Actividad 2

Conceptos y comandos básicos de la replicación en bases de datos NoSQL

Yesid Alexander Jiménez Vivas

Corporación Universitaria Iberoamericana
Facultad de Ingeniería
Bases de datos avanzadas

Ingeniería de Software
Mtro. Jorge Castañeda

Bogotá, Colombia
25 de noviembre de 2024

INTRODUCCIÓN

Este documento define los criterios de calidad no funcionales relacionados con la redundancia y disponibilidad 24x7 para el sistema de gestión de un torneo deportivo universitario, utilizando una base de datos MongoDB. Se establecerán las estrategias necesarias para garantizar que el sistema pueda operar de manera ininterrumpida, incluso en caso de fallos en algunos de los nodos del sistema, y que los datos estén siempre disponibles y accesibles.

REQUERIMIENTOS NO FUNCIONALES

Redundancia

Se debe garantizar que el sistema siga funcionando correctamente incluso si uno o más nodos fallan, mediante la duplicación de los datos en varios servidores.

Criterios:

- Los datos deberán ser replicados entre al menos tres nodos en diferentes servidores.
- En caso de fallo de uno de los nodos, el sistema debe poder seguir operando sin pérdida de datos ni interrupciones del servicio, utilizando los nodos restantes.

- La estrategia de replicación debe ser asíncrona para asegurar que los datos se actualicen en todos los nodos, pero sin afectar el rendimiento de las operaciones de lectura/escritura.

Disponibilidad 24x7

Se debe asegurar que el sistema esté disponible para los usuarios en todo momento, sin períodos de inactividad, incluso en caso de fallos de hardware o de red.

Criterios:

- El sistema debe ofrecer una disponibilidad mínima del 99.9%.
- Debe haber mecanismos de conmutación por error (o failover) para redirigir automáticamente las solicitudes a los nodos disponibles en caso de que uno falle.
- El sistema debe ser capaz de manejar actualizaciones de datos en tiempo real sin interrumpir la disponibilidad de los mismos.
- Los datos deben ser accesibles siempre, sin necesidad de mantenimiento programado prolongado que afecte la disponibilidad.

Escalabilidad

Se debe asegurar que el sistema pueda escalar fácilmente en términos de almacenamiento y rendimiento a medida que el volumen de datos y usuarios crece.

Criterios:

- La replicación debe ser escalable de modo que se puedan añadir más nodos al sistema sin afectar el rendimiento.
- La estrategia de replicación debe permitir el crecimiento del sistema a medida que se añaden nuevos eventos deportivos o nuevos participantes.

Seguridad

Se debe asegurar que el sistema de base de datos cumpla con los requisitos de seguridad y privacidad de datos aplicables como por ejemplo el RGPD (el Reglamento General de Protección de Datos) o HIPAA (la Ley de Responsabilidad y Transferibilidad de Seguros Médicos).

Criterios:

- Únicamente los usuarios autorizados accedan a la información por medio de sistemas de autenticación y autorización.
- El sistema debe ser capas de auditar y registrar las actividades de los usuarios, como la creación, modificación y eliminación de datos con el fin de prevenir y detectar posibles incidentes de seguridad.
- Los datos ingresados al sistema de base de datos deben cifrarse durante el tránsito y el reposo de la información para garantizar su integridad y confidencialidad.

ESTRATEGIA DE REPLICACIÓN

Definición de la Estrategia de Replicación

Para cumplir con los requerimientos de **redundancia** y **disponibilidad 24x7**, se utilizará una replicación de tipo **primaria-secundaria** en MongoDB.

- **Replica Set:** Se configurará un Replica Set que consistirá en 3 nodos:
  - **Nodo Primario (Primary):** Recibe todas las escrituras y actualizaciones de los datos.
  - **Nodos Secundarios (Secondary):** Replican los datos del nodo primario. Pueden servir para consultas de lectura, lo que mejora el rendimiento.
  - **Nodo Árbitro (Arbiter):** Este es opcional, pero ayuda a mantener la mayoría de votos en caso de partición de red para decidir qué nodo debe ser primario.

Configuración de la replicación:

- Se creará un Replica Set con tres nodos MongoDB distribuidos en diferentes servidores físicos o virtuales para garantizar la alta disponibilidad.
- La replicación será sincrónica para garantizar la consistencia de los datos entre nodos.

Comandos para el entorno de replicación

1. **Configurar las Carpetas de Datos**

   Se debe crear tres carpetas para almacenar los datos de cada instancia de MongoDB, para ello se hace uso de la consola de comandos de Windows (CMD):

   - mkdir C:\data\db1
   - mkdir C:\data\db2
   - mkdir C:\data\db3

2. **Inicia tres instancias de MongoDB**

   Ejecuta los siguientes comandos en tres terminales diferentes para iniciar cada instancia con un puerto y carpeta de datos distintos:

   - mongod --port 27017 --dbpath C:\data\db1 --replSet "rs0"
   - mongod --port 27018 --dbpath C:\data\db2 --replSet "rs0"
   - mongod --port 27019 --dbpath C:\data\db3 --replSet "rs0"

3. **Conectar a nodo primario**

   En una nueva terminal se debe ejecutar el siguiente comando

   - mongo --port 27017

4. **Configura el Replica Set**

   Dentro del shell de MongoDB, inicializa el Replica Set:

```
rs.initiate({
   _id: "rs0",
   members: [
      { _id: 0, host: "localhost:27017" },
      { _id: 1, host: "localhost:27018" },
      { _id: 2, host: "localhost:27019" }
   ]
})
```

5. **Verifica el Estado del Replica Set**

   Para asegurase de que los nodos estén conectados y funcionando correctamente se ejecuta el siguiente comando:

   - rs.status()

6. **Agregar un Árbitro**

   Este punto es opcional. Se hace con el fin de garantizar decisiones rápidas en caso de fallo, se puede agregar un nodo árbitro:

   - rs.addArb("localhost:27020")

   Se debe repetir los pasos del punto uno, en este caso para crear la carpeta el árbitro, después ejecutar el siguiente comando:

   - mongod --port 27020 --dbpath C:\data\arb --replSet "rs0"

CASOS DE PRUEBA

Casos de Prueba para Verificación de Redundancia y Disponibilidad 24x7

Caso de Prueba 1:

Comprobación de la Disponibilidad en Caso de Fallo de Nodo Primario

Verificar que el sistema siga funcionando sin interrupciones cuando el nodo primario falla.

Procedimiento:

- Iniciar el Replica Set con tres nodos.
- Realizar una escritura en el nodo primario.
- Detener el nodo primario y verificar que uno de los nodos secundarios se convierte en primario.
- Asegurarse de que las escrituras pueden continuar en el nuevo nodo primario.

Resultado Esperado: El sistema debe seguir operativo sin pérdida de datos, y el nuevo nodo primario debe aceptar las escrituras.

Caso de Prueba 2:

Comprobación de Redundancia y Replicación de Datos

Verificar que los datos se replican correctamente entre los nodos secundarios.

Procedimiento:

- Iniciar el Replica Set con tres nodos.
- Realizar varias escrituras en el nodo primario.
- Verificar que las escrituras se replican en los nodos secundarios.
- Detener uno de los nodos secundarios y verificar que el sistema sigue funcionando.

Resultado Esperado: Los datos deben estar disponibles en al menos dos nodos, incluso si uno de los nodos secundarios falla.

Caso de Prueba 3:

Prueba de Disponibilidad 24x7

Verificar que el sistema está disponible durante 24 horas, incluso con fallos en algunos nodos.

Procedimiento:

- Realizar operaciones de lectura y escritura durante un período prolongado (24 horas).
- Detener y reiniciar varios nodos durante este período.
- Comprobar que las operaciones continúan sin interrupciones.

Resultado Esperado: El sistema debe ser capaz de manejar las operaciones de manera continua sin caídas o períodos prolongados de inactividad.

EJECUCIÓN DE LOS CASOS DE PRUEBA Y ANÁLISIS DE RESULTADOS

Casos de Prueba Ejecutados

Caso de Prueba	Descripción	Resultado Esperado	Resultado Obtenido	Estado
Prueba 1: Lectura de Datos	Consultar los datos desde el nodo primario.	Los datos del torneo deben estar disponibles y mostrarse correctamente.	Los datos se mostraron correctamente.	**Aprobado**

Prueba 2: Fallo del Nodo Primario	Simular la desconexión del nodo primario y verificar el failover automático.	Uno de los nodos secundarios debe promoverse como primario automáticamente.	Uno de los nodos secundarios fue promovido como primario en menos de 5 segundos.	**Aprobado**
Prueba 3: Escritura en el Nodo Secundario	Intentar escribir datos en un nodo secundario.	El sistema debe rechazar la operación de escritura en un nodo secundario.	El intento de escritura fue rechazado, mostrando un error esperado.	**Aprobado**
Prueba 4: Restauración del Nodo Primario	Reconectar el nodo primario original y verificar la sincronización de datos.	El nodo original debe reintegrarse al Replica Set y sincronizar los datos automáticamente.	El nodo se reintegró y sincronizó los datos correctamente.	**Aprobado**
Prueba 5: Consistencia de Datos	Comparar los datos entre todos los nodos del Replica Set después de la sincronización.	Todos los nodos deben contener los mismos datos (consistencia eventual).	Los datos en todos los nodos coincidieron perfectamente después de la sincronización.	**Aprobado**

Análisis de Resultados

1. **Redundancia Verificada:**

   La replicación asegura que los datos se copian a todos los nodos secundarios. Esto garantiza que incluso si un nodo falla, los datos permanecen disponibles.

2. **Disponibilidad 24x7:**

   Las pruebas demostraron que el sistema se mantiene disponible incluso ante la falla del nodo primario, cumpliendo con el requerimiento de alta disponibilidad.

3. **Consistencia Eventual:**

   Los datos fueron sincronizados correctamente entre los nodos después de la reconexión de un nodo fallido. Esto valida la estrategia de replicación adoptada.

4. **Limitaciones Detectadas:**
    - Las operaciones de escritura solo son posibles en el nodo primario.
    - Durante el failover, hubo un retraso mínimo (3-5 segundos), pero esto es aceptable dentro de los estándares de tolerancia.
5. **Recomendaciones:**
    - Implementar monitoreo continuo del Replica Set para detectar fallos de manera proactiva.
    - Configurar alertas para notificar al equipo técnico cuando se produzcan failovers.

CONCLUSIÓN

El mecanismo de replicación implementado cumple con los requisitos de redundancia y disponibilidad 24x7 para la gestión del torneo deportivo. Todas las pruebas realizadas fueron exitosas, confirmando que la estrategia garantiza la continuidad del sistema ante fallos.

La implementación de un sistema de gestión para el torneo deportivo en MongoDB, utilizando técnicas avanzadas de replicación y asegurando criterios de redundancia y disponibilidad, demuestra la capacidad de las bases de datos NoSQL para satisfacer necesidades críticas de almacenamiento y procesamiento de datos en escenarios dinámicos y exigentes.

REFERENCIAS

Sarasa, A. (2016). Introducción a las bases de datos NoSQL usando MongoDB. Editorial UOC. https://elibro.net/es/lc/biblioibero/titulos/58524.