Bases de Datos Avanzadas

Actividad 3 - Conceptos y Comandos básicos del particionamiento en bases de datos NoSQL

Materia: Bases de Datos Avanzadas (28102024_C2_202434)

Estudiante: Lizeth Valentina Benitez ID de Estudiante: 100173953

Profesor: Jorge Isaac Castañeda Valbuena

Fecha: Diciembre de 2024

Introducción

En el contexto del caso práctico relacionado con la organización y gestión de un evento deportivo, se plantea la implementación de una base de datos NoSQL para manejar los datos generados por las diferentes actividades asociadas al evento. La base de datos `torneo_tenis` contiene varias colecciones, entre ellas `jugadores`, `encuentros`, `arbitros`, `entrenadores` y `tabla_posiciones`. Dado el incremento constante en la cantidad de datos y la necesidad de un rendimiento óptimo, se hace necesario implementar una estrategia de particionamiento (*sharding*).

Este documento describe los requerimientos no funcionales relacionados con el particionamiento y especifica el escenario que justifica su implementación. Además, se detallan las estrategias de particionamiento que permitirán cumplir con los objetivos de rendimiento, escalabilidad y disponibilidad.

Escenario de Uso

El evento deportivo genera un volumen significativo de datos, los cuales deben ser almacenados, procesados y consultados de manera eficiente. Entre los datos relevantes se incluyen:

1. **Jugadores**: Datos de identificación, estadísticas y perfiles de los jugadores inscritos.
2. **Encuentros**: Historial de partidos, horarios, resultados y puntuaciones.
3. **Arbitros y entrenadores**: Información de contacto, roles y asignaciones.
4. **Tabla de posiciones**: Clasificación actualizada en tiempo real.

El crecimiento esperado del volumen de datos, especialmente durante la fase activa del torneo, hace que una base de datos centralizada no sea suficiente para garantizar tiempos de respuesta rápidos. Además, se espera que haya:

- Múltiples consultas simultáneas realizadas por organizadores, jugadores y espectadores.
- Alta demanda de actualizaciones en tiempo real, especialmente en la colección `tabla_posiciones`.
- Requerimientos de alta disponibilidad para evitar interrupciones durante el torneo.

En este contexto, el particionamiento horizontal mediante *sharding* se presenta como la solución ideal para garantizar el cumplimiento de los objetivos.

Requerimientos No Funcionales

1. Rendimiento

- El sistema debe permitir consultas rápidas sobre las colecciones más consultadas, como `jugadores` y `encuentros`, con un tiempo de respuesta inferior a 100 ms.
- Las actualizaciones en tiempo real de la colección `tabla_posiciones` deben reflejarse en menos de 200 ms.
- El sistema debe soportar más de 500 consultas simultáneas sin degradación significativa del rendimiento.

2. Escalabilidad

- El sistema debe ser capaz de manejar el crecimiento de datos hasta 1 TB de información distribuida en múltiples nodos.
- La arquitectura debe permitir agregar nuevos nodos al clúster sin interrumpir las operaciones en curso.

3. Disponibilidad

- El sistema debe garantizar una disponibilidad del 99.9% durante las horas pico del torneo.
- Debe implementarse un mecanismo de replicación que permita la recuperación de datos en caso de fallos.

4. Balanceo de Carga

- Los datos deben distribuirse uniformemente entre los nodos del clúster para evitar sobrecarga en un solo nodo.
- El sistema debe balancear automáticamente las particiones en función del volumen de datos y la carga de consultas.

Estrategia de Particionamiento

1. Modelo de Sharding

Se utilizará un modelo de particionamiento horizontal (*sharding*) en la base de datos `torneo_tenis`. El particionamiento se realizará de la siguiente manera:

- **Clave de partición**:
  - En la colección `jugadores`: Campo `_id` utilizando un índice *hashed* para garantizar una distribución uniforme de los datos.
  - En la colección `encuentros`: Campo `fecha` para agrupar los datos por fechas, permitiendo consultas eficientes basadas en el calendario del torneo.
- **Estrategia de distribución**:
  - Distribuir los datos entre los nodos del clúster en función de la clave de partición.
  - Configurar un balanceador automático que ajuste las particiones para mantener el equilibrio.

2. Configuración del Clúster

El clúster se configurará de la siguiente manera:

- **Servidor de Configuración**:
  - Nodo configurado como `configsvr` para almacenar metadatos del clúster.
- **Shard Servers**:
  - Múltiples nodos configurados como `shardsvr`, cada uno con su propio replicaset para garantizar la disponibilidad y la tolerancia a fallos.
- **Router (mongos)**:
  - Un router que actúa como intermediario entre los clientes y los nodos del clúster.

3. Habilitación del Sharding

- Habilitar el sharding en la base de datos `torneo_tenis`.
- Configurar el particionamiento de las colecciones `jugadores` y `encuentros` según las claves de partición definidas.

Video con la explicación de los comandos: https://youtu.be/pvNN6No-dec

Conclusión

La implementación del particionamiento horizontal mediante *sharding* en la base de datos `torneo_tenis` garantiza el cumplimiento de los requerimientos no funcionales, como rendimiento, escalabilidad, disponibilidad y balanceo de carga. Esto asegura que el sistema pueda manejar de manera eficiente el creciente volumen de datos y las altas demandas de consultas durante el evento deportivo.

El enfoque propuesto también permite una extensibilidad futura al agregar nuevos nodos al clúster y asegura la continuidad operativa en escenarios de alto tráfico o fallos del sistema. La solución planteada está diseñada para cumplir con los objetivos del proyecto y garantizar una experiencia óptima para los usuarios y organizadores del torneo.