



Actividad 3 - Conceptos y Comandos básicos del particionamiento en bases de datos No SQL

Ingeniería de software (Corporación Universitaria Iberoamericana)



Escanea para abrir en Studocu

COMANDOS Y CONCEPTOS BASICOS DEL PARTICIONAMIENTO DE BASE DE DATOS NODQL

Estudiante:

KRHISTIAN ALEJANDRO GONZALEZ DUARTE

OSCAR MAURICIO GONZALEZ MONTENEGRO

Universidad:

Corporación Universitaria Iberoamericana

Profesor:

WILIAM RUIZ

Bogotá 16/04/2023

**COMANDOS Y CONCEPTOS BASICOS EL PARTICIONAMIENTO DE BASE DE
DATOS NOSQL**

AUTOR:

KRHISTIAN ALEJANDRO GONZALEZ DUARTE

OSCAR MAURICIO GONZALEZ MONTENEGRO

Presentado para optar el título de: Ingeniero de Software

Profesora:

WILIAM RUIZ

Universidad:

Corporación Universitaria Iberoamericana

Facultad: Diseño de Sistemas

Bogotá.

Bogotá 16/04/2023

Contenido

REQUERIMIENTOS NO FUNCIONALES EL PARTICIONAMIENTO DE BASE DE DATOS MONGODB	4
Requerimientos no funcionales:	4
Escalabilidad.....	4
Rendimiento:	4
Disponibilidad	4
Seguridad:.....	5
Escalabilidad geográfica:	5
Compatibilidad:	5
Definiendo estrategia de particionamiento de bade de datos:	5
PASO A PASO DE CREACION EL PARTICIONAMEITNO DE BASE DE DATOS MONGODB.....	6
Replicación:.....	6
Insertando datos sobre el balanceador:	8

Comprobación de la distribución de los datos en los nodos:	10
Primer nodo	11
Segundo Nodo	11
Tercer Nodo.....	11
Activando el Sharding	12
LINK REPOSITORIO GITHUB	14
LINK VIDEO.....	14

REQUERIMIENTOS NO FUNCIONALES EL PARTICIONAMIENTO DE BASE DE DATOS MONGODB

Requerimientos no funcionales:

Escalabilidad: El particionamiento debe permitir que la base de datos MongoDB crezca horizontalmente para manejar un mayor volumen de datos y una mayor cantidad de solicitudes concurrentes.

Rendimiento: El particionamiento debe estar diseñado para mejorar el rendimiento de la base de datos MongoDB al reducir la latencia de las consultas y mejorar el tiempo de respuesta.

Disponibilidad: El particionamiento debe garantizar la disponibilidad de la base de datos MongoDB en caso de fallos de hardware o software, y permitir la recuperación de datos sin pérdida.

Seguridad: El particionamiento debe proporcionar una separación adecuada de datos confidenciales y restringir el acceso a datos sensibles a usuarios no autorizados.

Costo: El particionamiento debe estar diseñado para minimizar los costos de infraestructura, almacenamiento y mantenimiento de la base de datos MongoDB.

Escalabilidad geográfica: El particionamiento debe permitir que la base de datos MongoDB se distribuya geográficamente para reducir la latencia de las consultas en diferentes regiones y cumplir con los requisitos de privacidad y cumplimiento normativo.

Compatibilidad: El particionamiento debe ser compatible con las herramientas y tecnologías de desarrollo utilizadas para interactuar con la base de datos MongoDB.

Definiendo estrategia de particionamiento de base de datos:

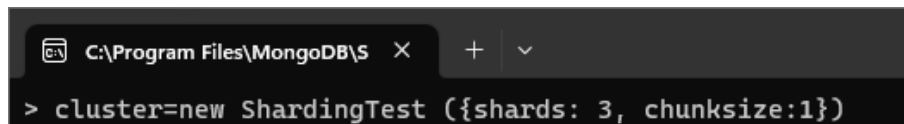
1. Analizar los requerimientos de la aplicación: Antes de comenzar a diseñar el particionamiento de la base de datos MongoDB, es necesario analizar los requerimientos de la aplicación, incluyendo el volumen de datos, el número de solicitudes concurrentes, el tiempo de respuesta, la escalabilidad y la disponibilidad.
2. Identificar las claves de partición: Las claves de partición son los atributos o campos que se utilizarán para distribuir los datos entre los nodos. Es importante identificar las claves de partición adecuadas para garantizar que los datos se distribuyan de manera equitativa y eficiente.
3. Seleccionar el método de particionamiento: MongoDB ofrece diferentes métodos de particionamiento, incluyendo el particionamiento por rango, el particionamiento por hash y el particionamiento por zona geográfica. Es necesario seleccionar el método de particionamiento adecuado en función de los requerimientos de la aplicación.

4. Establecer la estrategia de replicación: La replicación es un proceso esencial para garantizar la disponibilidad y la integridad de los datos. Es necesario establecer una estrategia de replicación adecuada que incluya el número de copias de los datos, la ubicación de las copias y la frecuencia de la sincronización.
5. Monitorear y ajustar el particionamiento: Una vez implementado el particionamiento, es necesario monitorear el rendimiento de la base de datos y ajustar la configuración según sea necesario para mejorar la escalabilidad, el rendimiento y la disponibilidad.

PASO A PASO DE CREACION EL PARTICIONAMEITNO DE BASE DE DATOS MONGODB

Replicación:

Para realizar la replicación, primero abriremos una consola mongodb, para poder ejecutar el siguiente comando:



```
C:\Program Files\MongoDB\> cluster=new ShardingTest ({shards: 3, chunksize:1})
```

Una vez ejecutado este comando, podremos visualizar en consola el arranque del servicio, como podemos observar a continuación:

```

C:\Program Files\MongoDB\B S  X  +  v
    print("ReplSetTest stopSet deleted all dbpaths");
  }

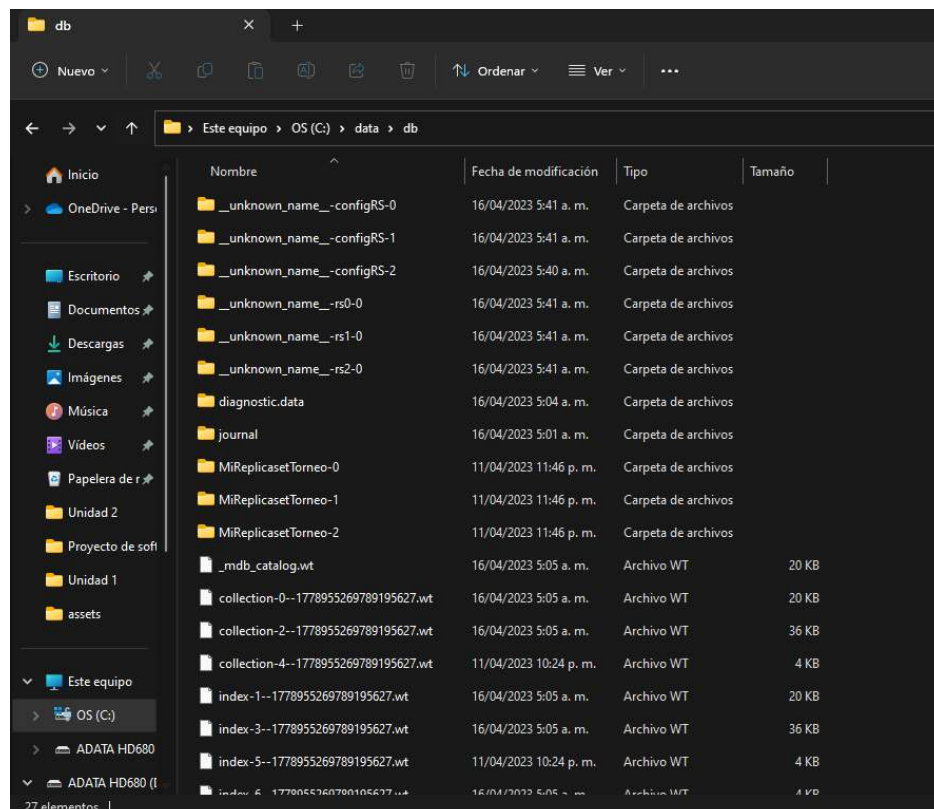
  _forgetReplSet(this.name);

  print('ReplSetTest stopSet *** Shut down repl set - test worked ****');
},
    "usesBridge" : function() {
    return _useBridge;
  },
    "waitForState" : function(node, state, timeout, reconnectNode) {
    _waitForIndicator(node, state, "state", timeout, reconnectNode);    },
    "waitForMaster" : function(timeout) {
    var master;
    assert.soonNoExcept(function() {
    return (master = self.getPrimary());
    }, "waiting for master", timeout);

    return master;
  },
    "name" : "__unknown_name__-rs0",
    "useHostName" : true,
    "host" : "Alejo",
    "oplogSize" : 40,
    "useSeedList" : false,
    "keyFile" : undefined,
    "protocolVersion" : undefined,
    "waitForKeys" : false,
    "seedRandomNumberGenerator" : true,
    "isConfigServer" : undefined,

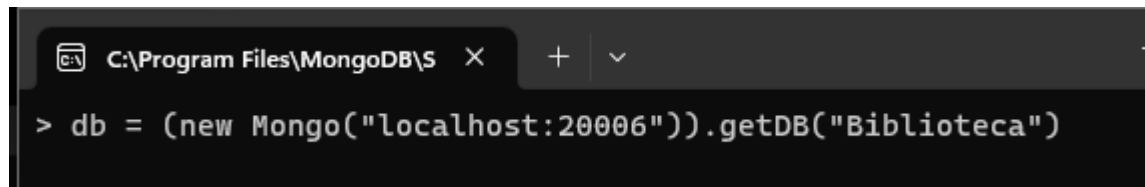
```

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:



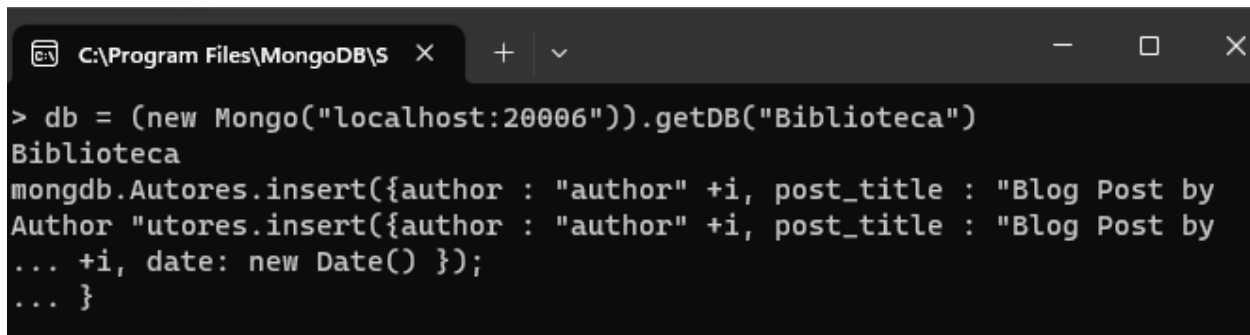
Insertando datos sobre el balanceador:

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:



```
C:\Program Files\MongoDB\S > db = (new Mongo("localhost:20006")).getDB("Biblioteca")
```

Ahora realizaremos la inserción de 10.000 registros o colecciones en la tabla autores, de la siguiente manera:



```
C:\Program Files\MongoDB\S > db = (new Mongo("localhost:20006")).getDB("Biblioteca")
Biblioteca
mongodb.Autores.insert({author : "author" +i, post_title : "Blog Post by
Author "utores.insert({author : "author" +i, post_title : "Blog Post by
... +i, date: new Date() });
... }
```

Cabe recalcar que este proceso puede tardar un poco, luego de que carguen todos los registros, nos dará un mensaje de confirmación de las inserciones

Realizaremos 5 inserciones adicionales, pero en la tabla editoriales, con el siguiente comando:

```

C:\Program Files\MongoDB\> >db.Editoriales.insertMany( [
... {id_editorial:9, nombre: 'McGraw Hill', pais: 'USA', grupo: 'McCallun y Cia.'},
... {id_editorial:10, nombre:'Addison Wesley', pais: 'USA', grupo: 'McCallun y
uncaught exception: SyntaxError: expected expression, got '>' :
@(shell):1:0
> Cia.'},
uncaught exception: SyntaxError: '' literal not terminated before end of script :
@(shell):1:7
> {id_editorial:11,nombre:'La oveja negra', pais: 'Colombia', grupo: 'Mejía y Cia'
... },
... {id_editorial:12, nombre: 'Aguilar', pais: 'México', grupo: 'Aguilar y Cia'
... },
... {id_editorial:13, nombre: 'Alfaguara', pais: 'Colombia', grupo: 'Grupo
uncaught exception: SyntaxError: unexpected token: ':' :
@(shell):1:23
> Empresarial'}}
uncaught exception: SyntaxError: '' literal not terminated before end of script :
@(shell):1:13
> ]];

```

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

```

undb.Editoriales.insertMany([id_editorial:9, nombre: 'McGraw Hill', pais: 'USA', grupo: 'McCallun y Cia.'],{id_editorial:10, nombre:'Addison Wesley
', pais: 'USA', grupo: 'McCallun y Cia.'],{id_editorial:11,nombre:'La oveja negra', pais: 'Colombia', grupo: 'Mejía y Cia'}, {id_editorial:12, nombr
e: 'Aguilar', pais: 'México', grupo: 'Aguilar y Cia'},{id_editorial:13, nombre: 'Alfaguara', pais: 'Colombia', grupo: 'Grupo Empresarial'}]); nombr
e:
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("643be2c98424e1e8fc1fa33d"),
    ObjectId("643be2c98424e1e8fc1fa33e"),
    ObjectId("643be2c98424e1e8fc1fa33f"),
    ObjectId("643be2c98424e1e8fc1fa340"),
    ObjectId("643be2c98424e1e8fc1fa341")
  ]
}
>

```

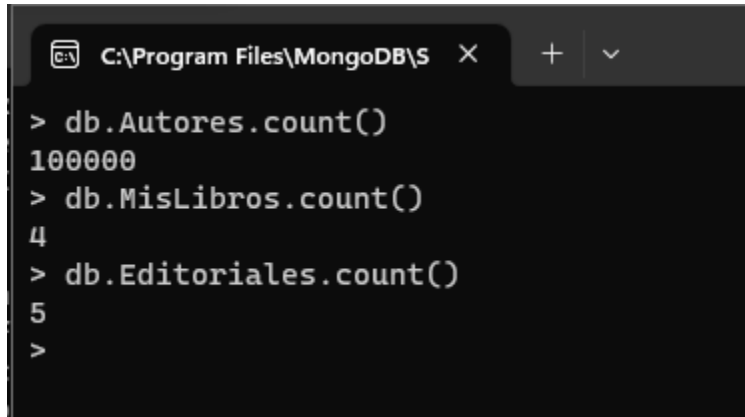
Ingresaremos posteriormente cuatro registros adicionales, pero en la colección MisLibros, de la siguiente manera:

```

C:\Program Files\MongoDB> dddb.MisLibros.insertMany([id_libro:1, nombre:'El Aleph', autor: 'Jorge Luis Borges', editoriales:['Planeta', 'Siglo XXI'],{id_libro:2, nombre:'El
Qdb.MisLibros.insertMany([id_libro:1, nombre:'El Aleph', autor: 'Jorge Luis Borges', editoriales:['Planeta', 'Siglo XXI'],{id_libro:2, nombre:'El
Quijote de la Mancha', autor: 'Miguel de Cervantes Saavedra', editoriales: ['Planeta']},{id_libro:3, nombre: 'Cien años de soledad', autor: 'Gabrie
l García Márquez',editoriales: ['Oveja Negra']},{id_libro:4, nombre: 'Crimen y castigo', autor: 'Fiodor Dostoyeski',editoriales: ['Addison Wesley ']}]);ia Márquez',editoriales: ['Oveja Negra']},{id_libro:4, nombre: 'Crimen y castigo', autor: 'Fiodor Dostoyeski',editoriales: ['Addison Wesley ']}]);
{
  "acknowledged" : true,
  "insertedIds" : [
    ObjectId("643be47c8424e1e8fc1fa342"),
    ObjectId("643be47c8424e1e8fc1fa343"),
    ObjectId("643be47c8424e1e8fc1fa344"),
    ObjectId("643be47c8424e1e8fc1fa345")
  ]
}
>

```

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



```
C:\Program Files\MongoDB\> db.Autores.count()
100000
> db.MisLibros.count()
4
> db.Editoriales.count()
5
>
```

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

Comprobación de la distribución de los datos en los nodos:

Desde el equilibrador, tenemos una perspectiva global de todos los datos, lo que nos permite observar cómo se ha distribuido la información entre los diferentes nodos de manera eficaz. Para acceder a los procesos individuales que conforman el Shard, podemos abrir una consola de Mongo separada y conectarnos a cada uno de los nodos utilizando objetos diferentes.

Nos conectamos al primer nodo y realizamos la verificación de la inserción de los registros:

Primer nodo

```
C:\Program Files\MongoDB\S  X  +  v
> shard1 = new Mongo("localhost:20000")
connection to localhost:20000
> shard1DB = shard1.getDB("Biblioteca")
Biblioteca
> shard1DB.Autores.count()
800000
>
```

Lo mismo lo realizamos para el segundo y tercer nodo

Segundo Nodo

```
> shard2= new Mongo("localhost:20001")
connection to localhost:20001
> shard2DB = shard2.getDB("Biblioteca")
Biblioteca
> shard2DB.Autores.count()
0
>
```

Tercer Nodo

```
> shard3= new Mongo("localhost:20002")
connection to localhost:20002
> shard3DB = shard3.getDB("Biblioteca")
Biblioteca
> shard3DB.Autores.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 del Shard, mientras que el segundo y tercer nodo no tienen ningún documento en dicha colección.

A partir de estos resultados, se podría pensar que el particionado de datos no está funcionando adecuadamente, ya que no se ha producido una distribución equitativa de los datos entre los diferentes nodos del Shard, lo que provoca un desequilibrio evidente.

Sin embargo, es importante tener en cuenta que la distribución de datos entre los nodos no se activa de manera automática al crear el objeto para las pruebas de Sharding en MongoDB. Es decir, si no se ha configurado explícitamente el particionado de datos, es normal que no se reparta la carga de información entre todos los nodos del Shard. Por lo tanto, antes de concluir que el Sharding en MongoDB no funciona, es necesario verificar si se ha activado el particionado de datos y asegurarse de que la distribución de los documentos se esté realizando correctamente.

Activando el Sharding

Para la activación de este, lo que realizaremos es regresar a nuestro balanceador y podremos validar el estado del sharding, de la siguiente manera:

```
sharding version: {
  "id" : 1,
  "minCompatibleVersion" : 5,
  "currentVersion" : 6,
  "clusterId" : ObjectId("634829124a540bec64ad614d")
}
shards:
  { "_id" : "unknown_name--rs0", "host" : "unknown_name--rs0/Metal2022:20000", "state" : 1 }
  { "_id" : "unknown_name--rs1", "host" : "unknown_name--rs1/Metal2022:20001", "state" : 1 }
  { "_id" : "unknown_name--rs2", "host" : "unknown_name--rs2/Metal2022:20002", "state" : 1 }
active mongoses:
  "4.2.23-rc0" : 1
autosplit:
  Currently enabled: no
```

En este caso se ha verificado que el balanceador, que es el proceso encargado de distribuir la carga de datos entre los diferentes nodos del Shard, no está activo. Esto se puede comprobar a través de las propiedades balancer: Currently enabled: no y balancer: Currently running: no.

Además, se ha observado que el particionado de datos para el shard0000, que es el nombre que se le ha dado automáticamente al Shard al crear el ShardingTest, no está activado, como lo indica la propiedad `partitioned: false`.

Para activar el Sharding, se puede utilizar la función `enableSharding()` sobre la base de datos que se desea distribuir entre todos los nodos del Shard. Esta función permitirá activar el particionado de datos y el balanceador, lo que permitirá una distribución equitativa de los documentos entre los diferentes nodos del Shard. Una vez que se ha activado el Sharding, se puede utilizar la función `sh.shardCollection()` para particionar la colección específica que se desea distribuir entre los nodos.



```
C:\Program Files\MongoDB\bin>enableSharding

{
  "raw" : {
    "___unknown_name___rs1/Meta12022:20001" : {
      "createdCollectionAutomatically" : false,
      "numIndexesBefore" : 1,
      "numIndexesAfter" : 2,
      "ok" : 1
    }
  },
  "ok" : 1,
  "operationTime" : Timestamp(1665683202, 7),
  "$clusterTime" : {
    "clusterTime" : Timestamp(1665683202, 7),
    "signature" : {
      "hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA="),
      "keyId" : NumberLong(0)
    }
  }
}
```

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

```
C:\Program Files\MongoDB\> db.Autores.ensureIndex({author : 1})
{
  "createdCollectionAutomatically" : false,
  "numIndexesBefore" : 1,
  "numIndexesAfter" : 2,
  "ok" : 1
}
```

Ahora se puede determinar la colección de la siguiente manera.

```
C:\Program Files\MongoDB\> sh.shardCollection("Biblioteca.Autores", {author: 1})
```

LINK REPOSITORIO GITHUB

Link: <https://github.com/1Alejo31/Actividad-1---Diseno-y-operaciones-CRUD-en-Bases-de-datos-NoSQL>

LINK VIDEO

Link: <https://www.youtube.com/watch?v=LInIHC4mbLo>