

2024

Bases de datos avanzadas NO-SQL

LIZETH PAOLA BUITRAGO QUINTERO,

IBEROAMERICANO

Replicas para mongo 1. Objetivo

El objetivo de este documento es establecer los criterios de calidad en cuanto a la redundancia y disponibilidad 24x7 para el sistema de gestión de participantes en un torneo deportivo. Estos criterios garantizarán un funcionamiento confiable y continuo del sistema, minimizando los tiempos de inactividad y asegurando la integridad de los datos en todo momento.

2. Criterios de Redundancia

- Los datos del sistema deben estar respaldados y almacenados en múltiples ubicaciones para garantizar la redundancia.
- Se implementará una estrategia de replicación en conjunto de réplicas con al menos 3 nodos para asegurar la disponibilidad de los datos.
- La replicación debe ser asíncrona para minimizar la latencia en la propagación de los cambios entre los nodos.
- Se deben realizar copias de seguridad periódicas de los datos replicados para facilitar la recuperación ante posibles fallas.

3. Criterios de Disponibilidad 24x7

- El sistema debe estar disponible las 24 horas del día, los 7 días de la semana, sin interrupciones planificadas para mantenimientos o actualizaciones.
- Se debe implementar un monitoreo constante del sistema para detectar y resolver de manera proactiva cualquier incidencia que pueda afectar la disponibilidad.
- Se deben establecer procedimientos de recuperación ante desastres para minimizar el tiempo de inactividad en caso de fallos del sistema o eventos imprevistos.
- La arquitectura del sistema debe ser escalable y capaz de gestionar cargas de trabajo variables sin comprometer la disponibilidad.

4. Consideraciones adicionales

- Se debe establecer un plan de contingencia para hacer frente a posibles interrupciones del servicio y garantizar una rápida recuperación.
- Se debe realizar un monitoreo constante de los recursos del sistema, como el rendimiento de los servidores, el consumo de almacenamiento y la utilización de la red, para prevenir posibles cuellos de botella y mantener un rendimiento óptimo.

Estos criterios de calidad en cuanto a la redundancia y disponibilidad 24x7 serán la base para el diseño e implementación del sistema de gestión de participantes en el torneo deportivo, asegurando que los datos estén protegidos, disponibles y accesibles en todo momento.

Creación del Replicaset

- `MiejReplicaSet = new ReplSetTest ({name: "MireplicaSet", nodes: 3})`
- `MiejReplicaSet = new ReplSetTest ({name: "MireplicaSet", nodes: 3}); print("hecho")`

```

    var master;
    assert.soonNoExcept(function() {
        return (master = self.getPrimary());
    }, "waiting for master", timeout);

    return master;
},
"name" : "MireplicaSet",
"useHostName" : true,
"host" : "LAPTOP-0C1A1M90",
"oplogSize" : 40,
"useSeedList" : false,
"keyFile" : undefined,
"protocolVersion" : undefined,
"waitForKeys" : undefined,
"nodeOptions" : {
    "n0" : undefined,
    "n1" : undefined,
    "n2" : undefined
},
"nodes" : [ ],
"ports" : [
    20000,
    20001,
    20002
]
}
}
> MieiReplicaSet = new ReplSetTest ({name: "MireplicaSet", nodes: 3}); print("hecho")
Starting new replica set MireplicaSet
hecho

```

Arrancar los procesos mongod de la replica

- MieiReplicaSet.startSet()

Arrancar el proceso de replica

- MieiReplicaSet.initiate()

Este equipo > Windows (C:) > data > db >					Buscar en db
Ordenar ▾ Ver ▾ ...					Detalles
Nombre	Fecha de modificación	Tipo	Tamaño		
journal	12/06/2023 16:09	Carpeta de archivos			
diagnostic.data	12/06/2023 16:19	Carpeta de archivos			
_unknown_name_-configRS-0	14/06/2023 17:21	Carpeta de archivos			
_unknown_name_-configRS-1	14/06/2023 17:21	Carpeta de archivos			
_unknown_name_-configRS-2	14/06/2023 17:21	Carpeta de archivos			
_unknown_name_-rs0-0	14/06/2023 17:21	Carpeta de archivos			
_unknown_name_-rs1-0	14/06/2023 17:21	Carpeta de archivos			
_unknown_name_-rs2-0	14/06/2023 17:21	Carpeta de archivos			
MireplicaSet-0	31/05/2024 22:50	Carpeta de archivos			
MireplicaSet-1	31/05/2024 22:50	Carpeta de archivos			
MireplicaSet-2	31/05/2024 22:50	Carpeta de archivos			

Prueba del grupo de replica

- conn=new Mongo("Metal2022:20003")

```

: NumberLong(1) }
ReplSetTest awaitReplication: checking secondary #0: LAPTOP-0C1A1M90:20004
ReplSetTest awaitReplication: secondary #0, LAPTOP-0C1A1M90:20004, is synced
ReplSetTest awaitReplication: checking secondary #1: LAPTOP-0C1A1M90:20005
ReplSetTest awaitReplication: secondary #1, LAPTOP-0C1A1M90:20005, is synced
ReplSetTest awaitReplication: finished: all 2 secondaries synced at optime { "ts" : Timestamp(1717213563, 1), "t" : NumberLong(1) }
d20003| 2024-05-31T22:46:03.980-0500 I COMMAND [conn1] Received replSetStepUp request
AwaitNodesAgreeOnPrimary: Waiting for nodes to agree on any primary.
d20003| 2024-05-31T22:46:03.980-0500 I ELECTION [conn1] Not starting an election for a replSetStepUp request, since we are not electable due to: Not standing for election again; already primary
AwaitNodesAgreeOnPrimary: Nodes agreed on primary LAPTOP-0C1A1M90:20003

[jsTest] ----
[jsTest] ReplSetTest stepUp: Finished stepping up LAPTOP-0C1A1M90:20003
[jsTest] ----

> conn=new Mongo("LAPTOP-0C1A1M90:20003")
d20003| 2024-05-31T22:46:42.510-0500 I NETWORK [listener] connection accepted from 192.168.3.5:58855 #27 (8 connections now open)
d20003| 2024-05-31T22:46:42.510-0500 I NETWORK [conn27] received client metadata from 192.168.3.5:58855 conn27: { application: { name: "MongoDB Shell" }, driver: { name: "MongoDB Internal Client", version: "4.2.24" }, os: { type: "Windows", name: "Microsoft Windows 10", architecture: "x86_64", version: "10.0 (build 22631)" } }
connection to LAPTOP-0C1A1M90:20003
> d20003| 2024-05-31T22:47:01.302-0500 I CONNPPOOL [Replication] Ending idle connection to host LAPTOP-0C1A1M90:20005 because the pool meets constraints; 1 connections to that host remain open
d20005| 2024-05-31T22:47:01.302-0500 I NETWORK [conn3] end connection 192.168.3.5:58828 (3 connections now open)

```

- testDB=conn.getDB("JUGADORES")
- testDB.isMaster()

```

> db.Entrenadores.insertOne({
...   nombre: "Jurgen Klopp",
...   edad: 54,
...   equipo: "Liverpool"
... })
{
  "acknowledged" : true,
  "insertedId" : ObjectId("64741aaea8f6c1fd7d838cbb")
}
> db.Entrenadores.insertOne({
...   nombre: "Jurgen Klopp",
...   edad: 54,
...   equipo: "Liverpool"
... })
{
  "acknowledged" : true,
  "insertedId" : ObjectId("64741af3a8f6c1fd7d838cbc")
}

```

- testDB.ENTRENADORES.count()

Comprobación de la réplica sobre los nodos secundarios

- connSecondary = new Mongo("LAPTOP-0C1A1M9O:20007 ")
- secondaryTestDB = connSecondary.getDB("Entrenadores")
- secondaryTestDB.isMaster()
- secondaryTestDB.Entrenadores.count();
- secondaryTestDB.Entrenadores.findOne()

Detener el nodo primario

- connPrimary = new Mongo("localhost:20003")
- primaryDB = connPrimary.getDB("Entrenadores")
- primaryDB.isMaster()

Comprobación del nuevo nodo primario

- connNewPrimary = new Mongo("localhost:20003")
- newPrimaryDB = connNewPrimary.getDB("Biblioteca")
- newPrimaryDB.isMaster()

Detener el ReplicaSet de pruebas

- MieiReplicaSet.stopSet()

Casos de pruebas en replicación bajo MongoDB

1. **Caso de Prueba:** Verificar la creación exitosa de las colecciones.
 - **Descripción:** Se verifica que las colecciones se creen correctamente en la base de datos.

- **Pasos:**
 1. Comprobar que las colecciones "Jugadores", "Entrenadores", "Árbitros", "Encuentros", "Resultados" y "TablaPosiciones" existan en la base de datos.
- **Resultado Esperado:** Todas las colecciones deben existir en la base de datos.
- 2. **Caso de Prueba:** Verificar la creación exitosa de los índices.
 - **Descripción:** Se verifica que los índices se creen correctamente en las colecciones correspondientes.
 - **Pasos:**
 1. Comprobar que los índices "equipo" existan en las colecciones "Jugadores", "Entrenadores" y "TablaPosiciones".
 - **Resultado Esperado:** Los índices "equipo" deben existir en las colecciones correspondientes.
- 3. **Caso de Prueba:** Verificar la inserción de jugadores.
 - **Descripción:** Se verifica que los jugadores se inserten correctamente en la colección "Jugadores".
 - **Pasos:**
 1. Insertar varios documentos de jugadores en la colección "Jugadores".
 2. Comprobar que los documentos se hayan insertado correctamente.
 - **Resultado Esperado:** Los documentos de jugadores deben estar presentes en la colección "Jugadores".
- 4. **Caso de Prueba:** Verificar la asignación de entrenadores a equipos.
 - **Descripción:** Se verifica que los entrenadores se asignen correctamente a los equipos en la colección "Entrenadores".
 - **Pasos:**
 1. Insertar varios documentos de entrenadores en la colección "Entrenadores" con el campo "equipo" adecuado.
 2. Comprobar que los documentos se hayan insertado correctamente y contengan el campo "equipo" correspondiente.
 - **Resultado Esperado:** Los documentos de entrenadores deben estar presentes en la colección "Entrenadores" y contener el campo "equipo" correctamente asignado.

Evidencias Github

Link: https://github.com/isabeldc13008/Bases_de_datos_avanzadas-NO-SQL

https://github.com/Lizeth-Buitrago/Bases-de-datos-avanzadas-No_SQL/upload/main

Evidencia Video

**Actividad 2 - Conceptos y comandos básicos de la replicación en bases de datos
NoSQL-20240531_224202-Grabación de la reunión.mp4**