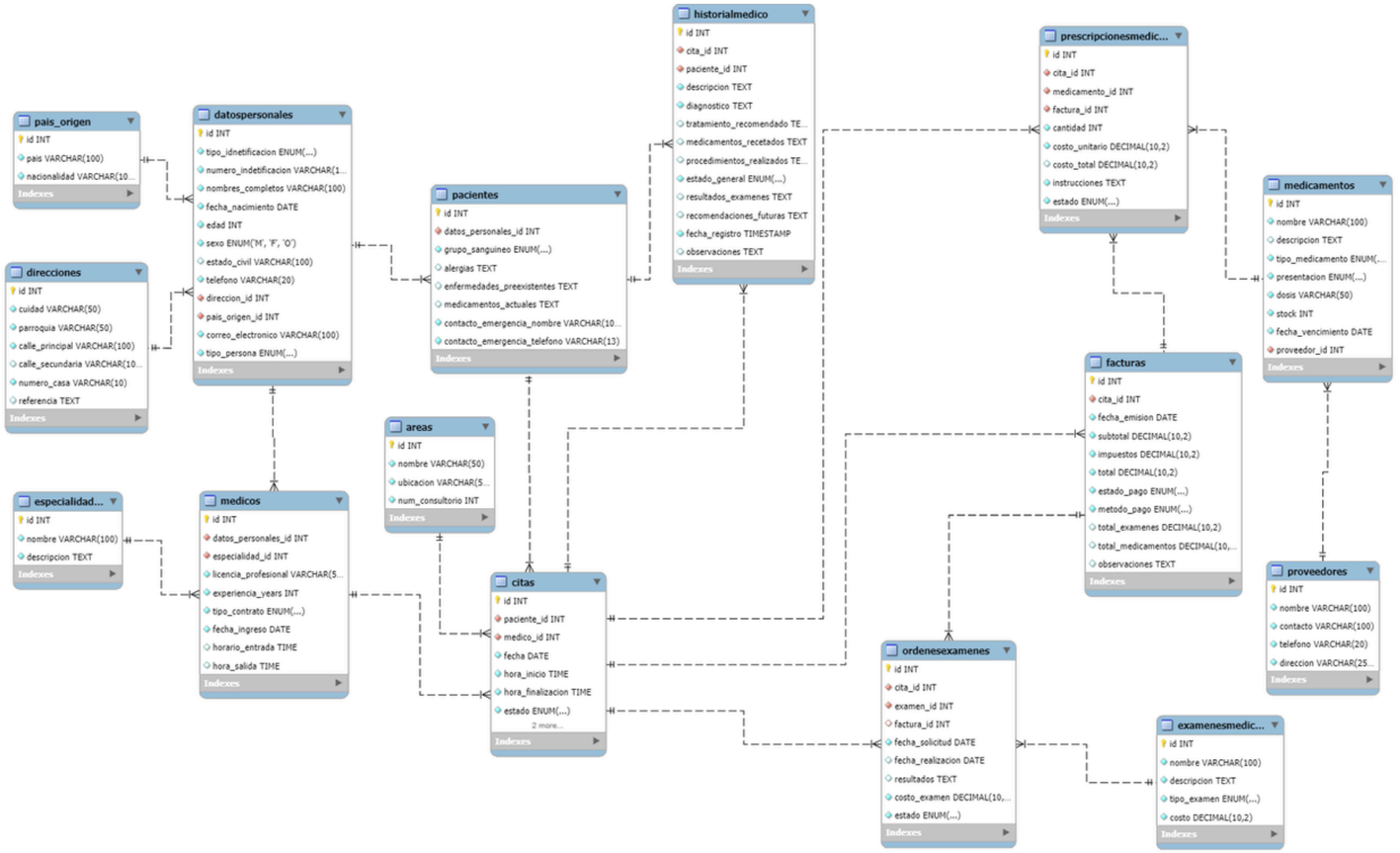
- Pacientes
- Médicos
- Especialidades
- Citas
- Historial Médico
- Facturas
- Medicamentos



Modelo logico

Tablas clave:

- Datos personales: Información de pacientes y médicos
- Direcciones y País de Origen: Información de ubicación
- Especialidades: Campos médicos del hospital
- Médicos: Relación con especialidades y datos personales
- Pacientes: Datos clínicos y contactos de emergencia
- Citas: Registro de consultas médicas
- Historial Médico: Diagnósticos, tratamientos y procedimientos
- Facturas: Gestión de pagos y cobros



Seguridad, Auditoría y Control de Acceso

Creación de Usuarios y Roles

Usuarios creados:

- admin_h: Acceso total a la base de datos
- medico_usuario: Acceso a pacientes, citas e historial médico
- enfermera_usuario: Acceso a consultas de citas, historial y pacientes
- administrativo_usuario: Gestión de datos personales, citas, médicos y facturas
- farmaceutico_usuario: Acceso a medicamentos y consultas de pacientes
- tecnico_laboratorio_usuario: Gestión de exámenes médicos y consultas de pacientes

Auditoría y Monitoreo de Seguridad

Registro y control de accesos

📌 Herramientas utilizadas:

- Consulta de usuarios activos: SELECT User, Host FROM mysql.user;
- Verificación de autenticación: SELECT user, host, authentication_string, plugin FROM mysql.user;
- Activación de registros de funciones: SET GLOBAL log_bin_trust_function_creators = 1;

Asignación de Permisos

Gestión de accesos mediante GRANT

📌 Ejemplo de permisos asignados:

- Médicos: SELECT, UPDATE sobre pacientes, citas y historialmedico
- Enfermeras: SELECT en citas, historialmedico y pacientes
- Administrativos: INSERT, UPDATE, DELETE en datospersonales, citas, medicos, pacientes y facturas
- Farmacéuticos: SELECT, UPDATE en medicamentos
- Técnicos de laboratorio: INSERT, UPDATE en examenesmedicos

Integridad y Seguridad de la Base de Datos

Medidas implementadas:

✓ Integridad referencial mediante claves primarias y foráneas

✓ Restricciones de datos (CHECK, UNIQUE, NOT NULL)

✓ Control de acceso con roles y permisos

✓ Cifrado de datos sensibles

✓ Registro de auditoría y monitoreo

```sql
CREATE TABLE datospersonales (
    id  INT AUTO_INCREMENT PRIMARY KEY,
    tipo_idnetificacion enum('cedula', 'pasaporte', 'licencia') not null,
    numero_indetificacion varchar(10) not null unique,
    nombres_completos VARCHAR(100) NOT NULL,
    fecha_nacimiento DATE NOT NULL,
    edad INT NOT NULL CHECK (edad >= 0),
    sexo ENUM('M','F','O') NOT NULL,
    estado_civil VARCHAR(100),
    telefono VARCHAR(20) NOT NULL CHECK (telefono REGEXP '^\+593[0-9]{9}$'),
    direccion_id INT NOT NULL,
    pais_origen_id INT NOT NULL,
    correo_electronico VARCHAR(100) UNIQUE NOT NULL CHECK (correo_electronico REGEXP '^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Za-z]{2,}$'),
    tipo_persona ENUM('Paciente', 'Medico') NOT NULL,
    CONSTRAINT fk_direccion FOREIGN KEY (direccion_id) REFERENCES direcciones(id),
    CONSTRAINT fk_pais_origen FOREIGN KEY (pais_origen_id) REFERENCES pais_origen(id)
);
```

Procedimientos Almacenados, Vistas y Triggers

Procedimientos Almacenados

Procedimientos implementados:

- RegistrarCita(): Inserta una nueva cita médica
- ActualizarEstadoCita(): Cambia el estado de una cita (Pendiente, Completada, Cancelada)
- RegistrarHistorialMedico(): Registra diagnóstico y tratamiento de un paciente
- FacturarCita(): Genera una factura por cita médica
- RegistrarPrescripcion(): Agrega una prescripción de medicamentos
- ConsultarCitasPaciente(): Muestra las citas de un paciente específico

Vistas

Vistas creadas:

- vista_pacientes: Datos personales y antecedentes médicos de pacientes
- vista_historial_medico: Registros médicos por paciente y cita
- vista_facturacion_citas: Detalles de facturación y pagos por consulta
- vista_citas_activas: Listado de citas médicas pendientes
- vista_medicamentos_stock: Medicamentos disponibles en inventario
- vista_examenes_paciente: Exámenes realizados por paciente
- vista_personal_medico: Información del personal médico y especialidad

Triggers

Triggers implementados:

- trg_no_datos_repetidos: Evita duplicidad en registros de pacientes y médicos
- trg_validar_disponibilidad_medico: Evita asignación de citas en horarios ocupados
- trg_actualizar_estado_cita: Marca automáticamente citas como "Completadas" al finalizar
- trg_validar_vencimiento_medicamentos: Impide la prescripción de medicamentos caducados
- trg_evitar_eliminar_paciente: No permite eliminar pacientes con citas pendientes

Optimización del Rendimiento

Optimización de SQL con EXPLAIN para identificar cuellos de botella en consultas

- Uso de JOIN para optimizar relaciones entre tablas
- Particionamiento de tablas para mejorar rendimiento en consultas de gran volumen
- Procedimientos almacenados y triggers para automatizar tareas repetitivas y mejorar la eficiencia del sistema

```sql
/* tabla_datos_personales */
CREATE INDEX idx_nombres_completos ON datospersonales(nombres_completos);
CREATE INDEX idx_correo_electronico ON datospersonales(correo_electronico);

/* tabla_medicos */
CREATE INDEX idx_especialidad_id ON medicos(especialidad_id);
CREATE INDEX idx_datos_personales_id_medicos ON medicos(datos_personales_id);

/* tabla_datos_personales */
CREATE INDEX idx_grupo_sanguineo ON pacientes(grupo_sanguineo);
CREATE INDEX idx_datos_personales_id_pacientes ON pacientes(datos_personales_id);

/* tabla_citas*/
CREATE INDEX idx_paciente_id ON citas(paciente_id);
CREATE INDEX idx_medico_id ON citas(medico_id);
CREATE INDEX idx_estado ON citas(estado);
```

Encriptación

SHA-2 (Secure Hash Algorithm 2) es una familia de funciones hash criptográficas que proporciona una forma segura de almacenar y verificar datos como contraseñas en bases de datos. MySQL ofrece soporte para funciones SHA-2 a través de SHA2().

¿Cómo funciona SHA-2 en MySQL?

- Generación de hash: SHA-2 toma una entrada (como una contraseña) y genera un valor hash de longitud fija (256 o 512 bits).
- Almacenaje seguro: En lugar de almacenar contraseñas en texto claro, se almacena el hash, haciendo que las contraseñas sean más seguras frente a accesos no autorizados.
- Verificación: Para verificar una contraseña, se genera el hash de la entrada proporcionada y se compara con el hash almacenado.

```
-- 1. Crear usuarios
CREATE USER 'admin_h'@'localhost' IDENTIFIED  WITH caching_sha2_password BY 'password_admin_h';
CREATE USER 'medico_usuario'@'localhost' IDENTIFIED  WITH caching_sha2_password BY 'password_medico';
CREATE USER 'enfermera_usuario'@'localhost' IDENTIFIED WITH     caching_sha2_password BY 'password_enfermera';
CREATE USER 'administrativo_usuario'@'localhost' IDENTIFIED  WITH caching_sha2_password BY 'password_administrativo';
CREATE USER 'farmaceutico_usuario'@'localhost' IDENTIFIED WITH caching_sha2_password BY 'password_farmaceutico';
CREATE USER 'tecnico_laboratorio_usuario'@'localhost' IDENTIFIED WITH caching_sha2_password BY 'password_tecnico_laboratorio';
```

admin_h	localhost	A005$[□Yv% !vq5oL~jPG3+B RFRiMxGHc4...	caching_sha2_password
administrativo_usuario	localhost	A005$/□(~!G□□YJp"5=50qTf□g9xaEbmz...	caching_sha2_password
enfermera_usuario	localhost	A005$□□g□'z%Q3^r□i□;<4O□XOpE9/...	caching_sha2_password
farmaceutico_usuario	localhost	A005$ □,□m*rzyZm[;a□□An;gfe9G/6WBS...	caching_sha2_password

Conclusiones

Optimización de la Gestión Hospitalaria

- La base de datos estructurada permite un manejo eficiente de la información de pacientes, médicos, citas, historiales médicos, facturación y medicamentos.
- Facilita el acceso rápido a la información médica, reduciendo tiempos de espera y mejorando la atención al paciente.

Eficiencia en la Atención Médica y Administrativa

- Implementación de procedimientos almacenados y triggers para automatizar tareas como la gestión de citas, actualización de historiales médicos y control de medicamentos.
- Uso de vistas optimizadas para consultas rápidas y simplificación del acceso a los datos clave.
- Implementación de índices para mejorar la velocidad en la recuperación de información.

Seguridad y Disponibilidad de la Información

- Definición de roles y permisos para proteger los datos y restringir el acceso según cada perfil de usuario (médico, enfermera, administrativo, farmacéutico, técnico de laboratorio).
- Aplicación de triggers de validación para evitar duplicidad de registros, eliminar pacientes con citas pendientes o prescribir medicamentos vencidos.
- Implementación de auditoría para registrar accesos y monitorear la seguridad de los datos.

Escalabilidad y Futuras Expansiones

- Diseño modular que permite agregar nuevas funcionalidades sin afectar el rendimiento del sistema.
- Uso de estrategias como particionamiento de tablas y optimización de consultas para mejorar la escalabilidad.
- Implementación de respaldos completos, incrementales y en caliente para garantizar la recuperación de datos en caso de fallos.
- Posibilidad de integrar sistemas adicionales, como aplicaciones móviles o servicios en la nube, para ampliar la cobertura y accesibilidad del hospital.