

**Actividad 3 – Conceptos y comandos básicos del particionamiento en bases de datos**

**NoSQL**

Carlos Gabriel Usama Cortez

100107522

cusamaco@ibero.edu.co

Docente: William Ruiz

Universidad Iberoamericana

Ingeniería De Software

Bases de Datos Avanzadas

02/06/2024

## TABLA DE CONTENIDO

<b>Documento de Requerimientos No Funcionales .....</b>	<b>4</b>
1. Introducción .....	4
2. Redundancia .....	4
• Definición.....	4
• Implementación en MongoDB .....	4
3. Disponibilidad 24x7 .....	4
• Definición.....	5
• Implementación en MongoDB:.....	5
▪ Sharding .....	5
4. Ejemplo de Particionamiento (Sharding) .....	5
• Shard Collection.....	6
• Key .....	6
5. Monitorización y Alertas .....	6
• Monitorización .....	6
• Alertas .....	6
6. Recuperación ante Desastres .....	6
• Backups Regulares .....	6
• Planes de Recuperación.....	6

Conclusión.....	7
-----------------	---

## Documento de Requerimientos No Funcionales

### 1. Introducción

Este documento describe los requerimientos no funcionales de calidad para la base de datos MongoDB diseñada para gestionar un torneo de fútbol profesional. Se enfoca en la redundancia y disponibilidad 24x7 para asegurar que el sistema funcione de manera continua y eficiente.

### 2. Redundancia

- **Definición:** La redundancia se refiere a la duplicación de componentes críticos del sistema para asegurar la continuidad del servicio en caso de fallos.
- **Implementación en MongoDB:** Se utilizará un clúster de réplicas (replica set) para asegurar la redundancia de la base de datos.
  - **Replica Set:** Se configurará un replica set con al menos tres nodos:
    - **Nodo Primario:** Maneja todas las operaciones de escritura y la mayoría de las lecturas.
    - **Nodos Secundarios:** Mantienen copias exactas del nodo primario y pueden tomar el control en caso de que el nodo primario falle.
  - **Electores de Árbitros:** Se configurarán para asegurar que los nodos secundarios puedan votar y elegir un nuevo nodo primario automáticamente en caso de fallo del nodo primario.

### 3. Disponibilidad 24x7

- **Definición:** La disponibilidad 24x7 garantiza que el sistema esté operativo y accesible en todo momento, sin interrupciones.
- **Implementación en MongoDB:**
  - **Sharding:** Se implementará el particionamiento de datos (sharding) para distribuir la carga entre varios servidores, mejorando la disponibilidad y escalabilidad.
    - **Shard Keys:** Se seleccionará una clave de particionamiento adecuada (por ejemplo, el ID de los encuentros) para distribuir los datos equitativamente.
    - **Config Servers:** Tres servidores de configuración (config servers) se usarán para almacenar los metadatos del cluster.
    - **Query Routers:** Los mongos se utilizarán como enrutadores de consultas para dirigir las operaciones a los shards correspondientes.
  - **Balanceo de Carga:** Se implementarán técnicas de balanceo de carga para asegurar que ninguna parte del sistema se sobrecargue.
  - **Mantenimiento Programado:** Se planificará el mantenimiento en horarios de menor actividad y se utilizarán nodos secundarios para evitar la interrupción del servicio.

#### 4. Ejemplo de Particionamiento (Sharding)

Para ilustrar el particionamiento en MongoDB, se presenta un ejemplo de cómo se configuraría el sharding para la colección "encuentros".

- **Shard Collection:** Se especifica la colección "encuentros" dentro de la base de datos "torneoFutbol".
- **Key:** Se selecciona la clave de particionamiento "id\_encuentro" con un índice hash para distribuir los datos uniformemente.

## 5. Monitorización y Alertas

- **Monitorización:** Se implementarán herramientas de monitorización (como MongoDB Cloud Manager o Prometheus) para vigilar el rendimiento, la disponibilidad y el estado de los nodos del clúster.
- **Alertas:** Se configurarán alertas para notificar a los administradores del sistema sobre cualquier problema potencial, como la caída de nodos o la degradación del rendimiento.

## 6. Recuperación ante Desastres

- **Backups Regulares:** Se realizarán copias de seguridad regulares de la base de datos y se almacenarán en ubicaciones geográficamente distribuidas.
- **Planes de Recuperación:** Se establecerán planes detallados de recuperación ante desastres para restaurar el sistema rápidamente en caso de fallo catastrófico.

## Conclusión

Este documento especifica los criterios de calidad en cuanto a redundancia y disponibilidad 24x7 para la base de datos MongoDB del torneo de fútbol profesional. La implementación de un replica set y el sharding asegurarán la continuidad del servicio y la capacidad de manejar grandes volúmenes de datos de manera eficiente.

## Particionamiento en MongoDB

El particionamiento en MongoDB se realiza con el objetivo de mejorar varios factores clave como la disponibilidad, la tolerancia a fallos y el rendimiento. Este proceso se conoce como sharding, el cual implica la distribución de datos a través de un conjunto de instancias para optimizar el manejo de grandes volúmenes de información.

## Configuración del Cluster de Sharding

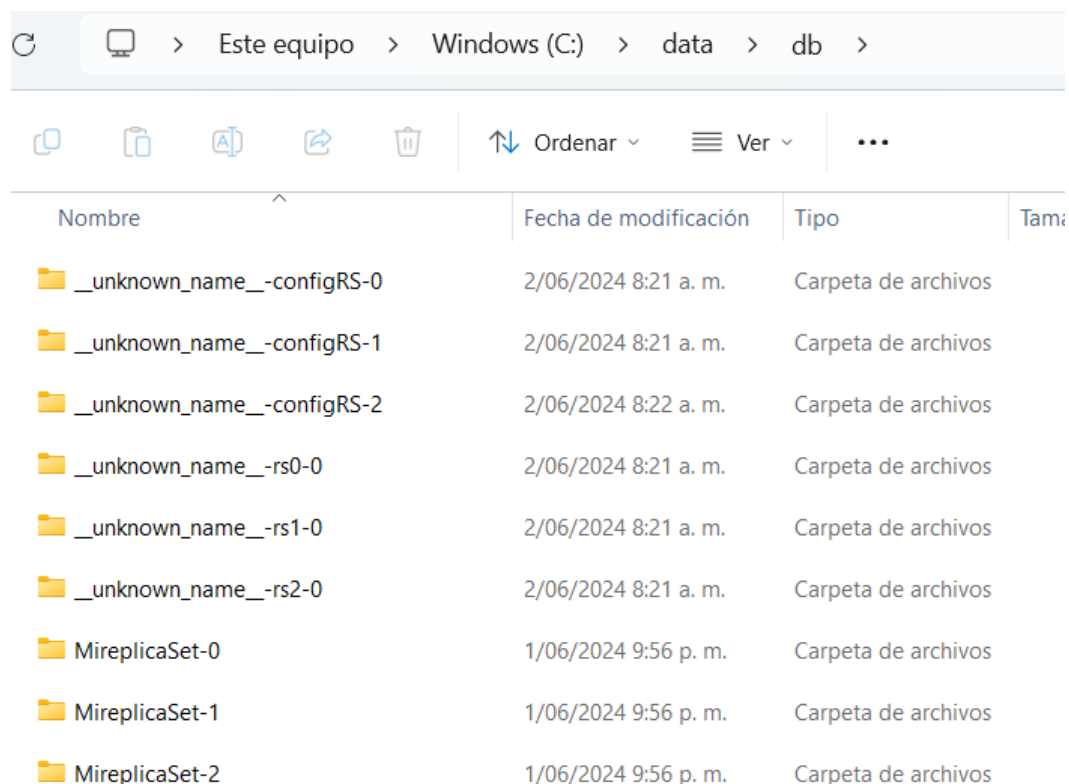
Para configurar un clúster de sharding, utilizaremos varias consolas. La primera consola se denominará "Base".

## Iniciar el Cluster de Réplica

Para levantar el clúster de réplica, desde la consola, ejecutaremos el siguiente comando:

Este comando crea un clúster de sharding con 3 nodos (shards).

Si nos dirigimos a la carpeta `data/db` en la unidad C, encontraremos las carpetas que representan las bases de datos particionadas.



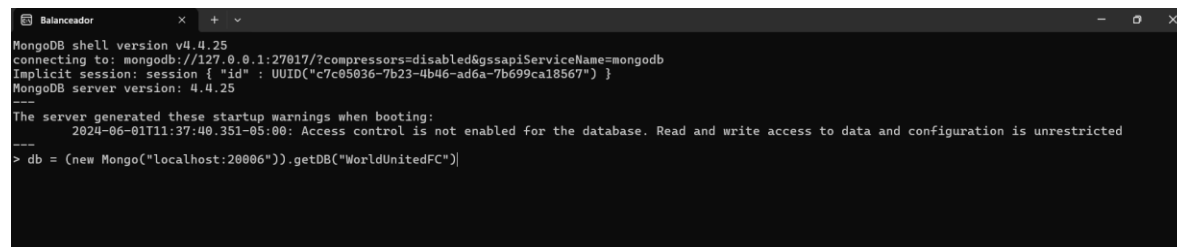
Nombre	Fecha de modificación	Tipo	Tamaño
__unknown_name__-configRS-0	2/06/2024 8:21 a. m.	Carpeta de archivos	
__unknown_name__-configRS-1	2/06/2024 8:21 a. m.	Carpeta de archivos	
__unknown_name__-configRS-2	2/06/2024 8:22 a. m.	Carpeta de archivos	
__unknown_name__-rs0-0	2/06/2024 8:21 a. m.	Carpeta de archivos	
__unknown_name__-rs1-0	2/06/2024 8:21 a. m.	Carpeta de archivos	
__unknown_name__-rs2-0	2/06/2024 8:21 a. m.	Carpeta de archivos	
MireplicaSet-0	1/06/2024 9:56 p. m.	Carpeta de archivos	
MireplicaSet-1	1/06/2024 9:56 p. m.	Carpeta de archivos	
MireplicaSet-2	1/06/2024 9:56 p. m.	Carpeta de archivos	

## Inserción de Datos en el Balanceador

Para actuar sobre el conjunto de sharding, que define a dónde van los datos, abriremos una nueva consola llamada "Balanceador".

```
db = (new Mongo("localhost:20006")).getDB("WorldUnitedFC")
```

Esto nos conecta al balanceador de MongoDB en el puerto 20006 y nos dirigimos a la base de datos "WorldUnitedFC".



```

MongoDB shell version v4.4.25
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("c7c05036-7b23-4b46-ad6a-7b699ca18567") }
MongoDB server version: 4.4.25
---
The server generated these startup warnings when booting:
  2024-06-01T11:37:40.351-05:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
> db = (new Mongo("localhost:20006")).getDB("WorldUnitedFC")

```

Conexión



```
MongoDB shell version v4.4.25
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("c7c05036-7b23-4b46-ad6a-7b699ca18567") }
MongoDB server version: 4.4.25
-----
The server generated these startup warnings when booting:
  2024-06-01T11:37:40.351-05:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
-----
> db = (new Mongo("localhost:20006")).getDB("WorldUnitedFC")
WorldUnitedFC
mongo>
```

## Inserción de Datos en la Colección Árbitros

Insertamos 500,000 documentos en la colección arbitros:

```
for (i = 0; i < 500000; i++) {db.arbitros.insert({nombre: "Árbitro" + i,edad: 45,
categoría:"Internacional"});}

```

[illegible]

## Culminación



```

> shard1 = new Mongo("localhost:20000")
connection to localhost:20000
> shard1DB = shard1.getDB("WorldUnitedFC")
WorldUnitedFC
> shard1DB.arbitros.count()
500000
> shard2 = new Mongo("localhost:20001")
connection to localhost:20001
> shard2DB = shard2.getDB("WorldUnitedFC")
WorldUnitedFC
> shard2DB.arbitros.count()
0
> shard3 = new Mongo("localhost:20002")
connection to localhost:20002
> shard3DB = shard3.getDB("WorldUnitedFC")
WorldUnitedFC
> shard3DB.arbitros.count()
0
>

Shard1
-Conexión al puerto 20000
shard1 = new Mongo("localhost:20000")
-Conexión a la base de datos WorldUnitedFC
shard1DB = shard1.getDB("WorldUnitedFC")
-verificación y conteo de documentos en arbitros
shard1DB.arbitros.count() // Debería retornar 500000

Shard2
-Conexión al puerto 20001
shard2 = new Mongo("localhost:20001")
-Conexión a la base de datos WorldUnitedFC
shard2DB = shard2.getDB("WorldUnitedFC")
-verificación y conteo de documentos en arbitros
shard2DB.arbitros.count() // Debería retornar el conteo de documentos en Shard2

Shard3
-Conexión al puerto 20002
shard3 = new Mongo("localhost:20002")
-Conexión a la base de datos WorldUnitedFC
shard3DB = shard3.getDB("WorldUnitedFC")
-verificación y conteo de documentos en arbitros
shard3DB.arbitros.count() // Debería retornar el conteo de documentos en Shard3

```

Al verificar, encontramos que todos los datos del particionamiento están en el shard 1, lo cual indica un desequilibrio en la distribución de los datos. Para solucionarlo, procederemos a activar el sharding.

Para esto, nos dirigimos a la segunda consola, previamente denominada "Balanceador", y ejecutamos el siguiente comando:

```

Balanceador
mongos> shard1 = new Mongo("localhost:20006")

```

```

Balanceador
mongos> shard1 = new Mongo("localhost:20006")
connection to localhost:20006
mongos>

```

Y comprobamos el estado de cada una de las instancias

sh.status()

```

Balanceador
mongos> shard1 = new Mongo("localhost:20006")
connection to localhost:20006
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
    "_id" : 1,
    "minCompatibleVersion" : 5,
    "currentVersion" : 6,
    "clusterId" : ObjectId("665c7177fa715ea638da44b6")
  }
  shards:
    { "_id" : "__unknown_name__-rs0", "host" : "__unknown_name__-rs0/Gabriel:20000", "state" : 1 }
    { "_id" : "__unknown_name__-rs1", "host" : "__unknown_name__-rs1/Gabriel:20001", "state" : 1 }
    { "_id" : "__unknown_name__-rs2", "host" : "__unknown_name__-rs2/Gabriel:20002", "state" : 1 }
  active mongoses:
    "4.4.25" : 1
  autosplit:
    Currently enabled: no
  balancer:
    Currently enabled: no
    Currently running: no
    Failed balancer rounds in last 5 attempts: 0
    Migration Results for the last 24 hours:
      No recent migrations
  databases:
    { "_id" : "WorldUnitedFC", "primary" : "__unknown_name__-rs0", "partitioned" : false, "version" : { "uuid" : UUID("9548ca35-27d6-471a-b2c0-3c90da41121f"), "lastMod" : 1 } }
    { "_id" : "config", "primary" : "config", "partitioned" : true }
mongos>

```

Nos

muestra la versión de sharding, junto con los identificadores de los shards rs0, rs1 y rs2, cada uno conectado a los puertos 20000, 20001 y 20002, respectivamente.

Además, indica que el balanceador no está activo ni en ejecución.

Para activar el sharding, utilizamos la siguiente secuencia de comandos, especificando la base de datos en la que se realizará el proceso:

```
sh.enableSharding("WorldUnitedFC")
```

y luego sobre la colección

```
db.arbitros.ensureIndex({arbitros: 1})
```

ahora procedemos a determinar la colección de acuerdo con árbitros

```
sh.shardCollection("WorldUnitedFC.arbitros", { nombre: 1 })
```

```

Balanceador
x + v
    "commitQuorum" : "votingMembers",
    "ok" : 1
  }
}
"ok" : 1,
"operationTime" : Timestamp(1717348542, 3),
"$clusterTime" : {
  "clusterTime" : Timestamp(1717348542, 3),
  "signature" : {
    "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
    "keyId" : NumberLong(0)
  }
}
}
}
mongos> sh.shardCollection("WorldUnitedFC.arbitros", { nombre: 1 })
{
  "ok" : 0,
  "errmsg" : "Please create an index that starts with the proposed shard key before sharding the collection",
  "code" : 72,
  "codeName" : "InvalidOptions",
  "operationTime" : Timestamp(1717348556, 4),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1717348556, 4),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
}
}
mongos> |

```

Ahora ya comprobamos que la base de datos ya está particionada

```

Balanceador
x + v
---
> db = (new Mongo("localhost:20006")).getDB("WorldUnitedFC")
WorldUnitedFC
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
    "_id" : 1,
    "minCompatibleVersion" : 5,
    "currentVersion" : 6,
    "clusterId" : ObjectId("665c7177fa715ea638da44b6")
  }
  shards:
    { "_id" : "__unknown_name__-rs0", "host" : "__unknown_name__-rs0/Gabriel:20000", "state" : 1 }
    { "_id" : "__unknown_name__-rs1", "host" : "__unknown_name__-rs1/Gabriel:20001", "state" : 1 }
    { "_id" : "__unknown_name__-rs2", "host" : "__unknown_name__-rs2/Gabriel:20002", "state" : 1 }
  active mongoses:
    "4.4.25" : 1
  autosplit:
    Currently enabled: no
  balancer:
    Currently enabled: no
    Currently running: no
    Failed balancer rounds in last 5 attempts: 0
    Migration Results for the last 24 hours:
      No recent migrations
  databases:
    { "_id" : "WorldUnitedFC", "primary" : "__unknown_name__-rs0", "partitioned" : true, "version" : { "uuid" : UUID("9548ca35-27d6-471a-b2c0-3c90da41121f"), "lastMod" : 1 } }
    { "_id" : "config", "primary" : "config", "partitioned" : true }
mongos> |

```

insertamos el comando para comprobar cuál es el estado del balanceador

```
sh.getBalancerState()
```

Procedemos a establecer el estado del balanceador en verdadero mediante el siguiente comando

```
sh.setBalancerState(true)
```

y nuevamente preguntamos cual es el estado

```
sh.getBalancerState()
```

y colocamos a correr el balanceador

```
sh.isBalancerRunning()
```

Luego procedemos a verificar como quedo el particionamiento de los datos

conectándonos a shard 1

```
shard1 = new Mongo("localhost:20000")
```

y conocer el estado de los nodos activos

```
mongos> db.adminCommand( { listShards: 1 } )
{
  "shards" : [
    {
      "_id" : "__unknown_name__-rs0",
      "host" : "__unknown_name__-rs0/Gabriel:20000",
      "state" : 1
    },
    {
      "_id" : "__unknown_name__-rs1",
      "host" : "__unknown_name__-rs1/Gabriel:20001",
      "state" : 1
    },
    {
      "_id" : "__unknown_name__-rs2",
      "host" : "__unknown_name__-rs2/Gabriel:20002",
      "state" : 1
    }
  ],
  "ok" : 1,
  "operationTime" : Timestamp(1717351608, 2),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1717351608, 2),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
mongos> |
```

