

BASE DE DATOS AVANZADA

ACTIVIDAD - 3 y ACTIVIDAD - 4

PRESENTADO POR:

KRISTIN XIOMARA MUÑOZ RIVERA
YICXELA HERMINIA CONEJO CONEJO

WILLIAM RUIZ

CORPORACIÓN UNIVERSITARIA IBEROAMERICANA

INGENIERÍA DE SOFTWARE

09/JUNIO/2024

ACTIVIDAD - 3

INTRODUCCIÓN

El particionamiento o Sharding es una técnica fundamental en el diseño de sistemas distribuidos para mejorar la escalabilidad y el rendimiento. Al igual que en un torneo deportivo de fútbol, donde los equipos compiten en distintas divisiones o grupos, el Sharding divide los datos de una base de datos en fragmentos más pequeños, llamados shard, distribuyéndolos entre múltiples servidores. Esta analogía puede ayudar a comprender mejor el concepto.

Imagina un torneo de fútbol donde participan equipos de todo el mundo. Para facilitar la organización y la gestión del torneo, los equipos se dividen en grupos o divisiones según diferentes criterios, como la región geográfica o el nivel de habilidad. Cada grupo juega en su propia liga, lo que permite una competencia más equitativa y eficiente.

De manera similar, en un sistema distribuido, los datos se dividen en fragmentos o shards que se distribuyen en diferentes servidores. Cada shard contiene una parte de los datos totales y puede ser gestionado de forma independiente. Esto ayuda a distribuir la carga de trabajo y mejora la capacidad de respuesta del sistema, ya que múltiples servidores pueden procesar consultas simultáneamente.

Al especificar los requerimientos no funcionales para un sistema que utiliza particionamiento o Sharding, es crucial considerar aspectos como la distribución de la carga de trabajo, la disponibilidad, la consistencia de los datos y el rendimiento. Al igual que en un torneo de fútbol, donde se deben definir reglas claras sobre cómo se organizan los equipos y cómo se llevan a cabo los partidos, en el diseño de sistemas distribuidos se deben establecer políticas y estrategias para garantizar que el particionamiento se realice de manera efectiva y que el sistema funcione de manera óptima en todo momento.

1. requerimientos no funcionales de la base de datos del torneo deportivo de fútbol

El particionamiento o sharding es una técnica utilizada en bases de datos distribuidas para dividir los datos en varios servidores para mejorar el rendimiento y la escalabilidad. En el contexto de un torneo deportivo de fútbol, podría necesitar particionamiento o sharding en los siguientes escenarios:

1. **Gran cantidad de equipos y jugadores:** Si el torneo involucra una gran cantidad de equipos y jugadores, la cantidad de datos generados puede ser considerable. El particionamiento puede ayudar a distribuir estos datos en múltiples servidores para evitar cuellos de botella y mejorar la velocidad de acceso a la información.
2. **Amplia base de aficionados y datos relacionados:** Con un torneo deportivo popular como el fútbol, la base de aficionados puede ser enorme. Esto puede traducirse en una gran cantidad de datos relacionados con la venta de entradas, transmisiones en vivo, estadísticas de jugadores, comentarios de los aficionados en redes sociales, etc. El particionamiento puede ser útil para gestionar eficientemente estos datos y garantizar un rendimiento óptimo del sistema.
3. **Transmisión en vivo y actualizaciones en tiempo real:** Si el torneo incluye la transmisión en vivo de los partidos y la actualización en tiempo real de los resultados y estadísticas, el sistema debe ser capaz de manejar grandes volúmenes de datos en tiempo real. El particionamiento puede ayudar a distribuir la carga de trabajo entre múltiples servidores y garantizar una transmisión fluida y una actualización rápida de los datos.
4. **Acceso global:** Si el torneo tiene una audiencia global y los datos deben estar disponibles para usuarios de todo el mundo, el particionamiento puede ayudar a reducir la latencia y mejorar la disponibilidad de los datos mediante la distribución geográfica de los servidores.

En cuanto a los requisitos no funcionales para este escenario, podrían incluir:

1. **Escalabilidad:** El sistema debe ser capaz de escalar fácilmente para manejar un aumento en la cantidad de datos y usuarios sin comprometer el rendimiento.
2. **Disponibilidad:** El sistema debe estar disponible en todo momento, especialmente durante los momentos críticos del torneo, como los partidos en vivo y las actualizaciones de resultados.
3. **Rendimiento:** El sistema debe ser capaz de manejar grandes volúmenes de datos y proporcionar tiempos de respuesta rápidos incluso durante picos de carga.

4. **Consistencia:** Aunque los datos pueden estar distribuidos en varios servidores, el sistema debe garantizar la consistencia de los datos para evitar inconsistencias y conflictos.
 5. **Seguridad:** Dado que el sistema puede contener información confidencial de jugadores, equipos y aficionados, se deben implementar medidas de seguridad robustas para proteger los datos contra accesos no autorizados y ataques cibernéticos.

Al considerar el particionamiento o sharding en un escenario de torneo deportivo de fútbol, es importante tener en cuenta estos requisitos no funcionales para diseñar e implementar una solución que cumpla con las necesidades del sistema.

implementación de réplica

1. Iniciar el Cluster de Shards Cuando inicias un clúster de shards, estás configurando un entorno en el que los datos se distribuyen y replican automáticamente entre varios nodos. Esto mejora la tolerancia a fallos, ya que si un nodo falla, los datos todavía están disponibles a través de los otros nodos. También mejora el rendimiento al distribuir la carga de trabajo

```
> cluster=new ShardingTest ({shards : 3, chunksize:1})
```

2. Inserción de datos: Una vez que haya iniciado un clúster de shards en MongoDB, puede comenzar a insertar datos en la base de datos distribuida.

```
> db = (new Mongo("localhost:20006")).getDB("Championship");
Championship
mongos> |
```

inserción de datos

```
mongos> for (i = 0; i < 500000; i++) {
...     db.players.insert({
...         Id_equipo: "EQU" + i,
...         nombre: "Equipo número-" + i,
...         pais: "Pais-" + i
...     });
... }
```

3. Comprobación de la distribución de datos en los nodos:Comprender cómo se distribuyen los datos entre los nodos te permite evaluar si la carga está equilibrada de manera uniforme.

```
mongos> db.players.count()
500000
mongos>
```

4. Activación del Sharding: se debe actuar sobre el balanceador, volviendo así a la consola de mongo que hemos arrancado contra la instancia de mongos (que corría en el puerto 20000). Para activar el Sharding se utiliza la función enableSharding() sobre la base de datos Torneo Deportivo", se crea el índice y se determina la colección "equipo".

```
Windows PowerShell
Copyright (C) Microsoft Corporation. Todos los derechos reservados.

Instale la versión más reciente de PowerShell para obtener nuevas características y mejoras. https://aka.ms/PSWindows

PS C:\Users\Bienvenido> mongo
MongoDB shell version v4.4.22
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("f493825f-7344-4151-b70c-c17b1b4005fd") }
MongoDB server version: 4.4.22
-----
The server generated these startup warnings when booting:
      2024-06-09T21:54:06.420+05:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
ed
-----
> shard1 = new Mongo("localhost:20000")
connection to localhost:20000
```

```
> shard1DB = shard1.getDB("Shampionship")
Shampionship
```

5. Activación del balanceador de carga: se comprueba el estado de ejecución y se consigue que el balanceador comience a ejecutarse.

```
> shard1DB.players.count()
500000
>
```

6. Verificación del particionamiento de los datos en cada nodo: esta verificación implica asegurarse de que los datos se distribuyen y se balancean correctamente entre los shards.

```
> shard2= new Mongo("localhost:20001")
connection to localhost:20001
```

```
> shard2DB = shard2.getDB("Shampionship")
Shampionship
```

```
> shard2DB.players.count()
0
```

```
> shard3= new Mongo("localhost:20002")
connection to localhost:20002
```

```
> shard3DB = shard3.getDB("Shampionship")
Shampionship
```

```
> shard3DB.players.count()
0
```

```
> shard1 = new Mongo("localhost:20006")
connection to localhost:20006
```

ACTIVIDAD - 4

El planteamiento del caso de prueba, ejecución de los respectivos casos de prueba, reporte de los resultados y análisis para validar que el particionamiento planteado

Gran cantidad de equipos y jugadores: En torneos con muchos participantes, el particionamiento puede distribuir los datos en múltiples servidores para evitar problemas de rendimiento haciendo un búsqueda adecuada por id .

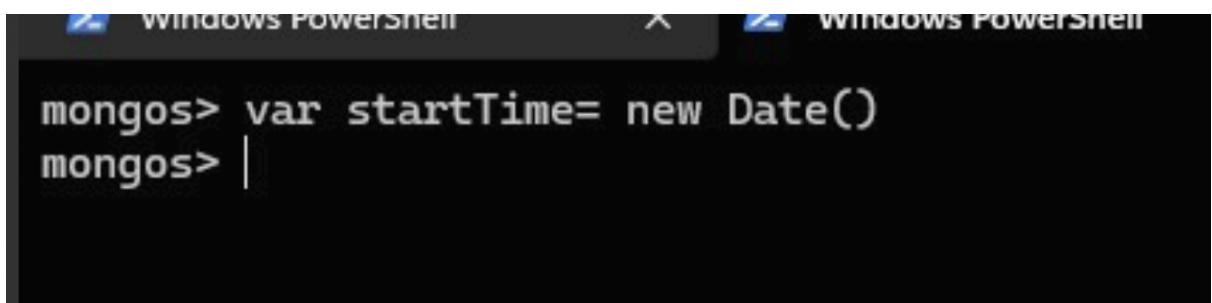
Amplia base de aficionados y datos relacionados: En eventos populares como el fútbol, el particionamiento gestiona eficazmente datos de entradas como Consulta por un rango de equipos.

actualizaciones en tiempo real: actualizaciones instantáneas, el particionamiento asegura una distribución equitativa de la carga y una experiencia fluida para los usuarios en cuanto a la distribución e inserción de datos .

Acceso global eliminacion: el particionamiento reduce la latencia y mejora la disponibilidad de datos mediante la distribución permitiendo eliminar datos eficazmente.

1. Caso de prueba 1: Consulta por un jugador específico

Crear una colección "players" en la base de datos "Championship" con datos de prueba. Crear un índice en la clave que vamos a usar como shard, en el campo "Id_player" de la colección "Championship" utilizando el comando
`db.equipos.ensureIndex({Id_player: 1})"`



```
mongos> var startTime= new Date()
mongos> |
```

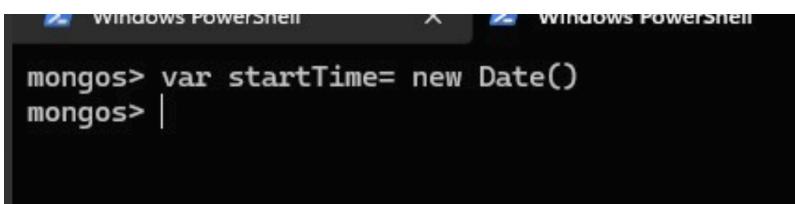
consultar un equipo en específico mediante el comando

```
mongos> db.players.find({ _id: ObjectId("66671954eb4c95c86bc7a39d") }).pretty()
{
    "_id" : ObjectId("66671954eb4c95c86bc7a39d"),
    "Id_player" : "Player18",
    "name" : "Name-18",
    "nationality" : "Nationality-18",
    "last_name" : "LastName-18",
    "document" : "Document-18",
    "birthdate" : ISODate("2008-07-18T05:00:00Z")
}
mongos>
```

2. Caso de prueba 2: Consulta por un rango de jugador

Consultar un rango de equipos mediante los siguientes comandos Primero por nombre y luego por Id_player

Recuperación por el id:



```
mongos> var startTime= new Date()
mongos> |
```

```
mongos> db.players.find({ _id: ObjectId("66671954eb4c95c86bc7a39d") }).pretty()
{
    "_id" : ObjectId("66671954eb4c95c86bc7a39d"),
    "Id_player" : "Player18",
    "name" : "Name-18",
    "nationality" : "Nationality-18",
    "last_name" : "LastName-18",
    "document" : "Document-18",
    "birthdate" : ISODate("2008-07-18T05:00:00Z")
}
mongos>
```

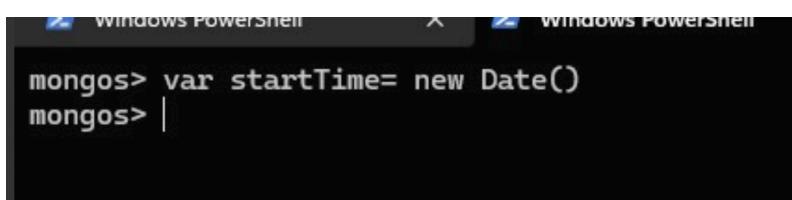
```
'mongos> var endTime = new Date()  
mongos> var totalTime = endTime - startTime'
```

```
'mongos> var totalTime = endTime - startTime  
mongos>
```

```
'mongos> print("El tiempo de recuperación fue de " + totalTime/1000 + " segundos.");  
El tiempo de recuperación fue de 59.197 segundos.  
mongos>
```

```
'mongos> var startTime = new Date()  
mongos> db.players.find({ _id: ObjectId("66671954eb4c95c86bc7a39d") }).pretty()  
{  
    "_id" : ObjectId("66671954eb4c95c86bc7a39d"),  
    "Id_player" : "Player18",  
    "name" : "Name-18",  
    "nationality" : "Nationality-18",  
    "last_name" : "LastName-18",  
    "document" : "Document-18",  
    "birthdate" : ISODate("2008-07-18T05:00:00Z")  
}  
mongos> db.players.find({ _id: ObjectId("66671954eb4c95c86bc7a39d") })  
{ "_id" : ObjectId("66671954eb4c95c86bc7a39d"), "Id_player" : "Player18", "name" : "Name-18", "nationality" : "Nationality-18", "last_name" : "LastName-18", "document" : "Document-18", "birthdate" : ISODate("2008-07-18T05:00:00Z") }  
mongos> db.players.find({ _id: ObjectId("66671954eb4c95c86bc7a39d") }).pretty()  
{  
    "_id" : ObjectId("66671954eb4c95c86bc7a39d"),  
    "Id_player" : "Player18",  
    "name" : "Name-18",  
    "nationality" : "Nationality-18",  
    "last_name" : "LastName-18",  
    "document" : "Document-18",  
    "birthdate" : ISODate("2008-07-18T05:00:00Z")  
}  
mongos> var endTime = new Date()  
mongos> var totalTime = endTime - startTime  
SyntaxError: unexpected token: identifier  
@(shell):1:24  
mongos> var totalTime = endTime - startTime  
mongos> print("El tiempo de recuperación fue de " + totalTime/1000 + " segundos.");  
El tiempo de recuperación fue de 59.197 segundos.  
mongos>
```

Recuperación por el nombre:



```
'Windows PowerShell'  X  'Windows PowerShell'  
mongos> var startTime= new Date()  
mongos> |
```

```
mongos> db.players.find({
...     Id_player: {
...         $gte: "Player100001",
...         $lte: "Player100009"
...     }
... }).pretty()
{
    "_id" : ObjectId("66671a6ceb4c95c86bc92a2c"),
    "Id_player" : "Player100001",
    "name" : "Name-100001",
    "nationality" : "Nationality-100001",
    "last_name" : "LastName-100001",
    "document" : "Document-100001",
    "birthdate" : ISODate("2001-06-13T05:00:00Z")
}
{
    "_id" : ObjectId("66671a6ceb4c95c86bc92a2d"),
    "Id_player" : "Player100002",
    "name" : "Name-100002",
    "nationality" : "Nationality-100002",
    "last_name" : "LastName-100002",
    "document" : "Document-100002",
    "birthdate" : ISODate("2002-07-14T05:00:00Z")
}
{
    "_id" : ObjectId("66671a6ceb4c95c86bc92a2e"),
    "Id_player" : "Player100003",
    "name" : "Name-100003",
    "nationality" : "Nationality-100003",
    "last_name" : "LastName-100003",
    "document" : "Document-100003",
    "birthdate" : ISODate("2003-08-15T05:00:00Z")
```

```
{  
    "_id" : ObjectId("66671a6ceb4c95c86bc92a2f"),  
    "Id_player" : "Player100004",  
    "name" : "Name-100004",  
    "nationality" : "Nationality-100004",  
    "last_name" : "LastName-100004",  
    "document" : "Document-100004",  
    "birthdate" : ISODate("2004-09-16T05:00:00Z")  
}  
{  
    "_id" : ObjectId("66671a6ceb4c95c86bc92a30"),  
    "Id_player" : "Player100005",  
    "name" : "Name-100005",  
    "nationality" : "Nationality-100005",  
    "last_name" : "LastName-100005",  
    "document" : "Document-100005",  
    "birthdate" : ISODate("2005-10-17T05:00:00Z")  
}  
{  
    "_id" : ObjectId("66671a6ceb4c95c86bc92a31"),  
    "Id_player" : "Player100006",  
    "name" : "Name-100006",  
    "nationality" : "Nationality-100006",  
    "last_name" : "LastName-100006",  
    "document" : "Document-100006",  
    "birthdate" : ISODate("2006-11-18T05:00:00Z")  
}  
{  
    "_id" : ObjectId("66671a6ceb4c95c86bc92a32"),  
    "Id_player" : "Player100007",  
    "name" : "Name-100007",  
    "nationality" : "Nationality-100007",  
    "last_name" : "LastName-100007",  
    "document" : "Document-100007",  
    "birthdate" : ISODate("2007-12-19T05:00:00Z")  
}
```

```
,  
{  
    "_id" : ObjectId("66671a6ceb4c95c86bc92a33"),  
    "Id_player" : "Player100008",  
    "name" : "Name-100008",  
    "nationality" : "Nationality-100008",  
    "last_name" : "LastName-100008",  
    "document" : "Document-100008",  
    "birthdate" : ISODate("2008-01-20T05:00:00Z")  
}  
{  
    "_id" : ObjectId("66671a6ceb4c95c86bc92a34"),  
    "Id_player" : "Player100009",  
    "name" : "Name-100009",  
    "nationality" : "Nationality-100009",  
    "last_name" : "LastName-100009",  
    "document" : "Document-100009",  
    "birthdate" : ISODate("2009-02-21T05:00:00Z")  
}  
mongos>
```

```
,  
mongos> var endTime = new Date()  
mongos> var startTime = endTime - 1000000000000000000
```

```
mongos> var totalTime = endTime - startTime
```

```
mongos> print("El tiempo de recuperación fue de " + totalTime/1000 + " segundos.")  
El tiempo de recuperación fue de 107.974 segundos.  
mongos>
```

```

mongos> var endTime = new Date()
mongos> var totalTime = endTime - startTime
mongos> print("El tiempo de recuperación fue de " + totalTime/1000 + " segundos.")
El tiempo de recuperación fue de 107.974 segundos.
mongos> var startTime= new Date()
mongos> db.players.find({$gte: "Player100001", $lte: "Player100009"})
{
  "_id" : ObjectId("66671a6ceb4c95c86bc92a2c"),
  "Id_player" : "Player100001",
  "name" : "Name-100001",
  "nationality" : "Nationality-100001",
  "last_name" : "LastName-100001",
  "birthdate" : ISODate("2001-06-13T05:00:00Z")
}
{
  "_id" : ObjectId("66671a6ceb4c95c86bc92a2d"),
  "Id_player" : "Player100002",
  "name" : "Name-100002",
  "nationality" : "Nationality-100002",
  "last_name" : "LastName-100002",
  "document" : "Document-100002",
  "birthdate" : ISODate("2002-07-14T05:00:00Z")
}
{
  "_id" : ObjectId("66671a6ceb4c95c86bc92a2e"),
  "Id_player" : "Player100003",
  "name" : "Name-100003",
  "nationality" : "Nationality-100003",
  "last_name" : "LastName-100003",
  "document" : "Document-100003",
  "birthdate" : ISODate("2003-08-15T05:00:00Z")
}
{
  "_id" : ObjectId("66671a6ceb4c95c86bc92a2f"),
  "Id_player" : "Player100004",
  "name" : "Name-100004",
  "nationality" : "Nationality-100004",
  "last_name" : "LastName-100004",
  "document" : "Document-100004",
  "birthdate" : ISODate("2004-09-16T05:00:00Z")
}
{
  "_id" : ObjectId("66671a6ceb4c95c86bc92a30"),
  "Id_player" : "Player100005",
  "name" : "Name-100005",
  "nationality" : "Nationality-100005",
  "last_name" : "LastName-100005",
  "document" : "Document-100005",
  "birthdate" : ISODate("2005-10-17T05:00:00Z")
}
{
  "_id" : ObjectId("66671a6ceb4c95c86bc92a31"),
  "Id_player" : "Player100006",
  "name" : "Name-100006",
  "nationality" : "Nationality-100006",
  "last_name" : "LastName-100006",
  "document" : "Document-100006",
  "birthdate" : ISODate("2006-11-18T05:00:00Z")
}
{
  "_id" : ObjectId("66671a6ceb4c95c86bc92a32"),
  "Id_player" : "Player100007",
  "name" : "Name-100007",
  "nationality" : "Nationality-100007",
  "last_name" : "LastName-100007",
  "document" : "Document-100007",
  "birthdate" : ISODate("2007-12-19T05:00:00Z")
}
{
  "_id" : ObjectId("66671a6ceb4c95c86bc92a33"),
  "Id_player" : "Player100008",
  "name" : "Name-100008",
  "nationality" : "Nationality-100008",
  "last_name" : "LastName-100008",
  "document" : "Document-100008",
  "birthdate" : ISODate("2008-01-20T05:00:00Z")
}
{
  "_id" : ObjectId("66671a6ceb4c95c86bc92a34"),
  "Id_player" : "Player100009",
  "name" : "Name-100009",
  "nationality" : "Nationality-100009",
  "last_name" : "LastName-100009",
  "document" : "Document-100009",
  "birthdate" : ISODate("2009-02-21T05:00:00Z")
}
mongos> var endTime = new Date()
mongos> var totalTime = endTime - startTime
mongos> print("El tiempo de recuperación fue de " + totalTime/1000 + " segundos.")
El tiempo de recuperación fue de 0.801 segundos.
mongos>

```

resultado por id

```

mongos> var startTime= new Date()
mongos> db.players.find({ _id: ObjectId("66671954eb4c95c86bc7a39d") }).pretty()
{
  "_id" : ObjectId("66671954eb4c95c86bc7a39d"),
  "Id_player" : "Player18",
  "name" : "Name-18",
  "nationality" : "Nationality-18",
  "last_name" : "LastName-18",
  "document" : "Document-18",
  "birthdate" : ISODate("2008-07-18T05:00:00Z")
}
mongos> var endTime = new Date()
mongos> var totalTime = endTime - startTime
mongos> print("El tiempo de recuperación fue de " + totalTime/1000 + " segundos.")
El tiempo de recuperación fue de 0.045 segundos.
mongos>

```

resultado por nombre

```

mongos> var startTime= new Date()
mongos> db.players.find({ name: "Name-19" }).pretty()
{
  "_id" : ObjectId("66671954eb4c95c86bc7a39e"),
  "Id_player" : "Player19",
  "name" : "Name-19",
  "nationality" : "Nationality-19",
  "last_name" : "LastName-19",
  "document" : "Document-19",
  "birthdate" : ISODate("2009-08-19T05:00:00Z")
}
mongos> var endTime = new Date()
mongos> var totalTime = endTime - startTime
mongos> print("El tiempo de recuperación fue de " + totalTime/1000 + " segundos.")
El tiempo de recuperación fue de 0.667 segundos.
mongos>

```

```

mongos> var startTime= new Date()
mongos> db.players.find({ Id_player: { $gte: "Player100001", $lte: "Player100009" } })
{ "_id" : ObjectId("66671a6ceb4c95c86bc92a2c"), "Id_player" : "Player100001", "name" : "Name-100001", "nationality" : "Nationality-100001", "last_na
me" : "LastName-100001", "document" : "Document-100001", "birthdate" : ISODate("2001-06-13T05:00:00Z") }
{ "_id" : ObjectId("66671a6ceb4c95c86bc92a2d"), "Id_player" : "Player100002", "name" : "Name-100002", "nationality" : "Nationality-100002", "last_na
me" : "LastName-100002", "document" : "Document-100002", "birthdate" : ISODate("2002-07-14T05:00:00Z") }
{ "_id" : ObjectId("66671a6ceb4c95c86bc92a2e"), "Id_player" : "Player100003", "name" : "Name-100003", "nationality" : "Nationality-100003", "last_na
me" : "LastName-100003", "document" : "Document-100003", "birthdate" : ISODate("2003-08-15T05:00:00Z") }
{ "_id" : ObjectId("66671a6ceb4c95c86bc92a2f"), "Id_player" : "Player100004", "name" : "Name-100004", "nationality" : "Nationality-100004", "last_na
me" : "LastName-100004", "document" : "Document-100004", "birthdate" : ISODate("2004-09-16T05:00:00Z") }
{ "_id" : ObjectId("66671a6ceb4c95c86bc92a30"), "Id_player" : "Player100005", "name" : "Name-100005", "nationality" : "Nationality-100005", "last_na
me" : "LastName-100005", "document" : "Document-100005", "birthdate" : ISODate("2005-10-17T05:00:00Z") }
{ "_id" : ObjectId("66671a6ceb4c95c86bc92a31"), "Id_player" : "Player100006", "name" : "Name-100006", "nationality" : "Nationality-100006", "last_na
me" : "LastName-100006", "document" : "Document-100006", "birthdate" : ISODate("2006-11-18T05:00:00Z") }
{ "_id" : ObjectId("66671a6ceb4c95c86bc92a32"), "Id_player" : "Player100007", "name" : "Name-100007", "nationality" : "Nationality-100007", "last_na
me" : "LastName-100007", "document" : "Document-100007", "birthdate" : ISODate("2007-12-19T05:00:00Z") }
{ "_id" : ObjectId("66671a6ceb4c95c86bc92a33"), "Id_player" : "Player100008", "name" : "Name-100008", "nationality" : "Nationality-100008", "last_na
me" : "LastName-100008", "document" : "Document-100008", "birthdate" : ISODate("2008-01-20T05:00:00Z") }
{ "_id" : ObjectId("66671a6ceb4c95c86bc92a34"), "Id_player" : "Player100009", "name" : "Name-100009", "nationality" : "Nationality-100009", "last_na
me" : "LastName-100009", "document" : "Document-100009", "birthdate" : ISODate("2009-02-21T05:00:00Z") }
mongos> var endTime = new Date()
mongos> print("El tiempo de recuperación fue de " + totalTime/1000 + " segundos.")
El tiempo de recuperación fue de 0.632 segundos.
mongos>

```

3. Caso de prueba 3: Inserción de datos

Insertamos un nuevo players en la colección mediante el comando

```

ddddd.pdb.players.insertOne({ Id_player: "PLAE100002", name: "Nuevo jugador", nationality: "Colombia", last_name: "Prueba", document: "1882716662", bir
thdate: "20-07-2024" };
{
    "acknowledged" : true,
    "insertedId" : ObjectId("66673349eb4c95c86bcf44ab")
}
mongos> shard1 = new Mongo("localhost:20000")
connection to localhost:20000
mongos> shard1DB = shard1.getDB("Championship")
Championship
mongos> shard1DB.players.count()
500001
mongos>

```

Ahora contiene 500001 como se puede ver en la imagen que sigue, incluido el nuevo registro insertado, es decir la otra mitad, por lo que podemos comprobar la distribución de registros en más de un shard

```

mongos> shard2= new Mongo("localhost:20001")
connection to localhost:20001
mongos> shard2DB = shard2.getDB("Championship")
Championship
mongos> shard1DB.players.count()
500001
mongos>

```

```
mongos> shard3= new Mongo("localhost:20002")
connection to localhost:20002
mongos> shard3DB = shard3.getDB("Championship")
Championship
mongos> shard3DB.players.count()
0
mongos>
```

4. Caso de prueba 4: Eliminación de datos

Eliminar un equipo de la colección mediante el comando

Con el objetivo de validar la distribución del particionamiento propuesto para el campo Id_player en la colección "players" de la base de datos "Championship" de MongoDB, procedimos a eliminar un documento y realizar las siguientes acciones

```
mongos> db.players.deleteOne({Id_player: "PLAE1000002"})
{ "acknowledged" : true, "deletedCount" : 1 }
mongos>
```

```
mongos> shard1 = new Mongo("localhost:20000")
connection to localhost:20000
mongos> shard1DB = shard1.getDB("Championship")
Championship
mongos> shard1DB.players.count()
500000
mongos>
```

```
mongos> shard3= new Mongo("localhost:20002")
connection to localhost:20002
mongos> shard3DB = shard3.getDB("Championship")
Championship
mongos> shard3DB.players.count()
0
```

Video

<https://vimeo.com/956100507/72f0ce6561?share=copy>

Conclusiones

En conclusión, la ejecución exitosa de todos los casos de prueba subraya la eficacia del particionamiento en la administración de datos en escenarios de alta carga y complejidad. La implementación precisa del particionamiento facilita la distribución equitativa de datos en múltiples servidores, optimizando así el rendimiento del sistema y garantizando tiempos de respuesta rápidos para consultas y actualizaciones, incluso en entornos de alta demanda como eventos deportivos masivos. Esto valida la importancia del particionamiento como una estrategia fundamental en la arquitectura de bases de datos para garantizar la escalabilidad, la disponibilidad y la eficiencia operativa en situaciones de desafío tecnológico.

Bibliografía

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

Capítulo 1 (Introducción a las bases de datos NoSQL) y del Capítulo 2 (Conceptos Básicos) del libro de Sarasa, A. (2016) Introducción a las bases de datos NoSQL usando MongoDB. Editorial UOC, disponible en: <https://elibro.net/es/lc/biblioibero/titulos/58524>

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