Justificación Arquitectónica de Bases de Datos

Este documento explica las decisiones técnicas detrás de la selección de PostgreSQL, MongoDB y Redis para optimizar el almacenamiento de datos en nuestro sistema. Cada elección se alinea con el tipo de información y los requisitos de rendimiento y consistencia.

La Columna Vertebral de los Datos: Una Visión General

La selección de PostgreSQL, MongoDB y Redis no es arbitraria; responde a las características intrínsecas de cada motor y su idoneidad para tipos específicos de información. A continuación, justificamos por qué cada motor es la elección ideal para su categoría de datos.

```
/sql/
   modelo_conceptual.pdf
   modelo_relacional.pdf
   create_tables.sql
   insert_data.sql
   queries_avanzadas.sql

/mongodb/
   diseño_colecciones.md
   inserts.json
   consultas_aggregation.md

/redis/
   comandos_basicos.txt
   operaciones_estructuras.txt
   casos_de_uso_redis.md

/documentacion/
   arquitectura_de_datos.pdf
   conexion_entre_las_3_bases.m

README.md
```

PostgreSQL (SQL): Integridad para Datos Críticos

PostgreSQL, un robusto sistema de gestión de bases de datos relacional, es el pilar para la información que demanda la máxima fiabilidad. Su cumplimiento con las propiedades ACID (Atomicidad, Consistencia, Aislamiento y Durabilidad) lo hace insustituible para datos financieros y transaccionales.

PostgreSQL: Garantía en Datos Financieros y Transaccionales

Cuentas Bancarias

Requieren integridad absoluta para saldos y movimientos. PostgreSQL impone restricciones como llaves primarias y foráneas, protegiendo la consistencia de forma que los motores NoSQL no pueden igualar en escenarios financieros críticos.

Transacciones (ACID)

Operaciones como depósitos o transferencias necesitan completarse íntegramente o no ejecutarse en absoluto. PostgreSQL garantiza esta seguridad transaccional, previniendo errores como "dinero fantasma" o registros incompletos.

PostgreSQL: Seguridad y Trazabilidad en el Sistema

Usuarios y Roles

Para sistemas con control de acceso granular, PostgreSQL ofrece un modelo de seguridad maduro que define con precisión quién puede acceder, modificar o administrar la información, asegurando la privacidad y el control.

Auditoría

El registro de todas las acciones es crucial. PostgreSQL facilita este proceso mediante triggers y logs, permitiendo un seguimiento exacto de cada operación para fines de cumplimiento y análisis de seguridad.

En resumen, PostgreSQL es la mejor opción cuando se necesitan altas garantías de integridad, seguridad y consistencia en operaciones financieras y datos críticos.

MongoDB (NoSQL – Documentos): Flexibilidad para Datos Evolutivos

MongoDB sobresale en el almacenamiento de información en documentos JSON, lo que lo hace ideal para manejar estructuras dinámicas y semi-estructuradas. Su capacidad para adaptarse a esquemas cambiantes es fundamental para la agilidad del desarrollo.

MongoDB: Adaptabilidad para la Experiencia del Usuario

Preferencias del Usuario

Cada usuario puede tener configuraciones únicas (temas, idiomas). MongoDB permite guardar estas estructuras variables sin necesidad de un esquema rígido, facilitando la personalización a gran escala.

Notificaciones

Las notificaciones son volátiles, cambian rápidamente y no requieren consistencia estricta. MongoDB ofrece alta velocidad de escritura y consulta, gestionando eficientemente grandes volúmenes de datos transitorios.

MongoDB: Eficiencia en Perfiles y Registros

Configuración del Perfil

Los perfiles a menudo incluyen listas y objetos anidados con campos opcionales. La naturaleza basada en documentos de MongoDB maneja estos datos de forma natural, optimizando el almacenamiento y la recuperación.

Registros de Login (Logs)

Los logs se generan con alta frecuencia y su estructura puede variar. MongoDB almacena millones de registros con excelente rendimiento sin impactar otros procesos críticos del sistema.

En conclusión, MongoDB es perfecto para información flexible, masiva y variable, donde la estructura no siempre es fija y la velocidad de escritura es una prioridad.

Redis (Cache / Key-Value): Velocidad para Datos Temporales

Redis es un motor de base de datos en memoria que destaca por su increíble velocidad y eficiencia. Su arquitectura clave-valor y su capacidad para operar como caché lo convierten en una pieza fundamental para optimizar el rendimiento de cualquier sistema.

Redis: Impulsando el Rendimiento y la Simplicidad

Caché de Alto Rendimiento

Redis se utiliza para almacenar datos a los que se accede frecuentemente, reduciendo la carga sobre las bases de datos primarias (PostgreSQL y MongoDB) y acelerando la respuesta de la aplicación.

Estructuras de Datos Versátiles

Desde cadenas y hashes hasta listas y conjuntos, Redis ofrece diversas estructuras de datos que permiten implementar soluciones rápidas para colas de mensajes, contadores en tiempo real o sesiones de usuario.

Sencillez y Rapidez

Su modelo de datos simple y su operación en memoria garantizan una latencia extremadamente baja, lo que es crucial para experiencias de usuario fluidas y funcionalidades que exigen respuestas instantáneas.

Conexión entre las Tres Bases de Datos: Una Arquitectura Integrada

La sinergia entre PostgreSQL, MongoDB y Redis crea una arquitectura de datos robusta y eficiente, optimizada para diferentes necesidades de datos y rendimiento. Cada base de datos juega un rol fundamental, trabajando en conjunto para soportar la agilidad y escalabilidad del sistema.

Sincronización de Datos

Aunque operan de forma independiente para sus datos primarios, puede haber escenarios donde los datos de PostgreSQL, al ser la fuente de verdad, necesiten replicarse o sincronizarse parcialmente con MongoDB, por ejemplo, para enriquecer perfiles de usuario con información transaccional clave.

Estrategia de Caché en Redis

Redis almacena datos de acceso frecuente, como tokens de sesión, configuraciones de usuario personalizadas o resultados de consultas complejas. Esto reduce la carga en PostgreSQL y MongoDB, y acelera drásticamente los tiempos de respuesta para el usuario final.

Flujo de Lectura y Escritura

Las solicitudes de lectura suelen ir primero a Redis. Si el dato no está cacheado, se busca en la base de datos correspondiente (PostgreSQL para datos relacionales, MongoDB para documentos). Las escrituras de datos críticos van directamente a PostgreSQL, mientras que las de datos flexibles se dirigen a MongoDB.

Consistencia Eventual

Adoptamos un modelo de consistencia eventual, especialmente para los datos en MongoDB y Redis. Esto significa que los datos pueden tardar un breve tiempo en propagarse completamente por todo el sistema, una compensación aceptable para la alta disponibilidad y rendimiento que ofrecen estas soluciones.