**Escenario de Particionamiento y Requerimientos No Funcionales**

1. Descripción del Escenario

Título: Gestión de un Torneo Deportivo en Tiempo Real

En el contexto de un torneo deportivo de gran escala, como una liga de fútbol, es esencial gestionar una gran cantidad de datos en tiempo real. En este torneo, los datos incluyen:

Información de deportistas (jugadores, entrenadores, árbitros).

Detalles de los encuentros deportivos (fechas, equipos participantes, ubicaciones).

Resultados de los partidos (goles, tarjetas amarillas, tarjetas rojas).

Estadísticas y clasificaciones en tiempo real.

Dada la magnitud de los datos generados, especialmente durante las temporadas altas, el uso de particionamiento es crucial. Este enfoque permite distribuir la carga de datos en diferentes nodos, mejorando el rendimiento de las consultas y la disponibilidad del sistema.

Escenario de Particionamiento Necesario

Tipo de Datos a Particionar: Datos de partidos, estadísticas de jugadores y resultados.

Criterio de Particionamiento: Se puede usar la fecha de los partidos (por ejemplo, por temporada) y el ID del equipo.

Objetivo: Asegurar que las consultas relacionadas con un periodo específico o un equipo particular no se vean afectadas por el aumento del volumen de datos.

2. Requerimientos No Funcionales

2.1. Disponibilidad

La base de datos debe estar disponible 24x7, garantizando que los usuarios puedan acceder a la información en tiempo real, especialmente durante los partidos.

2.2. Escalabilidad

La arquitectura debe permitir añadir más nodos fácilmente para manejar un incremento en el volumen de datos y solicitudes, especialmente durante los picos de tráfico (como durante los partidos).

2.3. Rendimiento

Las consultas sobre datos particionados deben realizarse en menos de 200 milisegundos. Esto incluye la lectura de estadísticas de jugadores y resultados de partidos en tiempo real.

2.4. Mantenimiento

El sistema debe facilitar el mantenimiento sin tiempo de inactividad significativo. Esto incluye la capacidad de agregar o eliminar particiones sin afectar la disponibilidad del sistema.

. Scripts para el Particionamiento

A continuación, se presentan los scripts para configurar el particionamiento en MongoDB:

use TorneoFutbol2024;

// Crear una colección para partidos

db.createCollection("partidos");

// Insertar datos de ejemplo

db.partidos.insertMany([

{ \_id: 1, equipoLocal: "Equipo A", equipoVisitante: "Equipo B", fecha: new Date("2024-12-01"), resultado: { local: 2, visitante: 1 } },

{ \_id: 2, equipoLocal: "Equipo C", equipoVisitante: "Equipo D", fecha: new Date("2024-12-02"), resultado: { local: 1, visitante: 1 } },

{ \_id: 3, equipoLocal: "Equipo A", equipoVisitante: "Equipo C", fecha: new Date("2024-12-03"), resultado: { local: 3, visitante: 2 } },

// ... más datos

]);

3.2. Crear un Índice para Particionamiento

Para facilitar el particionamiento, se recomienda crear un índice en la colección partidos basado en la fecha:

db.partidos.createIndex({ fecha: 1 });

3.3. Particionamiento Horizontal (Sharding)

Para realizar particionamiento horizontal (sharding) en MongoDB, debes tener un shard cluster. Aquí te muestro cómo configurarlo de manera básica:

3.3.1. Habilitar Sharding en la Base de Datos

sh.enableSharding("TorneoFutbol2024");

3.3.2. Configurar la Colección para el Sharding

Para configurar la colección partidos para el sharding, selecciona un campo (en este caso, fecha) que dividirá los datos

sh.shardCollection("TorneoFutbol2024.partidos", { fecha: 1 });

3.4. Resultados Obtenidos

Después de realizar los scripts, puedes verificar el estado de sharding con el siguiente comando:

sh.status();

4. Conclusiones

La implementación de un sistema de particionamiento permite gestionar efectivamente grandes volúmenes de datos en tiempo real, asegurando la disponibilidad y el rendimiento durante un torneo deportivo. Los requerimientos no funcionales garantizan que el sistema se mantenga operativo y eficiente bajo diversas cargas de trabajo.