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

Juan Sebastián Vacca Peña

ID 100098569

Corporación Universitaria Iberoamericana

Bases de Datos Avanzadas

03 diciembre de 203

Contenido

Objetivo

Este documento tiene como objetivo definir los requerimientos no funcionales relacionados con la redundancia y la disponibilidad 24x7 para el torneo de futbol.

Redundancia

El sistema debe tener una alta redundancia para garantizar que el evento se lleve a cabo sin interrupciones. Para lograr esto, se deben cumplir los siguientes requerimientos no funcionales:

- **Disponibilidad de múltiples servidores:** El sistema debe contar con múltiples servidores que puedan manejar el tráfico simultáneo de usuarios, y que estén configurados para proporcionar una redundancia adecuada. Los servidores deben estar ubicados en diferentes ubicaciones geográficas para garantizar la disponibilidad continua.

- **Respaldos frecuentes:** Se deben realizar respaldos frecuentes de la información del sistema en diferentes ubicaciones geográficas. Los respaldos deben ser almacenados en un lugar seguro y accesible en caso de una falla en el sistema principal.

- **Pruebas de continuidad del negocio:** El sistema debe contar con pruebas regulares de continuidad del negocio para garantizar que, en caso de una interrupción del servicio, se pueda restaurar el sistema en un tiempo razonable.

Disponibilidad 24x7

El sistema debe estar disponible las 24 horas del día, los 7 días de la semana, para garantizar la satisfacción del usuario y el éxito del torneo deportivo de fútbol. Para lograr esto, se deben cumplir los siguientes requerimientos no funcionales:

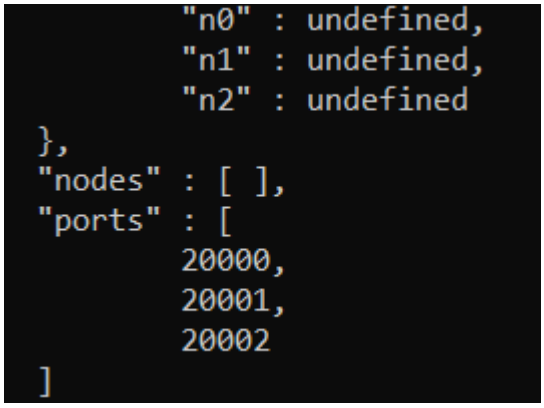
1. **Tiempo de actividad del sistema:** El sistema debe estar en funcionamiento continuo durante todo el evento, sin interrupciones, y estar disponible en todo momento para los usuarios.

2. **Monitoreo constante del sistema:** Se deben realizar monitoreos constantes del sistema para detectar cualquier problema que pueda afectar su disponibilidad, y solucionarlos en tiempo real.

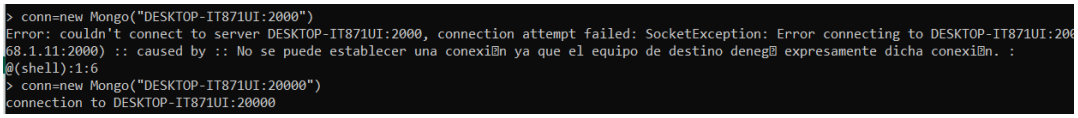
Pruebas de replicación

1. **Replicación**

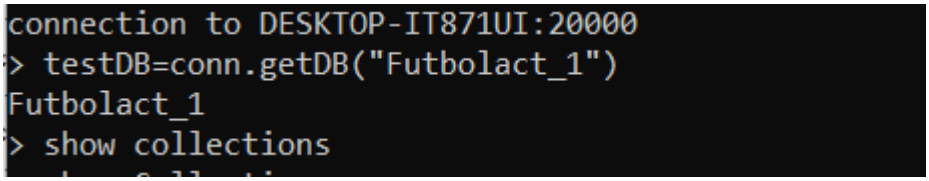
Verificar que se haya creado el nodo Maestro y los esclavos



Se crean tres nodos, y el sistema informa en que puerto se alojan y como se llama cada uno.



Nos conectamos al nodo maestro indicando su puerto

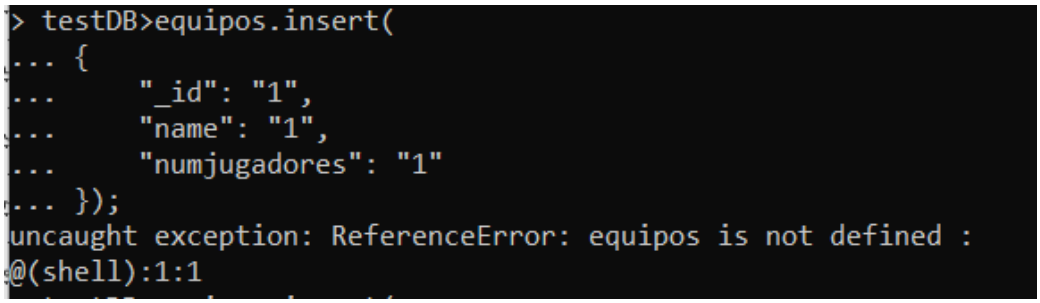


Nos conectamos a la base de datos para iniciar la replicación

Verificamos la creación y el estado del nodo

2. Disponibilidad

Ingresar documentos en las colecciones en el nodo maestro y verificar que todas las instancias tienen una réplica de los registros insertados



Se inicia a hacer inserciones, presentando alguna falla en la sintaxis

```
> testDB.Autores.insert(
... {
... "Id_autor": "0001",
... "nombre_autor": "William",
... "Genero": "literatura"
... });
WriteResult({ "nInserted" : 1 })
```

Se hacen inserciones con datos aleatorios para verificar el almacenamiento

```
> testDB.equipos.insert(
...   {
...   "Id_autor": "0001",
...   "nombre_autor": "William",
...   "Genero": "literatura"
...   });
WriteResult({ "nInserted" : 1 })
```

Verificamos el almacenamiento de los mismo por medio de find()

```
> testDB.Autores.find()
{ "_id" : ObjectId("656d454f2be999d185ad5680"), "Id_autor" : "0001", "nombre_autor" : "William", "Genero" : "literatura" }
{ "_id" : ObjectId("656d45702be999d185ad5681"), "Id_autor" : "0001", "nombre_autor" : "William", "Genero" : "literatura" }
> testDB.equipos.find()
{ "_id" : ObjectId("656d45ba2be999d185ad5682"), "Id_autor" : "0001", "nombre_autor" : "William", "Genero" : "literatura" }
```

3. Tolerancia a Fallos

Se ejecuta el comando shutdown para hacer desconexión del nodo primario y realizar la promoción de alguno de los secundarios a primario

```
> primaryDB.adminCommand({shutdown  : 1})
uncaught exception: Error: error doing query: failed: network error while at
DB.prototype.runCommand@src/mongo/shell/db.js:169:19
DB.prototype.adminCommand@src/mongo/shell/db.js:187:12
@(shell):1:1
```

Se genera la caída del nodo principal

```
> connNewPrimary = new Mongo("localhost:20001")
connection to localhost:20001
> newPrimaryDB = connNewPrimary.getDB("Futbolact_1")
Futbolact_1
> newPrimary.isMaster()
uncaught exception: ReferenceError: newPrimary is not defined :
@(shell):1:1
> newPrimaryDB.isMaster()
{
        "topologyVersion" : {
                "processId" : ObjectId("656d3a80f78494e5cbab9b2a"),
                "counter" : NumberLong(7)
        },
        "hosts" : [
                "DESKTOP-IT871UI:20000",
                "DESKTOP-IT871UI:20001",
                "DESKTOP-IT871UI:20002"
        ],
        "setName" : "MireplicaSet",
        "setVersion" : 3,
        "ismaster" : true,
        "secondary" : false,
        "primary" : "DESKTOP-IT871UI:20001",
        "me" : "DESKTOP-IT871UI:20001",
        "electionId" : ObjectId("7fffffff0000000000000002"),
        "lastWrite" : {
                "opTime" : {
                        "ts" : Timestamp(1701661654, 1),
                        "t" : NumberLong(2)
                },
                "lastWriteDate" : ISODate("2023-12-04T03:47:34Z"),
                "majorityOpTime" : {
```

Se hace la promoción de un nodo secundario como primario y se demuestra la disponibilidad de este

4. Disponibilidad

Verificar cual de los nodos secundarios quedó promovido como secundario

```
> newPrimaryDB.isMaster()
{
        "topologyVersion" : {
                "processId" : ObjectId("656d3a80f78494e5cbab9b2a"),
                "counter" : NumberLong(7)
        },
        "hosts" : [
                "DESKTOP-IT871UI:20000",
                "DESKTOP-IT871UI:20001",
                "DESKTOP-IT871UI:20002"
        ],
        "setName" : "MireplicaSet",
        "setVersion" : 3,
        "ismaster" : true,
        "secondary" : false,
        "primary" : "DESKTOP-IT871UI:20001",
        "me" : "DESKTOP-IT871UI:20001",
        "electionId" : ObjectId("7fffffff0000000000000002"),
        "lastWrite" : {
                "opTime" : {
                        "ts" : Timestamp(1701661654, 1),
                        "t" : NumberLong(2)
                },
                "lastWriteDate" : ISODate("2023-12-04T03:47:34Z"),
                "majorityOpTime" : {
                        "ts" : Timestamp(1701661654, 1),
                        "t" : NumberLong(2)
                },
                "majorityWriteDate" : ISODate("2023-12-04T03:47:34Z")
        },
        "maxBsonObjectSize" : 16777216,
```

Se demuestra que ahora el nodo maestro quedó alojado en el puerto 20001

Link de github

https://github.com/SebastianVacca/BD.git