Bases de Datos Avanzadas

Actividad 4 : Pruebas de particionamiento de 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

El presente documento describe de manera detallada los casos de prueba diseñados y ejecutados para validar la implementación del particionamiento (*sharding*) en la base de datos `torneo_tenis` de MongoDB. Este tipo de particionamiento permite la distribución horizontal de datos, optimizando el rendimiento, la escalabilidad y la disponibilidad de las operaciones de consulta y escritura en un clúster de bases de datos NoSQL.

La actividad responde a los retos comunes en sistemas que manejan grandes volúmenes de datos y que requieren una arquitectura que soporte un crecimiento dinámico. En este contexto, la configuración del particionamiento busca garantizar que los datos estén distribuidos uniformemente entre múltiples nodos, permitiendo consultas más rápidas y mejorando la tolerancia a fallos.

A través de este documento, se busca no solo evidenciar que la configuración del particionamiento cumple con los requisitos no funcionales planteados previamente, sino también proporcionar un análisis detallado de las pruebas ejecutadas. Estas incluyen desde la validación del estado del clúster hasta pruebas de balanceo de datos, asegurando que las colecciones `jugadores` y `encuentros` operen de manera eficiente en un entorno distribuido. Además, se documentan los procedimientos, resultados y conclusiones, sirviendo como referencia para futuras implementaciones en escenarios similares.

1. Objetivos de los Casos de Prueba

1. Validar la correcta distribución de los datos entre los shards configurados.
2. Garantizar que las operaciones de lectura y escritura funcionan de manera óptima en un entorno particionado.
3. Comprobar que el balanceador de MongoDB redistribuye los datos entre los shards cuando es necesario.
4. Evaluar el rendimiento de las consultas mediante el uso de índices y particionamiento.
5. Detectar errores o configuraciones incorrectas en el entorno de sharding.

2. Escenario de Pruebas

La base de datos `torneo_tenis` contiene dos colecciones principales:

1. **`jugadores`:** Contiene información sobre los jugadores del torneo.
2. **`encuentros`:** Registra los partidos jugados, incluyendo la fecha y los participantes.

La configuración de particionamiento utiliza las siguientes claves:

- **`jugadores`:** _id (hashed)
- **`encuentros`:** fecha (ranged)

3. Descripción de los Casos de Prueba

Caso de Prueba 1: Verificación del Estado del Clúster

- **Objetivo:** Validar que el clúster está configurado correctamente con los shards y el balanceador habilitados.
- **Procedimiento:**
    - Conectarse al router (`mongos`) utilizando `mongosh`.
    - Ejecutar el comando `sh.status()`.
- **Criterios de Éxito:**
    - Los shards configurados deben estar visibles en la salida del comando.
    - El balanceador debe estar habilitado.
- **Resultado Esperado:** Información completa del clúster con shards activos y balanceador habilitado.

Caso de Prueba 2: Inserción de Datos en la Colección `encuentros`

- **Objetivo:** Validar que la colección `encuentros` permite la inserción de datos sin errores y que los documentos son distribuidos correctamente entre los shards.
- **Procedimiento:**

Insertar 100 documentos en la colección utilizando un script:

```
for (let i = 1; i <= 100; i++) {

  db.encuentros.insert({

    fecha: new Date(),

    jugadorA: `Jugador${i}`,

    jugadorB: `Jugador${i + 1}`

  });
```

    - }
    - Verificar la distribución de los documentos con el comando `db.encuentros.getShardDistribution()`.
- **Criterios de Éxito:**
    - Los documentos deben ser distribuidos entre los shards configurados.
    - No se deben reportar errores durante la inserción.
- **Resultado Esperado:** Distribución uniforme de los documentos entre los shards.

Caso de Prueba 3: Validación del Particionamiento en la Colección `encuentros`

- **Objetivo:** Garantizar que la colección `encuentros` está correctamente particionada.
- **Procedimiento:**
    - Ejecutar el comando `sh.enableSharding("torneo_tenis")`.

    - Configurar el particionamiento con: sh.shardCollection("torneo_tenis.encuentros", { fecha: 1 });
    - Confirmar el particionamiento con `db.encuentros.getShardDistribution()`.
- **Criterios de Éxito:**
    - La colección debe aparecer como particionada.
    - Los datos deben estar distribuidos entre los shards.
- **Resultado Esperado:** Configuración exitosa del particionamiento.

Caso de Prueba 4: Prueba de Consulta Distribuida

- **Objetivo:** Evaluar el rendimiento de las consultas en el entorno particionado.
- **Procedimiento:**
    - Ejecutar una consulta en la colección `jugadores` con `explain("executionStats")`: db.jugadores.find().explain("executionStats");
    - Analizar el resultado para confirmar el acceso distribuido.
- **Criterios de Éxito:**
    - La consulta debe ejecutarse sin errores.
    - El resultado de `explain` debe mostrar operaciones distribuidas.
- **Resultado Esperado:** Las consultas deben ejecutarse de manera eficiente y distribuida.

Caso de Prueba 5: Simulación de Balanceo de Datos

- **Objetivo:** Verificar que el balanceador redistribuye los datos entre los shards al agregar nuevos nodos.
- **Procedimiento:**
    - Agregar un nuevo shard al clúster.
    - Verificar la redistribución de datos utilizando `sh.status()` y `db.encuentros.getShardDistribution()`.
- **Criterios de Éxito:**
    - Los datos deben redistribuirse entre los shards existentes y el nuevo shard.
- **Resultado Esperado:** Redistribución de los datos confirmada.

4. Resultados y Análisis

Los resultados de cada caso de prueba se documentan a continuación:

1. **Caso de Prueba 1:** El estado del clúster fue confirmado con `sh.status()`. Todos los shards y el balanceador estaban activos.
2. **Caso de Prueba 2:** Los 100 documentos fueron insertados correctamente en la colección `encuentros`. La distribución inicial mostró que los documentos estaban en un solo shard antes del particionamiento.

3. **Caso de Prueba 3:** Después de habilitar el sharding y configurar el particionamiento, los datos se distribuyeron entre los shards según la clave `fecha`.
4. **Caso de Prueba 4:** Las consultas en `jugadores` mostraron operaciones distribuidas en los shards configurados, confirmando la eficiencia del particionamiento.
5. **Caso de Prueba 5:** La redistribución de datos fue exitosa tras agregar un nuevo shard, demostrando que el balanceador funciona correctamente.

Video para visualizar la explicación del proceso: https://youtu.be/gMjZLL7Euw4

Video para visualizar el paso a paso del script: https://youtu.be/uFcj19DZrCo

5. Conclusiones

Los casos de prueba ejecutados demostraron que la configuración de particionamiento en la base de datos `torneo_tenis` funciona correctamente, permitiendo la distribución eficiente de datos, la ejecución óptima de consultas y el balanceo de carga entre los nodos del clúster. La configuración de sharding cumple con los criterios de calidad establecidos en los requerimientos no funcionales.