

Taller. N 4

Actividad 4: Pruebas de particionamiento de bases de datos NoSQL

**Nombres:**

Harvin Dubiel Acosta Triviño

NICOLAS VEGA CASTIBLANCO

JONATHAN NICOLAS MUÑOZ OSORIO

**Institución:**

Corporación Universitaria Iberoamericana Colombia

**Curso:**

Base de datos Avanzadas

**Instructor:**

WILLIAM RUIZ

**Fecha**

Junio 2024

# **Torneo de microfutbol NH Sports**

Introducción

Siguiendo con el proyecto anterior donde hablábamos de un torneo de microfutbol y la gestión de la información, realizaremos un proceso para llevar a cabo lo que es la replicación y así tener mayor disponibilidad de datos.

**Casos de Prueba de Replicación de Bases de Datos NoSQL**

|  |  |  |
| --- | --- | --- |
| Requisitos | PRUEBA | Resultado Esperado |
| Escalamiento horizontal  de datos. | Utilizaremos el comando sh.status() para consultar datos de partición y estado del clúster . | Muestra la distribución de documentos en los shards Implementados |
| Fragmentación de bases de datos. | Utilice el comando para consultar la distribución getShardDistribution(ShardDistribution() | Indica el espacio en  memoria asignada a cada shard y el peso que contienen |
| La escritura se debe  realizar desde un solo Servidor | Los intentos de escritura en nodos distintos al primero se comprobarán comenzando desde el primero. | No se puede escribir desde el segundo nodo, el primer nodo tiene permisos de escritura. |
| El grupo de partición  debe estar compuesto por servidores de  configuración,  enrutadores de consultas  y shards. | Se verifica que los puertos asignados a los servidores de configuración, enrutador y shards, se encuentren  activos usando el comanto netstat -a | Todos los puertos  asignados al conjunto de  replicación se encuentran  activos. |
| El sistema debe emplear  particionamiento basado  en hash. | con getShardDsitribution() se  revisan los pesos de los  chunks, se adicionan 3 documentos de forma sucesiva y se verifica  posteriormente los pesos | En la segunda prueba  debe aumentar los pesos  de los chunk  uniformemente. |

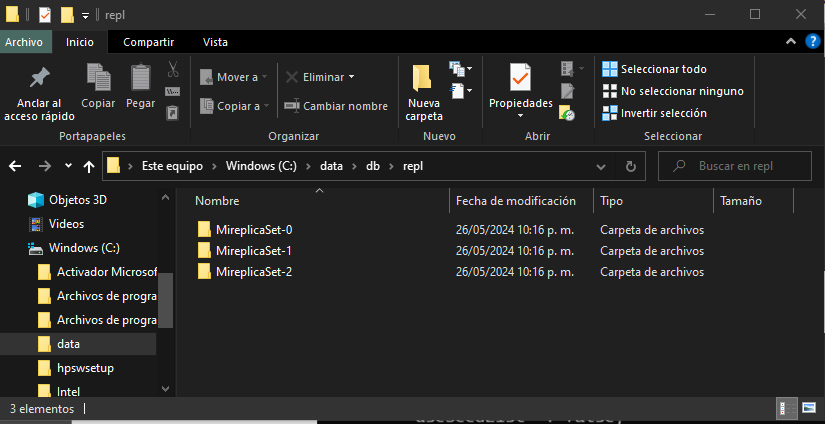
mpo de respuesta: Es necesario validar el tiempo de respuesta de la consulta de

datos en alguna de las colecciones, asegurándose de que sea inferior a un segundo.

Esto mejorará la eficacia del sistema y permitirá una mayor satisfacción del usuar

Pruebas de conversión de un replica set en un sharder clúster replicado

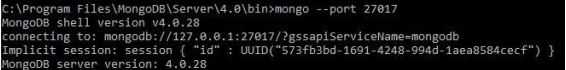
Creación de directorios:



Inicio de servidor primario



Activación shell servidor primario



Inicio de servidores y activación de shell secundarios

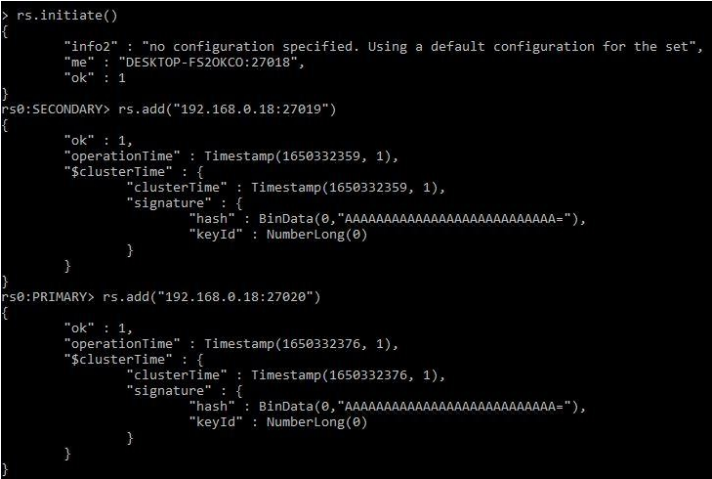








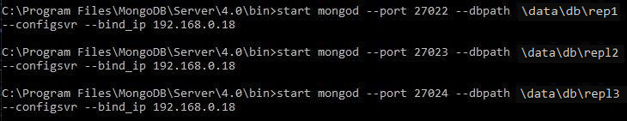
Creación de conjunto de replicas



Llamado a la base de datos



Inicio de servidores de configuración



Activación de servidores de configuración







Inicio enrutador



Respuesta enrutador



## Requerimientos no funcionales.

1. Como requerimiento principal es mantener el funcionamiento permanente de la base de datos
2. Disponibilidad: verificar que la información almacenada en la base de datos. Esto permitirá garantizar la disponibilidad y redundancia de los datos en caso de fallos en alguno de los nodos
3. Al estar los datos disponibles en muchos servidores a la vez, en menos posible que se presenten fallo. Ya que serán accesibles desde cualquier servidor nuestras replicas.
4. En caso de que el primario sea inaccesible, el algoritmo de elección hará un intento, para que el secundario quien tenga la prioridad más alta de las demás se encuentre disponible e invoque una elección y se elegirá un nuevo nodo Primario.
5. Disponibilidad: Se debe verificar que la información almacenada en la base de datos
6. se ha dividido físicamente en 2 shards y que la información de cada uno de ellos se
7. ha replicado como se hizo previamente con el esquema centralizado con réplicas.
8. Esto permitirá garantizar la disponibilidad y redundancia de los datos en caso de
9. fallos en alguno de los nodos o shards
10. Disponibilidad: Se debe verificar que la información almacenada en la base de datos
11. se ha dividido físicamente en 2 shards y que la información de cada uno de ellos se
12. ha replicado como se hizo previamente con el esquema centralizado con réplicas.
13. Esto permitirá garantizar la disponibilidad y redundancia de los datos en caso de
14. fallos en alguno de los nodos o shards
15. Tiempo de respuesta: se debe validar el tiempo de respuesta de la consulta de datos en alguna de las colecciones, asegurándose de que sea inferior a un segundo. Esto mejorará la eficacia del sistema y permitirá una mayor satisfacción del usuario.

Conclusiones:

1. Las particiones en el almacenamiento no relacional permiten que las consultas se indexen por rango o hash.

2. La capacidad de sombra se puede proporcionar configurando la ranura de cada chip.

3. Al particionar la base de datos, el costo de la máquina se puede reducir ya que es hipotético e ilimitado.

4.Es mejor distribuir los datos entre diferentes chips que el hash, pero el hash es mejor para distribuirlos por el conjunto de chips.

Link Video: <https://youtu.be/UZeKKFkjb3c>

Link Repositorio: <https://github.com/acosst97/Torneo-Deportivo.git>

Bibliografía.

* Sarasa, A. (2016). Introducción a las bases de datos NoSQL usando MongoDB. Editorial UOC. <https://elibro.net/es/lc/biblioibero/titulos/58524>
* https://web.archive.org/web/20110217092700/http://www.nosql.es/blog/