

Conceptos y Comandos básicos del particionamiento en bases de datos NoSQL

Adriana L. Giraldo B.

Héctor F. Muñoz M.

Luis E. Salcedo A.

Corporación Universitaria Iberoamericana

Facultad de Ingeniería

Ingeniería de Software

Base de Datos Avanzada

William R. Martínez

Bogotá, 10 de abril de 2023

Conceptos y Comandos básicos del particionamiento en bases de datos NoSQL

Enlace para el video: <https://youtu.be/7sCmJgTfQZM>

Enlace para el GitHub:

1. Requerimientos no funcionales.

El proyecto elaborado no solo requiere tener acceso a la base de datos con el nombre de Evento Deportivo las 24 horas los 7 días de la semana, adicionalmente se necesita escalar el almacenamiento de los datos que se manejan en nuestra base de datos, de esta manera realizaremos una distribución equitativa de la información para no recargar uno de los nodos, para este objetivo se tendrán en cuenta los siguientes requerimientos no funcionales:

- ✓ Como requerimiento principal es particionar la información de la base de datos **Evento Deportivo**.
- ✓ Se debe realizar la partición de las colecciones en tres shard's, en los cuales se almacenará la información de forma equitativa.
- ✓ Cada uno de los Shard's almacenara un porcentaje de información de las colecciones Deportistas, Jugadores y Equipos.
- ✓ Cada uno de los Shard's estará representado con numero de puerto asi:
 - “DESKTOP-KEHD2QP:20000”
 - “DESKTOP-KEHD2QP:20001”
 - “DESKTOP-KEHD2QP:20002”
- ✓ El sistema debe garantizar el ingreso de información en las colecciones mencionadas.

2. Particionamiento de la base de datos Evento Deportivo.

El particionamiento de la base de datos Evento Deportivo está configurada en cada instancia de mongoDB así:

Shard “DESKTOP-KEHD2QP:20001”

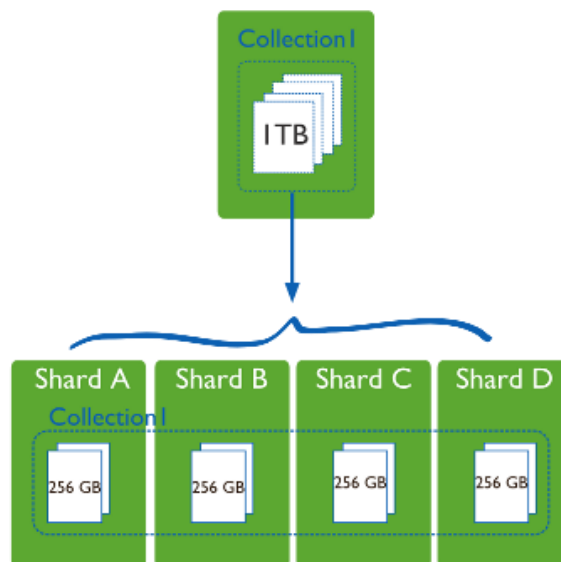
- Inicialmente almacenara toda la información ingresada en cada una de las colecciones.
- Permitirá mostrar la cantidad de información almacenada.

Shard “DESKTOP-KEHD2QP:20000”

Shard “DESKTOP-KEHD2QP:20002”

- Inicialmente no almacenarán información.
- Permitirán el particionamiento de la información almacenada en las colecciones de forma equitativa entre los tres Shard.

Diagrama de Particionamiento:



2.1. Scripts.

>cluster=new ShardingTest ({shards: 3, chunksize:1})







Este comando nos permite crear 3 instancias en las cuales se realizará la partición de la carga de información.

```
> cluster=new ShardingTest ({shards: 3, chunksize:1})
Starting new replica set __unknown_name__-rs0
ReplSetTest starting set
ReplSetTest n is : 0
{
  "useHostName" : true,
  "oplogSize" : 16,
  "keyFile" : undefined,
  "port" : 20000,
  "replSet" : "__unknown_name__-rs0",
  "dbpath" : "$set-$node",
  "useHostname" : true,
  "shardsvr" : "",
  "pathOpts" : {
    "testName" : "__unknown_name__",
    "shard" : 0,
    "node" : 0,
    "set" : "__unknown_name__-rs0"
  },
  "setParameter" : {
```

El sistema nos muestra los tres Shrad's habilitados para el ejercicio

```
61244771, 1), C: 1 }
s20006| 2023-04-11T15:26:16.513-0500 D1 NETWORK [shard-registry-reload] Started targeter for __unknown_name__-rs0/DESKTOP-KEHD2QP:20000
s20006| 2023-04-11T15:26:16.513-0500 D1 NETWORK [shard-registry-reload] Started targeter for __unknown_name__-rs1/DESKTOP-KEHD2QP:20001
s20006| 2023-04-11T15:26:16.513-0500 D1 NETWORK [shard-registry-reload] Started targeter for __unknown_name__-rs2/DESKTOP-KEHD2QP:20002
s20006| 2023-04-11T15:26:16.849-0500 D1 NETWORK [ReplicaSetMonitor-TaskExecutor] Refreshing replica set __unknown_name__-configRS took 1ms
s20006| 2023-04-11T15:26:16.964-0500 D1 TRACKING [replSetDistLockPinger] Cmd: NotSet, TrackingId: 6435c268200749423c25d744
s20006| 2023-04-11T15:26:17.179-0500 D1 TRACKING [UserCacheInvalidator] Cmd: NotSet, TrackingId: 6435c269200749423c25d746
s20006| 2023-04-11T15:26:17.302-0500 D1 TRACKING [Uptime-reporter] Cmd: NotSet, TrackingId: 6435c269200749423c25d748
s20006| 2023-04-11T15:26:17.875-0500 D1 NETWORK [ReplicaSetMonitor-TaskExecutor] Refreshing replica set __unknown_name__-rs0 took 0ms
d20001| 2023-04-11T15:26:17.884-0500 I CONNPPOOL [ShardRegistry] Ending idle connection to host DESKTOP-KEHD2QP:20005 because the pool meets co
```

En la carpeta data y la subcarpeta db encontraremos las carpetas creadas anteriormente:

 __unknown_name__-configRS-0	11/04/2023 3:26 p. m.	Carpeta de archivos
 __unknown_name__-configRS-1	11/04/2023 3:26 p. m.	Carpeta de archivos
 __unknown_name__-configRS-2	11/04/2023 3:26 p. m.	Carpeta de archivos
 __unknown_name__-rs0-0	11/04/2023 3:26 p. m.	Carpeta de archivos
 __unknown_name__-rs1-0	11/04/2023 3:26 p. m.	Carpeta de archivos
 __unknown_name__-rs2-0	11/04/2023 3:26 p. m.	Carpeta de archivos

```
> db = (new Mongo("DESKTOP-KEHD2QP:20006")).getDB("eventoDeportivo")
```

- En una consola diferente a la cual la tomaremos como el balanceador, en esta consola ejecutamos el comando anterior, este nos conecta a un puerto especial para manejar el balanceador y nos conecta con la base de datos **EventoDeportivo**.

```
> db = (new Mongo("DESKTOP-KEHD2QP:20006")).getDB("eventoDeportivo")
eventoDeportivo
mongos> for (i= 0; i < 80000; i++) {
...   db.Deportistas.insert({author : "author" +i, post_title : "Futbolistas de alto rendimiento "
...   +i, date: new Date() });
... }
WriteResult({ "nInserted" : 1 })
mongos> _
```

- En la colección **Deportistas**, ingresamos 80.000 registros, el sistema nos muestra que se realizaron los ingresos.

```
mongos> for (i= 0; i < 80000; i++) {
...   db.Deportistas.insert({author : "author" +i, post_title : "Futbolistas de alto rendimiento "
...   +i, date: new Date() });
... }
WriteResult({ "nInserted" : 1 })
mongos> _
```

L >db.Jugadores.insertMany

- Con este comando realizamos el ingreso de 6 registros en la colección **Jugadores**, el sistema nos indica que el ingreso se realizó de manera exitosa.

```
mongos> db.Jugadores.insertMany( [
... {id:1, nombre: 'Macnely Torres', equipo: 'A.Nacional', posicion: 'Delantero'},
... {id:2, nombre: 'Jason Zapata', equipo: 'D. Pasto', posicion: 'Medio Campo'},
... {id:3, nombre: 'Antonio Belez', equipo: 'A. de Cali', posicion: 'Defensa'},
... {id:4, nombre: 'Mario Benitez', equipo: 'D. Tolima', posicion: 'Portero'},
... {id:5, nombre: 'Camilo Vargas', equipo: 'Junior', posicion: 'Lateral'},
... {id:6, nombre: 'Luis Diaz', equipo: 'Millonarios', posicion: 'Delantero'}
... ]);
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("6436b89aca6ccec077ad053d"),
    ObjectId("6436b89aca6ccec077ad053e"),
    ObjectId("6436b89aca6ccec077ad053f"),
    ObjectId("6436b89aca6ccec077ad0540"),
    ObjectId("6436b89aca6ccec077ad0541"),
    ObjectId("6436b89aca6ccec077ad0542")
  ]
}
```

> db.Equipos.insertMany

- Con este comando realizamos el ingreso de 5 registros en la colección **Equipos**, el sistema nos indica que el ingreso se realizó de manera exitosa.

```
}
mongos> db.Equipos.insertMany( [
... {id:1, nombre: 'D. Cali', titulos: '12'},
... {id:2, nombre: 'A. Nacional', titulos: '22'},
... {id:3, nombre: 'A. de Cali', titulos: '9'},
... {id:4, nombre: 'Junior', titulos: '8'},
... {id:5, nombre: 'Millonarios', titulos: '12'}
... ]);
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("6436190977d705d2e652e9f7"),
    ObjectId("6436190977d705d2e652e9f8"),
    ObjectId("6436190977d705d2e652e9f9"),
    ObjectId("6436190977d705d2e652e9fa"),
    ObjectId("6436190977d705d2e652e9fb")
  ]
}
mongos> █
```

- Verificamos que los datos ingresados en cada una de las colecciones se ha registros de manera correcta.

```
}
mongos> db.Deportistas.count()
80000
mongos> db.Jugadores.count()
6
mongos> db.Equipos.count()
5
mongos> _
```

- En una nueva consola, realizaremos una comprobación si se está realizando la partición de datos entre los Shards, en primer lugar verificaremos la partición de los datos de la colección **Deportistas**, para este ejercicio ejecutamos los siguientes comandos:

```
>shard1 = new Mongo("DESKTOP-KEHD2QP:20000")
```

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

```
>shard1DB.Deportistas.count()
```

```
>shard2 = new Mongo("DESKTOP-KEHD2QP:20001")
```

```
>shard2DB = shard2.getDB("eventoDeportivo")
```

```
>shard2DB.Deportistas.count()
```

```
>shard3 = new Mongo("DESKTOP-KEHD2QP:20002")
```

```
>shard3DB = shard3.getDB("eventoDeportivo")
```

```
>shard3DB.Deportistas.count()
```

```

> shard1 = new Mongo("DESKTOP-KEHD2QP:20000")
connection to DESKTOP-KEHD2QP:20000
> shard1DB = shard1.getDB("eventoDeportivo")
eventoDeportivo
> shard1DB.Deportistas.count()
0
> shard2 = new Mongo("DESKTOP-KEHD2QP:20001")
connection to DESKTOP-KEHD2QP:20001
> shard2DB = shard2.getDB("eventoDeportivo")
eventoDeportivo
> shard2DB.Deportistas.count()
80000
> shard3 = new Mongo("DESKTOP-KEHD2QP:20002")
connection to DESKTOP-KEHD2QP:20002
> shard3DB = shard3.getDB("eventoDeportivo")
eventoDeportivo
> shard3DB.Deportistas.count()
0

```

- Como podemos ver el sistema nos permite ver que la información de la colección **Deportistas** solo se esta almacenando en el Shard 2, con puerto 20001; esto nos indica que aun no esta habilitado el sistema de partición entre los 3 Shards.
- Realizamos el mismo ejercicio con la colección **Jugadores**, utilizamos los siguientes comandos:

```

>shard1 = new Mongo("DESKTOP-KEHD2QP:20000")
>shard1DB = shard1.getDB("eventoDeportivo")
>shard1DB.Jugadores.count()

```

```

>shard2 = new Mongo("DESKTOP-KEHD2QP:20001")
>shard2DB = shard2.getDB("eventoDeportivo")
>shard2DB.Jugadores.count()

```



```
>shard3 = new Mongo("DESKTOP-KEHD2QP:20002")
```

```
>shard3DB = shard3.getDB("eventoDeportivo")
```

```
>shard3DB.Jugadores.count()
```

```
> shard1 = new Mongo("DESKTOP-KEHD2QP:20000")
connection to DESKTOP-KEHD2QP:20000
> shard1DB = shard1.getDB("eventoDeportivo")
eventoDeportivo
> shard1DB.Jugadores.count()
0
> shard2 = new Mongo("DESKTOP-KEHD2QP:20001")
connection to DESKTOP-KEHD2QP:20001
> shard2DB = shard2.getDB("eventoDeportivo")
eventoDeportivo
> shard2DB.Jugadores.count()
6
> shard3 = new Mongo("DESKTOP-KEHD2QP:20002")
connection to DESKTOP-KEHD2QP:20002
> shard3DB = shard3.getDB("eventoDeportivo")
eventoDeportivo
> shard3DB.Jugadores.count()
0
```

- Como podemos ver el sistema nos permite ver que la información de la colección **Jugadores** solo se está almacenando en el Shard 2, con puerto 20001; esto nos indica que aún no está habilitado el sistema de partición entre los 3 Shards.
- Realizamos el mismo ejercicio con la colección **Equipos**, utilizamos los siguientes comandos:

```
>shard1 = new Mongo("DESKTOP-KEHD2QP:20000")
```

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

```
>shard1DB.Equipos.count()
```

```
>shard2 = new Mongo("DESKTOP-KEHD2QP:20001")
```

```
>shard2DB = shard2.getDB("eventoDeportivo")
```

```
>shard2DB.Equipos.count()
```

```
>shard3 = new Mongo("DESKTOP-KEHD2QP:20002")
```

```
>shard3DB = shard3.getDB("eventoDeportivo")
```

```
>shard3DB.Equipos.count()
```

```
> shard1 = new Mongo("DESKTOP-KEHD2QP:20000")
connection to DESKTOP-KEHD2QP:20000
> shard1DB = shard1.getDB("eventoDeportivo")
eventoDeportivo
> shard1DB.Equipos.count()
0
> shard2 = new Mongo("DESKTOP-KEHD2QP:20001")
connection to DESKTOP-KEHD2QP:20001
> shard2DB = shard2.getDB("eventoDeportivo")
eventoDeportivo
> shard2DB.Equipos.count()
5
> shard3 = new Mongo("DESKTOP-KEHD2QP:20002")
connection to DESKTOP-KEHD2QP:20002
> shard3DB = shard3.getDB("eventoDeportivo")
eventoDeportivo
> shard3DB.Equipos.count()
0
```

- Como podemos ver el sistema nos permite ver que la información de la colección **Equipos** solo se está almacenando en el Shard 2, con puerto 20001; esto nos indica que aún no está habilitado el sistema de partición entre los 3 Shards.

```
A > shard1 = new Mongo("DESKTOP-KEHD2QP:20006")
```

```
> sh.status()
```

- Como observamos que aún no hay una partición de los datos vamos a verificar que el balancer este activo, para esto ejecutamos los dos comandos anteriores.

```

mongos> shard1 = new Mongo("DESKTOP-KEHD2QP:20006")
connection to DESKTOP-KEHD2QP:20006
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
    "_id" : 1,
    "minCompatibleVersion" : 5,
    "currentVersion" : 6,
    "clusterId" : ObjectId("6436b634a46f7e89a0f6e962")
  }
  shards:
    { "_id" : "__unknown_name__-rs0", "host" : "__unknown_name__-rs0/DESKTOP-KEHD2QP:20000", "state" : 1 }
    { "_id" : "__unknown_name__-rs1", "host" : "__unknown_name__-rs1/DESKTOP-KEHD2QP:20001", "state" : 1 }
    { "_id" : "__unknown_name__-rs2", "host" : "__unknown_name__-rs2/DESKTOP-KEHD2QP:20002", "state" : 1 }
  active mongoses:
    "4.2.23-rc0" : 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" : "config", "primary" : "config", "partitioned" : true }
      config.system.sessions
        shard key: { "_id" : 1 }
        unique: false
        balancing: true
        chunks:
          __unknown_name__-rs0      1
          { "_id" : { "$minKey" : 1 } } --> { "_id" : { "$maxKey" : 1 } } on : __unknown_name__-rs0
    { "_id" : "eventoDeportivo", "primary" : "__unknown_name__-rs1", "partitioned" : false, "version" : {
eb32e62c2"}, "lastMod" : 1 } }

```

- El status me informa que que el balancer no se encuentra activo.

> sh.enableSharding("eventoDeportivo")

- Este comando nos permite activar el particionamiento de la información que se encuentra en las colecciones.

```

mongos> sh.enableSharding("eventoDeportivo")
{
  "ok" : 1,
  "operationTime" : Timestamp(1681312338, 3),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1681312338, 3),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
mongos>

```

- El sistema nos muestra que OK, el particionamiento esta activo.

>db.Deportistas.ensureIndex({author : 1})

- Con este comando realizaremos un index, de esta manera enviara la colección Deportistas al shard 20001.

```
mongos> db.Deportistas.ensureIndex({author : 1})
{
  "raw" : {
    "__unknown_name__-rs1/DESKTOP-KEHD2QP:20001" : {
      "createdCollectionAutomatically" : false,
      "numIndexesBefore" : 1,
      "numIndexesAfter" : 2,
      "ok" : 1
    }
  },
  "ok" : 1,
  "operationTime" : Timestamp(1681312504, 2),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1681312504, 2),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
```

>sh.shardCollection("eventoDeportivo.Deportistas", {author: 1})

- Con este comando tomamos la colección Deportistas de la base de datos eventoDeportivo y le aplicamos el Shard.

```
mongos> sh.shardCollection("eventoDeportivo.Deportistas", {author: 1})
{
  "collectionsharded" : "eventoDeportivo.Deportistas",
  "collectionUUID" : UUID("e818bf17-d7dc-42f6-bbf9-725856510c52"),
  "ok" : 1,
  "operationTime" : Timestamp(1681312661, 9),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1681312661, 9),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
```

>sh.getBalancerState()

- Consultamos el estado del balanceador, usamos el comando anterior.

```
mongos> sh.getBalancerState()
false
mongos> _
```

>sh.setBalancerState(true)

- Para activar el balanceador usamos el comando balancerstate(true)

```
mongos> sh.setBalancerState(true)
{
  "ok" : 1,
  "operationTime" : Timestamp(1681313731, 3),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1681313731, 3),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
```

>sh.isBalancerRunning()

- Con este comando ejecutamos la solicitud de partición.

```
active mongoses:
  "4.2.23-rc0" : 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" : "Biblioteca", "primary" : "__unknown_name__-rs2", "partitioned" : true, "version" : { "uuid" : "ff2"), "lastMod" : 1 } }
  { "_id" : "config", "primary" : "config", "partitioned" : true }
    config.system.sessions
      shard key: { "_id" : 1 }
      unique: false
      balancing: true
      chunks:
        __unknown_name__-rs0    1
        { "_id" : { "$minKey" : 1 } } --> { "_id" : { "$maxKey" : 1 } } on : __unknown_name__-rs0 Ti
  { "_id" : "eventoDeportivo", "primary" : "__unknown_name__-rs1", "partitioned" : true, "version" : { "uu
b32e62c2"), "lastMod" : 1 } }
    eventoDeportivo.Deportistas
      shard key: { "author" : 1 }
      unique: false
      balancing: true
      chunks:
        __unknown_name__-rs1    1
        { "author" : { "$minKey" : 1 } } --> { "author" : { "$maxKey" : 1 } } on : __unknown_name__-
```

> cluster.stop()

- Detenemos la partición de los documentos.

```
ReplSetTest stop *** Mongod in port 20005 shutdown with code (0) ***
ReplSetTest stopSet stopped all replica set nodes.
ReplSetTest stopSet deleting all dbpaths
ReplSetTest stopSet deleting dbpath: /data/db/__unknown_name__-configRS-0
ReplSetTest stopSet deleting dbpath: /data/db/__unknown_name__-configRS-1
ReplSetTest stopSet deleting dbpath: /data/db/__unknown_name__-configRS-2
ReplSetTest stopSet deleted all dbpaths
2023-04-12T13:37:07.138-0500 I NETWORK [js] Removed ReplicaSetMonitor for replica set __unknown_name__-configRS
ReplSetTest stopSet *** Shut down repl set - test worked ****
ShardingTest stop deleting all dbpaths
*** ShardingTest __unknown_name__ completed successfully in 17447.412 seconds ***
```