Actividad 2 - Conceptos y comandos básicos de la replicación en bases de datos NoSQL

Anderson Devia Artunduaga.

IBEROAMERICANA CORPORACION UNIVERSITARIA

Ingeniería de software

Base de datos avanzadas

WILLIAM RUIZ

Diciembre 2023.

Tabla de Contenidos

Video Link: https://youtu.be/_5Cbo-Cr7ZA

GitHub: https://github.com/Anderdevia/torneoFutbol.git

Paso 1: Creación de la replica

Con el comando llamado: ReplicaSet, se crean las réplicas para MongoDB para que se puedan crear tres copias de los datos correspondientes.

```
> MiejReplicaSet = new ReplSetTest({name: "MireplicaSet", nodes: 3})
```

Este es el resultado que se genera y posteriormente la creación de la réplica con sus tres nodos y puertos correspondientes

```
        return master;
    },
        "name" : "MireplicaSet",
        "useHostName" : true,
        "host" : "DESKTOP-N2RUEH8",
        "oplogSize" : 40,
        "useSeedList" : false,
        "keyFile" : undefined,
        "protocolVersion" : undefined,
        "waitForKeys" : undefined,
        "seedRandomNumberGenerator" : true,
        "isConfigServer" : undefined,
        "nodeOptions" : {
                "n0" : undefined,
                "n1" : undefined,
                "n2" : undefined
        },
        "nodes" : [ ],
        "ports" : [
                20000,
                20001,
                20002
        ]
```

Paso 2: Levantar el proceso de la réplica en MONGODB

En esta parte se inicia el arranque de las instancias de los nodos creados anteriormente, la sentencia o el comando a ejecutar seria el siguiente:

```
> MiejReplicaSet.startSet()
```

Resultado nodo 0

```
> MiejReplicaSet.startSet()
ReplSetTest starting set 'MireplicaSet'
ReplSetTest n is : 0
{
        "useHostName" : true,
        "oplogSize" : 40,
        "keyFile" : undefined,
        "port" : 20000,
        "replSet" : "MireplicaSet",
        "dbpath" : "$set-$node",
        "waitForConnect" : false,
        "restart" : undefined,
        "pathOpts" : {
                "node" : 0,
                "set" : "MireplicaSet"
        },
        "setParameter" : {
                "writePeriodicNoops" : false,
                "numInitialSyncConnectAttempts" : 60
        }
}
```

Resultado nodo 1

```
ReplSetTest n is : 1
{
        "useHostName" : true,
        "oplogSize" : 40,
        "keyFile" : undefined,
        "port" : 20001,
        "replSet" : "MireplicaSet",
        "dbpath" : "$set-$node",
        "waitForConnect" : false,
        "restart" : undefined,
        "pathOpts" : {
                "node" : 1,
                "set" : "MireplicaSet"
        },
        "setParameter" : {
                "writePeriodicNoops" : false,
                "numInitialSyncConnectAttempts" : 60
        }
}
```

Resultado nodo 2

```
ReplSetTest n is : 2
{
        "useHostName" : true,
        "oplogSize" : 40,
        "keyFile" : undefined,
        "port" : 20002,
        "replSet" : "MireplicaSet",
        "dbpath" : "$set-$node",
        "waitForConnect" : false,
        "restart" : undefined,
        "pathOpts" : {
                "node" : 2,
                "set" : "MireplicaSet"
        },
        "setParameter" : {
                "writePeriodicNoops" : false,
                "numInitialSyncConnectAttempts" : 60
        }
}
```

1. **Paso 3: Iniciar el proceso de la réplica**

Se inicia el proceso de réplica de datos con el siguiente comando:

```
MiejReplicaSet.initiate()
```

Salida:

```
ReplSetTest startSet took 1617ms for 3 nodes.
[
        connection to DESKTOP-N2RUEH8:20000,
        connection to DESKTOP-N2RUEH8:20001,
        connection to DESKTOP-N2RUEH8:20002
]
> MiejReplicaSet.initiate()
{
        "replSetInitiate" : {
                "_id" : "MireplicaSet",
                "protocolVersion" : 1,
                "members" : [
                        {
                                "_id" : 0,
                                "host" : "DESKTOP-N2RUEH8:20000"
                        }
                ]
        }
}
```

Casos de pruebas para la verificación del mecanismo de replicación y disponibilidad 24x7

Casos de pruebas	
Replicación	Se verifica que los nodos se hallan creado de manera correcta
Disponibilidad	Insertar documentos en las colecciones de acuerdo al documento de requerimientos no funcionales para el nodo primario, también se verifica que todas las estancias tengan la réplica de los documentos agregados
Tolerancia a Fallos	Se realiza la prueba la consiste en desconectar el nodo primario y promoción de los nodos segundarios a primario
Disponibilidad	Comprobar cuál de los nodos corresponde al nodo primario

Replicación: Se verifica que los nodos se hallan creado de manera correcta

En una nueva terminal realizamos la conexión al nodo primario con el siguiente comando:

```
> conn=new Mongo("DESKTOP-N2RUEH8:20000")
```

Después verificamos la conexión con la base de datos TorneoDeportivo y ejecutamos el siguiente comando:

```
> TestBD=conn.getDB("equipos")
```

Resultado:

```
> conn=new Mongo("DESKTOP-N2RUEH8:20000")
connection to DESKTOP-N2RUEH8:20000
> TestBD=conn.getDB("equipos")
equipos
>
```

Comprobación nodo master

Se verifica si el nodo master es el que se está ejecutando con el siguiente comando:

```
> TestBD.isMaster()
```

```
        "topologyVersion" : {
                "processId" : ObjectId("656a3d19f641de3932ef6ed6"),
                "counter" : NumberLong(8)
        },
        "hosts" : [
                "DESKTOP-N2RUEH8:20000",
                "DESKTOP-N2RUEH8:20001",
                "DESKTOP-N2RUEH8:20002"
        ],
        "setName" : "MireplicaSet",
        "setVersion" : 3,
        "ismaster" : true,
        "secondary" : false,
        "primary" : "DESKTOP-N2RUEH8:20000",
        "me" : "DESKTOP-N2RUEH8:20000",
        "electionId" : ObjectId("7fffffff0000000000000001"),
        "lastWrite" : {
                "opTime" : {
                        "ts" : Timestamp(1701461318, 1),
                        "t" : NumberLong(1)
                },
                "lastWriteDate" : ISODate("2023-12-01T20:08:38Z"),
                "majorityOpTime" : {
                        "ts" : Timestamp(1701461318, 1),
                        "t" : NumberLong(1)
                },
                "majorityWriteDate" : ISODate("2023-12-01T20:08:38Z")
```

Disponibilidad: Insertar documentos en las colecciones de acuerdo al documento de requerimientos no funcionales para el nodo primario, también se verifica que todas las estancias tengan la réplica de los documentos agregados

Insertar y actualizar datos en el nodo primario

Insertar en equipos:

```
> TestBD.equipos.insert({nombre: "unicos", ciudad: "bogota", año: "2023"});
WriteResult({ "nInserted" : 1 })
>

> TestBD.equipos.insert({nombre: "toros", ciudad: "tolima", año: "2023"});
WriteResult({ "nInserted" : 1 })
>
```

Se verifica si se insertaron de manera correcta los registros con el siguiente comando:

```
> TestBD.equipos.count()
2
>
```

Mostramos el contenido agregado con el siguiente comando:

```
> TestBD.equipos.find().pretty()
```

```
> TestBD.equipos.find().pretty()
{
        "_id" : ObjectId("656a56515607859c26ac7788"),
        "nombre" : "unicos",
        "ciudad" : "bogota",
        "año" : "2023"
}
{
        "_id" : ObjectId("656a5b785607859c26ac7789"),
        "nombre" : "toros",
        "ciudad" : "tolima",
        "año" : "2023"
}
>
```

Insertar en deportistas:

```
> TestBD.deportistas.insert({
... cedula: 1245695,
... nombre: "alberto gomez",
... apellidos: "rodrigez",
... celuar: 456963545,
... edad: 41,
... nacionalidad: "mexicana",
... equipo:"unicos",
... ciudad:"mexico",
... año:"2023"});
WriteResult({ "nInserted" : 1 })
>
```

```
> TestBD.deportistas.insert({
... cedula: 74555266,
... nombre: "federico",
... apellidos: "romero",
... celular: 7485545,
... edad: 35,
... nacionalidad: "chilena",
... equipo: "unicos",
... ciudad: "chile",
... año: "2023"});
WriteResult({ "nInserted" : 1 })
>
```

Se verifica si se insertaron de manera correcta los registros con el siguiente comando:

```
WriteResult({ "nInserted" : 1 })
> TestBD.deportistas.count()
2
>
```

Mostramos el contenido agregado con el siguiente comando:

```
> TestBD.deportistas.find().pretty()
```

```
> TestBD.deportistas.find().pretty()
{
        "_id" : ObjectId("656a5fbf5607859c26ac778a"),
        "cedula" : 1245695,
        "nombre" : "alberto gomez",
        "apellidos" : "rodrigez",
        "celuar" : 456963545,
        "edad" : 41,
        "nacionalidad" : "mexicana",
        "equipo" : "unicos",
        "ciudad" : "mexico",
        "año" : "2023"
}
{
        "_id" : ObjectId("656a62995607859c26ac778b"),
        "cedula" : 74555266,
        "nombre" : "federico",
        "apellidos" : "romero",
        "celular" : 7485545,
        "edad" : 35,
        "nacionalidad" : "chilena",
        "equipo" : "unicos",
        "ciudad" : "chile",
        "año" : "2023"
}
```

Actualizar registros nombre en la colección deportistas con el siguiente comando:

```
> TestBD.deportistas.updateOne({'nombre':'alberto gomez'}, {$set: {nombre: 'domingo'}});
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
>
```

Ahora verificamos con el siguiente comando:

```
> TestBD.deportistas.find({'nombre': 'domingo'}).pretty();
```

```
> TestBD.deportistas.find({'nombre': 'domingo'}).pretty();
{
        "_id" : ObjectId("656a5fbf5607859c26ac778a"),
        "cedula" : 1245695,
        "nombre" : "domingo",
        "apellidos" : "rodrigez",
        "celuar" : 456963545,
        "edad" : 41,
        "nacionalidad" : "mexicana",
        "equipo" : "unicos",
        "ciudad" : "mexico",
        "año" : "2023"
}
>
```

Ahora verificamos la réplica en los nodos segundarios:
Procedemos a conectarnos al primer nodo segundario con el siguiente comando:

```
> connSecondary=new Mongo("DESKTOP-N2RUEH8:20001")
```

Respuesta:

```
> connSecondary=new Mongo("DESKTOP-N2RUEH8:20001")
connection to DESKTOP-N2RUEH8:20001
>
```

Conexión a la Base correspondiente con el siguiente comando:

```
> secondaryTestDB=connSecondary.getDB("TorneoDeportivo")
TorneoDeportivo
>
```

Verificamos si es un nodo segundario con el siguiente comando:

```
> secondaryTestDB.isMaster()
{
        "topologyVersion" : {
                "processId" : ObjectId("656a3d1969564edc5a268f9e"),
                "counter" : NumberLong(4)
        },
        "hosts" : [
                "DESKTOP-N2RUEH8:20000",
                "DESKTOP-N2RUEH8:20001",
                "DESKTOP-N2RUEH8:20002"
        ],
        "setName" : "MireplicaSet",
        "setVersion" : 3,
        "ismaster" : false,
        "secondary" : true,
        "primary" : "DESKTOP-N2RUEH8:20000",
        "me" : "DESKTOP-N2RUEH8:20001",
        "lastWrite" : {
                "opTime" : {
                        "ts" : Timestamp(1701471541, 1),
                        "t" : NumberLong(1)
                },
                "lastWriteDate" : ISODate("2023-12-01T22:59:01Z"),
                "majorityOpTime" : {
                        "ts" : Timestamp(1701471541, 1),
                        "t" : NumberLong(1)
                },
                "majorityWriteDate" : ISODate("2023-12-01T22:59:01Z")
```

Ahora verificamos que este nodo tenga los mismos datos del primario en la colección equipos y deportistas con el suigiente comando:

```
> TestBD.equipos.count()
```

Respuesta:

```
> TestBD.equipos.count()
2
>
```

```
> TestBD.deportistas.count()
```

Respuesta:

```
> TestBD.deportistas.count()
2
>
```

Ahora procedemos a ver el contenido de las colecciones:

```
> TestBD.equipos.find().pretty()
{
        "_id" : ObjectId("656a56515607859c26ac7788"),
        "nombre" : "unicos",
        "ciudad" : "bogota",
        "año" : "2023"
}
{
        "_id" : ObjectId("656a5b785607859c26ac7789"),
        "nombre" : "toros",
        "ciudad" : "tolima",
        "año" : "2023"
}
>
```

```
> TestBD.deportistas.find().pretty()
{
        "_id" : ObjectId("656a5fbf5607859c26ac778a"),
        "cedula" : 1245695,
        "nombre" : "domingo",
        "apellidos" : "rodrigez",
        "celuar" : 456963545,
        "edad" : 41,
        "nacionalidad" : "mexicana",
        "equipo" : "unicos",
        "ciudad" : "mexico",
        "año" : "2023"
}
{
        "_id" : ObjectId("656a62995607859c26ac778b"),
        "cedula" : 74555266,
        "nombre" : "federico",
        "apellidos" : "romero",
        "celular" : 7485545,
        "edad" : 35,
        "nacionalidad" : "chilena",
        "equipo" : "unicos",
        "ciudad" : "chile",
        "año" : "2023"
}
>
```

Ahora nos conectamos al segundo nodo segundario con el siguiente comando:

```
> connSecondary=new Mongo("DESKTOP-N2RUEH8:20002")
```

Respuesta:

```
> connSecondary=new Mongo("DESKTOP-N2RUEH8:20002")
connection to DESKTOP-N2RUEH8:20002
>
```

Conexión a la Base correspondiente con el siguiente comando:

```
> secondaryTestDB=connSecondary.getDB("TorneoDeportivo")
```

Respuesta:

```
> secondaryTestDB=connSecondary.getDB("TorneoDeportivo")
TorneoDeportivo
>
```

Se verifica si es el nodo segundario:

```
> connSecondary=new Mongo("DESKTOP-N2RUEH8:20002")
connection to DESKTOP-N2RUEH8:20002
> secondaryTestDB=connSecondary.getDB("TorneoDeportivo")
TorneoDeportivo
> secondaryTestDB.isMaster()
{
        "topologyVersion" : {
                "processId" : ObjectId("656a3d1a544611a0e75fe330"),
                "counter" : NumberLong(3)
        },
        "hosts" : [
                "DESKTOP-N2RUEH8:20000",
                "DESKTOP-N2RUEH8:20001",
                "DESKTOP-N2RUEH8:20002"
        ],
        "setName" : "MireplicaSet",
        "setVersion" : 3,
        "ismaster" : false,
        "secondary" : true,
        "primary" : "DESKTOP-N2RUEH8:20000",
        "me" : "DESKTOP-N2RUEH8:20002",
        "lastWrite" : {
                "opTime" : {
                        "ts" : Timestamp(1701471541, 1),
                        "t" : NumberLong(1)
```

Verificamos si tiene los registros con el siguiente comando:

```
> TestBD.equipos.count()
```

Respuesta:

```
> TestBD.equipos.count()
2
>
```

```
> TestBD.deportistas.count()
```

Respuesta:

```
> TestBD.deportistas.count()
2
>
```

Ahora procedemos a ver el contenido de las colecciones:

```
> TestBD.equipos.find().pretty()
{
        "_id" : ObjectId("656a56515607859c26ac7788"),
        "nombre" : "unicos",
        "ciudad" : "bogota",
        "año" : "2023"
}
{
        "_id" : ObjectId("656a5b785607859c26ac7789"),
        "nombre" : "toros",
        "ciudad" : "tolima",
        "año" : "2023"
}
>
```

```
> TestBD.deportistas.find().pretty()
{
        "_id" : ObjectId("656a5fbf5607859c26ac778a"),
        "cedula" : 1245695,
        "nombre" : "domingo",
        "apellidos" : "rodrigez",
        "celuar" : 456963545,
        "edad" : 41,
        "nacionalidad" : "mexicana",
        "equipo" : "unicos",
        "ciudad" : "mexico",
        "año" : "2023"
}
{
        "_id" : ObjectId("656a62995607859c26ac778b"),
        "cedula" : 74555266,
        "nombre" : "federico",
        "apellidos" : "romero",
        "celular" : 7485545,
        "edad" : 35,
        "nacionalidad" : "chilena",
        "equipo" : "unicos",
        "ciudad" : "chile",
        "año" : "2023"
}
>
```

Tolerancia a Fallos: Se realiza la prueba la consiste en desconectar el nodo primario y promoción de los nodos segundarios a primario

Se detiene el nodo primario para simular que esta caído y lo detenemos con los siguientes comandos:

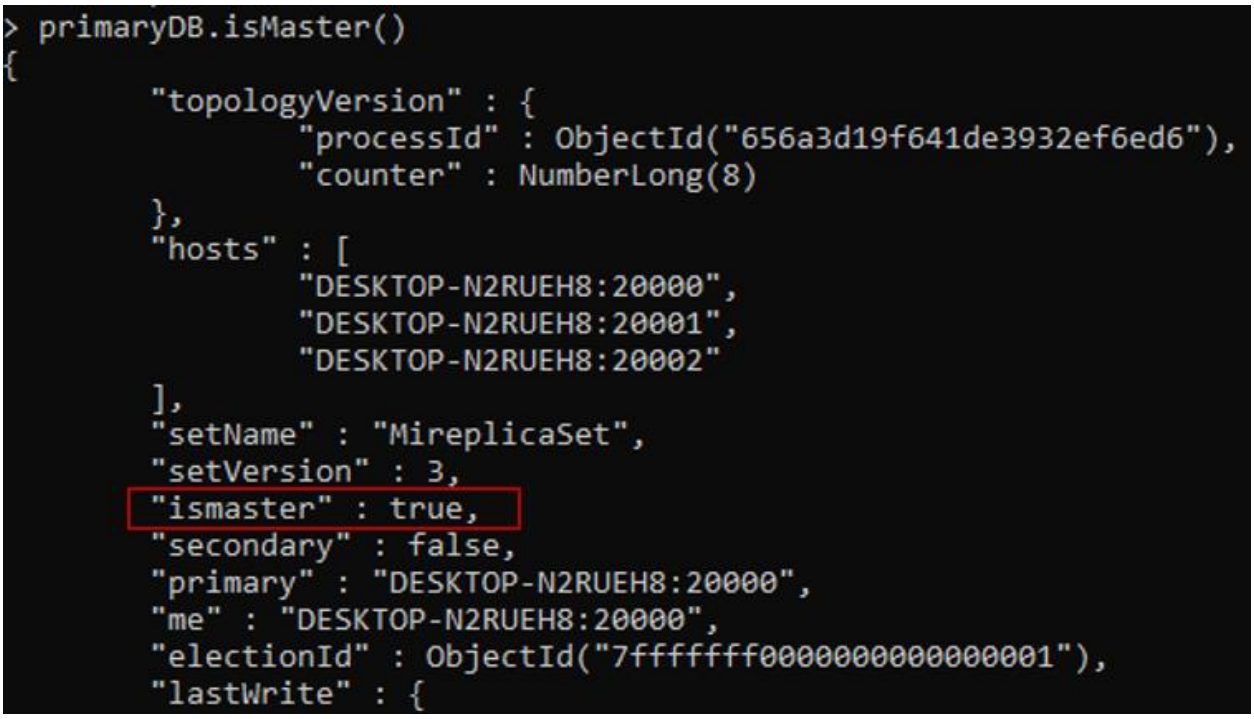
Primero nos conectamos:

```
> connPrimary = new Mongo("DESKTOP-N2RUEH8:20000")
connection to DESKTOP-N2RUEH8:20000
>
```

Ahora a la base:

```
> primaryDB = connPrimary.getDB("TorneoDeportivo")
TorneoDeportivo
>
```

Verificamos que estamos en Master:



Disponibilidad: Comprobar cuál de los nodos corresponde al nodo primario

Detenemos el nodo primario:

```
> primaryDB.adminCommand({shutdown:1})
```

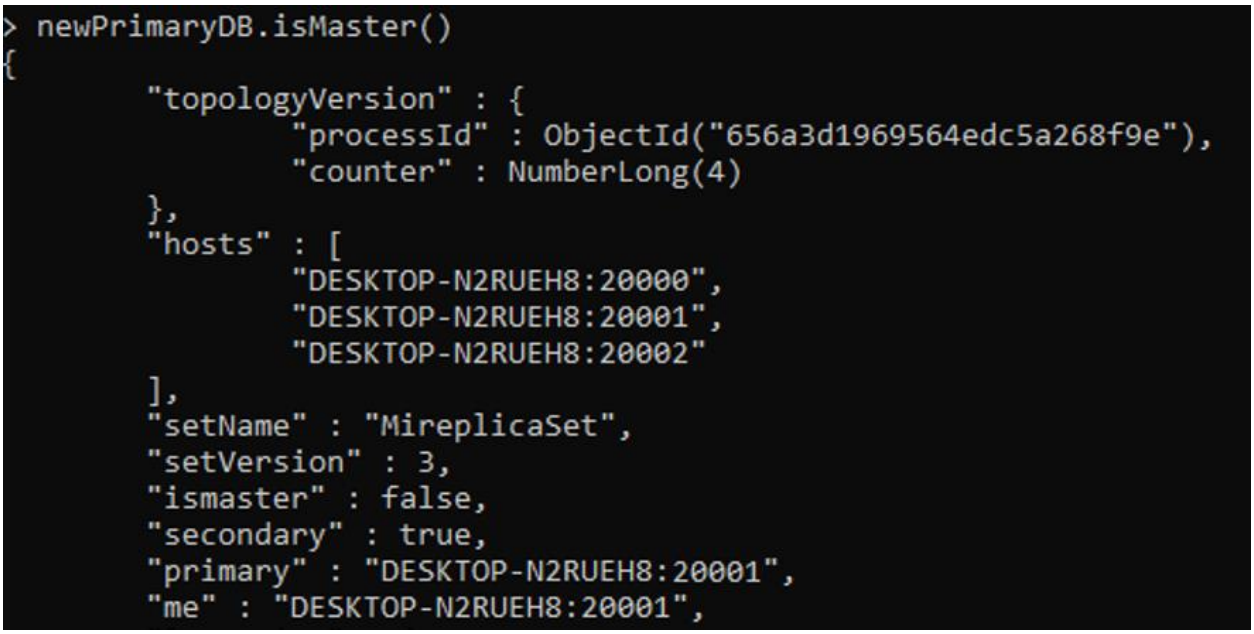
Disponibilidad 24x7 para verificar cual nodo segundario corresponde al nodo primario y promovemos el nuevo nodo (20001) a nodo primario con los siguientes comandos:

```
> connNewPrimary=new Mongo("DESKTOP-N2RUEH8:20001")
connection to DESKTOP-N2RUEH8:20001
>
```

Accedemos a la base:

```
> newPrimaryDB=connNewPrimary.getDB("Torneo")
Torneo
>
```

Ahora verificamos si el nodo elegido es verdaderamente el primario:



BILIOGRAFIAS

Sarasa, A. (2016). Introducción a las bases de datos NoSQL usando MongoDB. Editorial UOC. (Capitulo 7- Replicacion)