ANALISIS Y DISEÑO DE LA

ARQUITECTURA

DE BASE DE DATOS

LEVANTAMIENTO DE REQUERIMIENTOS FUNCIONALES Y NO FUNCIONALES

Ejercicio Propuesto:

Elaborar una lista de requerimientos funcionales y no funcionales para una biblioteca digital.

Lista de Requerimientos Funcionales para una Biblioteca Digital

1. Registrar usuarios (lectores, bibliotecarios y administradores).
2. Permitir el inicio de sesión y recuperación de contraseña.
3. Realizar búsquedas de libros por título, autor o categoría.
4. Permitir reservas y préstamos de libros digitales.
5. Mostrar disponibilidad de ejemplares en tiempo real.
6. Enviar notificaciones por correo cuando un préstamo esté por vencer.
7. Registrar el historial de préstamos y descargas.
8. Permitir calificaciones y comentarios sobre libros.
9. Administrar el catálogo de libros (agregar, editar, eliminar).
10. Generar reportes de uso, préstamos y usuarios activos.

Lista de Requerimientos No Funcionales para una Biblioteca Digital

1. Rendimiento: el sistema debe responder a cada solicitud en menos de 2 segundos.
2. Seguridad: cifrado de contraseñas y conexión HTTPS.
3. Usabilidad: interfaz intuitiva accesible desde computadoras y móviles.
4. Disponibilidad: el sistema debe estar activo 24/7 con un 99% de tiempo operativo.
5. Escalabilidad: capacidad para atender más de 10,000 usuarios concurrentes.
6. Compatibilidad: soporte para navegadores modernos (Chrome, Firefox, Edge).
7. Mantenibilidad: código modular y documentado.
8. Respaldo: copias automáticas de la base de datos cada 12 horas.

DISEÑO LÓGICO Y FÍSICO DE LA BASE DE DATOS

Ejercicio Propuesto:

Proponer un plan de indices y claves para optimizar una base de datos de e-commerce.

Claves primarias (Primary Keys)

- Clientes: id_cliente
- Productos: id_producto
- Pedidos: id_pedido
- Pagos: id_pago
- Carrito: id_carrito

Claves foráneas (Foreign Keys)

- Pedidos: id_cliente (referencia a Clientes), id_pago (referencia a Pagos).
- Carrito: id_cliente, id_producto.
- Pagos: id_pedido.

Índices propuestos

1. Índice en email dentro de Clientes → para agilizar búsquedas de login.
2. Índice compuesto en Productos (nombre, categoría) → para acelerar filtros.
3. Índice en fecha_pedido de Pedidos → para reportes y estadísticas.
4. Índice en método_pago de Pagos → para análisis de transacciones.
5. Índice en id_producto de Carrito → para obtener contenido rápidamente.

Estrategias adicionales

- Normalizar tablas hasta 3FN para evitar redundancia.
- Crear particiones de pedidos por año si el volumen es alto.
- Implementar cache para productos más consultados.
- Usar vistas para reportes predefinidos y optimizar consultas frecuentes.

SELECCIÓN DEL DBMS ADECUADO SEGÚN LAS NECESIDADES DEL PROYECTO

Ejercicio Propuesto:

Justificar el uso de un DBMS relacional frente a uno NoSQL en un caso específico

CASO:

Imaginemos que una universidad virtual está desarrollando un sistema de gestión académica que permita administrar cursos, docentes, estudiantes, calificaciones, y matrículas. Este sistema necesita manejar una gran cantidad de datos estructurados que incluyen:

- Usuarios: estudiantes, profesores, y administradores.
- Cursos: información detallada sobre cada curso (nombre, código, fechas, requisitos).
- Matrículas: qué estudiante se ha matriculado en qué curso.
- Calificaciones: registro de notas y comentarios de los profesores.

Este sistema debe ser capaz de realizar consultas complejas que incluyan la relación entre estudiantes, cursos, profesores y calificaciones. También debe ser posible generar reportes detallados de los estudiantes, su rendimiento académico, y el estado de las matrículas.

JUSTIFICACIÓN PARA EL USO DE UN DBMS RELACIONAL (SQL):

- Estructura Definida: Los datos tienen relaciones claras entre sí (por ejemplo, estudiantes, cursos y calificaciones), lo que hace que un DBMS relacional sea ideal para gestionarlos de manera eficiente.
- Integridad Referencial: La base de datos necesita mantener relaciones consistentes entre estudiantes, matrículas y calificaciones, lo cual se asegura con claves foráneas en un DBMS relacional.
- Transacciones Seguras (ACID): La necesidad de realizar operaciones como matricular a estudiantes y registrar calificaciones requiere la confiabilidad y seguridad que un DBMS relacional proporciona con sus propiedades ACID.
- Consultas Complejas: El sistema requiere consultas detalladas sobre estudiantes, cursos y calificaciones, lo cual es más adecuado para un DBMS relacional, que facilita operaciones como JOINs y filtros complejos.

CONCLUSIÓN:
Un DBMS relacional (como PostgreSQL) es la mejor opción para sistemas académicos o administrativos con relaciones complejas, necesidad de integridad y seguridad transaccional. NoSQL se reserva para casos con datos no estructurados o de alta variabilidad, como redes sociales o análisis de logs.