



ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS



BASES DE DATOS (TSDS)

ASIGNATURA:

Bases de Datos

PROFESOR:

Ing. Lorena Chulde

FECHA DE ENTREGA:

2025 - 02-05

PERÍODO ACADÉMICO:

2024-B

TÍTULO

PROYECTO FINAL



Estudiantes

Vargas Iza Johan Sebastián
Simbaña Sanguña Kevin Alexis

Instrucciones Generales

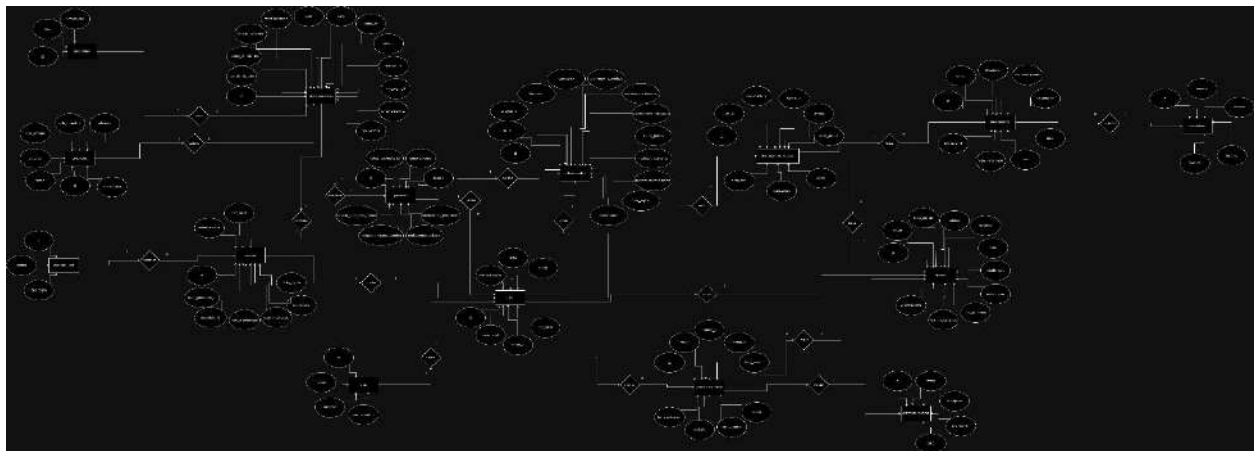
1. Modelado de Base de Datos y Diccionario de Datos

Objetivo: Crear un diseño eficiente y bien documentado para la base de datos, utilizando el modelado ER y un diccionario de datos completo.

Actividades:

Diseñar el modelo conceptual, lógico y físico.

Práctica: Crear un modelo entidad-relación que refleje las entidades clave (como Clientes, Vuelos, Reservas, Pagos) y sus relaciones.



Investigación: Buscar buenas prácticas sobre cómo hacer escalables los modelos de bases de datos para sistemas de reservas en hospitales.

Diseñar una base de datos escalable para un sistema de reservas en hospitales es esencial para garantizar un rendimiento eficiente y una experiencia de usuario satisfactoria. A continuación, se presentan algunas buenas prácticas clave para lograrlo:

1. **Normalización de Datos:** Organiza la información en tablas relacionadas para minimizar la redundancia y mejorar la integridad de los datos. Esto facilita la gestión y actualización de la información.
2. **Uso de Índices:** Implementa índices en columnas que se consultan con frecuencia para acelerar las operaciones de lectura. Sin embargo, es importante equilibrar el uso de índices, ya que pueden afectar el rendimiento de las operaciones de escritura.
3. **Particionamiento de Tablas:** Divide tablas grandes en particiones más pequeñas basadas en criterios como rangos de fechas o identificadores de pacientes. Esto

mejora el rendimiento de las consultas y facilita la gestión de grandes volúmenes de datos.

4. **Optimización de Consultas:** Escribe consultas eficientes y utiliza técnicas como la paginación para manejar grandes conjuntos de resultados. Evita operaciones costosas y asegúrate de que las consultas estén bien indexadas.
5. **Escalabilidad Horizontal:** Considera la posibilidad de distribuir la carga de trabajo entre múltiples servidores de bases de datos para manejar un mayor volumen de usuarios y transacciones simultáneas.
6. **Caché de Resultados:** Implementa mecanismos de caché para almacenar resultados de consultas frecuentes, reduciendo la carga en la base de datos y mejorando los tiempos de respuesta.
7. **Seguridad y Cumplimiento:** Asegura que la base de datos cumpla con las normativas de protección de datos personales, como la Ley de Protección de Datos Personales en Posesión de los Particulares en Ecuador. Implementa controles de acceso adecuados y encripta la información sensible.
8. **Monitoreo y Mantenimiento:** Establece procesos de monitoreo para identificar y resolver problemas de rendimiento. Realiza mantenimientos periódicos, como la actualización de estadísticas y la reconstrucción de índices, para mantener la eficiencia del sistema.

Importancia del Conocimiento: Conocer cómo diseñar bases de datos adecuadas asegura la eficiencia y fácil mantenimiento de la aplicación.

1. Desarrollar un diccionario de datos detallado.

Práctica: Crear un diccionario que defina todas las tablas, sus campos, relaciones y restricciones.

areas										
Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓	✓					✓		Identificador único del área médica.
nombre	VARCHAR(50)		✓							Nombre del área.
ubicacion	VARCHAR(50)		✓							Ubicación del área en el hospital.
num_consultorio	INT		✓							Número del consultorio dentro del área.

citas										
Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT		✓					✓		Identificador único de la cita médica.
paciente_id	INT			✓						Clave foránea que referencia al paciente.
medico_id	INT			✓						Clave foránea que referencia al médico.
fecha	DATE			✓						Fecha de la cita.
hora_inicio	TIME			✓						Hora de inicio de la cita.
hora_finalizacion	TIME			✓						Hora de finalización de la cita.
estado	ENUM('Pendiente', 'Completada', 'Cancelada')			✓						Estado de la cita (pendiente, completada, cancelada).
motivo	TEXT								NULL	Motivo de la cita.
area_id	INT								NULL	Clave foránea que referencia al área donde se realiza la cita.

datos personales												
Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment		
id	INT	✓	✓					✓		Identificador único del registro de datos personales.		
tipo_identificacion	ENUM('cédula', 'pasaporte', 'licencia')	✓								Tipo de documento de identificación (cédula, pasaporte, licencia).		
numero_identificacion	VARCHAR(10)	✓								Número de identificación, único en el sistema.		
nombres_completos	VARCHAR(100)	✓								Nombre completo de la persona.		
fecha_nacimiento	DATE	✓								Fecha de nacimiento.		
edad	INT	✓								Edad de la persona.		
sexo	ENUM('M', 'F', 'O')	✓								Sexo de la persona (Masculino, Femenino, Otro).		
estado_civil	VARCHAR(100)								NULL	Estado civil de la persona.		
telefono	VARCHAR(20)	✓								Número de teléfono.		
direccion_id	INT	✓								Clave foránea que referencia a la tabla de direcciones.		
pais_origen_id	INT	✓								Clave foránea que referencia a la tabla de países de origen.		
correo_electronico	VARCHAR(100)	✓										
tipo_persona	ENUM('Paciente', 'Medico')	✓								Especifica si la persona es paciente o médico.		

direcciones												
Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment		
id	INT	✓	✓					✓		Identificador único de la dirección.		
ciudad	VARCHAR(50)	✓								Nombre de la ciudad.		
parroquia	VARCHAR(50)	✓								Nombre de la parroquia.		
calle_principal	VARCHAR(100)	✓								Calle principal de la dirección.		
calle_secundaria	VARCHAR(100)								NULL	Calle secundaria de la dirección.		
numero_casa	VARCHAR(10)	✓								Número de la casa o edificio.		
referencia	TEXT								NULL	Información adicional sobre la ubicación.		

especialidades												
Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment		
id	INT	✓	✓					✓		Identificador único de la especialidad médica.		
nombre	VARCHAR(100)		✓							Nombre de la especialidad.		
descripcion	TEXT		✓							Descripción de la especialidad.		

exámenes médicos												
Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment		
id	INT	✓	✓					✓		Identificador único del examen médico.		
nombre	VARCHAR(100)		✓							Nombre del examen.		
descripcion	TEXT		✓							Descripción del examen.		
tipo_examen	VARCHAR(50)		✓							Tipo de examen (laboratorio, imagenología, etc.).		
costo	DECIMAL(10,2)		✓							Costo del examen.		

facturas												
Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment		
id	INT		✓	✓				✓				
cita_id	INT		✓									
fecha_emision	DATE		✓									
subtotal	DECIMAL(10,2)		✓									
impuestos	DECIMAL(10,2)		✓									
total	DECIMAL(10,2)		✓									
estado_pago	ENUM('Pagada', 'Pendiente', 'Cancelada')		✓									
metodo_pago	ENUM('Efectivo', 'Tarjeta de Crédito', 'Transferencia Bancaria', 'Seguro Médico')		✓									
total_examenes	DECIMAL(10,2)								'0.00'			
total medicamentos	DECIMAL(10,2)								'0.00'			
observaciones	TEXT								NULL			

historial médico												
Column name	DataType	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment		
id	INT	✓	✓					✓		Identificador único del historial médico.		
cita_id	INT		✓							Clave foránea que referencia a la cita médica.		
paciente_id	INT		✓							Clave foránea que referencia al paciente.		
descripcion	TEXT		✓							Descripción del historial médico.		
diagnostico	TEXT		✓							Diagnóstico realizado.		
tratamiento_recomendado	TEXT								NULL	Tratamiento recomendado.		
medicamentos_recetados	TEXT								NULL	Medicamentos recetados.		
procedimientos_realizados	TEXT								NULL	Procedimientos realizados.		
estado_general	ENUM('Bueno', 'Regular', 'Crítico')		✓							Estado general del paciente (bueno, regular, crítico).		
resultados_examenes	TEXT								NULL	Resultados de exámenes médicos.		
recomendaciones_futuras	TEXT								NULL	Recomendaciones para el futuro.		
fecha_registro	TIMESTAMP		✓						CURRENT_TIMESTAMP	Fecha y hora de registro del historial médico.		
observaciones	TEXT								NULL	Observaciones adicionales.		

medicamentos										
Column name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓	✓					✓		Identificador único del medicamento.
nombre	VARCHAR(100)		✓							Nombre del medicamento.
descripcion	TEXT								NULL	Descripción del medicamento.
tipo_medimento	ENUM('Analgesico', 'Antibiotico', 'Antiinflamatorio', 'Antipirético', 'Otros')		✓							Tipo de medicamento (analgesico, antibiotico, etc.).
presentacion	ENUM('Tabletas', 'Jarabe', 'Inyección', 'Cápsulas')		✓							Forma de presentación del medicamento (tabletas, jarabe, etc.).
dosis	VARCHAR(50)		✓							Dosis recomendada del medicamento.
stock	INT		✓							Cantidad de stock disponible.
fecha_vencimiento	DATE		✓							Fecha de vencimiento del medicamento.
proveedor_id	INT		✓							Clave foránea que referencia al proveedor del medicamento.

medicos										
column name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓	✓					✓		Identificador único del médico.
datos_personales_id	INT		✓							Clave foránea que referencia a los datos personales del médico.
especialidad_id	INT		✓							Clave foránea que referencia a la especialidad del médico.
licencia_profesional	VARCHAR(50)		✓							Número de licencia profesional del médico.
experiencia_years	INT		✓							Años de experiencia del médico.
tipo_contrato	ENUM('Permanente', 'Temporal', 'Por Servicio')		✓							Tipo de contrato del médico.
fecha_ingreso	DATE		✓							Fecha en que el médico ingresó al hospital.
horario_entrada	TIME								NULL	Hora de entrada del médico.
hora_salida	TIME								NULL	Hora de salida del médico.

ordenesexámenes										
Column name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓	✓					✓		Identificador único de la orden examen médico.
cita_id	INT		✓							Clave foránea que referencia a la cita médica.
examen_id	INT		✓							Clave foránea que referencia al examen médico.
factura_id	INT								NULL	Clave foránea que referencia a la factura.
fecha_solicitud	DATE		✓							Fecha de solicitud del examen.
fecha_realizacion	DATE								NULL	Fecha de realización del examen.
resultados	TEXT		✓							Resultados del examen.
costo_examen	DECIMAL(10,2)		✓							Costo del examen.
estado	ENUM('Pendiente', 'Realizado', 'Cancelado')		✓							Registra si la orden del examen está pendiente, realizado o cancelado.

pacientes										
column name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓	✓					✓		Identificador único del paciente.
datos_personales_id	INT		✓							Clave foránea que referencia a los datos personales del paciente.
grupo_sanguineo	ENUM('A+', 'A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-')		✓							Grupo sanguíneo del paciente.
alergias	TEXT								NULL	Alergias del paciente.
enfermedades_preexistentes	TEXT								NULL	Enfermedades preexistentes del paciente.
medicamentos_actuales	TEXT								NULL	Medicamentos actuales del paciente.
contacto_emergencia_nombre	VARCHAR(100)						✓			Nombre del contacto de emergencia.
contacto_emergencia_telefono	VARCHAR(13)						✓			Teléfono del contacto de emergencia.

pais_origen										
Column name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓	✓					✓		Identificador único del país de origen.
pais	VARCHAR(100)		✓							Nombre del país.
nacionalidad	VARCHAR(100)		✓							Nacionalidad asociada al país.

prescripcionesmedicas										
Column name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓	✓					✓		Identificador único de la prescripción médica.
cita_id	INT		✓							Clave foránea que referencia a la cita médica.
medicamento_id	INT		✓							Clave foránea que referencia al medicamento prescrito.
factura_id	INT		✓							Clave foránea que referencia a la factura.
cantidad	INT		✓							Cantidad de medicamento prescrita.
costo_unitario	DECIMAL(10,2)		✓							Costo total de la prescripción (calculado automáticamente).
costo_total	DECIMAL(10,2)								NULL	
instrucciones	TEXT		✓							Instrucciones de uso del medicamento.
estado	ENUM('Pendiente', 'Entregado')		✓							Estado de la prescripción (pendiente, entregado).

proveedores										
column name	Datatype	PK	NN	UQ	BIN	UN	ZF	AI	Default	Comment
id	INT	✓	✓					✓		Identificador único del proveedor.
nombre	VARCHAR(100)		✓							Nombre del proveedor.
contacto	VARCHAR(100)		✓							Persona de contacto del proveedor.
telefono	VARCHAR(20)		✓							Teléfono de contacto del proveedor.
direccion	VARCHAR(255)		✓							Dirección del proveedor.

Investigación: Investigar las mejores herramientas y métodos para generar diccionarios de datos y su uso en proyectos reales.

Un **diccionario de datos** es una herramienta esencial en la gestión de bases de datos, proporcionando una descripción detallada de los elementos de datos, sus relaciones y reglas de negocio. Facilita la comprensión, el mantenimiento y la evolución de los sistemas de información.

Herramientas para Generar Diccionarios de Datos:

Existen diversas herramientas que automatizan la creación y mantenimiento de diccionarios de datos, mejorando la eficiencia y reduciendo errores. A continuación, se presentan algunas de las más destacadas:

Dataedo

Permite documentar bases de datos, generar diccionarios de datos y diagramas entidad-relación. Es compatible con sistemas como Microsoft SQL Server, Oracle y MySQL.

SolarWinds Database Mapper

Facilita la visualización y documentación de la estructura de bases de datos, ayudando a comprender las relaciones entre tablas y objetos.

La implementación de un diccionario de datos en proyectos reales realmente contribuye de forma significativa a la mejora de la calidad de datos. Aquí te detallo cómo se puede lograr una mejora de calidad y otros beneficios que proporciona:

Mejora de la Calidad de Datos:

Un diccionario de datos proporciona un marco de referencia claro que establece cómo se deben almacenar, manejar y validar los datos. Algunos puntos clave que contribuyen a la mejora de la calidad de datos son:

1. **Establecimiento de Estándares de Datos:**

- Se definen convenciones claras para los nombres de los campos, formatos y tipos de datos, lo que garantiza que todos los equipos trabajen con una terminología común y evitando inconsistencias.

2. **Validaciones y Restricciones:**

- El diccionario puede especificar las validaciones necesarias (por ejemplo, restricciones de longitud de cadenas o formatos específicos para correos electrónicos, números de teléfono, etc.), lo que minimiza la introducción de datos erróneos o inconsistentes.

3. **Integridad Referencial:**

- Al especificar las relaciones entre las entidades, como claves primarias y foráneas, se asegura que los datos sean consistentes y completos en todo el sistema. Esto evita la creación de registros huérfanos y asegura que todas las dependencias sean respetadas.

4. **Normas de Enriquecimiento y Limpieza de Datos:**

- El diccionario de datos puede incluir reglas sobre cómo enriquecer los datos y cómo realizar la limpieza de datos (por ejemplo, eliminar duplicados, corregir datos incompletos), lo que resulta en una base de datos más precisa.

5. Reducción de Errores Humanos:

- Al contar con una documentación clara de los datos, las posibilidades de que los desarrolladores, analistas o usuarios finales cometan errores al ingresar o manipular datos disminuyen significativamente.

Importancia del Conocimiento: Un diccionario de datos permite mantener la consistencia y facilita la colaboración entre desarrolladores.

2. Definir las restricciones de integridad referencial.

Práctica: Establecer claves primarias y foráneas entre las tablas, asegurando la coherencia de los datos.

Tabla direcciones	
Restricción	Descripción
PRIMARY KEY	La columna id identifica de forma única cada dirección.
NOT NULL	Las columnas ciudad, parroquia, calle_principal, y numero_casa no pueden ser nulas.

Tabla pais_origen	
Restricción	Descripción
PRIMARY KEY	La columna id identifica cada país de forma única.
NOT NULL	Las columnas pais y nacionalidad son obligatorias.

Tabla datospersonales	
Restricción	Descripción
PRIMARY KEY	La columna id identifica cada registro de forma única.
NOT NULL	Las columnas como tipo_idnetificacion, nombres_completos, fecha_nacimiento, etc., son obligatorias.

UNIQUE (numero_indetificacion, correo_electronico)	Los valores deben ser únicos.
CHECK (edad >= 0)	La edad debe ser positiva.
CHECK (telefono REGEXP '^[0-9]{10}\$')	Valida que el número de teléfono tenga exactamente 10 dígitos.
CHECK (correo_electronico ...)	Valida el formato del correo electrónico.
FOREIGN KEY (direccion_id)	Relaciona la dirección de una persona.
FOREIGN KEY (pais_origen_id)	Relaciona el país de origen con pais_origen.

Tabla especialidades	
Restricción	Descripción
PRIMARY KEY	La columna id identifica de forma única cada especialidad.
NOT NULL	Las columnas nombre y descripcion son obligatorias.

Tablas pacientes	
Restricción	Descripción
PRIMARY KEY	La columna id identifica cada paciente.
NOT NULL	Las columnas datos_personales_id, grupo_sanguineo, y contacto_emergencia_nombre son obligatorias.
CHECK (contacto_emergencia_telefono REGEXP '^[0-9]{10}\$')	Valida el formato del teléfono de contacto de emergencia.
FOREIGN KEY (datos_personales_id)	Relaciona los datos personales con un paciente.

Tabla areas	
Restricción	Descripción

PRIMARY KEY	La columna id identifica cada área de forma única.
NOT NULL	Las columnas nombre, ubicación, y num_consultorio son obligatorias.

tabla citas	
Restricción	Descripción
PRIMARY KEY	La columna id identifica cada cita de forma única.
NOT NULL	Las columnas pacientes_id, medico_id, fecha, hora_inicio, estado, etc., son obligatorias.
FOREIGN KEY (paciente_id)	Relaciona una cita con un paciente.
FOREIGN KEY (medico_id)	Relaciona una cita con un médico.
FOREIGN KEY (area_id)	Relaciona una cita con un área específica.

Tabla historialmedico	
Restricción	Descripción
PRIMARY KEY	La columna id identifica cada historial médico.
NOT NULL	Las columnas cita_id, paciente_id, descripcion, estado_general son obligatorias.
FOREIGN KEY (cita_id)	Relaciona el historial con una cita específica.

FOREIGN KEY (paciente_id)	Relaciona el historial con una cita específica.
------------------------------	---

Tabla facturas	
Restricción	Descripción
PRIMARY KEY	La columna id identifica cada factura.
NOT NULL	Las columnas cita_id, subtotal, impuestos, y total son obligatorias.
CHECK (subtotal, impuestos, total >= 0)	Garantiza que estos valores sean positivos.
FOREIGN KEY (cita_id)	Relaciona una factura con una cita específica.

Tabla examenesmedicos	
Restricción	Descripción
PRIMARY KEY	La columna id identifica cada examen.
NOT NULL	Las columnas nombre, tipo_examen, y costo son obligatorias.
CHECK (costo >= 0)	El costo debe ser positivo o cero.

Tabla ordenesexamenes	
Restricción	Descripción
PRIMARY KEY	La columna id identifica cada orden de examen.
NOT NULL	Las columnas cita_id, examen_id, fecha solicitud, costo_examen, y estado son obligatorias.

FOREIGN KEY (cita_id)	Relaciona una orden con una cita específica.
FOREIGN KEY (examen_id)	Relaciona una orden con un examen médico.
CHECK (costo_examen >= 0)	Asegura que el costo del examen sea positivo o cero.

Tabla medicamentos	
Restricción	Descripción
PRIMARY KEY	La columna id identifica cada medicamento de forma única.
NOT NULL	Las columnas nombre, tipo_medicamento, dosis, stock, etc., son obligatorias.
CHECK (stock >= 0)	El stock debe ser positivo o cero.
FOREIGN KEY (proveedor_id)	Relaciona un medicamento con un proveedor.

Tabla proveedores	
Restricción	Descripción
PRIMARY KEY	La columna id identifica cada proveedor de forma única.
NOT NULL	Las columnas nombre, contacto, telefono, y direccion son obligatorias.

prescripcionesmedicas	
Restricción	Descripción
PRIMARY KEY	La columna id identifica cada prescripción médica.
NOT NULL	Las columnas cita_id, medicamento_id, cantidad, instrucciones, y estado son obligatorias.

FOREIGN KEY (cita_id)	Relaciona una prescripción con una cita específica.
FOREIGN KEY (medicamento id)	Relaciona una prescripción con un medicamento.
CHECK (cantidad > 0)	Asegura que la cantidad sea mayor a cero.
CHECK (costo_unitario >= 0)	El costo unitario debe ser positivo o cero.

Investigación: Investigar cómo la integridad referencial puede prevenir la pérdida de datos y mantener la consistencia.

La integridad referencial es crucial para mantener la salud de tu base de datos. Aquí te presentamos algunas razones por las que es tan importante:

Evita inconsistencias de datos. ¿Alguna vez has intentado buscar un dato en una tabla, solo para encontrar que la clave principal que apunta a él ya no existe en la tabla principal? Eso es lo que la integridad referencial ayuda a evitar. Sin ella, podrías terminar con una base de datos llena de datos huérfanos, lo que puede llevar a confusiones y errores.

Mantiene la relación entre las tablas. En una base de datos relacional, las tablas están, valga la redundancia, relacionadas entre sí. Estas relaciones son lo que permiten que los datos fluyan de una tabla a otra de manera coherente. Al asegurar que la clave foránea siempre apunta a una clave primaria válida, la integridad referencial asegura que estas relaciones se mantienen.

Es esencial para el funcionamiento de las bases de datos. Cada vez que realizas una operación en tu base de datos, estás confiando en la integridad referencial. Cada vez que agregas, actualizas o eliminas datos, estás contando con el hecho de que las claves foráneas y las claves primarias se mantengan en sincronía. Sin la integridad referencial, tus operaciones de base de datos podrían convertirse en un auténtico caos.

La eliminación en cascada

La eliminación en cascada es un subproducto de la integridad referencial. Esta permite que, cuando borras un registro de la tabla principal, todos los registros correspondientes en las tablas secundarias se borren automáticamente. Aunque puede

parecer útil, es importante tener cuidado. No siempre es la mejor opción y puede llevar a la pérdida de datos si no se utiliza correctamente. Asegúrate de entender bien sus implicaciones antes de decidir utilizar la eliminación en cascada en tus bases de datos.

Importancia del Conocimiento: Las restricciones de integridad aseguran que los datos no se corrompan.

2. Seguridad, Auditoría y Control de Acceso

Objetivo: Proteger los datos sensibles y controlar el acceso a la base de datos.

Actividades:

1. Implementar políticas de acceso y seguridad.

Práctica: Crear roles y permisos de usuario (por ejemplo, roles de Administrador, Usuario, Auditor) para controlar el acceso a las tablas y vistas.

```
-- 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';

-- 2. Asignar permisos a los usuarios
-- Permisos para el usuario admin_h
GRANT ALL PRIVILEGES ON hospital_buen_dia.* TO 'admin_h'@'localhost' WITH GRANT OPTION;

-- Permisos para el usuario medicos
GRANT USAGE ON hospital_buen_dia.* TO 'medico_usuario'@'localhost';
GRANT SELECT, UPDATE ON hospital_buen_dia.pacientes TO 'medico_usuario'@'localhost';
GRANT SELECT ON hospital_buen_dia.citas TO 'medico_usuario'@'localhost';
GRANT SELECT, INSERT, UPDATE ON hospital_buen_dia.historialmedico TO 'medico_usuario'@'localhost';

-- Permisos para el usuario enfermeras
GRANT USAGE ON hospital_buen_dia.* TO 'medico_usuario'@'localhost';
GRANT SELECT ON hospital_buen_dia.citas TO 'enfermera_usuario'@'localhost';
GRANT SELECT ON hospital_buen_dia.historialmedico TO 'enfermera_usuario'@'localhost';
GRANT SELECT ON hospital_buen_dia.pacientes TO 'enfermera_usuario'@'localhost';

-- Permisos para el usuario personal_administrativo
GRANT USAGE ON hospital_buen_dia.* TO 'administrativo_usuario'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON hospital_buen_dia.datospersonales TO 'administrativo_usuario'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON hospital_buen_dia.citas TO 'administrativo_usuario'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON hospital_buen_dia.medicos TO 'administrativo_usuario'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON hospital_buen_dia.pacientes TO 'administrativo_usuario'@'localhost';
GRANT SELECT, INSERT, UPDATE, DELETE ON hospital_buen_dia.factura TO 'administrativo_usuario'@'localhost';
```

```
-- Permisos para el farmaceuticos
GRANT USAGE ON hospital_buen_dia.* TO 'farmaceutico_usuario'@'localhost';
GRANT SELECT ON hospital_buen_dia.pacientes TO 'farmaceutico_usuario'@'localhost';
GRANT SELECT ON hospital_buen_dia.medicos TO 'farmaceutico_usuario'@'localhost';
GRANT SELECT ON hospital_buen_dia.citas TO 'farmaceutico_usuario'@'localhost';
GRANT SELECT, UPDATE ON hospital_buen_dia.medicamentos TO 'farmaceutico_usuario'@'localhost';

-- Permisos para el usuario tecnicos_laboratorio
GRANT USAGE ON hospital_buen_dia.* TO 'tecnico_laboratorio_usuario'@'localhost';
GRANT SELECT ON hospital_buen_dia.pacientes TO 'tecnico_laboratorio_usuario'@'localhost';
GRANT SELECT ON hospital_buen_dia.citas TO 'tecnico_laboratorio_usuario'@'localhost';
GRANT SELECT ON hospital_buen_dia.medicos TO 'tecnico_laboratorio_usuario'@'localhost';
GRANT SELECT, INSERT, UPDATE ON hospital_buen_dia.examenesmedicos TO 'tecnico_laboratorio_usuario'@'localhost';
```

Investigación: Investigar sobre los mejores enfoques para la seguridad en bases de datos en entornos de alta disponibilidad.

Monitorización y gestión proactiva

La monitorización continua y la gestión proactiva son esenciales para mantener la alta disponibilidad. Aquí hay algunas prácticas recomendadas:

Implementar monitorización integral: Utilizar herramientas de monitorización que proporcionen visibilidad completa sobre el estado y el rendimiento de los recursos de la infraestructura. Esto incluye servidores, bases de datos, aplicaciones y redes.

Configurar alertas proactivas: Establecer alertas basadas en umbrales definidos para detectar problemas potenciales antes de que afecten la disponibilidad. Las alertas deben ser configuradas para notificar a los equipos de TI de manera oportuna.

Automatización de respuestas: Utilizar herramientas de automatización para ejecutar respuestas predefinidas a eventos de monitorización. Por ejemplo, reiniciar servicios, escalar recursos o iniciar scripts de reparación automatizados.

Análisis de tendencias y prevención de problemas: Analizar los datos de monitorización para identificar tendencias y patrones que puedan indicar problemas futuros. Implementar acciones preventivas basadas en estos análisis para evitar fallos.

Importancia del Conocimiento: El control adecuado de acceso previene fugas de información y mejora la seguridad general.

2. Cifrado de datos sensibles.

Práctica: Cifrar información sensible como contraseñas y detalles de pago (por ejemplo, usando AES_ENCRYPT en MySQL).

```
/*cifrado de informacion*/
-- Definir la clave de cifrado para cada usuario
SET @clave_admin = 'claveAdminSegura!';
SET @clave_medico = 'claveMedico123!';
SET @clave_farmaceutico = 'claveFarmacia$!';
SET @clave_tecnico = 'claveTecnico#2023';

-- Aplicar el cifrado por usuario en la tabla `datospersonales`
UPDATE datospersonales
SET nombre = AES_ENCRYPT(nombre, @clave_admin),
    telefono = AES_ENCRYPT(telefono, @clave_admin);

-- Cifrado de información confidencial en la tabla `pacientes`
UPDATE pacientes
SET nombre = AES_ENCRYPT(nombre, @clave_medico),
    cedula = AES_ENCRYPT(cedula, @clave_admin);

-- Cifrado en la tabla `medicos`
UPDATE medicos
SET nombre = AES_ENCRYPT(nombre, @clave_admin),
    especialidad = AES_ENCRYPT(especialidad, @clave_admin);

-- Cifrado en `historialmedico` solo para médicos
UPDATE historialmedico
SET diagnostico = AES_ENCRYPT(diagnostico, @clave_medico);

-- Cifrado de detalles en `factura` solo accesible por personal administrativo
UPDATE factura
SET detalles_pago = AES_ENCRYPT(detalles_pago, @clave_admin);
```

Investigación: Explorar algoritmos de cifrado y su impacto en el rendimiento de la base de datos.

El cifrado de datos es un método para convertir datos de un formato legible (texto simple) a un formato codificado ilegible (texto cifrado). Los datos cifrados solo se pueden leer o procesar después de haberlos descifrado, mediante una clave o contraseña de descifrado. Solo el remitente y el destinatario de los datos deben tener acceso a la clave de descifrado.

El estándar de cifrado de datos (DES) es un algoritmo de cifrado simétrico que ya no está actualizado: se utiliza la misma clave para cifrar y descifrar un mensaje. DES utiliza una clave de cifrado de 56 bits (se eliminan 8 bits de paridad de la clave completa de 64 bits) y cifra los datos en bloques de 64 bits. Estos tamaños no suelen ser lo suficientemente grandes para los usos actuales. Por lo tanto, otros algoritmos de cifrado han sustituido a DES:

Triple DES : antiguamente era el algoritmo simétrico estándar. Triple DES emplea tres claves individuales de 56 bits cada una. La longitud total de la clave suma 168 bits, pero según la mayoría de los expertos, su fuerza de clave efectiva es de solo 112 bits.

RSA : un algoritmo de cifrado de clave pública (asimétrico) muy popular. Utiliza un par de claves: la clave pública, que se utiliza para cifrar el mensaje, y la clave privada, que se utiliza para descifrarlo.

Blowfish : un algoritmo de cifrado simétrico que divide los mensajes en bloques de 64 bits y los encripta de uno en uno. Blowfish es un algoritmo antiguo que todavía es eficaz, pero que ha sido reemplazado por Twofish.

Twofish : un sistema de cifrado simétrico que utiliza claves de hasta 256 bits de longitud. Twofish se utiliza en muchos entornos de software y hardware. Es rápido, está disponible de forma gratuita y no está patentado.

Estándar de cifrado avanzado (AES) : este algoritmo es el estándar aceptado actualmente por el gobierno de los EE. UU. UU. y otras organizaciones. Funciona bien en formato de 128 bits, sin embargo, AES puede utilizar claves de 192 y 256 bits. Se considera que AES es resistente a todos los ataques, excepto a los de fuerza bruta.

Criptografía de curva elíptica (ECC) : algoritmo utilizado como parte del protocolo SSL/TLS que encripta la comunicación entre los sitios web y sus visitantes. Proporciona mayor seguridad con longitudes de claves más cortas; una clave ECC de 256 bits proporciona el mismo nivel de seguridad que una clave RSA de 3072 bits.

Importancia del Conocimiento: El cifrado es esencial para la protección de la información confidencial de los usuarios.

3. **Habilitar auditoría y registrar eventos de base de datos.**

Práctica: Activar los logs de acceso y auditoría para monitorear las actividades de los usuarios (por ejemplo, registrar quién accedió a qué datos).

Investigación: Buscar cómo configurar herramientas de auditoría en MySQL o PostgreSQL.

Paso 1: Instalar el complemento MySQL Enterprise Audit

Primero, conéctese a su servidor MySQL utilizando el cliente de línea de comandos MySQL o cualquier herramienta de administración MySQL:

```
mysql -u raíz -p
```

Luego, instale el complemento de auditoría ejecutando el siguiente comando:

```
$ INSTALAR PLUGIN audit_log SONAME 'audit_log.so' ;
```

Paso 2: Verificar la instalación del complemento

Asegúrese de que el complemento esté instalado correctamente ejecutando:

```
MOSTRAR PLUGINS;
```

Debería ver audit_log entre los complementos instalados.

Paso 3: Configurar el registro de auditoría

De forma predeterminada, el archivo de registro de auditoría se escribe en el directorio de datos de MySQL. Para especificar una ubicación personalizada para el archivo de registro de auditoría, agregue la siguiente línea a su archivo de configuración de MySQL (my.cnf o my.ini):

```
[mysqld]  
archivo_registro_auditoría =/var/log/mysql/audit.log
```

Después de realizar este cambio, reinicie el servidor MySQL para aplicar la configuración:

```
$ sudo systemctl reiniciar mysqld
```

Paso 4: Habilitar y configurar la auditoría

Habilite el complemento de auditoría configurando la política de registro de auditoría. Puede elegir registrar todas las actividades o actividades específicas. Por ejemplo, para registrar todas las actividades, ejecute:

```
ESTABLECER GLOBAL audit_log_policy = 'TODOS' ;
```

— Opciones: 'TODOS', 'INICIAR SESIONES', 'CONSULTAS', 'NINGUNA'

Verifique el estado del registro de auditoría con:

MOSTRAR VARIABLES COMO 'audit%' ;

Paso 5: Personalizar las reglas de auditoría (opcional)

Puede crear reglas de auditoría personalizadas para registrar eventos específicos. Por ejemplo, para registrar todas las consultas SELECT, siga estos pasos:

```
ESTABLECER @filter_name = 'log_select';  
ESTABLECER @filter_rule = '{\"filtro\": {\"registro\": verdadero, \"evento\": \"CONSULTA\",  
\"seleccionar\": {\"coincidencia_cualquiera\": \"SELECCIONAR\"}}}' ;  
LLAMAR audit_log_filter_set_filter(@filter_name, @filter_rule);
```

Asignar el filtro a los usuarios:

```
LLAMAR audit_log_filter_set_user('%', @filter_name);
```

Importancia del Conocimiento: La auditoría permite rastrear cambios en los datos y detectar actividades sospechosas.

3. RespalDOS y Recuperación de Datos

Objetivo: Asegurar la integridad y disponibilidad de los datos mediante técnicas de respaldo confiables.

Actividades:

1. Crear respaldos completos (full backups).

Práctica: Utilizar mysqldump o herramientas similares para hacer respaldos completos de la base de datos.

Verificación de nuestra base de datos.

```

MySQL [(none)]> show databases;
+-----+
| Database |
+-----+
| bk_lbreria2 |
| bk_libreria1 |
| concesionarios_ambacar |
| data_personas |
| ecommerce |
| empresa2 |
| hospital_buen_di |
| hospital_buen_dia |
| information_schema |
| inmobiliaria |
| inmobiliaria2 |
| library |
| megamercado |
| mysql |
| performance_schema |
| ping_pong |
| prueba_libreria |
| redes_sociales |
| respaldo_empresa |
| respaldo_libreria |
| restricciones |
| sakila |
| sys |
| world |
+-----+
24 rows in set (0.012 sec)

MySQL [(none)]> use hospital_buen_dia;
Database changed
MySQL [hospital_buen_dia]> |

```

```

C:\xampp\mysql\bin>mysqldump -h localhost -u root -p hospital_buen_dia > bk_hospital_buen_dia.sql
Enter password: *****

```

- **-h localhost:** especifica el host al que deseas conectarte (en este caso, localhost para la conexión local).
- **-u root:** indica el usuario con el que deseas conectarte (en este caso, el usuario root).
- **-p:** te pedirá la contraseña de ese usuario.
- **hospital_buen_dia:** es el nombre de la base de datos de la cual deseas hacer el volcado.

bk_hospital_buen_dia.sql: redirige la salida a un archivo llamado bk_hospital_buen_dia.sql, donde se guardará el volcado de la base de datos.

ana_read_log	30/10/2023 17:33	Aplicación	4.371 KB
bk_hospital_buen_dia	6/2/2025 11:26	SQL Text File	65 KB

Investigación: Buscar estrategias de respaldo para bases de datos de gran tamaño y la mejor manera de gestionarlas.

Copia de seguridad incremental

Con este método, solo se guardan los cambios realizados desde la última copia de seguridad completa. Esto significa copias de seguridad más rápidas y que requieren menos almacenamiento. Sin embargo, la restauración de datos puede ser más compleja con este método, ya que requiere la recuperación de la última copia de seguridad completa, así como de todas las copias de seguridad incrementales posteriores.

Programación de copias de seguridad

Programe copias de seguridad completas durante los periodos de bajo uso, realice copias de seguridad incrementales diariamente o con mayor frecuencia si sus datos cambian con rapidez, y realice copias de seguridad diferenciales semanal o quincenalmente para equilibrar las cargas de tráfico de la red.

Importancia del Conocimiento: Los respaldos completos permiten restaurar toda la base de datos ante una falla.

2. Configurar respaldos incrementales.

Práctica: Realizar respaldos incrementales para reducir el tiempo y espacio de almacenamiento.

Investigación: Investigar cómo realizar respaldos incrementales y cuándo es más conveniente utilizarlos.

Proceso de realización de un backup incremental

El proceso para realizar un backup incremental consta de los siguientes pasos:

Realizar una copia completa inicial de todos los datos y archivos.

Identificar y respaldar únicamente los archivos que han sufrido modificaciones desde la última copia.

Actualizar y guardar los cambios en una ubicación segura destinada al almacenamiento de los backups.

Es importante tener en cuenta que el backup incremental se debe realizar de forma periódica para asegurar la protección de los datos más actualizados y minimizar la pérdida de información en caso de fallos o errores.

Como lo hemos mencionado, el respaldo incremental a menudo es más veloz y ocupa menos espacio de almacenamiento. Las empresas que deseen optimizar el espacio de almacenamiento y evitar añadir grandes volúmenes de datos modificados al almacenamiento desde el último respaldo obtendrán mayores beneficios de los respaldos incrementales. Sin embargo, ante una situación de recuperación ante desastres, la recuperación de respaldo incremental tomará más tiempo que una recuperación mediante respaldo diferencial. Esto se puede traducir en mayor tiempo de inactividad y una reanudación más lenta del proceso empresarial.

Importancia del Conocimiento: Los respaldos incrementales permiten optimizar los recursos y acelerar los tiempos de recuperación.

3. Implementar respaldos en caliente (Hot Backups).

Práctica: Hacer respaldos sin interrumpir el servicio (por ejemplo, usando Percona XtraBackup).

Investigación: Investigar cómo hacer respaldos sin detener la base de datos.

Las copias de seguridad en caliente se realizan mientras las bases de datos permanecen en línea y accesibles, lo que minimiza el tiempo de inactividad, pero requiere recursos adicionales del sistema. Este enfoque es adecuado para empresas con demandas de alta disponibilidad, ya que admite la interacción del usuario incluso durante las operaciones de copia de seguridad. Sin embargo, las copias de seguridad en caliente pueden generar una sobrecarga de rendimiento, lo que hace que la gestión de recursos sea fundamental.

Copias de seguridad físicas con MySQL Enterprise Backup

MySQL Enterprise Backup (MEB), una solución comercial incluida en MySQL Enterprise Edition y MySQL Cluster CGE de Oracle, realiza copias de seguridad físicas y captura copias exactas de los archivos de base de datos para lograr una recuperación rápida y confiable. MEB admite copias de seguridad en vivo y basadas en instantáneas, lo que permite a las organizaciones minimizar el tiempo de inactividad. Al centrarse en la duplicación de archivos físicos, MEB funciona bien con grandes conjuntos de datos, lo que garantiza tiempos de recuperación rápidos y consistentes incluso en entornos complejos.

MEB gestiona grandes volúmenes de transacciones de manera eficiente, aprovechando el procesamiento paralelo y las funciones de optimización para administrar el uso de los recursos. Sin embargo, la implementación de MEB puede ser compleja y requiere un conocimiento profundo de las configuraciones del servidor, ya que la ejecución precisa es crucial para realizar copias de seguridad exitosas.

Importancia del Conocimiento: Los respaldos en caliente son esenciales para bases de datos de producción que no pueden permitirse inactividad.

4. Optimización y Rendimiento de Consultas

Objetivo: Mejorar la eficiencia en la recuperación de datos mediante la optimización de consultas y el uso adecuado de índices.

Actividades:

1. Crear y gestionar índices.

Práctica: Implementar índices en las columnas más consultadas, como VueloID, ClienteID, etc.

```
/* 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);

/* ordenes de exámenes*/
CREATE INDEX idx_cita_id_orden ON ordenesexámenes(cita_id);
CREATE INDEX idx_examen_id ON ordenesexámenes(examen_id);
CREATE INDEX idx_estado_orden ON ordenesexámenes(estado);

/*preinscripcion_medica*/
CREATE INDEX idx_cita_id_prescripcion ON prescripcionesMedicas(cita_id);
CREATE INDEX idx_medimento_id ON prescripcionesMedicas(medimento_id);
CREATE INDEX idx_estado_prescripcion ON prescripcionesMedicas(estado);

/*Tabla_factura*/
CREATE INDEX idx_fecha_emision_factura ON facturas(fecha_emision);
CREATE INDEX idx_estado_factura ON facturas(estado_pago);
CREATE INDEX idx_cita_id_factura ON facturas(cita_id);

```

Investigación: Investigar sobre los tipos de índices más adecuados para bases de datos transaccionales y cómo afectan el rendimiento.

Existen varios tipos de índices de tablas en bases de datos que se pueden utilizar en función de nuestras necesidades. Los más comunes son el índice agrupado y el índice único:

Índice agrupado: en este tipo de índice, las filas de datos se almacenan en la tabla a partir de la clave del índice agrupado. Cada tabla puede tener solo un índice agrupado.

Índice único: este índice asegura que todos los valores en la columna del índice son únicos. Puedes tener varios índices únicos en una tabla, pero solo uno puede ser la clave primaria.

Se emplea un índice SQL para poder recuperar datos de una base de datos de una manera más rápida. El indexar una tabla o la vista es sin lugar a dudas, una de las mejores opciones de poder mejorar el rendimiento de las consultas y aplicaciones.

Un índice SQL es una tabla de búsqueda rápida para poder encontrar los registros que los usuarios necesitan buscar con mayor frecuencia. Ya que un índice es pequeño, rápido y optimizado para búsquedas rápidas. Además, que son muy útiles para conectar las tablas relacionales y la búsqueda de tablas grandes.

Importancia del Conocimiento: Los índices son cruciales para acelerar las consultas y mejorar el rendimiento general de la base de datos.

2. Optimizar consultas SQL.

Práctica: Utilizar herramientas como EXPLAIN para identificar cuellos de botella en las consultas y optimizarlas.

```
22 • EXPLAIN SELECT * FROM citas WHERE estado = 'Pendiente';
23
--
```

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
►	1	SIMPLE	citas	NULL	ref	idx_estado	idx_estado	1	const	2	100.00	Using index condition

```
24 • EXPLAIN ANALYZE SELECT * FROM citas WHERE paciente_id = 5;
25
--
```

	EXPLAIN
►	-> Index lookup on citas using fk_paciente_cita (paciente_id=5) (cost=0.35 rows=1) (actual time=0.0313..0.0375 rows=1 loops=1)

Investigación: Investigar cómo hacer uso eficiente de las uniones (JOIN), subconsultas, y optimizar las consultas complejas.

Las uniones se utilizan en consultas SQL para combinar datos de varias tablas en función de una columna relacionada entre ellas. Las subconsultas, por otro lado, son consultas anidadas dentro de una consulta principal que se pueden utilizar para recuperar datos de una sola tabla o de varias tablas.

Si bien las uniones y subconsultas son herramientas poderosas para recuperar y manipular datos, también pueden generar problemas de rendimiento si no se utilizan correctamente. Las uniones y subconsultas ineficientes pueden generar tiempos de ejecución de consultas lentos, consumo de recursos y un rendimiento deficiente de la aplicación en general.

Prácticas recomendadas para optimizar uniones y subconsultas

Evite las subconsultas anidadas

Las subconsultas anidadas pueden resultar ineficientes y ralentizar el rendimiento de las consultas. En lugar de utilizar subconsultas anidadas, considere la posibilidad de utilizar operaciones JOIN para combinar datos de varias tablas. Las uniones suelen ser más eficientes y pueden dar como resultado un mejor rendimiento en comparación con las subconsultas anidadas.

Optimizar las condiciones de unión

Optimizar las condiciones de unión es esencial para mejorar el rendimiento de las consultas. Asegúrese de que las condiciones de unión estén indexadas correctamente y utilice el tipo de unión adecuado (por ejemplo, INNER JOIN, LEFT JOIN, RIGHT JOIN) según sus requisitos específicos. Evite condiciones de unión innecesarias o redundantes que puedan aumentar la complejidad de la consulta y afectar el rendimiento.

Limitar el número de uniones

Reducir la cantidad de uniones en las consultas SQL puede ayudar a mejorar el rendimiento. Evite unir tablas innecesarias o incluir columnas innecesarias en la instrucción SELECT. Mantenga las consultas simples e incluya solo las uniones y columnas esenciales necesarias para recuperar los datos deseados.

Comprensión de las uniones SQL

Las uniones SQL se utilizan para combinar filas de dos o más tablas en función de una columna relacionada entre ellas. Existen varios tipos de uniones, entre ellos:

Unión interna: devuelve filas cuando hay al menos una coincidencia en ambas tablas

- Unión izquierda (o unión externa izquierda): devuelve todas las filas de la tabla izquierda y las filas coincidentes de la tabla derecha
- Unión derecha (o unión externa derecha): devuelve todas las filas de la tabla derecha y las filas coincidentes de la tabla izquierda
- Unión completa (o unión externa completa): devuelve filas cuando hay una coincidencia en una de las tablas

Las uniones son esenciales para recuperar datos de varias tablas, pero también pueden afectar el rendimiento de las consultas SQL, especialmente cuando se trabaja con grandes conjuntos de datos. Al optimizar las uniones y las subconsultas, puede mejorar la eficiencia de las consultas y reducir el tiempo que lleva recuperar los resultados.

Importancia del Conocimiento: Las consultas optimizadas aseguran un sistema rápido y eficiente, especialmente en sistemas con alta demanda.

3. Utilizar particionamiento de tablas.

Práctica: Dividir tablas grandes, como Reservas, en particiones según una clave (por ejemplo, por fecha).

```

48 • CREATE TABLE citas_particionadas (
49     id INT NOT NULL,
50     paciente_id INT NOT NULL,
51     medico_id INT NOT NULL,
52     fecha DATE NOT NULL,
53     hora_inicio TIME NOT NULL,
54     hora_finalizacion TIME NOT NULL,
55     estado ENUM('Pendiente', 'Completada', 'Cancelada') NOT NULL,
56     motivo TEXT,
57     area_id INT,
58     PRIMARY KEY (id, fecha)
59 ) PARTITION BY RANGE (YEAR(fecha)) (
60     PARTITION p_2022 VALUES LESS THAN (2023),
61     PARTITION p_2023 VALUES LESS THAN (2024),
62     PARTITION p_2024 VALUES LESS THAN (2025),
63     PARTITION p_futuras VALUES LESS THAN (MAXVALUE)
64 );
65
66
67 • SELECT * FROM citas_particionadas;
68 • SELECT * FROM citas_particionadas WHERE YEAR(fecha) = 2023;
69
70

```

Result Grid									
Filter Rows:									
	id	paciente_id	medico_id	fecha	hora_inicio	hora_finalizacion	estado	motivo	area_id
▶	6	106	206	2023-02-10	11:00:00	11:30:00	Completada	Dolor abdominal	3
	7	107	207	2023-05-15	13:00:00	13:45:00	Pendiente	Chequeo anual	1
	8	108	208	2023-07-22	15:30:00	16:00:00	Cancelada	Fiebre y malestar	2
	9	109	209	2023-09-30	17:00:00	17:30:00	Completada	Examen de sangre	3
	10	110	210	2023-12-05	08:00:00	08:30:00	Pendiente	Consulta oftalmológica	1
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Investigación: Investigar sobre los beneficios del particionamiento y cómo implementarlo en sistemas de bases de datos grandes.

La partición de bases de datos es el proceso de dividir una base de datos grande en segmentos más pequeños y manejables, denominados particiones. Cada partición contiene un subconjunto de los datos y puede almacenarse en un dispositivo físico o servidor diferente. Al particionar una base de datos, resulta más fácil administrarla y escalarla, ya que cada partición se puede optimizar para consultas o cargas de trabajo específicas.

La partición de bases de datos puede proporcionar varios beneficios para la gestión de grandes conjuntos de datos. Al particionar los datos, las consultas se pueden dirigir a particiones específicas que contienen datos relevantes, lo que reduce la cantidad de procesamiento de datos innecesarios y el tráfico de red. Esto puede generar tiempos de respuesta de consultas más rápidos y un mejor rendimiento general de la base de datos. Además, a medida que aumentan los volúmenes de datos, la partición permite una distribución y replicación de datos eficientes, lo que mejora la escalabilidad y la confiabilidad.

La partición de bases de datos es particularmente importante para las organizaciones que manejan grandes cantidades de datos, como sitios web de comercio electrónico, instituciones financieras y plataformas de redes sociales, ya que les permite almacenar y administrar datos de manera más eficiente. Sin embargo, no todas las bases de datos requieren particionamiento. Es importante evaluar el tamaño y la complejidad de la base de datos, y los requisitos de la empresa, para determinar si es necesario.

Si bien la partición de bases de datos puede ofrecer beneficios significativos, es importante evitar la ingeniería excesiva. Algunas organizaciones pueden implementar particiones innecesariamente, lo que genera mayor complejidad y costos. Es esencial evaluar los beneficios y costos de la partición y elegir el método adecuado para el caso de uso específico.

Importancia del Conocimiento: El particionamiento de tablas mejora la escalabilidad y el rendimiento en bases de datos con gran volumen de datos.

5. Procedimientos Almacenados, Vistas y Triggers

Objetivo: Mejorar la eficiencia y automatizar tareas mediante el uso de procedimientos almacenados, vistas y triggers.

Actividades:

1. Crear procedimientos almacenados.

Práctica: Crear un procedimiento para calcular el precio total de una reserva, aplicando descuentos y cargos adicionales.

```
-- Registrar nueva cita
DELIMITER //
CREATE PROCEDURE RegistrarCita(
    IN p_paciente_id INT,
    IN p_medico_id INT,
    IN p_fecha DATE,
    IN p_hora_inicio TIME,
    IN p_hora_finalizacion TIME,
    IN p_motivo TEXT,
    IN p_area_id INT
)
BEGIN
    INSERT INTO citas (paciente_id, medico_id, fecha, hora_inicio, hora_finalizacion, estado, motivo, area_id)
    VALUES (p_paciente_id, p_medico_id, p_fecha, p_hora_inicio, p_hora_finalizacion, 'Pendiente', p_motivo, p_area_id);
END;
//
DELIMITER ;
```

```
-- Registrar historial médico
DELIMITER //
CREATE PROCEDURE RegistrarHistorialMedico(
    IN p_cita_id INT,
    IN p_paciente_id INT,
    IN p_descripcion TEXT,
    IN p_diagnostico TEXT,
    IN p_tratamiento TEXT,
    IN p_medicamentos TEXT,
    IN p_estado_general ENUM('Bueno', 'Regular', 'Crítico')
)
BEGIN
    INSERT INTO historialmedico (cita_id, paciente_id, descripcion, diagnostico, tratamiento_recomendado, medicamentos_recetados, estado_general)
    VALUES (p_cita_id, p_paciente_id, p_descripcion, p_diagnostico, p_tratamiento, p_medicamentos, p_estado_general);
END;
//
DELIMITER ;
```

```
-- Registrar resultado de examen médico
DELIMITER //
CREATE PROCEDURE RegistrarResultadoExamen(
    IN p_examen_id INT,
    IN p_fecha_realizacion DATE,
    IN p_resultados TEXT
)
BEGIN
    UPDATE ordenesexámenes
    SET fecha_realizacion = p_fecha_realizacion, resultados = p_resultados, estado = 'Realizado'
    WHERE id = p_examen_id;
END;
//
DELIMITER ;
```

```
-- Registrar prescripción médica
DELIMITER //
CREATE PROCEDURE RegistrarPrescripcion(
    IN p_cita_id INT,
    IN p_medicamento_id INT,
    IN p_cantidad INT,
    IN p_costo_unitario DECIMAL(10,2),
    IN p_instrucciones TEXT
)
BEGIN
    INSERT INTO prescripcionesmedicas (cita_id, medicamento_id, cantidad, costo_unitario, instrucciones, estado)
    VALUES (p_cita_id, p_medicamento_id, p_cantidad, p_costo_unitario, p_instrucciones, 'Pendiente');
END;
//
DELIMITER ;
```

```
-- Registrar nuevo paciente
DELIMITER //
CREATE PROCEDURE RegistrarPaciente(
    IN p_datos_personales_id INT,
    IN p_grupo_sanguineo ENUM('A+', 'A-', 'B+', 'B-', 'AB+', 'AB-', 'O+', 'O-'),
    IN p_alergias TEXT,
    IN p_enfermedades_preexistentes TEXT,
    IN p_medicamentos_actuales TEXT,
    IN p_contacto_emergencia_nombre VARCHAR(100),
    IN p_contacto_emergencia_telefono VARCHAR(13)
)
BEGIN
    INSERT INTO pacientes (datos_personales_id, grupo_sanguineo, alergias, enfermedades_preexistentes, medicamentos_actuales, contacto_emergencia_nombre, contacto_emergencia_telefono)
    VALUES (p_datos_personales_id, p_grupo_sanguineo, p_alergias, p_enfermedades_preexistentes, p_medicamentos_actuales, p_contacto_emergencia_nombre, p_contacto_emergencia_telefono);
END;
//
DELIMITER ;
```

```
-- Actualizar stock de medicamentos
DELIMITER //
CREATE PROCEDURE ActualizarStockMedicamento(
    IN p_medimento_id INT,
    IN p_nueva_cantidad INT
)
BEGIN
    UPDATE medicamentos
    SET stock = p_nueva_cantidad
    WHERE id = p_medimento_id;
END;
//
DELIMITER ;
```

Investigación: Explorar cómo los procedimientos almacenados pueden mejorar la reutilización de código y la eficiencia.

Los procedimientos almacenados son básicamente conjuntos de comandos SQL que se guardan y se ejecutan en el servidor de base de datos. En lugar de enviar varias consultas individuales, puede llamar a un único procedimiento almacenado que ejecute un conjunto predefinido de operaciones.

Beneficios de utilizar procedimientos almacenados

Estos tres beneficios muestran por qué debería utilizar procedimientos almacenados.

Rendimiento mejorado

Una de las principales ventajas de los procedimientos almacenados es su capacidad para mejorar el rendimiento de las bases de datos. Como están precompilados, los procedimientos almacenados se ejecutan más rápido que las consultas SQL ad hoc. Esto da como resultado:

Tiempos de respuesta más rápidos

Uso más eficiente de los recursos del servidor

Reutilización y mantenibilidad

Dado que los procedimientos almacenados se pueden llamar varias veces, promueven la reutilización del código. Esto reduce la necesidad de repetir el código en diferentes aplicaciones, lo que facilita el mantenimiento y las actualizaciones. Además, los cambios realizados en un procedimiento

almacenado se reflejarán automáticamente en todas las aplicaciones que lo utilicen.

Esto ayuda a garantizar la coherencia y reduce la posibilidad de errores.

Importancia del Conocimiento: Los procedimientos almacenados centralizan la lógica y pueden mejorar el rendimiento al ejecutarse directamente en el servidor.

2. Crear vistas para simplificar consultas complejas.

Práctica: Crear vistas que presenten información de varias tablas de manera unificada.

```
/*Vistas*/
-- Información básica de pacientes
CREATE VIEW vista_pacientes AS
SELECT p.id AS paciente_id,dp.nombres_completos AS nombre_paciente,dp.numero_identificacion AS identificacion,dp.telefono,dp.correo_electronico,p.grupo_sanguineo,p.alergias,
p.enfermedades_preexistentes
FROM
    pacientes p
JOIN datospersonales dp ON p.datos_personales_id = dp.id;

-- Historial médico por paciente
CREATE VIEW vista_historial_medico AS
SELECT h.paciente_id,dp.nombres_completos AS nombre_paciente,h.fecha AS fecha_cita,h.descripcion,h.diagnostico,h.tratamiento_recomendado,h.estado_general
FROM
    historialmedico h
JOIN citas c ON h.cita_id = c.id
JOIN datospersonales dp ON h.paciente_id = dp.id;

-- Facturación detallada por cita
CREATE VIEW vista_facturacion_citas AS
SELECT f.id AS factura_id,c.id AS cita_id,dp.nombres_completos AS nombre_paciente,f.fecha_emision,f.subtotal,f.impuestos,f.total,f.total_examenes,f.total_medicamentos,
f.estado_pago
FROM
    facturas f
JOIN citas c ON f.cita_id = c.id
JOIN pacientes p ON c.paciente_id = p.id
JOIN datospersonales dp ON p.datos_personales_id = dp.id;

-- Citas médicas activas
CREATE VIEW vista_citas_activas AS
SELECT c.id AS cita_id,dp.nombres_completos AS paciente,dp.m.nombres_completos AS medico,c.fecha,c.hora_inicio,c.hora_finalizacion,c.estado
FROM
    citas c
JOIN pacientes p ON c.paciente_id = p.id
JOIN datospersonales dp_p ON p.datos_personales_id = dp_p.id
JOIN medicos m ON c.medico_id = m.id
JOIN datospersonales dp_m ON m.datos_personales_id = dp_m.id
WHERE c.estado = 'Pendiente';

-- Medicamento por stock
CREATE VIEW vista_medicamentos_stock AS
SELECT m.id AS medicamento_id,m.nombre,m.tipo_medicamento,m.presentacion,m.dosis,m.stock,m.fecha_vencimiento,p.nombre AS proveedor
FROM
    medicamentos m
JOIN proveedores p ON m.proveedor_id = p.id
WHERE m.stock > 0;

-- Exámenes realizados por paciente
CREATE VIEW vista_examenes_paciente AS
SELECT o.cita_id,dp.nombres_completos AS nombre_paciente,e.nombre AS examen,o.resultados,o.fecha_realizacion,o.costo_examen
FROM
    ordenesexamenes o
JOIN examenesmedicos e ON o.examen_id = e.id
JOIN citas c ON o.cita_id = c.id
JOIN pacientes p ON c.paciente_id = p.id
JOIN datospersonales dp ON p.datos_personales_id = dp.id
WHERE o.estado = 'Realizado';
```

```
-- Información del personal médico
CREATE VIEW vista_personal_medico AS
SELECT m.id AS medico_id, dp.nombres_completos AS nombre_medico, e.nombre AS especialidad,
m.experiencia_years, m.fecha_ingreso, m.tipo_contrato
FROM
    medicos m
JOIN datospersonales dp ON m.datos_personales_id = dp.id
JOIN especialidades e ON m.especialidad_id = e.id;
```

Investigación: Investigar las ventajas de usar vistas en lugar de consultas complejas repetitivas.

Aunque las vistas no mejoran inherentemente el rendimiento de las consultas y, a veces, incluso pueden obstaculizarlo, se pueden usar estratégicamente para ajustar el rendimiento. Las vistas materializadas, que se almacenan en el disco, pueden calcular previamente y almacenar consultas complejas. Cuando se consulta una vista materializada, se accede a datos agregados previamente, lo que puede mejorar significativamente el rendimiento de las operaciones de lectura intensiva. Sin embargo, es importante usarlos con prudencia, ya que pueden consumir almacenamiento adicional y requerir mantenimiento.

1- **Vistas materializadas:** las vistas materializadas son un tipo especial de vista que se almacena en el disco. Precalculan y almacenan consultas complejas, lo que da como resultado datos preagregados. Cuando consulta una vista materializada, está accediendo a datos que ya se han procesado, lo que puede mejorar significativamente las operaciones de lectura intensiva.

2- **Beneficios de las vistas materializadas:** Aumento del rendimiento: al acceder a datos preagregados, las consultas se ejecutan más rápido. Carga reducida: las vistas materializadas descargan trabajo de la ejecución de la consulta principal. Resultados consistentes: los datos permanecen consistentes hasta que se actualiza la vista.

3- **Consideraciones:** Sobrecarga de almacenamiento: las vistas materializadas consumen espacio de almacenamiento adicional. Mantenimiento: actualice periódicamente las vistas materializadas para mantenerlas actualizadas.

Importancia del Conocimiento: Las vistas ayudan a simplificar el acceso a datos complejos y pueden mejorar la seguridad al limitar el acceso directo a las tablas.

3. Implementar triggers para auditoría y control de cambios.

Práctica: Crear triggers que registren cambios en las tablas de Reservas y Pagos cada vez que un registro se actualiza o elimina.

```

/*datos_personales*/
-- control datos duplicados
DELIMITER //
CREATE TRIGGER trg_no_datos_repetidos
BEFORE INSERT ON datospersonales
FOR EACH ROW
BEGIN
    DECLARE cuenta INT;

    -- Contar si ya existe una coincidencia exacta en número de identificación y nombre
    SELECT COUNT(*) INTO cuenta
    FROM datospersonales
    WHERE numero_indetificacion = NEW.numero_indetificacion
        AND nombres_completos = NEW.nombres_completos;

    -- Si ya existe, generar un error
    IF cuenta > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'El registro con estos datos personales ya existe.';
    END IF;
END;
//
DELIMITER ;

/*Preinscripcion*/
-- Evitar medicamentos vencidos en prescripciones
DELIMITER //
CREATE TRIGGER trg_validar_vencimiento_medicamentos
BEFORE INSERT ON prescripcionesmedicas
FOR EACH ROW
BEGIN
    DECLARE fecha_venc DATE;

    SELECT fecha_vencimiento INTO fecha_venc
    FROM medicamentos
    WHERE id = NEW.medicamento_id;

    IF fecha_venc < CURDATE() THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'El medicamento seleccionado está vencido.';
    END IF;
END;
//
DELIMITER ;

```



```

/*Pacientes*/
-- Evitar eliminación de pacientes con citas pendientes
DELIMITER //
CREATE TRIGGER trg_evitar_eliminar_paciente
BEFORE DELETE ON pacientes
FOR EACH ROW
BEGIN
    DECLARE cuenta INT;

    SELECT COUNT(*) INTO cuenta
    FROM citas
    WHERE paciente_id = OLD.id AND estado = 'Pendiente';

    IF cuenta > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'No se puede eliminar el paciente porque tiene citas pendientes.';
    END IF;
END;
//
DELIMITER ;

```

```

/*Citas*/
DELIMITER //
-- Validación de disponibilidad de médicos en citas
CREATE TRIGGER trg_validar_disponibilidad_medico
BEFORE INSERT ON citas
FOR EACH ROW
BEGIN
    DECLARE cuenta INT;

    SELECT COUNT(*) INTO cuenta
    FROM citas
    WHERE medico_id = NEW.medico_id
        AND fecha = NEW.fecha
        AND ((NEW.hora_inicio BETWEEN hora_inicio AND hora_finalizacion)
            OR (NEW.hora_finalizacion BETWEEN hora_inicio AND hora_finalizacion));

    IF cuenta > 0 THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'El médico ya tiene una cita en este horario.';
    END IF;
END;
//
DELIMITER ;

```

```

-- Actualizar el estado de citas automáticamente
delimiter //
CREATE TRIGGER trg_actualizar_estado_cita
AFTER UPDATE ON citas
FOR EACH ROW
BEGIN
    IF NEW.hora_finalizacion < CURTIME() AND NEW.estado = 'Pendiente' THEN
        UPDATE citas SET estado = 'Completada' WHERE id = NEW.id;
    END IF;
END;
//
delimiter ;

```

Investigación: Investigar cómo utilizar triggers para mantener un historial de cambios en la base de datos.

Un trigger (también conocido como disparador o desencadenador en español) es una especie de "escucha" programada dentro de una base de datos que se activa en respuesta a ciertos eventos de manipulación de datos (DML). En términos más técnicos, un trigger es un procedimiento almacenado que se ejecuta automáticamente cuando un evento específico ocurre dentro de una tabla o vista. Estos eventos suelen ser operaciones de INSERT, UPDATE, o DELETE. Una vez que se activan, los triggers pueden realizar una variedad de funciones, como la modificación de datos en la misma tabla o en otras tablas, la aplicación de restricciones y reglas de negocio, y muchas más tareas relacionadas con la gestión de datos.

Importancia y Aplicaciones en el Mundo Real

Los triggers tienen una amplia gama de aplicaciones:

Automatización de Tareas: Imagina que cada vez que se registra un nuevo usuario en tu sistema, también debes actualizar una tabla de auditoría. Un trigger se encargará de esto automáticamente después de cada inserción en la tabla de usuarios.

Integridad de Datos: Los triggers pueden aplicar reglas que protegen la integridad de tus datos. Por ejemplo, podrían impedir que se elimine un registro que aún está siendo referenciado en otra tabla.

Auditoría y Registro: Los triggers son útiles para mantener un registro histórico de los datos. Por ejemplo, podrías querer saber cuándo se modificó por última vez un campo específico.

Cascading Actions: Los triggers pueden ejecutar acciones en cascada. Si borras un registro en una tabla, un trigger podría eliminar automáticamente los registros relacionados en otras tablas para mantener la integridad referencial.

Utiliza activadores para el registro y la auditoría.

Los desencadenantes son herramientas excelentes para mantener registros de los cambios en los datos. Al registrar automáticamente los cambios, puedes crear una pista de auditoría que te ayude a rastrear las modificaciones de tus datos, lo que resulta especialmente útil para el cumplimiento de la normativa y la resolución de problemas. Por ejemplo, puedes hacer un disparador que registre cualquier operación de actualización o eliminación en una tabla sensible.

Importancia del Conocimiento: Los triggers permiten automatizar tareas como la auditoría y validación de datos.

6. Monitoreo y Optimización de Recursos



Objetivo: Controlar el rendimiento de la base de datos, identificando y solucionando problemas de recursos.

Actividades:

1. Monitorear el rendimiento de consultas.

Práctica: Usar herramientas como SHOW PROCESSLIST para detectar consultas lentas y optimizarlas.

```
70 • SHOW PROCESSLIST;
71
72
```

Result Grid								
Filter Rows:								
Export:  Wrap Cell Content: 								
	Id	User	Host	db	Command	Time	State	Info
▶	5	event_scheduler	localhost	NULL	Daemon	445628	Waiting on empty queue	NULL
	8	root	localhost:63378	bck_prueba_libreria_corregida	Sleep	84		NULL
	9	root	localhost:63379	hospital_buen_dia	Query	0	init	SHOW PROCESSLIST

70 • SHOW ENGINE INNODB STATUS;

71

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

Type	Name	Status
InnoDB	===== 2025-02-06 10:48:54 0xea48 INNODB MONITOR OUTPUT ===== Per second averages calculated from ...	

71 • EXPLAIN SELECT * FROM citas WHERE fecha = '2023-01-01';

72

Result Grid | Filter Rows: | Export: | Wrap Cell Content: |

	id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
▶	1	SIMPLE	citas	NULL	ALL	NULL	NULL	NULL	NULL	10	10.00	Using where

Investigación: Investigar las mejores prácticas para monitorear el rendimiento de las consultas en producción.

La monitorización de bases de datos, o la monitorización del rendimiento de las bases de datos, es la práctica de monitorizar bases de datos en tiempo real. Al realizar un seguimiento de métricas específicas, la monitorización de bases de datos permite a los equipos comprender el estado y el comportamiento de sus sistemas de bases de datos. Esto, a su vez, ayuda a solucionar problemas y a encontrar formas de optimizar el rendimiento de las bases de datos.

Uso de Splunk para la supervisión del rendimiento de las consultas de bases de datos

Las consultas de base de datos lentas pueden ser la causa de problemas más amplios de disponibilidad del servicio. Con Rendimiento de consultas de base de datos, puede supervisar el impacto de sus consultas de base de datos en la disponibilidad del servicio directamente en Splunk APM. De esta manera, puede identificar rápidamente consultas de larga duración, no optimizadas o pesadas y mitigar los problemas que puedan estar causando, sin tener que instrumentar sus bases de datos.

Mejorar y mantener el rendimiento de la base de datos

El rendimiento de una base de datos describe la velocidad a la que una base de datos proporciona información a los usuarios que la solicitan. Se considera que una base de datos que satisface esta demanda a un ritmo elevado tiene un buen rendimiento. Si provoca cuellos de botella en un proceso empresarial o una aplicación, se considera que tiene un rendimiento deficiente.

Hay muchos factores que influyen en el rendimiento de las bases de datos, pero cinco son especialmente importantes:

Carga de trabajo: La carga de trabajo se refiere al volumen total de solicitudes realizadas por los usuarios y las aplicaciones de una base de datos. Puede incluir varios tipos de consultas, trabajos por lotes, transacciones en línea,

comandos del sistema y todas las demás demandas que se le imponen al sistema en un momento determinado. Las cargas de trabajo fluctúan drásticamente con el tiempo, incluso de un segundo a otro. Ocasionalmente, se puede predecir la carga de trabajo (por ejemplo, una mayor demanda durante las compras de temporada o el procesamiento de nóminas de fin de mes y una menor demanda después del horario comercial), pero la mayoría de las veces la carga de trabajo es impredecible.

Rendimiento: el rendimiento describe el volumen de trabajo realizado por la base de datos a lo largo del tiempo, que normalmente se mide como la cantidad de consultas ejecutadas por segundo, minuto u hora. Si el rendimiento de una base de datos es menor que la cantidad de consultas entrantes, puede sobrecargar el servidor y generar mayores tiempos de respuesta de las consultas, lo que a su vez ralentiza un sitio web o una aplicación. Los problemas de rendimiento pueden indicar la necesidad de optimizar las consultas o actualizar el servidor.

Recursos: Los recursos son herramientas de hardware y software que utiliza la base de datos. Entre ellos se encuentran la CPU, la memoria, los controladores de caché y el microcódigo. Los recursos a disposición de la base de datos afectan drásticamente a todos los demás factores de rendimiento de la base de datos.

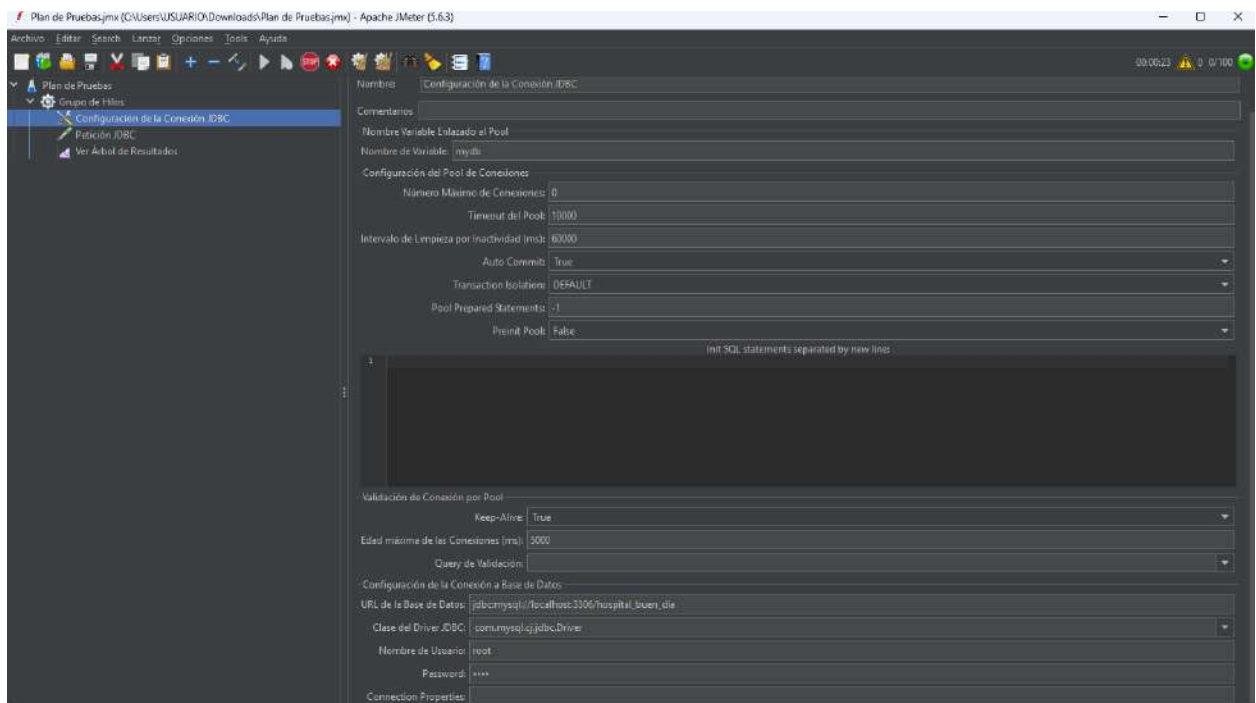
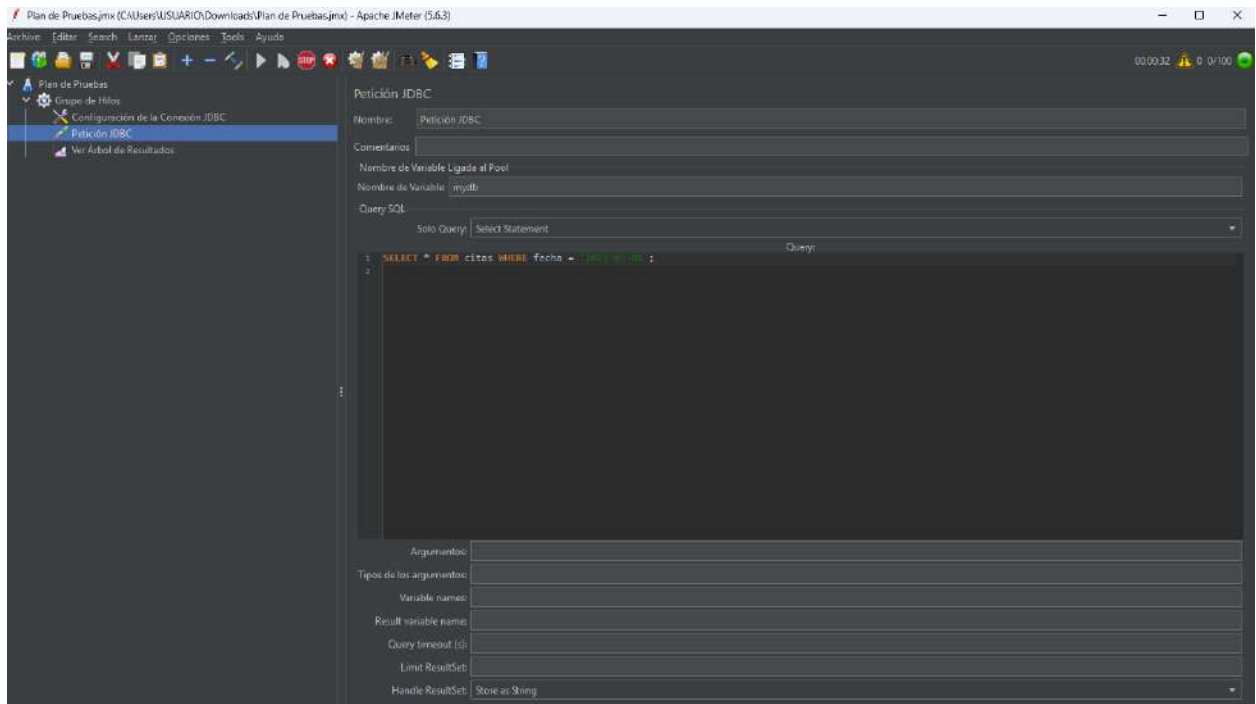
Optimización: la optimización se refiere a las estrategias utilizadas para aumentar la velocidad y la eficiencia con la que se recupera la información de la base de datos. Las prácticas de optimización incluyen la eliminación de tablas no utilizadas, la garantía de una indexación adecuada, el uso de tipos de datos apropiados y otras técnicas de ajuste de la base de datos. La optimización es un proceso continuo que requiere un seguimiento, un análisis y una mejora continuos.

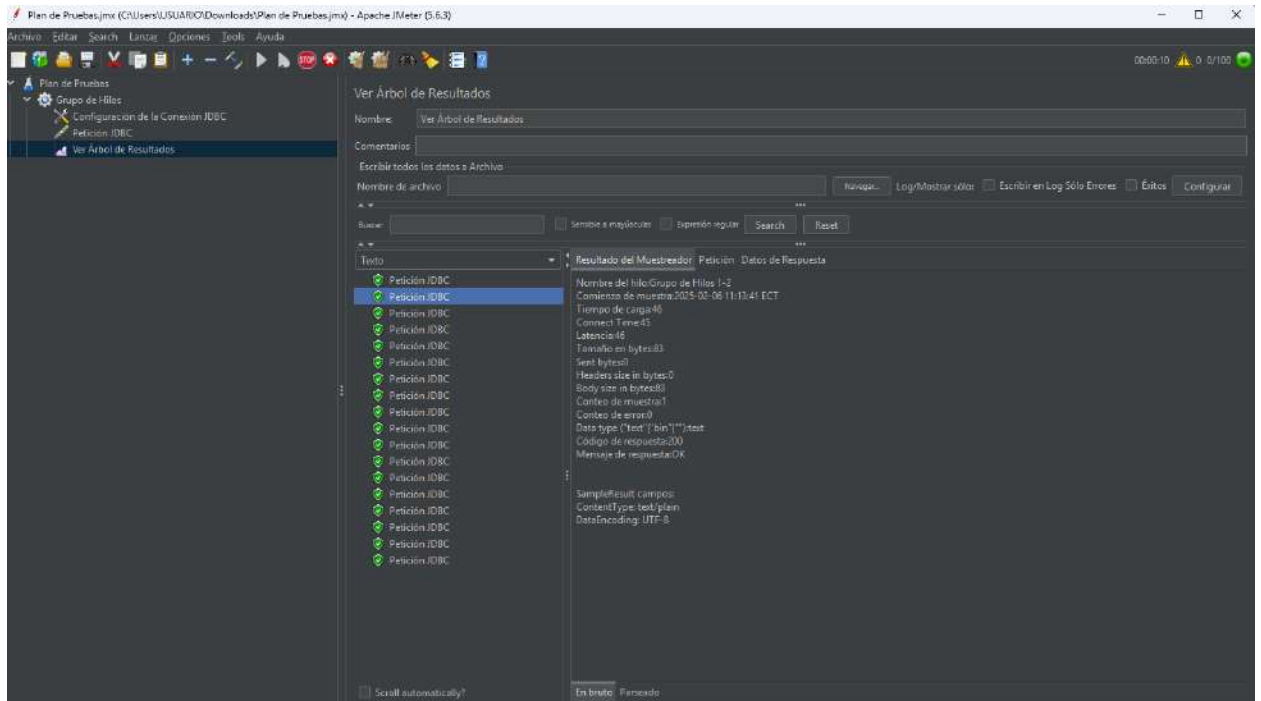
Contención: la contención se produce cuando dos o más procesos de carga de trabajo intentan acceder a los mismos datos al mismo tiempo. En una base de datos SQL, por ejemplo, la contención se produce cuando varias transacciones intentan actualizar la misma fila simultáneamente. Si una transacción intenta actuar sobre datos que están en proceso de ser modificados por otra, la base de datos tiene que prohibir el acceso a los datos, o "bloquearlos", hasta que se complete el cambio para garantizar la precisión y la coherencia de esos datos. A medida que aumenta la contención, como es probable durante los períodos de alta demanda, el rendimiento disminuye.

Importancia del Conocimiento: El monitoreo proactivo puede identificar cuellos de botella antes de que afecten el rendimiento del sistema.

2. Realizar pruebas de carga.

Práctica: Simular múltiples usuarios concurrentes usando herramientas como Apache JMeter para ver cómo responde la base de datos bajo alta carga.





Investigación: Investigar cómo realizar pruebas de estrés y carga en bases de datos de alto rendimiento.

La prueba de carga implica evaluar cómo un sistema se desempeña bajo cargas de trabajo esperadas o anticipadas. El objetivo principal es garantizar que la aplicación pueda manejar la concurrencia de usuarios y el volumen de datos esperados sin comprometer el rendimiento. Ayuda a identificar cuellos de botella, áreas de mejora y posibles optimizaciones.

La prueba de estrés, por otro lado, va más allá de las condiciones operativas normales. Somete al sistema a límites extremos mediante condiciones como tráfico intenso, agotamiento de recursos o picos concurrentes de usuarios. El objetivo es evaluar cómo se comporta el sistema bajo estrés y descubrir vulnerabilidades y debilidades.

Criterio	Prueba de carga	Prueba de estrés
----------	-----------------	------------------

Propósito	Evaluar el rendimiento bajo condiciones esperadas	Evaluar la robustez del sistema bajo condiciones extremas
Carga Aplicada	Carga de usuario esperada o anticipada	Más allá de la capacidad del sistema, escenarios extremos
Objetivo	Identificar cuellos de botella, optimizaciones	Descubrir vulnerabilidades, observar la recuperación del sistema
Ejemplos	Simulación de acciones concurrentes de usuarios	Simulación de repuntes repentinos de tráfico, agotamiento de recursos

Cómo realizar pruebas de carga:

Identificar Métricas de Rendimiento: Define las métricas que deseas medir, como tiempo de respuesta, rendimiento y uso de recursos.

Crear Escenarios de Prueba: Desarrolla escenarios de prueba realistas que imiten el comportamiento esperado del usuario y el volumen de datos.

Ejecutar Pruebas: Ejecuta pruebas con cargas gradualmente crecientes, monitoreando el rendimiento del sistema en cada etapa.

Analizar Resultados: Evalúa los resultados de las pruebas para identificar cuellos de botella, áreas de mejora y el rendimiento general del sistema.

Cómo realizar pruebas de estrés:

Definir Escenarios de Estrés: Identificar posibles escenarios de estrés, como picos repentinos de tráfico, agotamiento de recursos o cargas extremas de usuarios.

Configurar el Entorno de Prueba: Establecer el entorno de prueba para simular condiciones de estrés, asegurándose de que refleje escenarios del mundo real.

Aplicar Estrés: Aumentar gradualmente los factores de estrés, monitoreando el comportamiento y rendimiento del sistema bajo condiciones extremas.

Evaluar el Impacto: Valorar el impacto del estrés en el sistema, identificando puntos de fallo, vulnerabilidades y mecanismos de recuperación.

Importancia del Conocimiento: Las pruebas de carga aseguran que el sistema sea capaz de manejar tráfico alto y crecimiento de datos.

3. Optimizar el uso de recursos y gestionar índices.

Práctica: Identificar índices no utilizados y eliminarlos para liberar recursos y mejorar la velocidad de las operaciones de escritura.

```
73 -- Mostrar todos los índices existentes en una tabla (por ejemplo, en la tabla 'citas')
74 • SHOW INDEX FROM citas;
75
76
```

Table	Non_unique	Key_name	Seq_in_index	Column_name	Collation	Cardinality	Sub_part	Packed	Null	Index_type	Comment	Index_comment	Visible	Expression
citas	0	PRIMARY	1	id	A	10				BTREE			YES	
citas	1	fk_paciente_cita	1	paciente_id	A	10				BTREE			YES	
citas	1	fk_medico_cita	1	medico_id	A	8				BTREE			YES	
citas	1	fk_area_cita	1	area_id	A	5			YES	BTREE			YES	
citas	1	idx_estado	1	estado	A	3				BTREE			YES	
citas	1	idx_paciente_id	1	paciente_id	A	10				BTREE			YES	

```
76 -- Consultar el uso de índices utilizando la tabla performance_schema
77 • SELECT
78     OBJECT_SCHEMA, -- Esquema de la tabla
79     OBJECT_NAME,    -- Nombre de la tabla
80     INDEX_NAME,      -- Nombre del índice
81     COUNT_STAR       -- Número de veces que se ha utilizado el índice
82 FROM performance_schema.table_io_waits_summary_by_index_usage
83 WHERE COUNT_STAR = 0; -- Filtrar índices que no se han utilizado
84
85
```

OBJECT_SCHEMA	OBJECT_NAME	INDEX_NAME	COUNT_STAR
mysql	schemata	PRIMARY	0
mysql	schemata	catalog_id	0
mysql	schemata	default_collation_id	0
mysql	tablespace_files	tablespace_id	0
mysql	tablespace_files	file_name	0
mysql	tablespaces	PRIMARY	0
mysql	tablespaces	name	0
mysql	check_constraints	PRIMARY	0
mysql	check_constraints	schema_id	0
mysql	check_constraints	table_id	0
mysql	column_type_ele...	PRIMARY	0

```

84
85 -- Eliminar el índice 'idx_paciente_id' en la tabla 'citas'
86 • DROP INDEX idx_paciente_id ON citas;
87

```

Output

#	Time	Action	Message
47	10:49:37	EXPLAIN SELECT * FROM citas WHERE fecha = '2023-01-01'	1 row(s) returned
48	11:20:52	SHOW INDEX FROM citas	6 row(s) returned
49	11:21:12	SELECT OBJECT_SCHEMA, -- Esquema de la tabla OBJECT_NAME, -- Nombre de la tabla INDEX_NAME, -- Nombre del índice COU...	175 row(s) returned
50	11:21:39	DROP INDEX idx_paciente_id ON citas	0 row(s) affected Records: 0 Duplicates: 0 Warnings: 0

```

88 -- Usar EXPLAIN para ver cómo se ejecuta la consulta antes de eliminar el índice
89 • EXPLAIN SELECT * FROM citas WHERE paciente_id = 123;
90
91

```

Result Grid

id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	SIMPLE	citas	NULL	ref	fk_paciente_cita	fk_paciente_cita	4	const	1	100.00	NULL

Investigación: Investigar cómo ajustar el número de índices según el tipo de consulta (lectura/escritura).

Limitaciones de la característica de índices que faltan

Cuando el optimizador de consultas genera un plan de consulta, analiza cuáles son los mejores índices para una condición de filtro concreta. Si no existen los mejores índices, el optimizador de consultas sigue generando un plan de consulta mediante los métodos de acceso menos costosos disponibles, pero también almacena información sobre estos índices. La característica de índices que faltan le permite tener acceso a la información acerca de los mejores índices posibles para poder decidir si deberían implementarse.

La optimización de consultas es un proceso que depende del tiempo, por lo que hay limitaciones en la característica de índices que faltan. Entre las limitaciones se incluyen:

Las sugerencias de índices que faltan se basan en las estimaciones realizadas durante la optimización de una sola consulta, antes de la ejecución de la consulta. Las sugerencias de índices que faltan no se prueban ni actualizan después de la ejecución de la consulta.

La característica de índices que faltan sugiere solo índices de almacén de filas no agrupados basados en disco. No se sugieren índices únicos ni filtrados.

Se sugieren columnas de clave, pero la sugerencia no especifica un orden para esas columnas. Para obtener información sobre cómo ordenar columnas, consulte la sección Aplicar sugerencias de índices que faltan de este artículo.

Se sugieren columnas incluidas, pero SQL Server no realiza ningún análisis de costo-beneficio con respecto al tamaño del índice resultante cuando se sugiere un gran número de columnas incluidas.

Las solicitudes de índices que faltan pueden ofrecer variaciones similares de índices en la misma tabla y columna en todas las consultas. Es importante revisar las sugerencias de índices y combinarlas siempre que sea posible.

No se realizan sugerencias para planes de consulta triviales.

En las consultas que implican solo predicados de desigualdad, la información de costos es menos precisa.

Se recopilan las sugerencias para un máximo de 600 grupos de índices que faltan. Una vez alcanzado este umbral, no se recopilan más datos del grupo de índices que faltan.

Debido a estas limitaciones, las sugerencias de índices que faltan se tratan mejor como uno de varios orígenes de información al realizar análisis de índices, diseño, ajuste y pruebas. Las sugerencias de índices que faltan no son recomendaciones para crear índices exactamente como se sugiere.

Importancia del Conocimiento: La optimización de los recursos asegura un uso eficiente del hardware y mejora la escalabilidad.

7. Git y Control de Versiones

Objetivo: Asegurar que el código relacionado con la base de datos esté versionado y que el equipo pueda colaborar de manera eficiente.

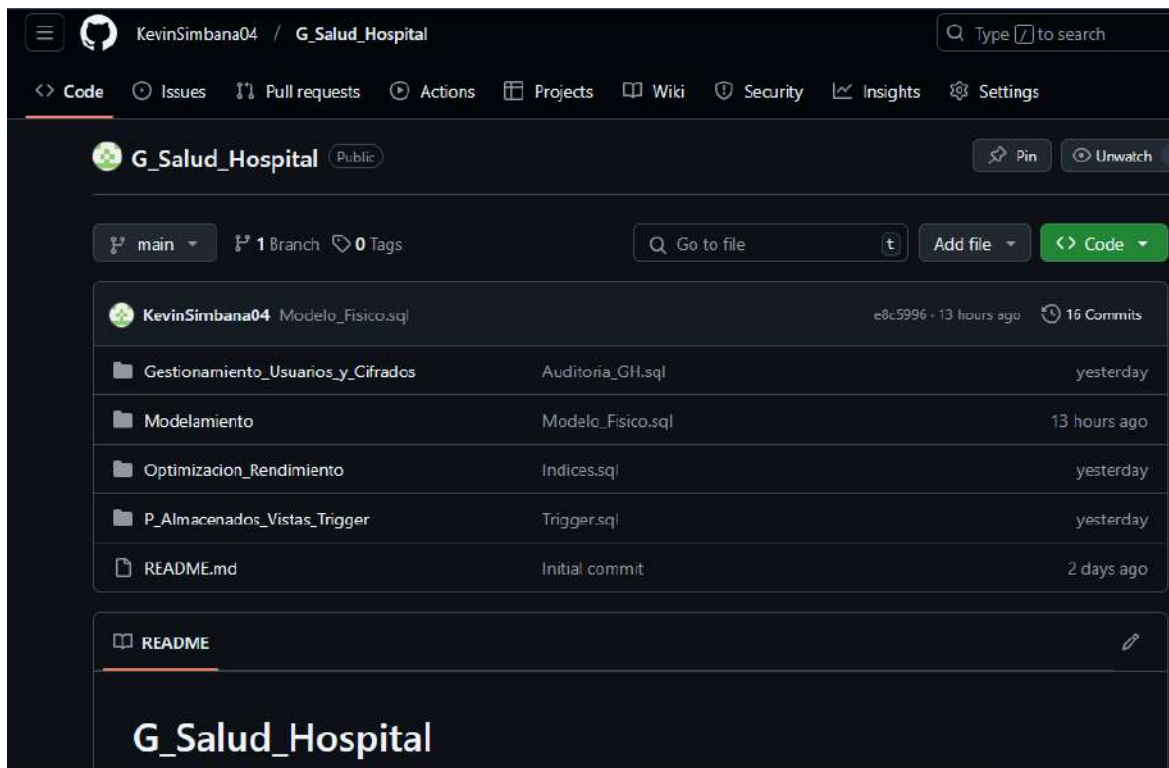
Actividades:

1. **Configurar un repositorio de Git para el proyecto.**

Práctica: Inicializar un repositorio en Git y subir los archivos de definición de la base de datos, scripts de SQL y procedimientos almacenados.

Link Repositorio:

https://github.com/KevinSimbana04/G_Salud_Hospita



Investigación: Investigar buenas prácticas de flujo de trabajo en Git (por ejemplo, uso de ramas, git merge).

1. Elige un flujo de trabajo.

Es importante que conozcas o elijas un flujo de trabajo / workflow ya sea si colaboras en equipo o lo haces solo. Para eso es importante que identifiques las fases más importantes que ocurren mientras desarrollas, que analices el procedimiento a ejecutar cuando se necesita hacer cambios de emergencia contra bugs y muy importante que entiendas de qué forma es más fácil identificar los cambios del proyecto.

Existen numerosos flujos de trabajo que se adaptan a cada necesidad, te listo algunos:

Git Feature Branch Workflow:

Todos los features deben tener una rama dedicada en lugar de concentrar todo en master.

Gitflow Workflow:

Es similar al anterior solo que la estructura de ramas está diseñado alrededor de las entregas del proyecto.

Forking Workflow:

En lugar de usar un solo repositorio centralizado le da la opción a cada colaborador de tener uno propio.

2. Utiliza una buena estructura en los mensajes de commits

Se recomienda que tus commits tengan una estructura que permita identificar el trabajo que realizaste en el proyecto. Entre más descriptivos sean tus mensajes mucho mejor.

Puedes emplear algunos sufijos como:

feat:

Describe si trabajaste en un nuevo feature

fix:

Describe si solucionaste un bug

docs:

Dice si hiciste algún cambio en la documentación

test:

Indica si añadiste un test

refactor:

Nos muestra que se ejecutó algún refactor en el código

De esta forma se puede tener mejor organizado todo el control de cambios además que lo hacen mucho más fácil de entender.

3. Envía tus cambios en pull-request

Un pull-request es una solicitud que hacemos al repositorio cuando enviamos código con cambios, idealmente el propietario del repositorio revisa el código y decide si es apto para ser integrado.

Tener una revisión previa del código antes de ser integrado es muy saludable no importa si estas en un equipo grande o pequeño enviar cambios sin aprobación puede provocar inconsistencias en el código.

Importancia del Conocimiento: Git permite la colaboración y el manejo eficiente de cambios en el código, especialmente cuando se trabaja en equipo.

2. Realizar commits frecuentes y con mensajes claros.

Práctica: Hacer commits regularmente, describiendo claramente los cambios realizados en los scripts SQL y la estructura de la base de datos.

Investigación: Investigar cómo utilizar git rebase y git pull para evitar conflictos.

Git Rebase

Realizar un rebase a una rama (branch) en Git es una forma de mover la totalidad de una rama a otro punto del árbol. El ejemplo más simple es mover una rama más arriba en el árbol. Digamos que tenemos una rama que se separó de la rama master en el punto A:

```
    /o----o--o-o----o----- branch
--o-o-A--o--o--o--o----o-o-o-o-- master
```

Cuando se realiza rebase se puede mover así:

```
                /o----o--o-o----o----- branch
--o-o-A--o--o--o--o----o-o-o-o master
```

Para realizar rebase, asegúrate de tener todos los commits que quieras en el rebase en tu rama master. Revisar la rama en la que quieres hacer el rebase y escribe `git rebase master` (donde master es la rama en la que quieres hacer el rebase).

También es posible hacer rebase en una rama diferente, de modo que, por ejemplo, una rama que se basaba en otra rama (llamémosla feature) se rebasa en master:

```
            /---o-o branch
        /---o-o-o-o---o-o----- feature
---o-o-o-A---o--o--o-o-o-o-o-o- master
```

Después de `git rebase master branch` o `git rebase master` si te encuentras (checked out) en branch, obtendrás:

```
        /---o-o-o-o---o-o----- feature
---o-o-o-A---o--o--o-o-o-o-o-o- master
                \---o-o branch
```

Usando git pull

Usa `git pull` para actualizar un repositorio local del repositorio remoto correspondiente. Por ejemplo: Mientras trabajas localmente en main, ejecuta `git pull` para actualizar la copia local de main y actualizar las otras ramas remota de seguimiento remoto. (Más información sobre referencias de rama remota en la siguiente sección).

Sin embargo, hay algunas cosas que hay que tener en cuenta para que ese ejemplo sea cierto:

El repositorio local tiene un repositorio remoto vinculado.

Confirma esto ejecutando `git remote -v`

Si existen múltiples remotos, `git pull` podría no ser suficiente información. Es posible que debas ingresar `git pull origin` o `git pull upstream`.

La rama a la que te has movido tiene una rama de seguimiento remoto correspondiente.

Revisa esto ejecutando `git status`. Si no hay una rama de seguimiento remota, Git no sabe de dónde extraer la información.

Importancia del Conocimiento: Un flujo de trabajo claro en Git mejora la colaboración y la gestión de versiones.

3. Automatizar pruebas con GitHub Actions.

Práctica: Crear flujos de trabajo de CI/CD que automaticen las pruebas de las consultas SQL y otros scripts relacionados con la base de datos.

Investigación: Investigar sobre integración continua y cómo aplicarla en bases de datos con GitHub Actions.

Cuando se envía el código, las herramientas de CI verifican cada integración mediante la creación de una iteración de la compilación y la ejecución de una batería de pruebas automatizadas para detectar y abordar los errores de integración con mayor rapidez.

La CI se creó como respuesta a los retos del desarrollo de software tradicional, en concreto los procesos asociados a la integración y la implementación. En el desarrollo tradicional, cada desarrollador era responsable de integrar de forma manual el nuevo código en las nuevas iteraciones de una aplicación o servicio, lo que hacía que la integración fuera un proceso lento y propenso a errores, especialmente para los grandes equipos de desarrollo.

Las diferentes piezas de código no siempre funcionaban bien juntas y los desarrolladores integraban sus cambios en diferentes plazos (y a veces en el último minuto), por lo que los comentarios sobre los problemas de integración se retrasaban a menudo. Los retrasos relacionados con integraciones incoherentes hicieron más difícil para los equipos averiguar qué cambio introdujo el fallo, por lo que la depuración también se convirtió en un proceso arduo.

Configurar MySQL para GitHub Actions puede implicar algo de prueba y error, pero con una investigación cuidadosa, experimentación y las configuraciones adecuadas, puede lograr un entorno de base de datos estable y funcional. Al compartir mi experiencia, espero ayudar a otros desarrolladores a superar desafíos similares y agilizar el proceso de integración de MySQL en sus propios flujos de trabajo de GitHub Actions

Importancia del Conocimiento: Las pruebas automáticas aseguran que las bases de datos se mantengan consistentes y funcionales a lo largo del tiempo.

Bibliografía

- [1] T. Naeem, “¿Qué es la integridad de datos en una base de datos? ¿Por qué lo necesitas?”, *Astera*, 31-oct-2020. .
- [2] “¿Qué es la integridad de los datos?”, *Ibm.com*, 12-jul-2024. .
- [3] N. Emilio, “Integridad de datos: ¿Cómo verificar la integridad de los datos?”, *Bismart.com*, 05-ago-2024. .
- [4] “¿Qué es la prevención de pérdida de datos (DLP)?”, *Ibm.com*, 11-oct-2024. .
- [5] A. Casero, “Integridad referencial: ¿Qué es y para qué sirve?”, *KeepCoding Bootcamps*, 18-jul-2023. .
- [6] K. Jackson, “Alta disponibilidad frente a recuperación ante desastres: diferencias clave”, *Trilio*, 28-oct-2024. [En línea]. Disponible en: <https://trilio.io/es/resources/high-availability-vs-disaster-recovery/>. [Consultado: 06-feb-2025].
- [7] Agustin, “Alta disponibilidad en Cloud Hosting: Conceptos y técnicas esenciales”, *SysAdminOK*, 01-ago-2024. .
- [8] B. Robertson y L. Murray, “Data encryption”, *Learning Center*. [En línea]. Disponible en: <https://www-imperva-com.translate.goog/learn/data-security/data-encryption/? x tr sl=en& x tr tl=es& x tr hl=es& x tr pto=sge>. [Consultado: 06-feb-2025].
- [9] “Estrategias de copia de seguridad de datos: Consejos para garantizar la continuidad de negocio y reducir el tiempo de inactividad”, *Arcserve*, 24-jun-2024. [En línea]. Disponible en: <https://www.arcserve.com/es/blog/data-backup-strategies-tips-ensuring-business-continuity-and-minimizing-downtime>. [Consultado: 06-feb-2025].
- [10] G. Latour, “8 steps to creating and implementing a data backup plan”, *Groupe SL Inc*, 27-may-2024. [En línea]. Disponible en: <https://www-groupesl-com.translate.goog/en/news/how-to-create-data-backup-plan/? x tr sl=en& x tr tl=es& x tr hl=es& x tr pto=sge>. [Consultado: 06-feb-2025].
- [11] “Backup Incremental: La Mejor Estrategia Para Proteger Tus Datos”, *Datos 101*, 28-dic-2023. .
- [12] “¿Cuál es la diferencia entre respaldo incremental, diferencial y completo?”, *Acronis*, 30-may-2023. [En línea]. Disponible en: <https://www.acronis.com/es-mx/blog/posts/incremental-differential-backups/>. [Consultado: 06-feb-2025].
- [13] C. De, “Respaldo en caliente (respaldo dinámico)”, *ComputerWeekly.es*, 25-nov-2014. [En línea]. Disponible en: <https://www.computerweekly.com/es/definicion/Respaldo-en-caliente-respaldo-dinamico>. [Consultado: 06-feb-2025].
- [14] S. Straub, “Top 7 MySQL backup methods and how to choose”, *N2WS*, 27-ene-2025. [En línea]. Disponible en: <https://n2ws-com.translate.goog/blog/mysql-backup-methods? x tr sl=en& x tr tl=es& x tr hl=es& x tr pto=sge>. [Consultado: 06-feb-2025].
- [15] L. Vileikis, “Hot, warm and cold backups in MySQL”, *Severalnines*, 14-ene-2021. [En línea]. Disponible en: <https://severalnines-com.translate.goog/blog/hot-warm-and-cold-backups-in-mysql/? x tr sl=en& x tr tl=es& x tr hl=es& x tr pto=sge>. [Consultado: 06-feb-2025].
- [16] A. Casero, “Índices de tablas en bases de datos: cómo y cuándo usarlos”, *KeepCoding Bootcamps*, 20-jul-2023. .

- [17] B. Petrovic, “Visión general y estrategias de índices SQL”, *SQL Shack - articles about database auditing, server performance, data recovery, and more*, 10-may-2019. [En línea]. Disponible en: <https://www.sqlshack.com/es/vision-general-y-estrategias-de-indices-sql/>. [Consultado: 06-feb-2025].
- [18] V. Crudu, “How to optimize joins and subqueries in SQL queries for better performance?”, *Moldstud.com*, 07-dic-2024. .
- [19] D. Kapur, “Database partitioning 🧑 : A comprehensive guide to best practices and benefits”, *Medium*, 20-may-2023. [En línea]. Disponible en: <https://deepam-kapur.medium.com/database-partitioning-a-comprehensive-guide-to-best-practices-and-benefits-933dc5d63150>. [Consultado: 06-feb-2025].
- [20] A. Chia, “What are stored procedures?”, *Splunk*. [En línea]. Disponible en: https://www.splunk.com/en_us/blog/learn/stored-procedures.html. [Consultado: 06-feb-2025].
- [21] Walther, “Cómo Funciona un Trigger en SQL: Un Tutorial Paso a Paso”, *Tutoriales Dongee*, 30-ago-2023. [En línea]. Disponible en: <https://www.dongee.com/tutoriales/como-funciona-un-trigger-en-sql-un-tutorial-paso-a-paso/>. [Consultado: 06-feb-2025].
- [22] *Datacamp.com*. [En línea]. Disponible en: <https://www.datacamp.com/es/tutorial/sql-triggers>. [Consultado: 06-feb-2025].
- [23] “Materialized views”, *Databricks*, 06-jun-2020. [En línea]. Disponible en: <https://www.databricks.com/glossary/materialized-views>. [Consultado: 06-feb-2025].
- [24] “¿Cuáles son las principales ventajas de usar vistas de base de datos en las consultas SQL?”, *Linkedin.com*, 11-abr-2024. [En línea]. Disponible en: <https://www.linkedin.com/advice/1/what-key-benefits-using-database-views-airhf?lang=es&lang=es&originalSubdomain=es>. [Consultado: 06-feb-2025].
- [25] S. Watts, “Database monitoring: Basics & introduction”, *Splunk*. [En línea]. Disponible en: https://www.splunk.com/en_us/blog/learn/database-monitoring.html. [Consultado: 06-feb-2025].
- [26] “5 recomendaciones para optimizar el rendimiento de datos en tu empresa”, *E-dea.co*. [En línea]. Disponible en: <https://www.e-dea.co/blog/rendimiento-base-de-datos>. [Consultado: 06-feb-2025].
- [27] “¿Cómo puede optimizar su base de datos para operaciones de lectura y escritura?”, *Linkedin.com*, 13-may-2024. [En línea]. Disponible en: <https://www.linkedin.com/advice/0/how-can-you-optimize-your-database-both-vtucc?lang=es&lang=es&originalSubdomain=es>. [Consultado: 06-feb-2025].
- [28] “¿Cuál es la diferencia entre las pruebas de carga y de estrés?”, *Qalified.com*. [En línea]. Disponible en: <https://qalified.com/es/blog/pruebas-carga-vs-pruebas-estres/>. [Consultado: 06-feb-2025].
- [29] “Buenas prácticas en Git y Github”, *Platzi*, 20-may-2019. [En línea]. Disponible en: <https://platzi.com/blog/buenas-practicas-git-y-github/>. [Consultado: 06-feb-2025].
- [30] J. Carrillo, “La Guía Definitiva para Git Merge y Git Rebase”, *freecodecamp.org*, 07-feb-2021. [En línea]. Disponible en: <https://www.freecodecamp.org/espanol/news/la-guia-definitiva-para-git-merge-y-git-rebase/>. [Consultado: 06-feb-2025].
- [31] O. Montiel, “Git pull explicado”, *freecodecamp.org*, 20-feb-2021. [En línea]. Disponible en: <https://www.freecodecamp.org/espanol/news/git-pull-explicado/>. [Consultado: 06-feb-2025].
- [32] Hariharan, “How to Enable MYSQL DB audit logs”, *Medium*, 12-nov-2024. [En línea]. Disponible en: <https://medium.com/@princehari.selvaraj/how-to-enable-mysql-db-audit-logs-a997b63264e3>. [Consultado: 06-feb-2025].

CONSIDERACIONES

Sugerencia para mejorar el trabajo en equipo y habilidades blandas:

Para optimizar la colaboración, sugiero crear una **tabla de responsabilidades y capacitación**. Esta tabla permitirá monitorear quién es responsable de cada tema, qué actividades se han realizado para capacitar a los compañeros y cuándo se realizaron. Esto fomenta la responsabilidad individual y la transparencia en el equipo.

Ejemplo de tabla de seguimiento:

Responsable	Tema Asignado	Fecha de asignación	Fecha de culminación	Fecha de Capacitación	Capacitación a Compañeros	Observaciones
Guissela Franco	Consultas			01/12/2025	Índices y optimización	Uso de EXPLAIN
Danna López	Seguridad			03/12/2025	Cifrado y control de acceso	Implementación

Mejoras en habilidades blandas:

Comunicación efectiva: Promover reuniones de retroalimentación para que todos los miembros intercambien ideas y soluciones. Fomentar la participación activa en las discusiones y evitar el trabajo mecánico.

Investigación y curiosidad: Incentivar a los miembros a investigar profundamente sobre los temas, identificando problemas no documentados y buscando soluciones innovadoras.

Colaboración activa: Fomentar un ambiente de colaboración, promoviendo sesiones de brainstorming y revisiones entre compañeros, y asegurando que todos estén alineados con el progreso del proyecto.

Responsabilidad colectiva: Asegurar que los miembros del equipo no solo sean responsables de sus tareas individuales, sino también del éxito global del proyecto. Esto incluye apoyar a los compañeros en su aprendizaje.

Temáticas Disponibles

- El grupo es libre de elegir una temática sin repetirse con los demás grupos, seguir el ejemplo indicado

Entregables

```
/Project-AEROLINEAS
/Presentaciones
  Proyecto.pptx
/Informe
  Informe_Proyecto.pdf
/Modelados
  Modelo_ER_Conceptual.png
  Modelo_ER_Logico.png
  Modelo_ER_Fisico.png
/Diccionario_De_Datos
  diccionario_datos.xlsx
/Responsabilidades
  responsabilidades_equipo.xlsx
/Scripts
  /Modelado
    crear_tablas.sql
    relaciones_integridad.sql
  /Seguridad
    crear_rols.sql
    cifrado_datos.sql
  /Auditoria
    activar_auditoria.sql
  /Optimización
    crear_indices.sql
    optimizar_consultas.sql
README.md
```

EXPLICACION

Presentación (PPT):

Crear una carpeta llamada Presentaciones donde se suba el archivo .ppt o .pptx correspondiente a la explicación del proyecto, los objetivos, las actividades, y los resultados alcanzados.

Documento Informe:

- Subir el informe detallado del proyecto en formato .docx o .pdf, incluyendo:
 - Resumen ejecutivo
 - Descripción de cada fase del proyecto
 - Resultados obtenidos
 - Conclusiones

Modelados (ER):

Crear una carpeta llamada Modelados para almacenar los diagramas de modelado ER. Estos pueden estar en formatos como .png, .jpg, .pdf.

- Incluir versiones del modelo conceptual, lógico y físico.

Diccionario de Datos:

Subir un archivo en formato .xlsx o .csv que contenga el diccionario de datos. Este debe incluir detalles como:

- Nombre de la tabla
- Descripción de la tabla
- Campos (nombre, tipo de datos, restricciones, etc.)
- Relación con otras tablas

Responsabilidades:

Subir un archivo que detalle las responsabilidades de cada miembro del equipo, indicando qué tareas corresponden a cada uno. Este archivo puede ser una tabla en formato .xlsx o .docx.

Script Actividades a Realizar:

Subir los scripts de las actividades, como la creación de la base de datos, la implementación de procedimientos almacenados, vistas, triggers, etc. Estos archivos pueden ser .sql o .sh (si son scripts de shell para automatizar tareas). Estos scripts deben estar organizados en carpetas según la actividad, por ejemplo, Scripts/Modelado, Scripts/Seguridad, Scripts/Auditoría, etc.

RUBRICA

Criterio	Descripción	Puntos
1. Modelado de Base de Datos y Diccionario de Datos		8
Diseño del Modelo Conceptual, Lógico y Físico	El modelo ER refleja las entidades y relaciones correctamente.	4
Diccionario de Datos	El diccionario de datos es detallado, claro y bien estructurado, incluye tablas, campos, relaciones y restricciones.	2
Restricciones de Integridad Referencial	Se definen correctamente las claves primarias y foráneas entre las tablas.	1
Escalabilidad y Mejores Prácticas	El modelo incluye prácticas recomendadas para la escalabilidad y la integración de sistemas de reservas.	1
2. Seguridad, Auditoría y Control de Acceso		8
Políticas de Acceso y Seguridad	Roles y permisos bien definidos para controlar el acceso a las tablas y vistas.	3
Cifrado de Datos Sensibles	Se implementa correctamente el cifrado de datos sensibles, como contraseñas y detalles de pago.	2

Auditoría y Registro de Eventos	Se habilitan herramientas de auditoría para monitorear el acceso y las actividades de los usuarios en la base de datos.	3
3. Respaldos y Recuperación de Datos		5
Respaldos Completos e Incrementales	Los respaldos completos e incrementales están implementados y explicados correctamente.	3
Respaldos en Caliente	Se implementa correctamente el respaldo sin interrumpir el servicio (hot backups).	2
4. Optimización y Rendimiento de Consultas		5
Índices y Optimización de Consultas	Se implementan índices apropiados y se optimizan consultas SQL con herramientas como EXPLAIN.	3
Particionamiento de Tablas	El particionamiento de tablas está correctamente implementado y explicado.	2
5. Procedimientos Almacenados, Vistas y Triggers		5
Procedimientos Almacenados	Se crean procedimientos almacenados para cálculos y tareas recurrentes.	2
Vistas y Simplificación de Consultas	Se crean vistas para simplificar consultas complejas y mejorar el acceso a datos.	2
Triggers para Auditoría y Control de Cambios	Se implementan triggers para mantener un historial de cambios y automatizar tareas.	1
6. Monitoreo y Optimización de Recursos		2

7. Git y Control de Versiones		2
Uso de Git y Control de Versiones	El repositorio está correctamente configurado, con commits claros y frecuentes.	1
Colaboración y Flujo de Trabajo en Git	Se siguen buenas prácticas en el flujo de trabajo (uso de ramas, fusión, etc.).	1
Total		35