



## **Actividad 3 - Conceptos y Comandos básicos del particionamiento en bases de datos NoSQL**

Juan Sebastián Vacca Peña

ID 100098569

Corporación Universitaria Iberoamericana

Bases de Datos Avanzadas

13 diciembre de 2023

## Contenido

Requerimientos funcionales.....	2
Escalabilidad.....	2
Rendimiento.....	2
Disponibilidad.....	2
Seguridad .....	2
Respaldo.....	2
Comandos para realizar particionamiento.....	3
Github .....	8

## **Requerimientos funcionales**

### **Escalabilidad**

La base de datos debe ser capaz de escalar horizontalmente, aumentando su tamaño en función al aumento de usuarios, sin generar interrupciones ni conflictos al momento de generar consultas.

### **Rendimiento**

La base de datos deberá mantener un rendimiento eficiente manejando tráfico de diferentes magnitudes.

### **Disponibilidad**

El sistema debe estar en funcionamiento continuo durante todo el evento, sin interrupciones, y estar disponible en todo momento para los usuarios.

### **Seguridad**

La base de datos debe ser segura y confiable, para conservar temas de confidencialidad dentro del campeonato.

### **Respaldo**

La base de datos tiene respaldo de información al ser replicada y alojada en nodos secundarios que garantizan la información de la base de datos.

## Comandos para realizar particionamiento

1. Levantamiento del cluster: `cluster=new ShardingTest ({shards: 3, chunksize:1})`

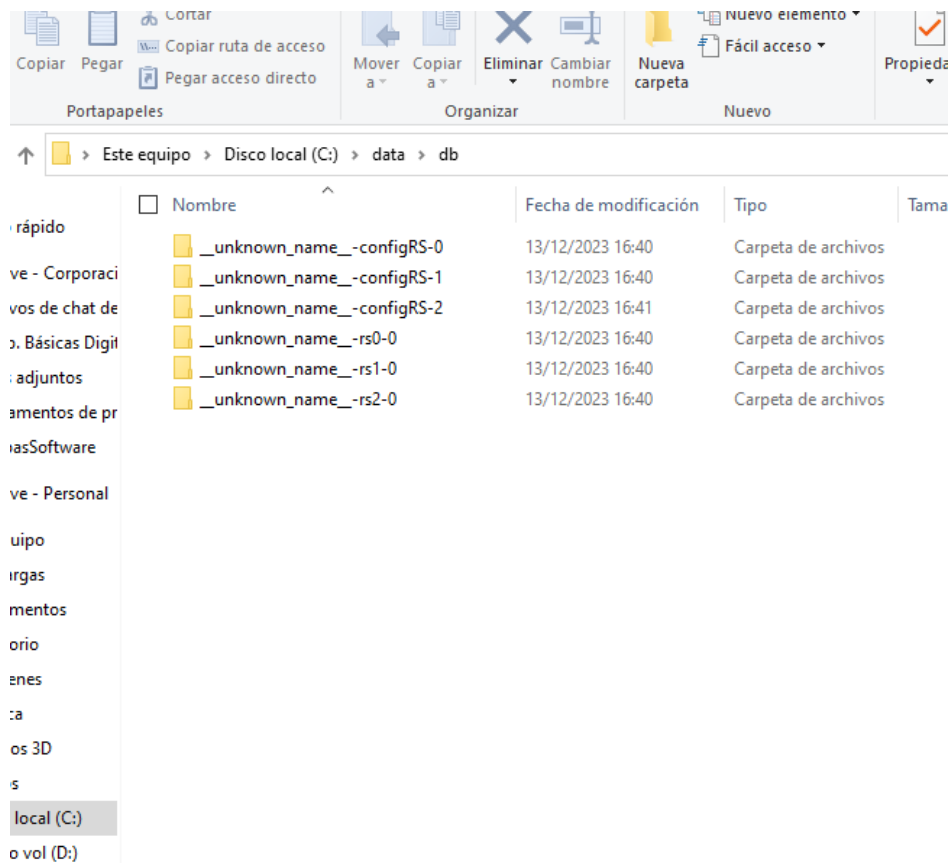
```

    return _callIsMaster() == false;
  }, msg, timeout);
},
  "getSecondaries" : function(timeout) {
    var master = this.getPrimary(timeout);
    var secs = [];
    for (var i = 0; i < this.nodes.length; i++) {
      if (this.nodes[i] != master) {
        secs.push(this.nodes[i]);
      }
    }
    return secs;
  },
  "getSecondary" : function(timeout) {
    return this.getSecondaries(timeout)[0];
  },
  "getArbiters" : function() {
    let arbiters = [];
    for (let i = 0; i < this.nodes.length; i++) {
      const node = this.nodes[i];

      let isArbiter = false;

      assert.retryNoExcept(() => {
        isArbiter = isNodeArbiter(node);
        return true;
      }, `Could not call 'isMaster' on ${node}.`, 3, 1000);
    }
  }
}

```



2. Creación el balanceador: `db = (new Mongo("localhost:20006")).getDB("Futbolact_1")`

```

mongo shell
MongoDB shell version v4.4.25
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("4526743c-882d-44ad-b113-787902949c0b") }
MongoDB server version: 4.4.25
---
The server generated these startup warnings when booting:
  2023-12-11T13:47:27.871-05:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
> db = (new Mongo("localhost:20006")).getDB("Futbolact_1")
uncaught exception: SyntaxError: missing ) after argument list :
@(shell):1:42
> db = (new Mongo("localhost:20006")).getDB("Futbolact_1")
Futbolact_1
mongos>

```

3. Ingresamos registros a las tablas que se alojan en la base de datos

```

mongo shell
MongoDB shell version v4.4.25
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("4526743c-882d-44ad-b113-787902949c0b") }
MongoDB server version: 4.4.25
---
The server generated these startup warnings when booting:
  2023-12-11T13:47:27.871-05:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
> db = (new Mongo("localhost:20006")).getDB("Futbolact_1")
uncaught exception: SyntaxError: missing ) after argument list :
@(shell):1:42
> db = (new Mongo("localhost:20006")).getDB("Futbolact_1")
Futbolact_1
mongos> for(i=0;i<50000;i++){
...   db.arbitros.insert({idarbitro:"idarbitro" +i, nombreArbitro:"Recuento de arbitros"
...   +i, date: new Date()});
... }

```

*1 inserción de datos en la tabla árbitros*

```

mongos> for(i=0;i<25000;i++){
...   db.equipos.insert({name:"name" +i, numjugadores:"numjugadores"
...   +i, date: new Date()});
... }

```

*2 inserción de datos en tabla equipos*

```
WriteResult({ "nInserted" : 1 })
mongo> for(i=0;i<20000;i++){
...   db.goles.insert({equipo:"equipo" +i, goles:"Recuento de goles"
...   +i, date: new Date()});
... }
```

*3 inserción de datos en tabla goles*

4. En una nueva ventana de mongo Shell nos conectamos con el shard en el puerto 20000.

```
mongo shell
MongoDB shell version v4.4.25
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("79444805-ca9c-4fff-bc89-12e5e4157ebd") }
MongoDB server version: 4.4.25
---
The server generated these startup warnings when booting:
  2023-12-11T13:47:27.871-05:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
> shar1 = new Mongo("localhost:20000")
connection to localhost:20000
> shard1DB = shard1.getDB("Futbolact_1")
uncaught exception: ReferenceError: shard1 is not defined :
@(shell):1:1
> shard1 = new Mongo("localhost:20000")
connection to localhost:20000
>
```

Luego nos conectamos con la base de datos.

```
> shard1DB = shard1.getDB("Futbolact_1")
Futbolact_1
>
```

5. Verificamos la inserción de los registros en la tabla y luego de eso nos conectamos con los otros shards y repetimos el proceso.

```
> shard1DB.arbitros.count()
50000
>
```

```
> shard2 = new Mongo("localhost:20001")
connection to localhost:20001
>
```

```
> shard2 = new Mongo("localhost:20001")
connection to localhost:20001
> shard2DB = shard2.getDB("Futbolact_1")
Futbolact_1
> shard2DB.arbitros.count()
0
>
```

## 6. Activamos el sharding en la base de datos

```

mongos> sh.enableSharding("Futbolact_1")
{
  "ok" : 1,
  "operationTime" : Timestamp(1702507751, 3),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1702507751, 3),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
mongos>

```

## 7. Creamos un índice que se usara como shard

```

mongo shell
mongos> db.arbitros.ensureIndex({idarbitro : 1})
{
  "raw" : {
    "__unknown_name__-rs0/DESKTOP-IT871UI:20000" : {
      "createdCollectionAutomatically" : false,
      "numIndexesBefore" : 1,
      "numIndexesAfter" : 2,
      "commitQuorum" : "votingMembers",
      "ok" : 1
    }
  },
  "ok" : 1,
  "operationTime" : Timestamp(1702508019, 6),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1702508019, 6),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
mongos>

```

luego activamos la colección que está alojada dentro de la base de datos “Futbolact\_1”

```

mongos> sh.shardCollection("Futbolact_1.arbitros", {idarbitro: 1})
{
  "collectionsharded" : "Futbolact_1.arbitros",
  "collectionUUID" : UUID("8f02ed3f-f4b9-4032-9d35-4bb745cece3d"),
  "ok" : 1,
  "operationTime" : Timestamp(1702508335, 2),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1702508335, 2),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
mongos>

```

Colocamos el balanceador de carga a funcionar

```

mongo shell
mongos> sh.getBalancerState()
false
mongos> sh.setBalancerState(true)
{
  "ok" : 1,
  "operationTime" : Timestamp(1702508599, 3),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1702508599, 3),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
mongos>

```

8. Confirmamos el status del balanceador

```

active mongoses:
  "4.4.25" : 1
autosplit:
  Currently enabled: yes
balancer:
  Currently enabled: yes
  Currently running: no
  Failed balancer rounds in last 5 attempts: 0
  Migration Results for the last 24 hours:
    No recent migrations
databases:
  { "_id" : "Futbolact_1", "primary" : "__unknown_name__-rs0", "partitioned" : true, "version" : { "u
id" : UUID("80bedce9-97eb-4957-9c06-878634c0e3b1"), "lastMod" : 1 } }
    Futbolact_1.arbitros
      shard key: { "idarbitro" : 1 }
      unique: false
      balancing: true
      chunks:
        __unknown_name__-rs0    1
        { "idarbitro" : { "$minKey" : 1 } } --> { "idarbitro" : { "$maxKey" : 1 } } on : __unkn
own_name__-rs0 Timestamp(1, 0)
        { "_id" : "config", "primary" : "config", "partitioned" : true }
mongos>

```



9. Verificamos los shards, que están activos

```

false
mongos> db.adminCommand({listshards: 1})
{
  "shards" : [
    {
      "_id" : "__unknown_name__-rs0",
      "host" : "__unknown_name__-rs0/DESKTOP-IT871UI:20000",
      "state" : 1
    },
    {
      "_id" : "__unknown_name__-rs1",
      "host" : "__unknown_name__-rs1/DESKTOP-IT871UI:20001",
      "state" : 1
    },
    {
      "_id" : "__unknown_name__-rs2",
      "host" : "__unknown_name__-rs2/DESKTOP-IT871UI:20002",
      "state" : 1
    }
  ],
  "ok" : 1,
  "operationTime" : Timestamp(1702509021, 3),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1702509021, 3),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
mongos>

```

## Github

<https://github.com/SebastianVacca/BD>

