ACTIVIDAD 3 - CONCEPTOS Y COMANDOS BÁSICOS DEL PARTICIONAMIENTO EN BASES DE DATOS NOSQL

ANDREY LEANDRO VARONA ORJUELA

LUIGY GILBERTO CICUA CABALLERO

CORPORACIÓN UNIVERSITARIA IBEROAMERICANA

FACULTAD DE INGENIERÍA

BASE DE DATOS AVANZADA

IBAGUE – TOLIMA

2 DE JUNIO DEL 2024

Disponibilidad 24/7 para el Torneo Deportivo de Tenis

El sistema debe estar disponible las 24 horas del día, los 7 días de la semana, para garantizar la satisfacción del usuario y el éxito del torneo de tenis. Para lograr esto, se deben cumplir los siguientes requerimientos no funcionales:

1. **Tiempo de actividad del sistema**: El sistema debe estar en funcionamiento continuo durante todo el evento, sin interrupciones, y disponible en todo momento para los usuarios.
2. **Monitoreo constante del sistema**: Se deben realizar monitoreos constantes del sistema para detectar cualquier problema que pueda afectar su disponibilidad y solucionarlos en tiempo real.
3. **Personal de soporte técnico disponible**: Se debe contar con personal de soporte técnico disponible las 24 horas del día, los 7 días de la semana, para responder rápidamente a cualquier problema técnico que pueda surgir.

Además, el sistema debe contar con:

- **Infraestructura de alta disponibilidad y escalabilidad** para soportar un alto volumen de tráfico.
- **Balanceador de carga** para distribuir el tráfico de manera equilibrada entre los servidores.
- **Configuración de servidores** que permita la actualización y el mantenimiento en vivo del sistema, sin interrupciones en el servicio.
- **Medidas de seguridad y recuperación de desastres** para garantizar la continuidad del servicio en caso de incidentes.

Redundancia

Para asegurar la máxima disponibilidad y fiabilidad del sistema durante el torneo de tenis, se implementarán las siguientes estrategias:

- **Redundancia de componentes críticos**: Todos los componentes críticos del sistema, como servidores, bases de datos, y redes, tendrán redundancias implementadas. Esto significa que si un componente falla, otro tomará su lugar inmediatamente, garantizando así la continuidad del servicio sin interrupciones.
- **Balanceo de carga redundante**: Se implementarán múltiples balanceadores de carga que trabajen en conjunto para distribuir el tráfico de manera equilibrada y, en caso de fallo de uno, otro puede asumir su función sin afectar la experiencia del usuario.
- **Infraestructura escalable**: La infraestructura estará diseñada para escalar horizontalmente, añadiendo más recursos según sea necesario para manejar el aumento de tráfico durante el torneo sin comprometer el rendimiento del sistema.

Con estas medidas de replicación y redundancia, junto con los requerimientos no funcionales previamente mencionados, se garantiza una alta redundancia y disponibilidad 24/7 del sistema para el torneo de tenis, asegurando así una experiencia fluida y satisfactoria para todos los participantes y espectadores.

Definiendo estrategia de particionamiento de bade de datos:

• **Evaluación de las necesidades de la aplicación**: Al comenzar el diseño del particionado de la base de datos (shard), es crucial examinar las necesidades de la aplicación. Esto incluye aspectos como el volumen de datos, la cantidad de solicitudes simultáneas, el tiempo de respuesta, la escalabilidad y la disponibilidad.

• **Determinación de las claves de partición**: Las claves de partición son los atributos o campos que se emplearán para repartir los datos entre los nodos. Es vital determinar las claves de partición correctas para asegurar una distribución de datos justa y eficaz.

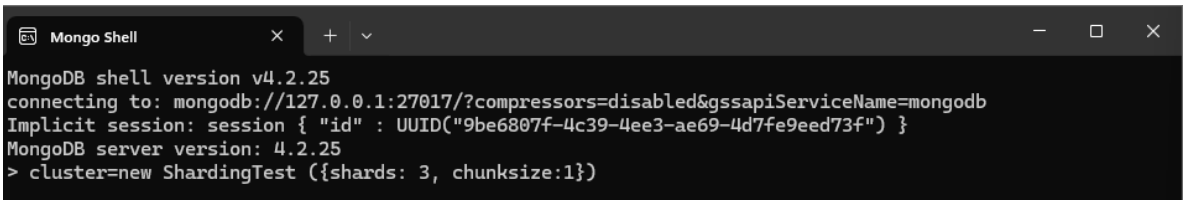
• **Elección del método de particionamiento**: MongoDB proporciona varios métodos de particionamiento, como el particionamiento por rango, el particionamiento por hash y el particionamiento por zona geográfica. Es imprescindible elegir el método de particionamiento más adecuado en función de las necesidades de la aplicación.

• Definición **de la estrategia de replicación**: La replicación es un proceso fundamental para asegurar la disponibilidad e integridad de los datos. Es necesario definir una estrategia de replicación apropiada que contemple el número de copias de los datos, la localización de las copias y la frecuencia de sincronización.

• Monitoreo **y ajuste del particionamiento**: Tras la implementación del particionamiento, es esencial supervisar el rendimiento de la base de datos y ajustar la configuración según sea necesario para optimizar la escalabilidad, el rendimiento y la disponibilidad.

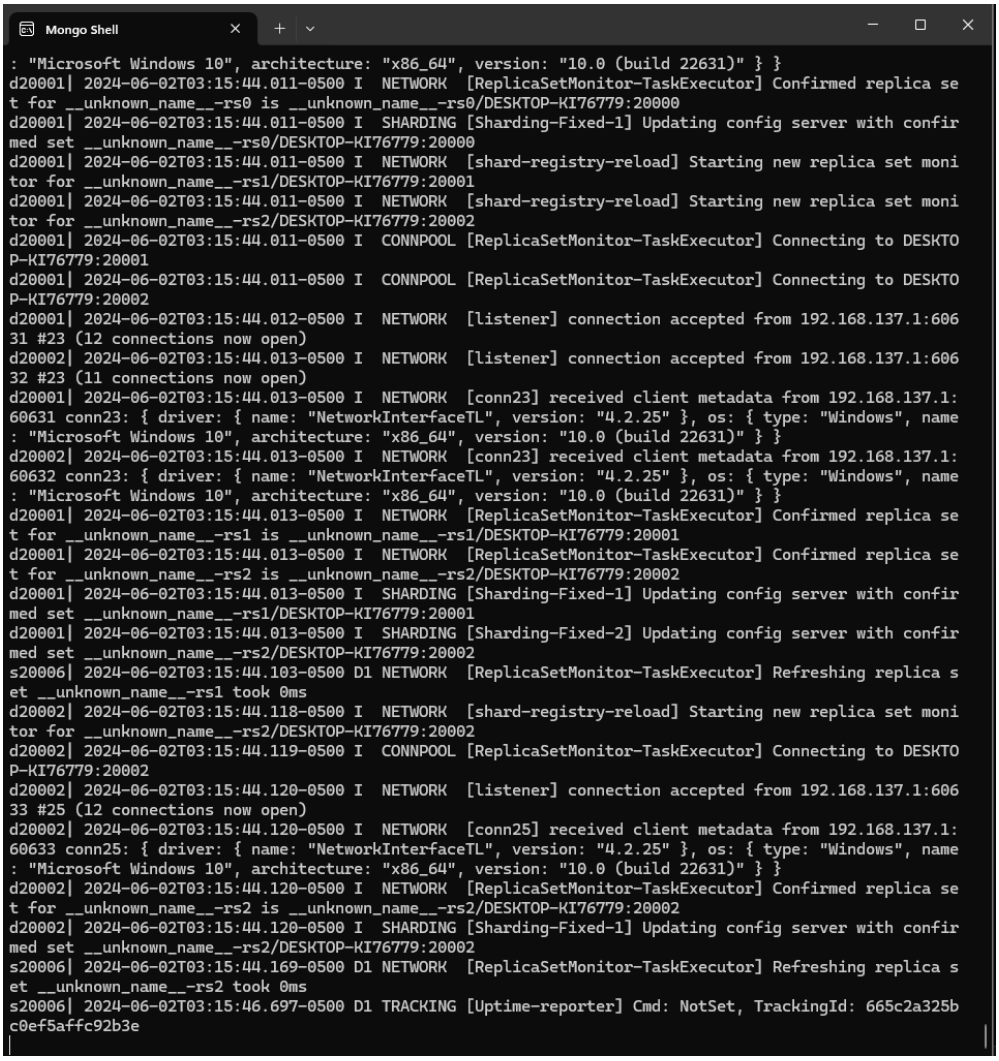
Replicación:

Pare realizar la replicación, primer abriremos una consola mongodb, para poder ejecutar el siguiente comando:



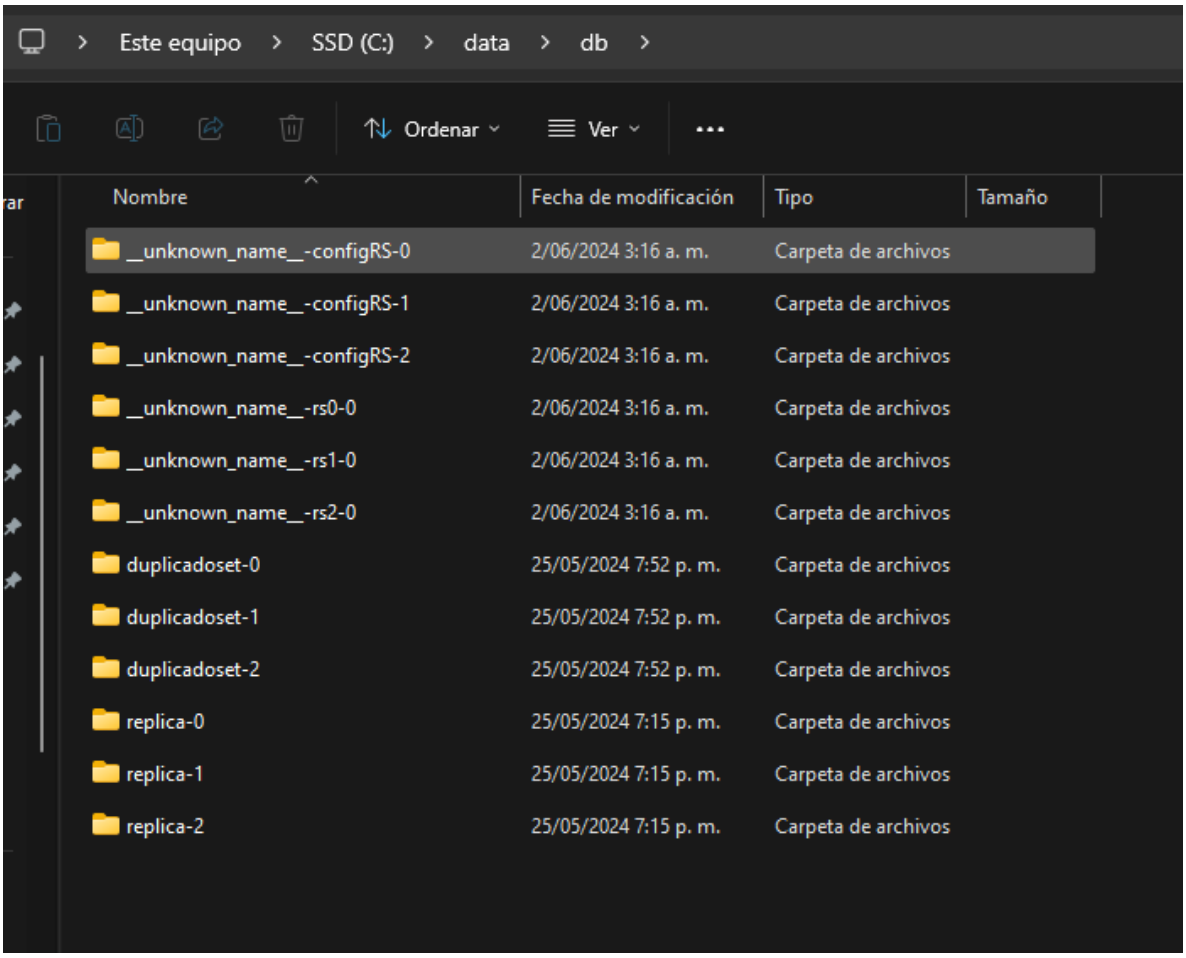
Una vez ejecutado este comando, podremos visualizar en consola el arranque del servicio, como

podemos observar a continuación:



Luego de la espera de la ejecución, podremos ver las instancias que fueron creadas, en la ruta

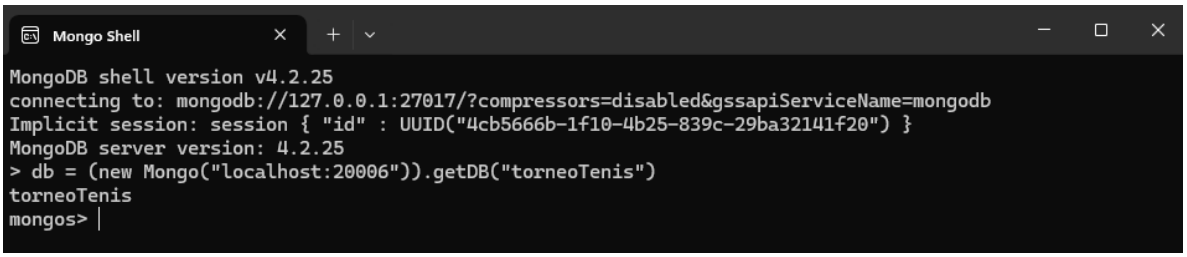
C:\data\db, como se puede apreciar a continuación:



Insertando datos sobre el balanceador:

Para este siguiente paso es muy importante tener en cuenta, que donde ejecutamos el primer comando, esta consola no deberá ser cerrada, porque de ser así tendremos que iniciar todo nuevamente, por lo cual, para insertar datos en el Sharding, deberemos abrir otra nueva consola

de mongodb, una vez realizado esto, ejecutaremos el siguiente comando contra el balanceador:



Ahora realizaremos la inserción de registros o colecciones en la tabla deportistas, de la siguiente manera:

```
mong    { "nombre": "Rafael Nadal", "edad": 35, "nacionalidad": "Español", "ranking": 3, "partidos_jug
ados    { "nombre": "Serena Williams", "edad": 39, "nacionalidad": "Estadounidense", "ranking": 8, "pa
rtid    { "nombre": "Novak Djokovic", "edad": 34, "nacionalidad": "Serbio", "ranking": 1, "partidos_ju
gado    { "nombre": "Ashleigh Barty", "edad": 25, "nacionalidad": "Australiana", "ranking": 2, "partid
os_j    { "nombre": "Roger Federer", "edad": 40, "nacionalidad": "Suizo", "ranking": 7, "partidos_juga
dos": 20 }"nombre": "Roger Federer", "edad": 40, "nacionalidad": "Suizo", "ranking": 7, "partidos_juga
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("665c2c4ba536263e44e73ad7"),
                ObjectId("665c2c4ba536263e44e73ad8"),
                ObjectId("665c2c4ba536263e44e73ad9"),
                ObjectId("665c2c4ba536263e44e73ada"),
                ObjectId("665c2c4ba536263e44e73adb")
        ]
}
mongos>
```

Realizaremos 5 inserciones adicionales, pero en la tabla equipos y entrenadores, con el siguiente comando:

Luego de insertados las colecciones, nos dará un mensaje de confirmación como el siguiente:

```
mongos> db.entrenadores.insertMany([
...     { "nombre": "Carlos Moya", "equipo": "Equipo A", "especialidad": "Tenis de Campo" },
...     { "nombre": "Patrick Mouratoglou", "equipo": "Equipo C", "especialidad": "Técnica y Preparació
n Fí    { "nombre": "Craig Tyzzer", "equipo": "Equipo D", "especialidad": "Coordinación y Estrategia"
},.     { "nombre": "Severin Luthi", "equipo": "Equipo E", "especialidad": "Táctica y Preparación Físi
ca" }   { "nombre": "Severin Luthi", "equipo": "Equipo E", "especialidad": "Táctica y Preparación Físi
... ])
{
        "acknowledged" : true,
        "insertedIds" : [
                ObjectId("665c2d49a536263e44e73ae1"),
                ObjectId("665c2d49a536263e44e73ae2"),
                ObjectId("665c2d49a536263e44e73ae3"),
                ObjectId("665c2d49a536263e44e73ae4"),
                ObjectId("665c2d49a536263e44e73ae5")
        ]
}
```

Ahora en este paso, realizaremos la verificación de la inserción de los datos, de la siguiente manera:

```
mongos> db.deportistas.count()
5
mongos> db.equipos.count()
5
mongos> db.entrenadores.count()
5
```

Se puede evidenciar que, en las tres colecciones diferentes, se han insertado los registros correctamente.

Verificación de la distribución de datos entre los nodos:

A través del balanceador, obtenemos una visión completa de todos los datos, lo que nos facilita ver cómo se ha repartido la información entre los distintos nodos de manera eficiente. Para interactuar con los procesos individuales que componen el Sharp, podemos iniciar una consola de Mongo independiente y establecer conexión con cada uno de los nodos utilizando distintos objetos. Nos enlazamos al primer nodo y llevamos a cabo la comprobación de la inserción de los registros.

Primer Nodo

```
mongos> shard1 = new Mongo("localhost:20006")
connection to localhost:20006
mongos> shard1DB = shard1.getDB("torneoTenis")
torneoTenis
mongos> shard1DB.deportistas.count()
5
```

Segundo Nodo

```
mongos> shard2 = new Mongo("localhost:20000")
connection to localhost:20000
mongos> shard2DB = shard2.getDB("torneoTenis")
torneoTenis
mongos> shard2DB.deportistas.count()
0
```

Tercer Nodo

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

Después de realizar una comprobación, se ha determinado que todos los documentos que se han insertado en la colección Autores se han almacenado únicamente en el primer nodo de

Shard, mientras que el segundo y tercer nodo no tienen ningún documento en dicha colección.

Basándonos en estos resultados, uno podría suponer que el particionamiento de datos no está funcionando correctamente, dado que no se ha logrado una distribución equitativa de los datos entre los distintos nodos del Shard, resultando en un desbalance notable. Sin embargo, es crucial recordar que la distribución de datos entre los nodos no se activa automáticamente al crear el objeto para las pruebas de Sharding en MongoDB. En otras palabras, si no se ha configurado explícitamente el particionamiento de datos, es común que la carga de información no se distribuya entre todos los nodos del Shard. Por lo tanto, antes de asumir que el Sharding en MongoDB no está funcionando, es esencial verificar si se ha habilitado el particionamiento de datos y asegurarse de que la distribución de los documentos se esté llevando a cabo de manera adecuada. Para activar el Sharding, lo que haremos es volver a nuestro balanceador y podremos validar el estado del sardina de la siguiente forma:

```
mongos> sh.status()
--- Sharding Status ---
  sharding version: {
        "_id" : 1,
        "minCompatibleVersion" : 5,
        "currentVersion" : 6,
        "clusterId" : ObjectId("665c2a0b83cb4d0e6b7286f9")
  }
  shards:
        {  "_id" : "__unknown_name__-rs0",  "host" : "__unknown_name__-rs0/DESKTOP-KI76779:20000",  "st
ate" : 1 }
        {  "_id" : "__unknown_name__-rs1",  "host" : "__unknown_name__-rs1/DESKTOP-KI76779:20001",  "st
ate" : 1 }
        {  "_id" : "__unknown_name__-rs2",  "host" : "__unknown_name__-rs2/DESKTOP-KI76779:20002",  "st
ate" : 1 }
  active mongoses:
        "4.2.25" : 1
  autosplit:
        Currently enabled: no
  balancer:
        Currently enabled:  no
        Currently running:  no
        Failed balancer rounds in last 5 attempts:  0
        Migration Results for the last 24 hours:
                No recent migrations
  databases:
        {  "_id" : "config",  "primary" : "config",  "partitioned" : true }
                config.system.sessions
                        shard key: { "_id" : 1 }
                        unique: false
                        balancing: true
                        chunks:
                                __unknown_name__-rs0    1
                        { "_id" : { "$minKey" : 1 } } -->> { "_id" : { "$maxKey" : 1 } } on : __unknown
_name__-rs0 Timestamp(1, 0)
        {  "_id" : "torneoTenis",  "primary" : "__unknown_name__-rs1",  "partitioned" : false,  "versio
n" : {  "uuid" : UUID("a5b043a5-eb6c-422f-8236-49fb5a4548b9"),  "lastMod" : 1 } }
```

En este escenario, se ha confirmado que el balanceador, que es el proceso responsable de distribuir la carga de datos entre los distintos nodos del Shard, no está en funcionamiento. Esto se puede verificar a través de las propiedades del balanceador: Currently enabled: no y balancer: Currently running: no.

Además, se ha notado que el particionamiento de datos para el shard0000, que es el nombre asignado automáticamente al Shard al crear el ShardingTest, no está habilitado, como lo señala la propiedad partitioned: false. Para habilitar el Sharding, se puede emplear la función enableSharding() en la base de datos que se desea repartir entre todos los nodos del Shard. Esta función permitirá habilitar el particionamiento de datos y el balanceador, lo que facilitará una distribución equitativa de los documentos entre los distintos nodos del Shard. Una vez que se ha habilitado el Sharding, se puede utilizar la función sh.shardCollection() para particionar la colección específica que se desea distribuir entre los nodos.

```
mongos> sh.enableSharding(torneoTenis)
2024-06-02T03:52:53.935-0500 E  QUERY    [js] uncaught exception: ReferenceError: torneoTenis is not de
fined :
@(shell):1:1
mongos> sh.enableSharding("torneoTenis")
{
        "ok" : 1,
        "operationTime" : Timestamp(1717318443, 3),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1717318443, 3),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
```

Previamente habilitaremos los fragmentos para que se pueda crear un índice en la clave que desea usar como shard:

```
mongos> db.deportistas.ensureIndex({deportista :1})
{
        "raw" : {
                "__unknown_name__-rs1/DESKTOP-KI76779:20001" : {
                        "createdCollectionAutomatically" : false,
                        "numIndexesBefore" : 1,
                        "numIndexesAfter" : 2,
                        "ok" : 1
                }
        },
        "ok" : 1,
        "operationTime" : Timestamp(1717321702, 2),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1717321702, 2),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
```

Ahora se puede determinar la colección de la siguiente manera.

```
mongos> sh.shardCollection("torneoTenis.deportista", {deportista: 1})
{
        "collectionsharded" : "torneoTenis.deportista",
        "collectionUUID" : UUID("b79d6f7d-6881-4178-98de-150ce50536e9"),
        "ok" : 1,
        "operationTime" : Timestamp(1717321829, 12),
        "$clusterTime" : {
                "clusterTime" : Timestamp(1717321829, 12),
                "signature" : {
                        "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
                        "keyId" : NumberLong(0)
                }
        }
}
```

Enlace repositorio: https://github.com/LuigyC/TorneoDeportivo/tree/main/Actividad%203

Enlace Video: Actividades-20240602_181002-Grabación de la reunión.mp4