23-11-2024

Bases de datos no relacionales

PEC2

Juan Luis Acebal Rico
GRADO DE CIENCIA DE DATOS APLICADA

Indice

Ejercicio 1. Casos de uso de una base de datos

1. Razona por qué los modelos relacional y de grafos no son la mejor opción en este caso. Concluye por qué los modelos de agregación son mejores en este caso.

El **modelo relacional** quizás no es el mejor por las siguientes razones:

- **Estructura rígida**. A mi parecer es el motivo más importante, ya que las bases de datos relacionales deben predefinirse su esquema y sus tipos de datos antes de su uso, en una red social, los datos tienden a aumentar y evolucionar con el tiempo, entonces un sistema rígido y muy tipado, si las necesidades del negocio cambian, será un problema adaptar una base de datos relacional al nuevo entorno.
- **Operaciones costosas**. Las relaciones entre tablas pueden implicar múltiples operaciones JOIN, y para no perjudicar la performance, se requeriría muchas estrategias para mejorarla, haciendo muy costoso su uso y mantenimiento, como por ejemplo múltiples índices, uso masivo de caché, particionados, escalados horizontales a gran escala, que llevaría a escalados verticales que se volverían demasiado costosos para la organización y que llevaría todo esto a una situación límite y complicada de solucionar en entornos de producción.
- **Escalabilidad limitada**. Como he nombrado en el punto anterior, el hecho de ser una necesidad escalar se empezaría por un escalado horizontal que es menos costoso, pero requiere más equipo humano para preparar y mantener el sistema de bases de datos. Sin embargo, no sería suficiente, ya que las bases de datos relacionales no están diseñadas para ser escaladas horizontalmente, se necesitaría escalar verticalmente y, no sería una opción hacerlo con las cantidades masivas de datos que usa Instagram en forma de fotos y videos, sobre todo.

El **modelo de grafos**, si bien a priori me parece una mejor opción, tampoco creo que sea una buena opción:

- **Complejidad en escalabilidad**. Aunque los grafos están diseñados para representar relaciones (por ejemplo, una publicación es amada por los usuarios 1, 2 y 3... o también los seguidores y seguidos, etc…), las bases de datos de grafos tendrían dificultad ya que se necesitaría una alta interconectividad de los datos. No sé yo hasta qué punto distribuir un grafo muy grande y altamente conectado puede no degradar el rendimiento, los costos y yo creo que se volvería muy complicado de gestionar.
- **Requerimientos de rendimiento**. Las consultas de grafos complejas pueden ser muy intensivas computacionalmente y en una aplicación de alto volumen de lecturas como Instagram, seguro que afectaría a la latencia y la experiencia del usuario.

Los **modelos de agregación**, que una explicación sencilla sería un índice para un conjunto de documentos, o datos, es decir, BBDD orientadas a documentos o BBDD clave-valor, yo lo veo una excelente elección. En este caso se tendría que usar una combinación de ambos. Si es verdad que los datos estarían menos estructurados que en una BBDD relacional o de grafos, pero tendría la gran ventaja del rendimiento, escalabilidad, costos y desde mi punto de vista, sobre todo, el mantenimiento, ya que en entornos productivos el mayor problema es montar un sistema, que luego se vuelve muy costoso de mantener, con muy mal rendimiento, y que el costo de migrar es aún mayor. Entre sus ventajas, destacaría:

- **Escalabilidad horizontal**. Están diseñados para escalar casi infinitamente, la razón es dando una pequeña comparación o símil, la infraestructura de internet. Cuando entras en un sitio web, tienes un DNS que resuelve tu nombre del sitio web a visitar (la clave), y te da la IP (el valor) de donde te tienes que comunicar. Es un claro ejemplo de una base de datos de agregación con miles de millones de registros con latencias cercanas a cero para acceder al registro. Es decir, son sistemas preparados para agregar nodos al sistema y poder manejar grandes cantidades de tráfico.
- **Esquema flexible**. Permiten almacenar datos de todo tipo, siguiendo el ejemplo anterior, un DNS puede almacenar una IP, o a su vez otro nombre de dominio, el único límite es el uso para el cual ha sido diseñado.
- **Eficiencia en lecturas y escrituras**. Están optimizados tanto para la lectura y escritura mediante la agrupación de datos relacionados en datos agregados, lo cual elimina la necesidad de operaciones de estilo JOIN, es decir, para una zona geográfica se podrían agrupar y agregar los datos, para un tipo de cuenta igual, incluso para tipos de contenidos.
- **Están optimizados para acceso por su clave**. Los modelos de agregación permiten acceder a datos por claves, es muy útil para recuperar documentos tales que post de usuarios, o contenido multimedia.

La conclusión sería que están optimizados para escalar, ser flexibles y dar rendimiento en operaciones tanto de lectura como escritura y serían una excelente opción antes que usar un modelo relacional o de grafos.

2. Explicar qué estrategia de fragmentación (partición) y de replicación de los datos sería la óptima.

La estrategia de fragmentación y replicación óptima podría estar basada en diferentes criterios, pero los que yo veo más adecuados son por regiones geográficas, zonas horarias, y también por usuarios. Con esto podemos reducir la latencia ya que los datos estarán cerca del usuario. Veo una serie de ventajas, en principio estas podrían ser:

- **Escalabilidad**. Podemos añadir usuarios a otros nodos, o añadir nodos según la cantidad de usuarios aumenta, o dividir los usuarios de una zona geográfica que tengan más de una zona horaria entre estos, con esto se conseguiría equilibrar el uso más aún en las horas pico.
- La **carga** estaría **distribuida** de forma eficiente, y ayudaría a balancear el tráfico entre servidores.
- **Cercanía**: el hecho de que los servidores se encuentren cerca del usuario haría disminuir la latencia que otro tipo de partición.

La estrategia de replicación podría ser maestra-esclavo, con múltiples réplicas que serían dependiendo el nivel de rendimiento. Las lecturas se harían desde las réplicas. Siempre hay más lecturas que escrituras, ya que el hecho de escribir asegura que al menos será leído de 1 a N veces, al tener múltiples nodos para esta función, permitiría reducir la carga de cada nodo, y tener un rendimiento general adecuado. Sin embargo, las escrituras se harían en el nodo maestro. Así podemos mantener la consistencia de los datos, y después de cada escritura, el nodo maestro replicaría a cada nodo esclavo los nuevos registros. Es exactamente, como puse en el apartado anterior, lo que hace un DNS, que utilizan cada día casi toda la población del planeta. Sus ventajas son:

- **Disponibilidad**. Si un nodo falla, hay otros que atienden a las solicitudes.
- **Rendimiento**. Las lecturas se pueden hacer desde tantas replicas como sea necesario, asegurando un rendimiento adecuado.
- **Coherencia** del caso de uso para una red social. En una red social, es aceptable un pequeño retraso en la propagación de datos, igual que pasa en el DNS, ya que una vez se crea un post o una publicación, igual que cuando se modifica un registro DNS, es tolerable unos minutos en los que quizás, para algunos usuarios, sobre todo de otras zonas geográficas, no esté disponible aún la publicación. No se trata de transacciones financieras que requieran requerimientos de un modelo transaccional de tipo ACID (atomicidad, consistencia, aislamiento, durable) propias de un sistema relacional.

3. Indicar el modelo transaccional más adecuado para los requisitos de la aplicación descrita y justificar el porqué.

Como ya he descrito en el anterior punto en la coherencia, no necesitamos un sistema ACID, sino un sistema BASE(básicamente disponible, estado blando, consistencia eventual). Básicamente disponible significa dar siempre una respuesta, aunque no sea inmediata, estado blando (soft state), es que el sistema pueda cambiar la información en el tiempo, debido a la propagación a las réplicas que hace que ciertas actualizaciones no estén disponibles y se ofrezca al usuario versiones anteriores. Por último, consistencia eventual es que, a medio plazo, tanto el nodo maestro como los esclavos tendrán consistencia en sus datos, aunque inicialmente no, ya que se replicarán éstos.

El razonamiento es el alto volumen de lecturas donde hay que priorizar la disponibilidad con tolerancia a errores temporales y con un buen rendimiento. Por tanto, puedo decir que, el modelo BASE es el adecuado para una red social como Instagram, ya que busca el equilibrio entre rendimiento, disponibilidad y admitiendo ciertas inconsistencias temporales pero que no afectan en gran medida al usuario.

4. Supongamos que se ha optado por el uso de quórums como sistema de replicación con N=5, ¿qué valores asignarías a W y a R?

De entrada, lo veo una pregunta abierta en la que no hay una respuesta correcta, sino que hay que valorar ciertos puntos más subjetivos tales como la experiencia y uso de la red social del usuario y el rendimiento. Entiendo que se pretende que sea justificado más que un número concreto.

Sin embargo, en líneas generales, en un sistema con 5 réplicas, para garantizar una consistencia fuerte “(W+R)>N”, el número mínimo de réplicas que deben confirmar una escritura debería de ser al menos 2 o 3, y el número mínimo de réplicas que deben confirmar una lectura debería ser al menos 3 o 4. Es complicado decir un número exacto ya que más alto sea W y R (número de confirmaciones de Write y Read), más consistencia, pero menos rendimiento. Si queremos una consistencia más ligera, pero mucho más rendimiento, podríamos tomar W=2 y R=1, por ejemplo, ya que hay más operaciones de lectura que de escritura, y así tendríamos un rendimiento mucho mejor y no comprometeríamos tanto la consistencia de los datos.

Ejercicio 2

Afirmación 1. **Durante una transacción ACID, todas las operaciones de la transacción se anulan si hay una falla. En este caso, se deshacen todos los cambios desde el inicio y hasta el momento del fallo en la transacción.**

CIERTA. Las propiedades ACID garantizan que una transacción se complete de manera completa o no se ejecute nada. Todas las operaciones hasta el momento del fallo se anulan y se deja la BD Enel lugar donde estaba.

(Pagina 10 de gestionTransacciones.pdf, apuntes donde describe las transacciones ACID (…*o bien se ejecutan todas las operaciones de la transacción (y, en este caso, la transacción confirma los resultados) o bien no se ejecuta ninguna en absoluto*…))

Afirmación 2.**La fragmentación vertical (usar subconjuntos del esquema para distribuir datos) puede complicar las operaciones de inserción y modificación, y comprometer la consistencia de los datos.**

CIERTA. La fragmentación vertical implica dividir una tabla en subconjuntos de columnas, que complica especialmente las operaciones de escritura. Si no existen suficientes mecanismos de sincronización, y control entre tablas, puede comprometer la consistencia de los datos.

(pagina 45 de Distributed databases.pdf, 4.1.4 Fragmentation in Oracle 2)Verticalfragmentation: explica cómo se puede poner en peligro la consistencia de los datos el aumento de inserciones y modificaciones: (…updating/inserting data in a replicated copy takes more time, and synchronizing such writings may not be trivial…) y (…As a result, consistency of copies may be affected…))

Afirmación 3. **El teorema CAP es especialmente importante en el contexto NoSQL, ya que NoSQL ofrece distintos tipos de consistencia para dar soporte a distintas necesidades de disponibilidad de datos que se adaptan a distintas estrategias de distribución de datos.**

CIERTA. El teorema CAP establece que un sistema distribuido solamente puede garantizar 2 de las 3 propiedades de este. Es decir, Consistencia(C), Disponibilidad(A) y Tolerancia a Particiones(P). En NoSQL los sistemas tienen distintas combinaciones de estas propiedades según las necesidades específicas, en general CP o AP, y en BD relacionales es CA y ofrecen diferentes niveles de consistencia y disponibilidad para adaptarse a diversas estrategias de distribución de datos.

Es decir, si bien la disponibilidad total no hay, y eso define el teorema CAP, nunca puede ser las 3 partes del sistema, pero en el contexto de la afirmación, las bases de datos NoSQL ofrecen distintos tipos de consistencia precisamente por eso, para adaptarse a la disponibilidad, que da una consistencia fuerte, o eventual para adaptarse a las estrategias de distribución de datos.

(Pagina 9 de B3_T7_BDD_BASE.pdf (…Desde la perspectiva de NoSQL, el teorema CAP captura la idea de que la consistencia puede no ser compatible con la gestión de datos altamente distribuidos …))

Afirmación 4. ***L*****a principal ventaja del modelo de replicación maestro-esclavo es la consistencia. No permite que diferentes clientes lean valores diferentes de un dato desde diferentes nodos secundarios.**

FALSA. No se cumple para la política maestro esclavo asíncrona. En un modelo de replicación maestro-esclavo la principal ventaja en general es la disponibilidad y el rendimiento, no es la consistencia, que sería una desventaja, o al menos, no garantiza la consistencia al 100%, y dependiendo el uso, no podría utilizarse un sistema de replicación maestro-esclavo.

(Página 7 de B3_T7_BDD_BASE.pdf (…Las operaciones de lectura se pueden servir a partir de cualquier réplica y siempre se obtendrá el valor más recientemente confirmado…)

Afirmación 5. **En un sistema P2P, siendo una red lógica que se superpone sobre una red física, todos los nodos pueden ser diferentes en hardware y nunca puede existir una correlación entre un nodo y los datos que este almacena.**

FALSA. En sistemas P2P, es decir, usuario a usuario, aunque si pueden variar a nivel físico, a nivel lógico puede haber o no una correlación entre un nodo y la información que almacena. La distribución de datos en una red P2P estructurada tiene relación entre los datos que almacena, en una no estructurada, no. Un ejemplo es blockchain, que es una distribución de datos que es igual en todos los nodos, por tanto, respecto a la pregunta, un sistema P2P puede, o no, tener correlación entre los datos que almacena.

(Pagina 10 de B3_T5_2_BDD_Arquitecturas.pdf (…En una red P2P no estructurada no existen restricciones acerca de en qué nodos del sistema se tienen que almacenar los datos… y …en una red P2P estructurada, existe un procedimiento que determina qué datos almacena cada participante…))

Ejercicio 3

Después de una lectura detallada, no hay una respuesta única, estamos ante 4 casos de uso diferentes, en los que en los dos primeros las lecturas son más frecuentes que las escrituras, el tercero es igual de importancia en lectura y escritura. El ultimo es más importante la escritura. La consistencia es en el primero y tercero fuerte, y las otras dos es a final de tiempo, no le importa al sistema que no tenga consistencia eventual.

A. Sistema de gestión de historiales médicos (consistencia fuerte, lecturas más frecuentes que escrituras).

Este sistema se caracteriza por necesitar una consistencia fuerte, con lecturas más frecuentes que escrituras. Quizás el más apropiado sería C5: N=7, W=4, R=5.

La razón es que se debería poder cumplir que todas las lecturas vean los datos más recientes. Para cumplir eso tendría que ser (W+R)>N, así se asegura que cualquier lectura al menos estará en una réplica confirmada por la escritura. Además, los historiales médicos normalmente se abren, y luego se modifican en cada visita o en casi todas las visitas, entonces el equilibrio con un poco más de prioridad a la lectura, es adecuado. Como segunda opción elegiría C1: N=5, W=5, R=5

B. Un periódico online que es visitado por lectores de diferentes lugares de un país (consistencia final en el tiempo, lecturas más frecuentes que escrituras)

Aquí se requiere una consistencia final en el tiempo, y las lecturas son más frecuentes, por tanto, C6: N=5, W=2, R=2.

No se requiere que las lecturas sean inmediatamente consistentes con las escrituras, y se permite la tolerancia a retrasos en la propagación de los datos. Las lecturas seria W+R = 4 < 5, no garantiza una consistencia fuerte, pero es adecuado para una consistencia eventual. Lo veo adecuado para la gran cantidad de lecturas, y la pequeña cantidad de escrituras. También podríamos elegir segunda opción C4: N=7, W=1, R=5

C. Un sistema de control aéreo (consistencia fuerte, escrituras y lecturas de la misma importancia)

Aquí necesitamos una consistencia fuerte y misma importancia de R/W por tanto, la solución adecuada es C1: N=5, W=5, R=5 o si no estuviera disponible elegiría C7: N=7, W=5, R=4 en ambos tenemos que las confirmaciones en los nodos de lectura y escritura sea mayor que el número de nodos. Por último, la C1 tiene una consistencia muy fuerte (y más latencia también) respecto a C7 que tiene una consistencia fuerte, que es valorable si queremos más consistencia aún.

D. Un sistema de sensores para analizar la calidad del aire de una ciudad (consistencia final en el tiempo, escrituras más frecuentes que lecturas)

Aquí tenemos la necesidad de consistencia al final de tiempo, y escrituras más que lecturas. De las combinaciones dadas puede ser C6: N=5, W=2, R=2 y si C6 no estuviera disponible elegiría C7: N=7, W=5, R=4, ya que la primera tiene una consistencia eventual baja, es decir, tiene mayor rendimiento, y tiene W=R en C6, que si bien no es W>R , es de las opciones dadas la más correcta.

Sin embargo, en C7 hay más W que R, y no cumple que tenga consistencia al final del tiempo, como he decidido poner para cada sistema la segunda mejor opción, la pongo, pero ninguna se adapta aparte de la C6 ya que o tienen consistencia fuerte, o priorizan las lecturas a las escrituras.

Por último, no he podido utilizar ni C2 ni C3 ya que parecen erróneas, bien es parte de la actividad saber que no es posible C2 ni C3, o bien es una errata, pues no puede ser más nodos que confirmaciones de lectura o escritura, es decir N<R o N<W ya que estaríamos diciendo que tenemos 8 nodos de lectura, 7 de escritura, y en total tenemos 3 nodos, no es coherente, por último, el concepto de R y W es cuantos nodos han confirmado, entonces la suma de los dos si puede ser mayor a N siempre y cuando ninguno de estos individualmente sea mayor a N, es decir es correcta o posible una configuración solo si R<=N & W<=N.