Comparativa de Tecnologias de Disponibilidad en Bases de Datos

Carlos Maldonado, Alfredo Huillca, Daniela Soto, Alexander Huallpa, Laura Condori

September 19, 2021

I. Resumen

La asignación de objetos relacionales es una técnica que le permite asignar cada una de las filas de nuestras tablas en la base de datos de objetos, donde las columnas de la tabla corresponden a las propiedades de estos objetos. La técnica, utilizada en la programación para convertir los tipos de datos con los que trabajan en un lenguaje orientado a objetos, los tipos de datos con los que trabajan en un sistema de base de datos relacional para la persistencia de datos en el objeto de mapeo-relacional

II. Abstract

The assignment of relational objects is a technique that would assign each of the rows of our tables in the object database, where the columns of the table correspond to the properties of these Objects. The technique, used in programming to convert the data types they work with into a object-oriented language, the data types that you work with in a relational database system for data persistence in the relational-mapping object

III. Introduccion

ORM es el mapeo objeto-relacional (más conocido por su nombre en inglés, Object- Relational mapping), consiste en una técnica de programación para convertir datos entre el lenguaje de programación orientado a objetos utilizado y el sistema de base de datos relacional utilizado en el desarrollo de aplicaciones. Esto posibilita el uso de las características propias de la orientación a objetos (básicamente herencia y polimorfismo). Entre estos paquetes comerciales tenemos una lista alfabética de los principales motores de mapeo objeto relacional, tales como: ColdFusion, Common Lisp, Java, JavaScript, .NET, Perl, PHP, Python, Ruby, Smalltalk, C++. El problema que surge, porque hoy en día prácticamente todas las aplicaciones están diseñadas para usar la Programación Orientación a Objetos (POO), mientras que las bases de datos más extendidas son del tipo relacional y estas solo permiten guardar tipos de datos primitivos (enteros, cadenas de texto) por lo que no se puede guardar de El problema que surge, porque hoy en día prácticamente todas las aplicaciones están diseñadas para usar la Programación Orientación a Objetos (POO), mientras que las bases de datos más extendidas son del tipo relacional y estas solo permiten guardar tipos de datos primitivos (enteros, cadenas de texto) por lo que no se puede guardar de

IV. Desarrollo

i. Tecnologia Sharding

Una de las técnicas para el manejo de bases de datos que está tomando vida en la comunidad criptográfica actual es el Sharding. El sharding es una forma de segmentar los datos de una base de datos de forma horizontal, es decir, partir la base de datos principal en varias en bases de datos más pequeñas y repartiendo la información. De esta forma lo que se consigue es

una partición de datos en diferentes bases que tengan cierta homogeneidad, para conseguir una escalabilidad mucho más rápida.

El sharding se creó con la finalidad de permitir una mayor escalabilidad en sistemas distribuidos y descentralizados. Pero en la actualidad, su aplicación en la tecnología blockchain podría mejorar considerablemente los problemas de escalabilidad a los que se enfrentan redes como Bitcoin y Ethereum.

i.1 ¿Como Funciona el Sharding?

Básicamente lo que hace es dividir la red en grupos de nodos más pequeños, para que estos se concentren en procesar transacciones de un grupo limitado de usuarios o smart contracts. Con ello el sharding logra usar de forma más eficiente el poder de los nodos de la red y ofrecer respuestas más rápidas a las transacciones.

Estos nodos se encuentran orquestados por un protocolo de funcionamiento, que evita que los grupos de nodos puedan solaparse en sus funciones. Una medida que agrega seguridad a la red y evita conflictos en su funcionamiento. Finalmente, el aumento en la eficiencia tanto en el procesamiento de transacciones, como en la validación y almacenamiento de las mismas, repercute en la capacidad global de la red. Esto en forma de un aumento significativo en la escalabilidad, velocidad de procesamiento, almacenamiento, redundancia y funciones de la red en general.

i.2 Sharding y los desafíos a superar

Sin duda el sharding es una opción poderosa para escalar la tecnología blockchain. Pero su aplicación tiene una serie de desafíos que deben resolverse.

En primer lugar, usar sharding hace que cada pequeño grupo de nodos sea más susceptible de ser atacado. Esto se debe a que cada pequeño grupo de nodos mantendría una copia de su sub-cadena y al ser más pequeño, un atacante puede dirigir un ataque a estos de forma más sencilla para reescribirla o realizar una denegación de servicio. Problemas como el Ataque de 51 por ciento, el Ataque de carrera o el Ataque de Vector 76 se hacen más sencillos de llevar a cabo. Después de todo, no atacas a una red con miles de nodos, sino algo mucho menor.

Por ejemplo, para realizar un ataque de 51 por ciento en una blockchain con sharding solo necesitamos tener bajo control el 5,1 por ciento de los nodos totales de la red. Esto se debe, a que las redes que usan sharding serían susceptibles al llamado Ataque del 1 por ciento. Lo que significa que solo basta con controlar el 1 por ciento de un grupo para controlar el poder un grupo de nodos en sharding. Esto es un enorme problema de seguridad. Uno que tiene que atenderse antes de poder aplicar el sharding en blockchains como Ethereum.

En segundo lugar, nos encontramos con el problema de la selección de nodos para la validación. Siendo usuarios de la blockchain podemos emitir una transacción y esperamos que esta sea atendida por algún nodo dentro del sharding. Sin embargo, la estructura del sharding abre las puertas para que un atacante que controle cierta cantidad de nodos pueda validar la transacción en diferentes segmentos de la red al mismo tiempo. Como resultado, el atacante podría llevar a cabo ataques de doble gasto.

Sin embargo, esto tiene ya una solución práctica. Esta pasa por crear un sistema que permite asignar las transacciones de forma aleatoria a los nodos que forman parte de la red. De esta forma, la red velará por asignar transacciones a distintos grupos de nodos y que estos procesan la transacción en su momento. El proceso sin embargo introduce una fuerte latencia y otros problemas de seguridad no estudiados a profundidad aún.

i.3 Ventajas del sharding

En la práctica lo primero que vamos a notar es que el sharding nos ofrece unos accesos de gran velocidad. No importa, si por ejemplo disponemos de servidores en distintas zonas del mundo, dado que el sharding se ocupa de que se reduzca el volumen de latencia y

el rendimiento sea superior. Con el sharding vamos a poder gestionar mejor las empresas en las que se ha llegado a alcanzar una cantidad de datos enorme que ha llevado a que la velocidad de acceso se reduzca de forma considerable. Si por el contrario nuestro servidor no tiene tantos datos, pero sí registra una gran cantidad de escrituras, también nos veremos beneficiados de ello. Al fin y al cabo, hay que tener en cuenta que las escrituras se interpretan como bloqueos en el uso de los recursos, lo que acaba derivando en que haya problemas de rendimiento en la forma en la que los usuarios interactúan con nuestro sistema.

El sharding se encuentra presente en la mayor parte de los sistemas actuales, desde Hibernate hasta Apache, MongoDB o MySQL a través de distintas implementaciones. Por ejemplo, Oracle NoSQL Database introduce el sharding de manera automatizada con expansión online, mientras que la base de datos Spanner de Google lo ofrece por medio de distintas máquinas Paxos.

Otras de las ventajas de este escalado residen en cómo se reduce la cantidad de filas en las tablas de las bases de datos, minimizando de manera simultánea el tamaño de los índices. El efecto posterior de esto no es solo el aumento de espacio liberado, sino también que vamos a comprobar cómo las búsquedas se realizan con una mayor rapidez. Además, teniendo en cuenta que el sharding utiliza distintas máquinas para los shards, esto también beneficia a un aumento en el rendimiento.

i.4 Desventajas del Sharding

Hay expertos que no recomiendan utilizar el sharding en el inicio de un proyecto debido a que hay que entender sobre todo cuándo utilizarlo y dentro de qué límites de magnitud aprovecharlo. En parte por algunas de las desventajas que podemos comentar, como que la complejidad del SQL se incrementa, llevando a que los desarrolladores tengan que escribir código más elaborado. Este volumen de complejidad no solo se aplica al código, sino que también deriva en que la integridad del sistema se pueda encontrar en mayor riesgo, con distintos factores que pueden sufrir problemas, como el balance y las particiones.

Otra desventaja importante es cómo solo la corrupción que se pueda sufrir en uno de los shards del sistema puede llevar a que se produzca un fallo del sistema generalizado. Además, los servidores de recuperación están obligados a contar con copias de los shards de la base de datos, los backups son más complicados de realizar y se introduce una complejidad operacional que lo hace todo más complicado en términos globales.

ii. Herramienta Entity Framework

Entity Framework es el ORM de Microsoft, con versiones tanto para la plataforma .NET "tradicional" como para .NET Core. Como vimos en el articulo del enlace anterior, en el que se expliocaba con detalle que es un ORM, este tipo de software puede funcionar de varias maneras diferentes a la hora de "mapear" las clases de nuestro programa orientado a objetos y las tablas en la base de datos Entity Framework no es una excepcion, y nos ofrece diversas maneras de trabajar con los datos desde nuestros porgramas. Cadda una tieneun enfoque diferente y es interesante para ciertos casos concretos, ademas de tener sus beneficios y problemas. Es importante tener en cuenta que las capacidades de Entity Framework en .NET "tradicional" y en .NET core son copletamente diferentes. Asi los tres modos de trabajo descritos a continuacion estan completamente soportados en EF6, pero EF Core solamente soporta "Code First" en .NET Core ni esta ni se le espera.

V. Conclusiones

El uso de un ORM es una alternativa sumamente efectiva a la hora de trasladar el modelo conceptual(orientado a objetos) al esquema relacional nativo de la bas bases de datos SQL. Evita la inclusion de sentencias SQL embebidas en el codigo de la aplicacion, lo que a su vez facilita la migracion hacia otro sistemas gestor

de bases de datos, Incorpora una capa de abstraccion. Al ser realizado, en esta capa, de manera automatica la conversion de instrucciones orientadas a objetos, a sentencias SQL, minimiza la ocurrencia de errores humanos.

De cualquier modo, utilizar un ORM no debe ser considerado una panacea, sino que debe usarse a discrecion; teniendo en cuenta las particularidades de cada problema a modela. En determinados casos no es recomendable el uso de un ORM, sobre todo cuando se imponen tiempos de respuesta minimos o se requiere una menor sobrecarga. En estos casos lo mas conveniente es el uso de un microORM; evitando siempre que sea posible las inyecciones de SQL Inline.

VI. Recomendaciones

References

[1] Raquel García (2020) Sharding, ventajas y desventajas de su aplicación, Recuperado 19 de Setiembre del 2021, de: https://blog.mdcloud.es/sharding-ventajas-y-desventajas/

[2]

[3]

[4]

[5]

[6]

[7]

[8]

[9]