Corporación Universitaria Iberoamericana

Facultad de Ingeniería de Software

Bases de datos avanzadas (28102024_C2_202434)

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

Profesor:
Jorge Castañeda

Estudiantes:
Saulo Andres Franco Torres

1 de diciembre de 2024

Introducción.

El presente documento describe la implementación de una solución de bases de datos para la gestión de un torneo de fútbol regional en formato de liga. Este sistema tiene como objetivo proporcionar un soporte eficiente y seguro para la administración de equipos, jugadores, entrenadores, árbitros y encuentros, con la posibilidad de generar informes detallados sobre el desempeño del torneo. El diseño propuesto se basa en el uso de **MongoDB**, una base de datos NoSQL, que ofrece flexibilidad y escalabilidad, características esenciales para el almacenamiento de datos en entornos dinámicos como los de un torneo deportivo.

El sistema está estructurado en varias colecciones, cada una destinada a almacenar diferentes tipos de datos relacionados con el torneo, tales como los detalles de los equipos, los jugadores, los entrenadores, los árbitros y los encuentros. Además, se contempla una funcionalidad para el seguimiento de los resultados de los partidos, la generación de la tabla de posiciones y la elaboración de estadísticas de jugadores. En este contexto, la alta disponibilidad y la redundancia son aspectos cruciales, especialmente considerando la necesidad de un servicio que esté disponible **24x7**.

Para cumplir con estos requerimientos, se propone implementar una **replicación de bases de datos** mediante un conjunto de réplicas de **MongoDB**, lo que garantiza la disponibilidad continua de los datos incluso en el caso de fallos en algunos de los nodos del sistema. A lo largo de este documento, se explicarán los pasos necesarios para configurar el entorno de replicación, incluyendo la creación de nodos, la configuración de un conjunto de réplicas, el uso de **SCRAM-SHA-256** para la autenticación segura y la verificación de la correcta implementación de la replicación.

Asimismo, se detallarán los procedimientos para la creación y verificación de pruebas de redundancia y disponibilidad, asegurando que el sistema pueda manejar fallos sin interrumpir la experiencia de los usuarios. Finalmente, se proporcionará información adicional sobre la configuración del sistema, incluyendo la creación de índices, la configuración de seguridad, y cómo realizar consultas y obtener estadísticas del torneo en tiempo real.

Documento de Requerimientos No Funcionales

Título: Criterios de calidad para redundancia y disponibilidad 24x7

Objetivo General
Hay que asegurar que la base de datos del torneo de fútbol regional proporcione un servicio confiable, ininterrumpido y capaz de manejar fallos en nodos sin pérdida de datos o interrupciones notables para los usuarios.

1. Redundancia

Descripción:
El sistema debe garantizar la existencia de múltiples copias de los datos distribuidas en varios nodos, de modo que un fallo en uno o más nodos no comprometa la operación del sistema.

Requisitos específicos:
1. Los datos deben replicarse automáticamente entre al menos tres nodos del sistema.
2. Debe implementarse un *Replica Set* en MongoDB, con:
    - Un nodo primario que maneje las escrituras.
    - Dos nodos secundarios que actúen como copias sincronizadas del primario.
3. Configuración de un *hidden node* opcional para respaldos o pruebas sin afectar el rendimiento del sistema en producción.

Métricas de calidad:
- **RPO (Recovery Point Objective):** Los datos deben replicarse con un retraso máximo de 1 segundo en los nodos secundarios.
- **Tolerancia a fallos:** El sistema debe soportar la caída simultánea de un nodo sin interrumpir la operación.

2. Disponibilidad 24x7

Descripción:
La base de datos debe estar operativa en todo momento, con mecanismos automáticos que garanticen la continuidad del servicio, incluso en caso de fallos.

Requisitos específicos:
1. Configuración de *failover* automático:
    - En caso de caída del nodo primario, uno de los nodos secundarios debe ser promovido automáticamente como nuevo primario.
2. Los clientes deben redirigirse automáticamente al nodo primario activo en caso de cambios en el estado de los nodos.

3. Despliegue distribuido en diferentes servidores físicos o ubicaciones para garantizar alta disponibilidad.

Métricas de calidad:
- **SLA de disponibilidad:** 99.95% de tiempo de actividad anual.
- **Tiempo de recuperación (RTO - Recovery Time Objective):** El failover debe completarse en menos de 10 segundos.

3. Escalabilidad

Descripción:
El sistema debe ser capaz de escalar horizontalmente mediante la adición de nuevos nodos secundarios o configuraciones de *sharding* si el volumen de datos aumenta.

Requisitos específicos:
1. Permitir agregar nodos secundarios al *Replica Set* sin detener el servicio.
2. Configurar *arbiter nodes* si es necesario para optimizar la gestión de quórum.

Métricas de calidad:
- Tiempo para agregar un nuevo nodo: Menos de 30 minutos.
- Impacto en el rendimiento durante la sincronización de un nuevo nodo: No mayor al 5%.

4. Consistencia

Descripción:
El sistema debe garantizar que las operaciones de escritura sean consistentes, mientras que las lecturas pueden ser configuradas para utilizar consistencia eventual.

Requisitos específicos:
1. Las escrituras deben realizarse siempre en el nodo primario.
2. Las lecturas de datos críticos deben realizarse desde el nodo primario para garantizar consistencia.
3. Las lecturas de datos no críticos pueden realizarse desde los nodos secundarios para balancear la carga.

Métricas de calidad:
- **Latencia de escritura:** Menor a 50 ms en operaciones estándar.
- **Retraso de replicación:** Menor a 1 segundo en nodos secundarios.

5. Seguridad

Descripción:
El sistema debe proteger la información del torneo y garantizar que solo usuarios autorizados puedan acceder a los datos.

Requisitos específicos:

1. Configurar autenticación mediante *SCRAM-SHA-256* en todos los nodos del *Replica Set*.
2. Asegurar la conexión entre nodos mediante cifrado TLS/SSL.
3. Implementar roles y permisos para limitar el acceso a los datos.

Métricas de calidad:

- Tiempo de configuración de un nuevo nodo seguro: Menor a 15 minutos.
- Pruebas de acceso no autorizado: 0 incidentes exitosos.

6. Tolerancia a Desastres

Descripción:
En caso de desastres que afecten una ubicación completa, el sistema debe continuar operativo desde otros nodos en ubicaciones separadas.

Requisitos específicos:

1. Desplegar nodos secundarios en diferentes ubicaciones geográficas.
2. Configurar respaldos automáticos en un nodo oculto o servicio de almacenamiento externo.

Métricas de calidad:

- Pérdida máxima de datos en caso de desastre: 1 segundo.
- Tiempo de recuperación total (RTO): Menor a 30 minutos.

7. Monitoreo

Descripción:
El sistema debe contar con herramientas de monitoreo para supervisar el estado de los nodos y recibir alertas en caso de fallos.

Requisitos específicos:

1. Implementar herramientas como *MongoDB Atlas* o *Prometheus* para monitoreo.
2. Configurar alertas automáticas para eventos como:
    - Pérdida de conexión con un nodo.
    - Retraso en la replicación mayor a 1 segundo.
    - Fallo en la promoción de un nodo secundario a primario.

Métricas de calidad:

- Frecuencia de alertas por eventos críticos: Inmediata (< 5 segundos).
- Tiempo de respuesta a problemas detectados: Menor a 15 minutos.

Configuración del Entorno de Replicación en MongoDB

Estrategia de replicación en MongoDB

Objetivo:
Configurar el *Replica Set* en MongoDB con un nodo primario y dos secundarios, utilizando MongoDB para la replicación y su estado.

Creación de directorios para nodos.

En el CMD crearemos los 3 directorios para la ubiaccion de los nodos.

```
md c:\mongodb\repset\rs1
md c:\mongodb\repset\rs2
md c:\mongodb\repset\rs3
```

Inicialización de los nodos.

- Iniciamos cada nodo de MongoDB con el parámetro --replSet desde la línea de comandos para permitir la configuración de un *Replica Set*.

```
start mongod --bind_ip localhost --dbpath c:\mongodb\repset\rs1 --port 20001 --replSet myrs

start mongod --bind_ip localhost --dbpath c:\mongodb\repset\rs2 --port 20002 --replSet myrs

start mongod --bind_ip localhost --dbpath c:\mongodb\repset\rs3 --port 20003 --replSet myrs
```

Para verificar si MondoDB esta iniciando correctamente en alguno de estos puertos debemos introducir el comando:

```
netstat -ano | findstr :20001
```

Inicializar nodo principal:

En cada servidor, ejecuta lo siguiente en la terminal (reemplaza <IP_DEL_NODO> con las IP reales de cada nodo):

mongod --replSet rs0 --port 27017 --dbpath c:\data\db --bind_ip localhost,127.0.0.1:20001

Inicializar el *Replica Set* desde la shell de MongoDB:

Iniciamos la Shell de mongoDB desde el nodo principal con:

mongosh --host localhost --port 20001

- Una vez que esté conectado a uno de los nodos desde MongoDB , accedo a la *Mongo Shell* y ejecuta el siguiente comando para inicializar el *Replica Set*.

```
rs.initiate({
   _id: "myrs",
   members: [
      { _id: 0, host: "127.0.0.1:20001" },
      { _id: 1, host: "127.0.0.1:20002" },
      { _id: 2, host: "127.0.0.1:20003" }
   ]
})
```

Este comando inicia el *Replica Set* y agrega los nodos a la configuración.

5. Verificar el estado del Replica Set: Una vez configurado, puedes verificar el estado del *Replica Set* desde la shell de MongoDB con el siguiente comando:

rs.status();

Esto devolverá el estado actual del *Replica Set*, mostrando cuál nodo es primario y cuáles son secundarios.

Pasos para confirmar funcionamiento de nodos

Ingresar a la base de datos desde alguno de los nodos:

Ejemplo: en el nodo primario 20001

mongosh --host localhost --port 20001

1. **Selecciona la base de datos**

Usa el comando use <nombre_de_base_de_datos> para seleccionar la base de datos en la que deseas insertar la colección. Si la base de datos no existe, se creará automáticamente cuando insertes un documento.

En este caso:

use torneo_deportes;

2. **Inserta un documento en la colección**

Usar el método insertOne() para insertar un documento en la colección. Si la colección no existe, se creará automáticamente.

```
db.jugadores.insertOne
({
_id:"jugador_1",
nombre:"Cristiano Ronaldo",
edad:"20",
posicion:"delantero",
_id_equipo:"001",
goles:"9",
asistencias:"6",
 }
 )
```

O bien, si deseas insertar varios documentos a la vez, puedes usar insertMany():

```
db.jugadores.insertMany
(

{
```

```
_id:"jugador_1",
nombre:"Cristiano Ronaldo",
edad:"20",
posicion:"delantero",
_id_equipo:"001",
goles:"9",
asistencias:"6",
 },
{
_id:"jugador_2",
nombre:"Roberto Carlos",
edad:"32",
posicion:"medio campo",
_id_equipo:"001",
goles:”14",
asistencias:"8",
}
]
)
```

3. Consulta una colección específica
Usa el método find() para ver los documentos dentro de una colección específica. Por ejemplo, si quieres ver los documentos de la colección equipos:

```
db.jugadores.find();
```

Esto mostrará todos los documentos de la colección.

4. cerraremos mondo y lo abriremos nuevamente en un nodo segundario.
Cerraremos y abriremos nuevamente mondo en uno de los nodos segundarios y realizaremos la tarea de búsqueda de colección para confirmar que si esta funcionando el Replica Set.

5. realizar prueba CRUD y confirmar cambios en los nodos segundarios.
Realizaremos pruebas de CRUD en el nodo principal y se verificara que todos los cambios realizados se reflejen en los nodos segundarios para reporte final de la data y confirmar los resultados obtenidos.

Actualizar:

```
db.jugadores,updateMany
(
{nombre:”Cristiano Ronaldo”}, {$set:{goles:17}}
)
```

Borrar:

db.jugadores.deleteOne
(
{nombre:”Roberto Carlos”}
)

Mostrar las colecciones de la base de datos jugadores para confirmar los cambios en el nodo principal y segundarios con el comando find();

Crear y Elimina la colección
Usando los siguientes comandos:

db.createCollections (“nombre de colección”); //Crea una coleccion
db.equipos.drop(); *// Elimina la colección 'equipos'*

LINK DEL REPOSITORIO EN GITHUB

https://github.com/ZauloFranco/Actividad-2---Conceptos-y-comandos-b-sicos-de-la-replicaci-n-en-bases-de-datos-NoSQL

Referencias.

- libro Aramburu Cabo, M. J. y Sanz Blasco, I. (2012). Bases de datos avanzadas. D - Universitat Jaume I. Servei de Comunicació i Publicacions, https://elibro.net/es/lc/biblioibero/titulos/51741

- Capítulo 7 (Replicación) del libro de Sarasa, A. (2016) Introducción a las bases de datos NoSQL usando MongoDB. Editorial UOC, disponible en: https://elibro.net/es/lc/biblioibero/titulos/58524