**109. 13. Ejercicios – Sharding**

**110. Ejercicio 1**

En este ejercicio vamos a necesitar el fichero **week6\_\_hw6.1\_m102\_52b491d5e2d4237593ca1d3a.js** proporcionado.

Vamos a comenzar trabajando con un servidor **MongoDB** en modo **Standalone**, para convertirlo posteriormente en un cluster con dos **Shards**, y luego haremos **Sharding** de una de nuestras colecciones.

Para ello vamos a crear un entorno **dev**, sin **Replica Sets** ni **Config Servers**.

Empezamos levantando una instancia de **mongod**, y posteriormenta nos conectamos con **mongo** shell cargando el script mencionado arriba, a la base de datos **week6**:

mongo --shell localhost/week6 week6\_\_hw6.1\_m102\_52b491d5e2d4237593ca1d3a.js

mongo --shell Hutz3/week6 week6\_\_hw6.1\_m102\_52b491d5e2d4237593ca1d3a.js

Ejecutar **homework.init()**, esto insertara unos cuantos documentos en nuestra base de datos. homework.init()

Ahora, ejecutamos **db.trades.stats()** para confirmar que todo se ha realizado correctamente db.trades.stats()

A estas alturas, disponemos de un unico proceso **mongod**, que vamos a convertir en un **Sharded Cluster**, sin perder los datos que tenemos en **week6.trades**.

Paramos el proceso y volvemos a levantarlo con la opcion **--shardsvr**. Al hacer esto, el puerto por defecto de **mongod** se convierte en el **27018** mongod --shardsvr --dbpath ...

Ahora, vamos a levantar un **Config Server**. Al usar **--configdb** nuestro puerto por defecto se convierte en el **27019**, y el directorio de datos **/data/configdb** mongod --configsvr ...

Vamos a levantar ahora un **mongos**, diciendole donde debe buscar el **Config Server**. mongos –configdb nombrehost:27019

Con todo levantado, vamos a conectarnos al proceso **mongos** mediante la shell **mongo**, cargando el script proporcionado.

mongo --shell localhost/week6 week6\_\_hw6.1\_m102\_52b491d5e2d4237593ca1d3a.js

Vamos a anadir el primer **Shard** a nuestro cluster:

sh.addShard("nombrehost:27018")

Verificamos que podemos acceder correctamente a los datos:

db.trades.find().pretty()

db.trades.count()

db.trades.stats()

Para confirmar que todo ha ido sin problemas, ejecutar **homework.a()**. homework.a() .Que devuelve dicho metodo?

**RESULTADO**: 1000001

**111. Ejercicio 2**

Vamos a activar ahora el **Sharding** para la base de datos **week6**. Si no te acuerdas como hacerlo,

recuerda que puedes consultar la ayuda con **sh.help()**.

Ahora, vamos a **Shard** la coleccion **trades**, sobre una **ShardKey** que queremos que sea compuesta,

usando los campos **ticker** y **time**. Para ello antes tendremos que crear el indice sobre estos campos.

db.trades.createIndex({ticker:1, time:1})

Con el indice creado, ya puedes hacer **Shard** de la coleccion. Una vez hecho, echa un vistazo a los

**Chunks** creados:

use config

db.chunks.find({},{min:1,max:1,shard:1,\_id:0,ns:1})

Ejecuta **homework.b()**. .Que valor devuelve?

**RESULTADO**: 3

**112. Ejercicio 3**

Vamos a anadir ahora un nuevo **Shard**. Ejecuta una nueva instancia **mongod** con la opcion

**--shardsvr** sobre un puerto no usado.

Ahora, desde el **mongos**, anade el **Shard** al cluster (recuerda, si no sabes como hacerlo, consulta

**sh.help()**)

Puedes confirmar que todo esta funcionando correctamente ejecutando:

homework.check1()

Ahora espera a que el **Balancer** distribuya los datos entre los **Shards**. Al cabo de un rato, los **Chunks** deberian haberse distribuido de manera correcta entre los **Shards**.

Podemos ver en que estado estan en todo momento mediante:

db.chunks.find( { ns:"week6.trades" }, {min:1,max:1,shard:1,\_id:0} ).sort({min:1})

O tambien con:

db.chunks.aggregate( [

{ $match : { ns : "week6.trades" } } ,

{ $group : { \_id : "$shard", n : { $sum : 1 } } }

] )

Cuando tengamos la certeza de que ya hemos acabado de redistribuir los **Chunks**, ejecutemos

**homework.c()**. .Que resultado obtenemos? **RESULTADO**: 2