Bases de datos NoSQL 2

IT BOARDING

BOOTCAMP

En este módulo vamos a explicar los algunos conceptos teóricos del diseño en las bases de datos NoSql y sharding.

IT BOARDING

BOOTCAMP

Índice

IT BOARDING

BOOTCAMP

// Diseño de bases de datos no relacional

IT BOARDING

BOOTCAMP

Armado

Alguna vez te preguntaste “si las bases de datos relacionales son tan prácticas, ¿en qué situaciones es buena idea trabajar con las no relacionales?”. Si algo tienen de malo las bases de datos relacionales, es que tienen que saber de antemano qué es y cómo es lo que van a almacenar. En cambio, las bases de datos no relacionales son más flexibles con los datos y no les importa su estructura.

Imaginemos que hemos mandado unas máquinas al espacio para que nos reporten qué es lo que encuentran en su viaje. Obviamente, no sabemos a ciencia cierta qué se van a encontrar. De alguna forma, tienen una inteligencia artificial instalada que reconoce los objetos con los que se va encontrando y también tienen sensores instalados. Pero no sabemos bien qué miden, ya que cada máquina tiene sensores diferentes. Cada 24 horas envían un resumen de lo que han visto durante el día.

JSON


```
{
 "maquina_id":1,
 "timestamp":149992693000,
 "coordenadas":"75988823.567, 55375867.098, 12676444.311",
 "encontrado":[
  "roca",
  "agua",
  "roca",
  "roca",
  "algo que parece un animal",
  "roca"
 ],
 "temperatura":{
  "min":-50,
  "max":-49
 },
 "ruido":{
  "min":72,
  "max":4549
 }
}
```

JSON

```
{
 "maquina_id":2,
 "timestamp":1499925677000,
 "coordenadas":"66635675.920, 78021134.727, 53580995.751",
 "temperatura":{
  "min":-50,
  "max":-49
 },
 "humedad":{
  "min":2%,
  "max":5%
 }
}
```

Cada uno de estos documentos JSON contiene la información reportada en cada envío por cada máquina. La máquina con identificador 1 está reportando datos de temperatura y ruido, mientras que la de identificador 2 reporta temperatura y humedad. No sabemos qué sensores tendrá instalados la siguiente y, mucho menos, qué y cómo reportarán las máquinas que aún no se han mandado y los ingenieros están montando.

No merece la pena ponerse a diseñar una base de datos relacional para almacenar esta información. En este caso, lo mejor es dejar a una base de datos no relacional que se trague todo lo que las máquinas reportan, tal cual.

Además, la finalidad del sistema es meramente científica y no se contempla la existencia de usuarios a los que se les deba la garantía de consistencia que ofrecen las bases de datos SQL. Simplemente, se quiere almacenar todo para un futuro análisis.

Una vez almacenados en la base de datos no relacional se podrá pedir y visualizar la información de diferentes maneras. Y si en algún momento se necesita consumir los datos de una forma más estructurada, siempre podremos procesar y volcar la información a una base de datos relacional. Pero es que, muy probablemente, no sea necesario.

Paso de relacional a no relacional

NO SQL

SQL

student_id	age	score
1	12	77
2	12	68
3	11	75


```
[
 {
  "student_id":1,
  "age":12,
  "score":77
 },
 {
  "student_id":2,
  "age":12,
  "score":68
 },
 {
  "student_id":3,
  "age":11,
  "score":75
 }
]
```

Los dos esquemas de la imagen contienen exactamente la misma información. Lo único que cambia aquí es el **formato**: cada **documento** de la **figura** de la **derecha** es una **fila** de la figura de la **izquierda**.

¿Es más fácil verlo en una tabla no?

Lo que pasa es que a menudo en una base de datos no relacional una unidad de **datos** puede llegar a ser demasiado **compleja** como para **plasmarlo** en una **tabla**. Por ejemplo, en el documento **JSON** de la imagen que se muestra a continuación, al tener elementos jerárquicos, es más difícil plasmarlo en una tabla plana. Una solución sería plasmarlo en varias tablas y, por tanto, necesitar de relaciones.

JSON

```
[
{
 "student_id":1,
 "age":12,
 "subjects":{
  "mathematics":{
   "scores":[7,8,7,10],
   "final_score":8
  },
  "biology":{
   "scores":[6,6,5,7],
   "final_score":6
  }
 }
}
]
```

Esto explica por qué las bases de datos relacionales suelen servirse de tablas y las no relacionales de documentos JSON. En cualquier caso, a día de hoy, las bases de datos más competitivas suelen permitir, de una forma u otra, operaciones de los dos tipos. Por ejemplo, el servicio de base de datos en la nube BigQuery que ofrece Google es, en principio, una base de datos de lenguaje de consulta SQL, por lo que permite fácilmente relacionar varias tablas, pero, a su vez, permite insertar elementos jerárquicos JSON, más propios de bases de datos no relacionales.

La diferencia entre el éxito y el fracaso recae, sobre todo, en el diseño del modelo. Es decir, si se decide que el mejor enfoque es usar una base de datos relacional, no es suficiente con meter la información a lo bruto en una base de datos relacional y esperar a que se relacione sola, porque eso no va a ocurrir. De nada sirve elegir la base de datos más apropiada para nuestro sistema, si luego no se hace un buen diseño.

BASES DE DATOS NO SQL 2

//Sharding

IT BOARDING

BOOTCAMP

// Sharding

El sharding es un concepto que se puso de moda dentro de la comunidad criptográfica, debido a los grandes problemas de escalabilidad que tienen las principales plataformas como Bitcoin o Ethereum.

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.

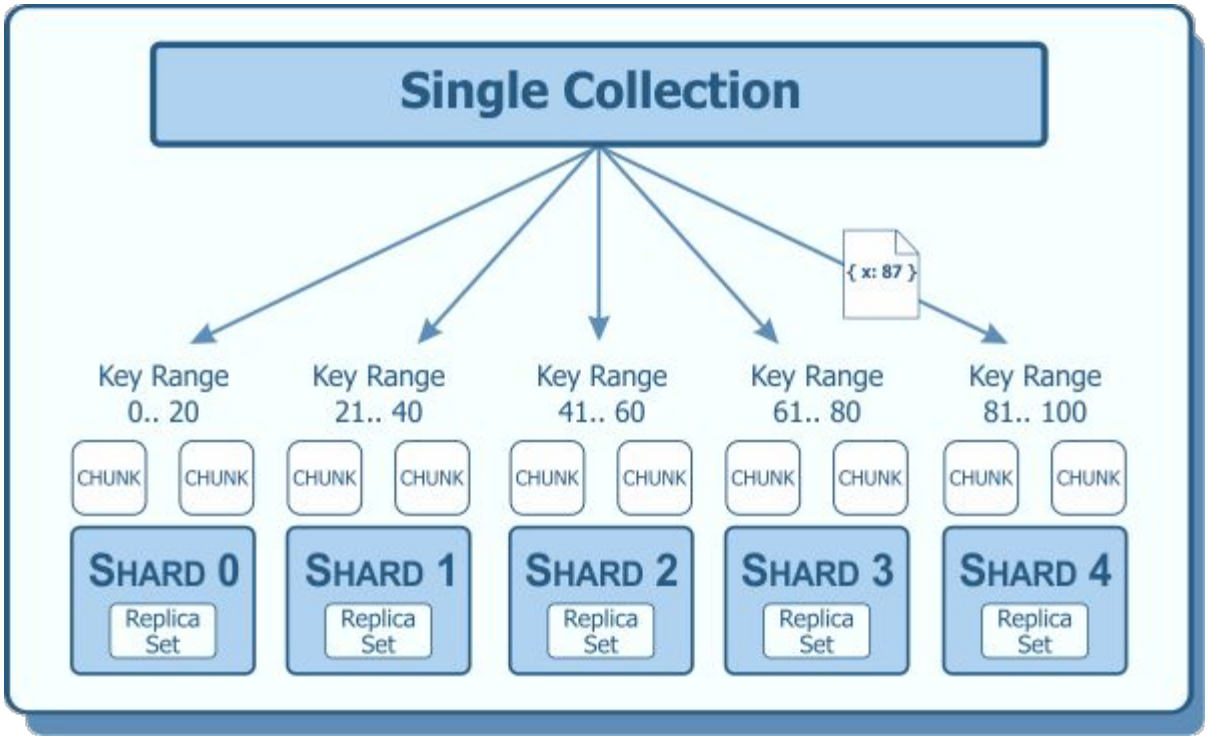
Blockchain

Es un gigantesco libro de cuentas en los que los registros (los bloques) están enlazados y cifrados para proteger la seguridad y privacidad de las transacciones. Es, en otras palabras, una base de datos distribuida y segura (gracias al cifrado) que se puede aplicar a todo tipo de transacciones que no tienen por qué ser necesariamente económicas.

Esa cadena de bloques tiene un requisito importante: debe haber varios usuarios (nodos) que se encarguen de verificar esas transacciones para validarlas y que así el bloque correspondiente a esa transacción (en cada bloque hay un gran número de transacciones que eso sí, es variable) se registre en ese gigantesco libro de cuentas.

¿Como funciona?

Imaginemos una única tabla donde se contienen todos los datos de los usuarios, por razón de edad, sexo, capacidad económica, y geografía. Y ahora pensemos en otro caso donde apliquemos el sharding y dividamos en diferentes tablas a los usuarios por sexo, edad, localización geográfica, y capacidad económica, y yendo aún más allá, dentro de cada rango los dividiremos en más fases, por ejemplo por localización geográfica separemos en diferentes tablas a los usuarios según país de nacimiento, por edades separaremos las tablas por franjas comprendidas entre 18 – 30 años, 31 – 45 años, 46 – 65 años, y 66 en adelante, por capacidad económica haremos lo mismo y diferenciaremos por tablas a las personas con un poder adquisitivo de 0 – 10.000€, 10.001€ a 24.000€, de 24.001€ a 50.000€, de 50.001€ a 120.000€ y de 120.001€ en adelante.


Single Collection
{ x: 87 }
Key Range
0.. 20
Key Range
21.. 40
Key Range
41.. 60
Key Range
61.. 80
Key Range
81.. 100
CHUNK
CHUNK
CHUNK
CHUNK
CHUNK
CHUNK
CHUNK
CHUNK
CHUNK
CHUNK
SHARD 0
Replica Set
SHARD 1
Replica Set
SHARD 2
Replica Set
SHARD 3
Replica Set
SHARD 4
Replica Set

Como podemos comprobar, en el primer ejemplo, tenemos a todos los usuarios en una única tabla, por lo que cuando una base de datos empieza a ser muy numerosa, nos encontramos ante grandes problemas al tener los datos de forma desorganizada y se vuelve inmanejable.

Sin embargo, si aplicamos el sharding como en el segundo ejemplo, podemos encontrar una infinidad de tablas horizontales clasificadas en un cierto grado de homogeneidad, de esta forma, al final lo que estamos consiguiendo es fragmentar una única tabla en múltiples tablas con menos datos cada uno, con esto conseguimos mejorar la escalabilidad al agrupar menos datos en tablas más pequeñas, de esta forma el acceso a ellos es mucho más rápido, y también se consigue mejorar la monitorización de los mismos.

IMPLEMENTACIONES ACTUALES

Para entender mejor este concepto en términos de la blockchain, hay que saber en en la actualidad, la red de Ethereum que es la que más problemas presenta de escalabilidad, funciona de forma que todos los datos que contiene cada transacción deben ser validados por todos los nodos de la cadena, por eso, al fragmentar los datos en shards, lo que estamos consiguiendo es que los nodos se repartan los shards y sólo tengan que validar una parte de los datos que contiene cada transacción, es decir, cada uno valida una parte de la información (se reparten la información) y luego toda la información una vez validada se incorpora a la cadena de forma completa, por lo que conseguimos mantener toda la información en el bloque, con la diferencia de que los nodos tan sólo contienen una parte de la información y no toda como hasta ahora. Con este método se consigue mejorar la eficacia y la escalabilidad al fragmentar la información.

La fragmentación es un tema que aún está en desarrollo para algunas blockchain como Ethereum y Cardano. Los desarrolladores de estos proyectos están estudiando la forma en cómo pueden implementar este método para optimizar sus redes. Pero otras criptomonedas como Zilliqa (ZIL) ya han implementado sharding en sus blockchains. Algo que le ha permitido alcanzar en su red de pruebas un total de 2.828 transacciones por segundo.

Otro interesante proyecto que ha mostrado el poder del sharding es **Telegram Open Network** (TON). Su actual desarrollo, aunque inicial ya ha demostrado que la blockchain TON hace uso de sharding para garantizar el mayor nivel de escalabilidad posible. Sin embargo, debido a problemas con la SEC, Telegram se ha visto imposibilitado de hacer público el lanzamiento de su red. Por esa razón, solo podemos acceder a sus herramientas de desarrollo y testeo.

Curiosamente, la misma situación aplica a **Libra de Facebook**, un proyecto que pese a sus enorme problemas de privacidad, sorprendía a nivel técnico debido al uso del sharding en su red de validadores.

Práctica

Cuestionario

1. **En las bases de datos datos relacionales no es necesario saber de antemano los datos con los datos con los que voy a trabajar**

   a) Verdadero

   b) Falso

1. **Las bases de datos no relacionales son:**

   a) Flexibles

   b) Estructuradas

   c) No estructuradas

   d) Ninguna de las anteriores

1. **¿Cuales de los siguientes son conceptos están relacionados con el Sharding?**

   a) Segmentar en 2 bases datos

   b) Segmentar los datos de una base de datos de forma horizontal.

   c) Centralizado

   d) Escalabilidad lenta

Gracias.

IT BOARDING

BOOTCAMP