ACTIVIDAD 5 - CONCEPTOS Y COMANDOS BÁSICOS DEL PARTICIONAMIENTO EN BASES DE DATOS NOSQL

BRANDON SNEIDER FRANCO QUINTERO
CC1090506954

UNIVERSIDAD IBEROAMERICANA

BASE DE DATOS AVANZADAS

OCTUBRE 2022

BOGOTA

ACTIVIDAD 5 - CONCEPTOS Y COMANDOS BÁSICOS DEL PARTICIONAMIENTO EN BASES DE DATOS NOSQL

DOCENTE
WILLIAN RUIZ MARTINES

ESTUDIANTE
BRANDON SNEIDER FRANCO QUINTERO
CC1090506954

UNIVERSIDAD

IBEROAMERICANABASE DE

DATOS AVANZADAS OCTUBRE

2022

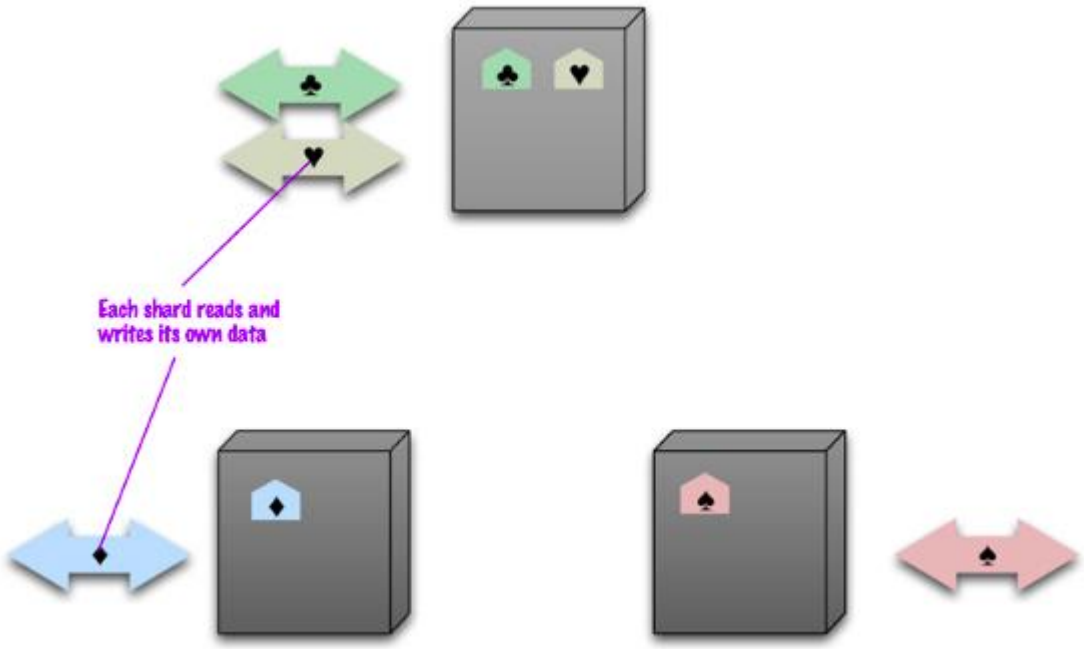
BOGOTA

INTRODUCCION

Dado el modo en el que se estructuran las bases de datos relacionales, normalmente escalan verticalmente - un único servidor que almacena toda la base de datos para asegurar la disponibilidad continua de los datos. Esto se traduce en costes que se incrementan rápidamente, con un límite definido por el propio hardware, y en un pequeño número de puntos críticos de fallo dentro de la infraestructura de datos.

La solución es escalar horizontalmente, añadiendo nuevos servidores en vez de concentrarse en incrementar la capacidad de un único servidor. Este escalado horizontal se conoce como Sharding o Particionado.

PARTICIONADO



El particionado no es único de las bases de datos NoSQL. Las bases de datos relacionales también lo soportan. Si en un sistema relacional queremos particionar los datos, podemos distinguir entre particionado:

Horizontal: diferentes filas en diferentes particiones.

Vertical: diferentes columnas en particiones distintas.

En el caso de las bases de datos NoSQL, el particionado depende del modelo de la base de datos:

Los almacenes clave-valor y las bases de datos documentales normalmente se particionan horizontalmente.

Las bases de datos basados en columnas se pueden particionar horizontal o verticalmente.

Escalar horizontalmente una base de datos relacional entre muchas instancias de servidores se puede conseguir, pero normalmente conlleva el uso de SANs [1] y otras triquiñuelas para hacer que el hardware actúe como un único servidor.

Como los sistemas SQL no ofrecen esta prestación de forma nativa, los equipos de desarrollo se las tienen que ingeniar para conseguir desplegar múltiples bases de datos relacionales en varias máquinas. Para ello:

Los datos se almacenan en cada instancia de base de datos de manera autónoma

El código de aplicación se desarrolla para distribuir los datos y las consultas y agregar los resultados de los datos a través de todas las instancias de bases de datos

Se debe desarrollar código adicional para gestionar los fallos sobre los recursos, para realizar joins entre diferentes bases de datos, balancear los datos y/o replicarlos, etc…

Además, muchos beneficios de las bases de datos como la integridad transaccional se ven comprometidos o incluso eliminados al emplear un escalado horizontal.

AUTO-SHARDING

Por contra, las bases de datos NoSQL normalmente soportan auto-sharding, lo que implica que de manera nativa y automáticamente se dividen los datos entre un número arbitrario de servidores, sin que la aplicación sea consciente de la composición del pool de servidores. Los datos y las consultas se balancean entre los servidores.

El particionado se realiza mediante un método consistente, como puede ser:

Por rangos de su id: por ejemplo "los usuarios del 1 al millón están en la partición 1" o "los usuarios cuyo nombre va de la A a la E" en una partición, en otra de la M a la Q, y de la R a la Z en la tercera.

Por listas: dividiendo los datos por la categoría del dato, es decir, en el caso de datos sobre libros, las novelas en una partición, las recetas de cocina en otra, etc..

Mediante una función hash, la cual devuelve un valor para un elemento que determine a que partición pertenece.

CUANDO PARTICIONAR

El motivo para particionar los datos se debe a:

limitaciones de almacenamiento: los datos no caben en un único servidor, tanto a nivel de disco como de memoria RAM.

rendimiento: al balancear la carga entre particiones las escrituras serán más rápidas que al centrarlas en un único servidor.

disponibilidad: si un servidor está ocupado, otro servidor puede devolver los datos. La carga de los servidores se reduce.

No particionaremos los datos cuando la cantidad sea pequeña, ya que el hecho de distribuir los datos conlleva unos costes que pueden no compensar con un volumen de datos insuficiente. Tampoco esperaremos a particionar cuando tengamos muchísimos datos, ya que el proceso de particionado puede provocar sobrecarga del sistema.

La nube facilita de manera considerable este escalado, mediante proveedores como Amazon Web Services el cual ofrece virtualmente una capacidad ilimitada bajo demanda, y preocupándose de todas las tareas necesarias para la administración de la base de datos.

Los desarrolladores ya no necesitamos construir plataformas complejas para nuestras aplicaciones, de modo que nos podemos centrar en escribir código de aplicación. Una granja de servidores puede ofrecer el mismo procesamiento y capacidad de almacenamiento que un único servidor de alto rendimiento por mucho menos coste.

REQUERIMIENTOS NO FUNCIONALES:

- Se debe garantizar la seguridad de la información.
- Debe permitir el acceso de varios usuarios al mismo tiempo (concurrencia)
- Se debe garantizar la fiabilidad de la información.
- Escale los datos horizontalmente.
- Fragmentación de la base de datos.
- La escritura debe hacerse desde un servidor.
- Un grupo de fragmentos debe constar de un servidor de configuración, un enrutador de consultas y shards.

COMANDOS PARA LA PARTICION:

Comando	Función
md	**Crea directorio en ruta especifica**
--port	**Indica el puerto al que se hace el llamado**
--dbpath	**Indica el directorio que contiene los datos**
--configsvr	**Especifica el servidor de configuración**
--configdb	**Asigna los enrutadores de consulta**
--shardsvr	**Crea un shard en una ruta especifica**

Creación de ficheros

```
   bindIp: 127.0.0.1
   port: 28017
sharding:
   clusterRole: configsvr
replication:
   replSetName: "replicaConfSvr"
storage:
   dbPath: "C:/Program Files/MongoDB/Server/3.4/data/cluster/cfgsvr1/"
systemLog:
   destination: file
   path: "C:/Program Files/MongoDB/Server/3.4/log/cfgsvr1.log"
   logAppend: true
```

```
   bindIp: 127.0.0.1
   port: 28117
sharding:
   clusterRole: configsvr
replication:
   replSetName: "replicaConfSvr"
storage:
   dbPath: "C:/Program Files/MongoDB/Server/3.4/data/cluster/cfgsvr2/"
systemLog:
   destination: file
   path: "C:/Program Files/MongoDB/Server/3.4/log/cfgsvr2.log"
   logAppend: true
```

```
   bindIp: 127.0.0.1
   port: 28217
sharding:
   clusterRole: configsvr
replication:
   replSetName: "replicaConfSvr"
storage:
   dbPath: "C:/Program Files/MongoDB/Server/3.4/data/cluster/cfgsvr3/"
systemLog:
   destination: file
   path: "C:/Program Files/MongoDB/Server/3.4/log/cfgsvr3.log"
   logAppend: true
```

inicialización de los servers

```
mongod --config cfgsvr1.cfg --install --serviceName
mongod --config "C:\Program Files\MongoDB\Server\3.4\etc\cfgsvr1.cfg" --install --serviceName Mongocfgsvr1
--serviceDisplayName "MongoDB Config Server 1"
mongod --config "C:\Program Files\MongoDB\Server\3.4\etc\cfgsvr2.cfg" --install --serviceName Mongocfgsvr2
--serviceDisplayName "MongoDB Config Server 2"
mongod --config "C:\Program Files\MongoDB\Server\3.4\etc\cfgsvr3.cfg" --install --serviceName Mongocfgsvr3
--serviceDisplayName "MongoDB Config Server 3"
```

Conexión a local host del server para iniciar la réplica set

```
rs.initiate(
  {
    _id: "replicaConfSvr",
    configsvr: true,
    members: [
      { _id : 0, host : "localhost:28017" },
      { _id : 1, host : "localhost:28117" },
      { _id : 2, host : "localhost:28217" }
    ]
  }
)
```

Replica set del Shard

```
   bindIp: 127.0.0.1
   port: 29017
sharding:
   clusterRole: shardsvr
replication:
   replSetName: "replicaShard1"
storage:
   dbPath: "C:/Program Files/MongoDB/Server/3.4/data/cluster/shard1pri/"
systemLog:
   destination: file
   path: "C:/Program Files/MongoDB/Server/3.4/log/shard1pri.log"
   logAppend: true
   bindIp: 127.0.0.1
   port: 29117
sharding:
   clusterRole: shardsvr
replication:
   replSetName: "replicaShard1"
storage:
   dbPath: "C:/Program Files/MongoDB/Server/3.4/data/cluster/shard1sec/"
systemLog:
   destination: file
   path: "C:/Program Files/MongoDB/Server/3.4/log/shard1sec.log"
   logAppend: true
   bindIp: 127.0.0.1
   port: 29217
sharding:
   clusterRole: shardsvr
replication:
   replSetName: "replicaShard1"
storage:
   dbPath: "C:/Program Files/MongoDB/Server/3.4/data/cluster/shard1arb/"
systemLog:
   destination: file
   path: "C:/Program Files/MongoDB/Server/3.4/log/shard1arb.log"
   logAppend: true
```

Se inician los servidores del Shard

```
mongod --config "C:\Program Files\MongoDB\Server\3.4\etc\shard1pri.cfg" --install --serviceName Mongosh1pri
--serviceDisplayName "MongoDB Shard 1 Primary"
mongod --config "C:\Program Files\MongoDB\Server\3.4\etc\shard1sec.cfg" --install --serviceName Mongosh1sec
--serviceDisplayName "MongoDB Shard 1 Secondary"
mongod --config "C:\Program Files\MongoDB\Server\3.4\etc\shard1arb.cfg" --install --serviceName Mongosh1arb
--serviceDisplayName "MongoDB Shard 1 Arbiter"
```

Segundo Shard Replica Set

```
   bindIp: 127.0.0.1
   port: 29317
sharding:
   clusterRole: shardsvr
replication:
   replSetName: "replicaShard2"
storage:
   dbPath: "C:/Program Files/MongoDB/Server/3.4/data/cluster/shard2pri/"
systemLog:
   destination: file
   path: "C:/Program Files/MongoDB/Server/3.4/log/shard2pri.log"
   logAppend: true
   bindIp: 127.0.0.1
   port: 29417
sharding:
   clusterRole: shardsvr
replication:
   replSetName: "replicaShard2"
storage:
   dbPath: "C:/Program Files/MongoDB/Server/3.4/data/cluster/shard2sec/"
systemLog:
   destination: file
   path: "C:/Program Files/MongoDB/Server/3.4/log/shard2sec.log"
   logAppend: true
   dbPath: "C:/Program Files/MongoDB/Server/3.4/data/cluster/shard2arb/"
systemLog:
   destination: file
   path: "C:/Program Files/MongoDB/Server/3.4/log/shard2arb.log"
   logAppend: true
```

Inicialización de servidores

```
mongod --config shard2pri.cfg
mongod --config shard2sec.cfg
mongod --config shard2arb.cfg
```

```
rs.initiate(
  {
    _id: "replicaShard2",
    members: [
      { _id : 0, host : "localhost:29317", priority : 20 },
      { _id : 1, host : "localhost:29417", priority : 10 },
      { _id : 2, host : "localhost:29517", arbiterOnly : true }
    ]
  }
)
```

Fichero de configuración de routers

```
   bindIp: 127.0.0.1
   port: 27017
sharding:
   configDB: replicaConfSvr/localhost:28017,localhost:28117,localhost:28217
systemLog:
   destination: file
   path: "C:/Program Files/MongoDB/Server/3.4/log/router1.log"
   logAppend: true
```

Inicializar Mongo y conectarse al router.

```
mongos --config router1.cfg
```

```
admin = db.getSiblingDB("admin")
admin.createUser(
  {
    user: "admin",
    pwd: "Password1",
    roles: [ { role: "userAdminAnyDatabase", db: "admin" }, { role: "clusterAdmin", db: "admin" }, { role:
    "readAnyDatabase", db: "admin" }, { role: "readWriteAnyDatabase", db: "admin" }, { role:
    "userAdminAnyDatabase", db: "admin" } ]
  }
)
```

Autenticación.

```
db.getSiblingDB("admin").auth("admin", "Password1" )
```

```
mongo -u "fred" -p "changeme1" --authenticationDatabase "admin"
```

Añadir fragmentos al clúster

```
sh.addShard( "replicaShard1/localhost:29017")
```

```
> sh.enableSharding ("Torneo")
```

Fragmentar una colección

```
sh.shardCollertion("torneo.arbitro",{"nombre":"Brandon"})
```

Link GITHUB: https://github.com/Brandfranco/ACTIVIDAD-5-DBS

BIBLIOGRAFIA

- Sarasa, A. (2016). Introducción a las bases de datos NoSQL usando MogoDB. Editorial UOC. https://elibro.net/es/lc/biblioibero/titulos/58524

- Medrano, A. (s. f.). NoSQL. Recuperado 16 de octubre de 2022, de http://expertojava.ua.es/si/nosql/nosql01.html

- ¿Qué es el particionamiento de bases de datos? (s. f.). Microsoft Azure. Recuperado 16 de octubre de 2022, de https://azure.microsoft.com/es-es/resources/cloud-computing-dictionary/what-is-database-sharding/