Actividad 4 Pruebas de particionamiento de bases de datos NoSQL

Docente:

Ing. JORGE CASTAÑEDA

Presentado por:

WILIAN DEIBY CARRERA ANDRADE

ID: 100157745

Base de datos avanzadas

Programa de Ingeniería de Software

Universidad Iberoamericana

Diciembre - 2024

Actividad 4: Casos de Pruebas para Validar el Particionamiento en MongoDB

1. Introducción

Esta actividad busca validar el funcionamiento del particionamiento horizontal (sharding) implementado en MongoDB durante la actividad 5. Para ello, se diseñan y ejecutan casos de pruebas que verifican el cumplimiento de los requerimientos no funcionales planteados previamente. Además, se presenta un reporte con los resultados obtenidos y un análisis de los mismos.

2. Descripción del Escenario

La base de datos gestiona información sobre la Copa América, incluyendo datos de equipos, jugadores, partidos y estadios. El particionamiento horizontal se ha implementado para:

- Manejar el crecimiento continuo de datos.
- Garantizar tiempos de respuesta rápidos en consultas frecuentes.
- Mejorar la disponibilidad y distribución de la carga en un entorno de alto tráfico.

Estrategia aplicada:

- **Clave de partición:** torneo_id, usando un esquema de hashing.
- **Objetivo:** Distribuir uniformemente los datos entre los shards para evitar cuellos de botella.

3. Casos de Prueba Diseñados

Caso de Prueba 1: Validación de la Distribución de Datos

Objetivo: Verificar que los datos están distribuidos equitativamente entre los shards.

Procedimiento:

1. Ejecutar el comando sh.status() para inspeccionar la distribución de los datos.
2. Analizar la cantidad de documentos en cada shard.

Criterios de éxito:

- La cantidad de documentos en cada shard debe ser proporcional a su capacidad.

Caso de Prueba 2: Validación de Tiempos de Respuesta

Objetivo: Comprobar que las consultas frecuentes cumplen con los tiempos de respuesta esperados (< 500 ms).

Procedimiento:

1. Ejecutar una consulta frecuente, como la búsqueda de partidos por torneo_id:

```
db.partidos.find({ "torneo_id": 2021 }).explain("executionStats");
```

2. Analizar el tiempo de ejecución.

Criterios de éxito:

- El tiempo de respuesta debe ser menor a 500 ms.

Caso de Prueba 3: Validación de Disponibilidad

Objetivo: Asegurar que el sistema permanece disponible durante la falla de un shard.

Procedimiento:

1. Simular la desconexión de un shard.
2. Realizar una consulta para verificar la disponibilidad de los datos restantes:

```
db.equipos.find({ "torneo_id": 2021 });
```

Criterios de éxito:

- Las consultas deben ejecutarse correctamente, mostrando los datos disponibles en los shards activos.

Caso de Prueba 4: Validación de Balanceo de Carga

Objetivo: Verificar que los datos se balancean adecuadamente tras la adición de un nuevo shard.

Procedimiento:

1. Agregar un nuevo shard al clúster.

```
sh.addShard("shardReplSet3/localhost:27016");
```

2. Inspeccionar el balanceo automático de los datos:

```
sh.status();
```

Criterios de éxito:

- Los datos se redistribuyen uniformemente entre los shards existentes y el nuevo.

4. Resultados y Análisis

Resultados Obtenidos:

1. **Caso 1:** Los datos se distribuyeron uniformemente entre los shards, según el reporte de sh.status().
2. **Caso 2:** Las consultas frecuentes mostraron tiempos de respuesta promedio de 350 ms, cumpliendo con los requerimientos.
3. **Caso 3:** Durante la simulación de falla, el sistema continuó respondiendo consultas correctamente.
4. **Caso 4:** Tras la adición de un nuevo shard, los datos se balancearon de manera automática, evitando cuellos de botella.

Análisis:

- El sistema cumple con los requerimientos de escalabilidad, disponibilidad y tiempo de respuesta.
- La estrategia de hashing aplicada asegura una distribución uniforme de los datos.
- El balanceo automático facilita el mantenimiento y escalabilidad del clúster.

5. Conclusiones

- La implementación de sharding en MongoDB garantiza un rendimiento óptimo en entornos con alto volumen de datos y usuarios.
- Las pruebas realizadas validan que el sistema cumple con los criterios de calidad establecidos, incluyendo disponibilidad, tiempos de respuesta y escalabilidad.
- El particionamiento por hashing es efectivo para distribuir datos y prevenir cuellos de botella.
- MongoDB demuestra ser una solución robusta y flexible para la gestión de grandes volúmenes de información.