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

AUTOR:

Andres Gregorio Rojas Barrera

Facultad de Ingeniería

Corporación Universitaria Iberoamericana

Bases de datos Avanzadas

Tutor:

WILLIAM RUIZ MARTINEZ

Número de ID:

100132525

Fecha de entrega:

26/05/2024

Introducción

El siguiente documento contiene el desarrollo de la actividad 2, la cual corresponde a conceptos y comandos básicos de replicación en las bases de datos no SQL, esto con el fin de evitar fallos y cumplir los requerimientos funcionales como lo son la disponibilidad, redundancia de datos, base de datos de calidad, para ellos se crean los correspondientes nodos secundarios los cuales nos permiten cumplir dichos objetivos, en este caso el ejercicio va a ser aplicado a una base de datos de un torneo de Futsal

El trabajo se guarda en el repositorio de Github

https://github.com/arojasb5/Bases-de-datos-avanzadas

y el video debido al peso se guarda en la siguiente ruta de One Drive

Actividad 2 Base de datos Avanzadas

1. Requerimientos no funcionales (Base de datos torneo Futsal)

REQNF # 01	La base de datos debe contar con disponibilidad 7*24 debido a las consultas realizadas por aficionados, jugadores, jueces y árbitros
	Solicitante : Cliente

REQNF # 02	La base de datos debe contar con 2 nodos secundarios los cuales permitan tener redundancia de datos, esto con el fin de no perder información en caso de desastres
	Solicitante : Cliente

REQNF # 03	Los datos actualizados deben ser disponibles para todos los usuarios
	Solicitante : Cliente

2. Proceso de replicación

Comandos

1. Creación de las instancias

   Utilizamos el siguiente comando el cual nos permite crear el número de instancias necesarias, en este caso creamos 3 nodos disponibles

```
> CreacionReplica = new ReplSetTest ({name: "Creacionreplica", nodes: 3})
Starting new replica set Creacionreplica
{
        "kDefaultTimeoutMS" : 600000,
        "getReadConcernMajorityOpTimeOrThrow" : function(conn) {
        const majorityOpTime = _getReadConcernMajorityOpTime(conn);
        if (friendlyEqual(majorityOpTime, {ts: Timestamp(0, 0), t: NumberLong(0)})) {
            throw new Error("readConcern majority optime not available");
```

```
                "n0" : undefined,
                "n1" : undefined,
                "n2" : undefined
        },
        "nodes" : [ ],
        "ports" : [
                20000,
                20001,
                20002
        ]
}
>
```

   CreacionReplica = new ReplSetTest ({name: "Creacionreplica", nodes: 3})

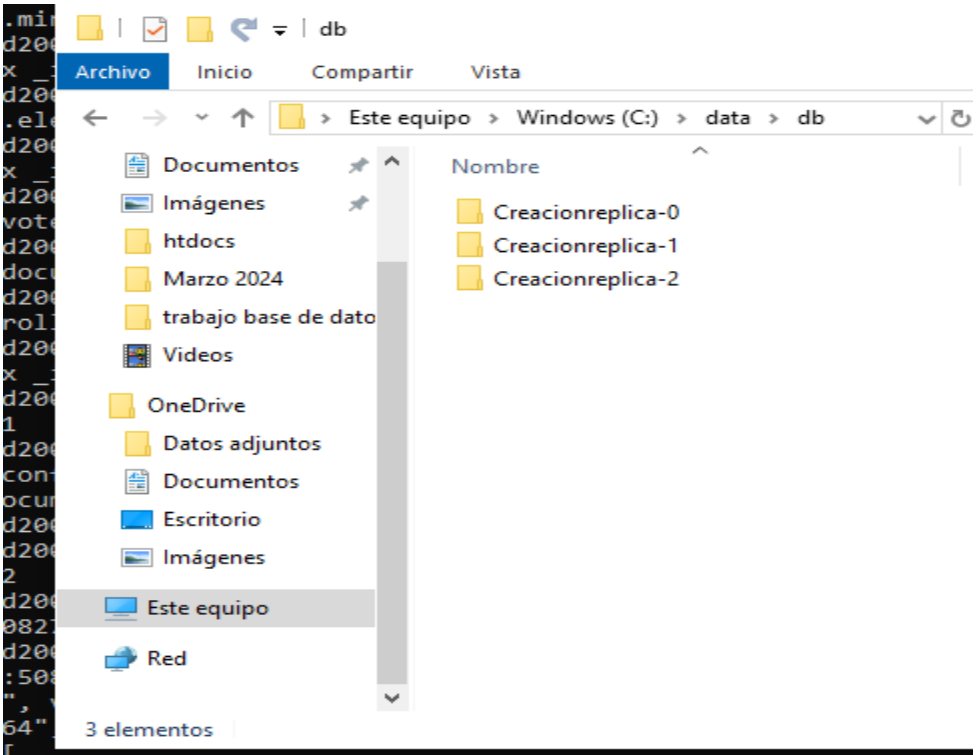
2. Y inicializamos los nodos generados, esto con el fin de iniciarlos dado que solo se a creado el espacio en la variable

   Para esto utilizamos el objeto generado anteriormente junto al comando .startSet()

   CreacionReplica.startSet()

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

donde se evidencia la generación de los repositorios para los nodos




```
[
        connection to LAPTOP-QC7509DO:20000,
        connection to LAPTOP-QC7509DO:20001,
        connection to LAPTOP-QC7509DO:20002
]
[
        connection to LAPTOP-QC7509DO:20000,
        connection to LAPTOP-QC7509DO:20001,
        connection to LAPTOP-QC7509DO:20002
]
```

3. Ya con los repositorios generados y los nodos inicializados, iniciamos el proceso de replicación, para esto utilizamos el siguiente comando

   CreacionReplica.initiate()

```
> CreacionReplica.initiate()
{
        "replSetInitiate" : {
                "_id" : "Creacionreplica",
                "protocolVersion" : 1,
                "members" : [
                        {
                                "_id" : 0,
                                "host" : "LAPTOP-QC7509DO:20000"
                        }
                ]
        }
}
```

   Y nos indica la finalización del proceso

```
AwaitNodesAgreeOnPrimary: Waiting for nodes to agree on any primary.
AwaitNodesAgreeOnPrimary: Nodes agreed on primary LAPTOP-QC7509DO:20000


[jsTest] ----
[jsTest] ReplSetTest stepUp: Finished stepping up LAPTOP-QC7509DO:20000
[jsTest] ----


>
```

Pruebas

Se realiza conexión del nodo primario (en este caso denominado con 0 en el puerto 20000) con el siguiente comando, el cual nos permite conectarnos a dicho nodo

conn=new Mongo(“nombreconexión:puerto”)

conn=new Mongo(“localhost:20000”)

```
> conn=new Mongo("localhost:20000")
connection to localhost:20000
>
```

Y para asegurar que se conecta a la base de datos, realizamos la prueba con la base de datos del torneo

```
> testDB=conn.getDB("TorneoFutbolSala")
TorneoFutbolSala
>
```

Y realizamos la validación que los servicios estén en el nodo maestro

testBD.isMaster()

```
> testDB.isMaster()
{
        "hosts" : [
                "LAPTOP-QC7509DO:20000",
                "LAPTOP-QC7509DO:20001",
                "LAPTOP-QC7509DO:20002"
        ],
        "setName" : "Creacionreplica",
        "setVersion" : 2,
        "ismaster" : true,
        "secondary" : false,
        "primary" : "LAPTOP-QC7509DO:20000",
        "me" : "LAPTOP-QC7509DO:20000",
        "electionId" : ObjectId("7fffffff0000000000000001"),
        "lastWrite" : {
                "opTime" : {
                        "ts" : Timestamp(1716767546, 1),
                        "t" : NumberLong(1)
                },
                "lastWriteDate" : ISODate("2024-05-26T23:52:26Z"),
                "majorityOpTime" : {
                        "ts" : Timestamp(1716767546, 1),
                        "t" : NumberLong(1)
                },
                "majorityWriteDate" : ISODate("2024-05-26T23:52:26Z")
        },
        "maxBsonObjectSize" : 16777216,
        "maxMessageSizeBytes" : 48000000,
        "maxWriteBatchSize" : 100000,
        "localTime" : ISODate("2024-05-27T00:15:31.515Z"),
        "logicalSessionTimeoutMinutes" : 30,
        "connectionId" : 29,
```

Y realizamos la inserción de algún documento para verificar que se este replicando en los nodos

```
> testDB.equipos.insert(
... {
... id_equipo:"5",
... nombre:"sur FC",
... numero_jugadores:"7",
... puntos:"0"
... }
... )
WriteResult({ "nInserted" : 1 })
>
```

Y revisamos que se encuentre el registro

```
> testDB.equipos.count()
1
> testDB.equipos.find().pretty()
{
        "_id" : ObjectId("6653d7ffcac5b08ed70954fe"),
        "id_equipo" : "5",
        "nombre" : "sur FC",
        "numero_jugadores" : "7",
        "puntos" : "0"
}
>
```

Ahora nos conectamos a nodo 2 y verificamos que se halla replicado el documento

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

```
> testDBdos = connNodo.getDB("equipos")
equipos
>
```

```
> connNodo.setSecondaryOk()
> testDBdos.equipos.find().pretty()
{
        "_id" : ObjectId("6653d7ffcac5b08ed70954fe"),
        "id_equipo" : "5",
        "nombre" : "sur FC",
        "numero_jugadores" : "7",
        "puntos" : "0"
}
>
```

Y se evidencia que en el segundo nodo también se encuentra el registro

Y en el nodo 3 se replica el evento

```
> connNodo = new Mongo("localhost:20002")
connection to localhost:20002
> testDBdos = connNodo.getDB("TorneoFutbolSala")
TorneoFutbolSala
> connNodo.setSecondaryOk()
> testDBdos.equipos.find().pretty()
{
        "_id" : ObjectId("6653d7ffcac5b08ed70954fe"),
        "id_equipo" : "5",
        "nombre" : "sur FC",
        "numero_jugadores" : "7",
        "puntos" : "0"
}
>
```

Para cumplir la funcionalidad de disponibilidad bajamos el nodo

```
> connPrimario = new Mongo("localhost:20000")
connection to localhost:20000
> primarioDB=connPrimario.getDB("TorneoFutbolSala")
TorneoFutbolSala
> primarioDB.adminCommand({shutdown:1})
```

Y ponemos el nodo 2 como master

```
> connNodo=new Mongo("localhost:20001")
connection to localhost:20001
> primaryDB=connNodo.getDB("TorneoFutbolSala")
TorneoFutbolSala
> primaryDB.isMaster()
{
        "hosts" : [
                "LAPTOP-QC7509DO:20000",
                "LAPTOP-QC7509DO:20001",
                "LAPTOP-QC7509DO:20002"
        ],
        "setName" : "Creacionreplica",
        "setVersion" : 2,
        "ismaster" : true,
        "secondary" : false,
        "primary" : "LAPTOP-QC7509DO:20001",
        "me" : "LAPTOP-QC7509DO:20001",
        "electionId" : ObjectId("7fffffff0000000000000002"),
        "lastWrite" : {
```

```
> primaryDB.equipos.insert(
... {
... id_equipo: "6",
... nombre: "atlantico",
... numero_jugadores: "6",
... puntos: "0"
... }
... );
WriteResult({ "nInserted" : 1 })
> primaryDB.equipos.find().pretty()
{
        "_id" : ObjectId("6653ed2678eb39b73e13167b"),
        "id_equipo" : "6",
        "nombre" : "atlantico",
        "numero_jugadores" : "6",
        "puntos" : "0"
}
```

Bibliografía

Sarasa, A. (2016). *Introducción a las bases de datos NoSQL usando MongoDB*. Editorial UOC