[](https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=2ahUKEwjp65HqrrngAhUMna0KHdKNAdgQjRx6BAgBEAU&url=https://eltentempie.wordpress.com/2013/06/19/presenta-udlap-programa-de-verano-para-ninos-jovenes-y-papas/&psig=AOvVaw01fiQVxQt5VtX8Q5zWmYrR&ust=1550169834224916)

**Laboratorio de Manejadores de Bases de Datos**

**Reporte de Práctica 1 y 2: MongoDB**

**Francisco Ramos Omaña 157330**

**Erick Siordia Nagaya 157504**

**Carlos Andrés Reyes Evangelista 157068**

**26 de febrero de 2019**

Contenido

[Práctica 1 3](#_Toc2029024)

[Objetivos 3](#_Toc2029025)

[Desarrollo 3](#_Toc2029026)

[Discusión 11](#_Toc2029027)

[Práctica 2 12](#_Toc2029028)

[Objetivos 12](#_Toc2029029)

[Desarrollo 12](#_Toc2029030)

[Parte 1 12](#_Toc2029031)

[Parte 2 14](#_Toc2029032)

[Discusión 18](#_Toc2029033)

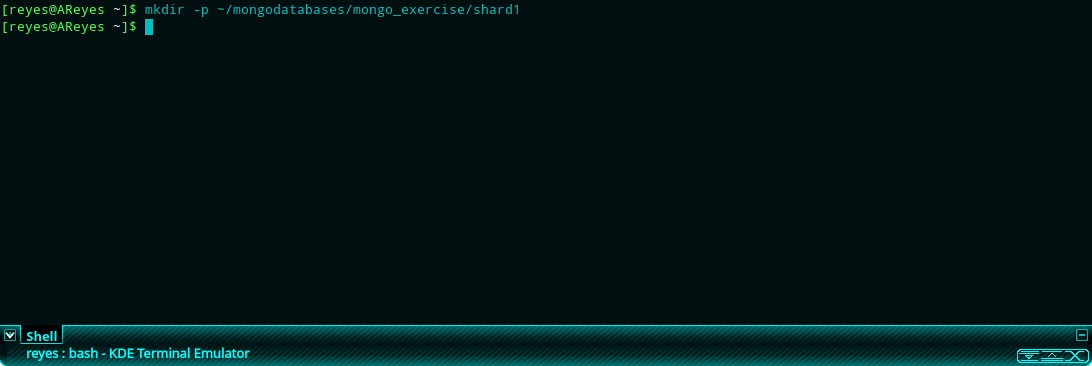
# Práctica 1

## Objetivos

Para la realización de esta práctica se hará uso de la versión 2.6.12 de MongoDB. El objetivo de esta sección en particular es empezar a utilizar MongoDB desde sus operaciones básicas para usuarios neófitos. En específico las tareas a ejecutar son: conectarse a una instancia activa de mongo, crear una base de datos, agregar ciertos registros específicos y realizar consultas, actualizaciones, búsquedas y eliminaciones sobre ellos.

## Desarrollo

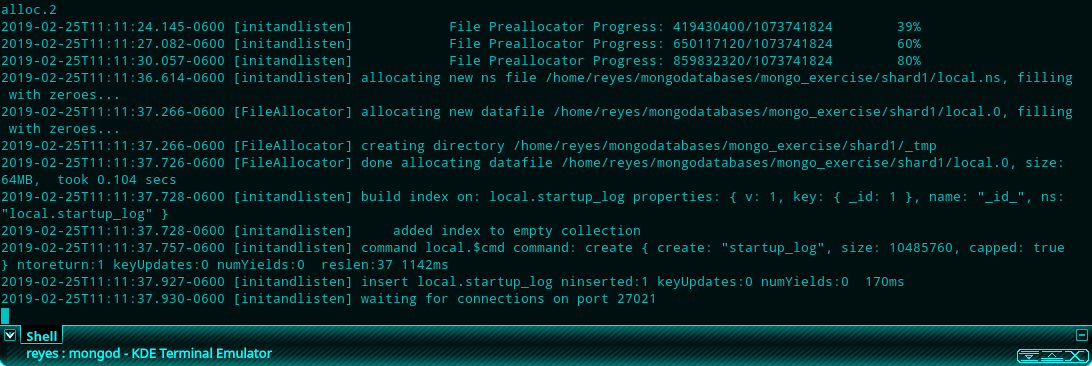
Para comenzar, se creó un directorio local donde guardaremos los datos de nuestra base de datos



Inmediatamente se utilizó la instrucción siguiente para crear una nueva instancia del MongoDB Server en un puerto específico



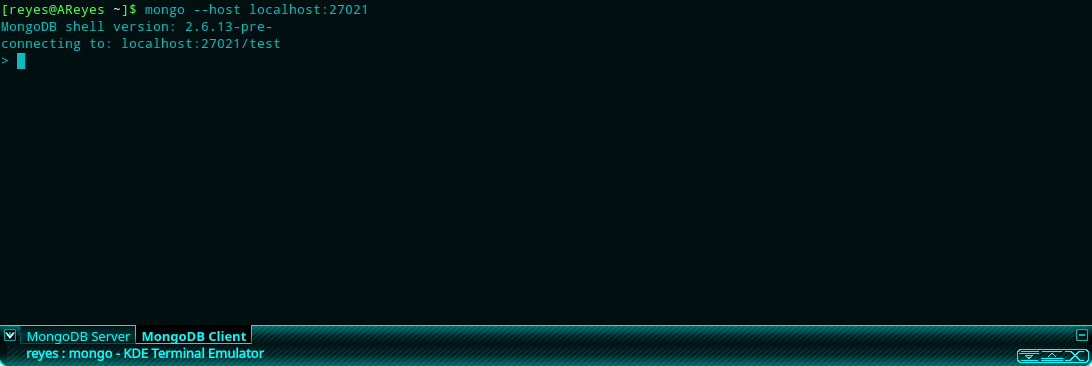
Cuando el daemon de MongoDB consigue iniciar su instancia se muestra la siguiente pantalla, donde destaca la espera de conexiones a través de ese puerto



Etiquetamos la terminal anterior como “MongoDB Server” y proseguiremos en una terminal independiente para realizar nuestra conexión a esa instancia utilizando para ello la siguiente instrucción.



Ésta se ejecuta y conecta exitosamente en seguida



**Sección II: Creando una base de datos, una colección y añadiendo registros a ella**

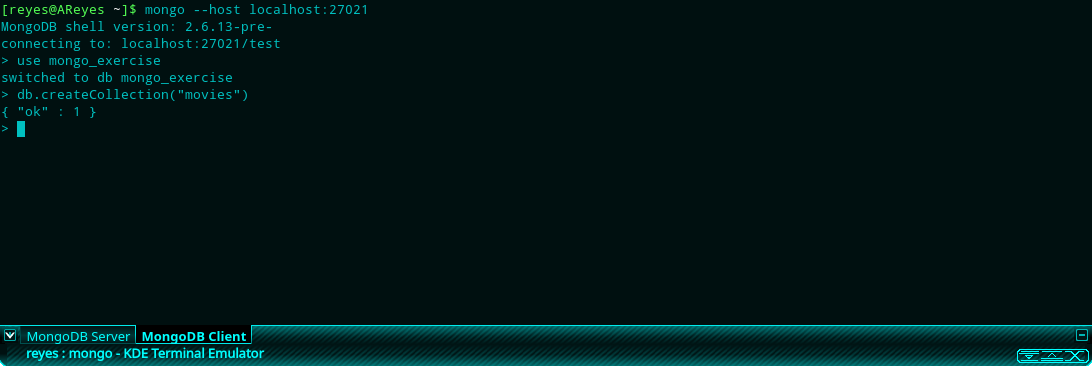
Una vez conectados a la instancia de MongoDB, se creará una base de datos -si es que no existe- utilizando la instrucción dentro de la *shell* de Mongo



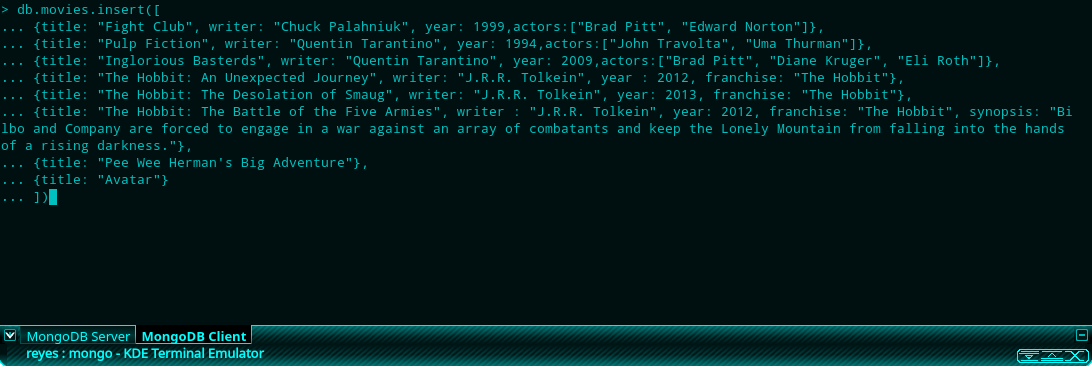
La instrucción anterior nos ubicará directamente en la base de datos indicada, donde se creará la colección requerida para este ejercicio



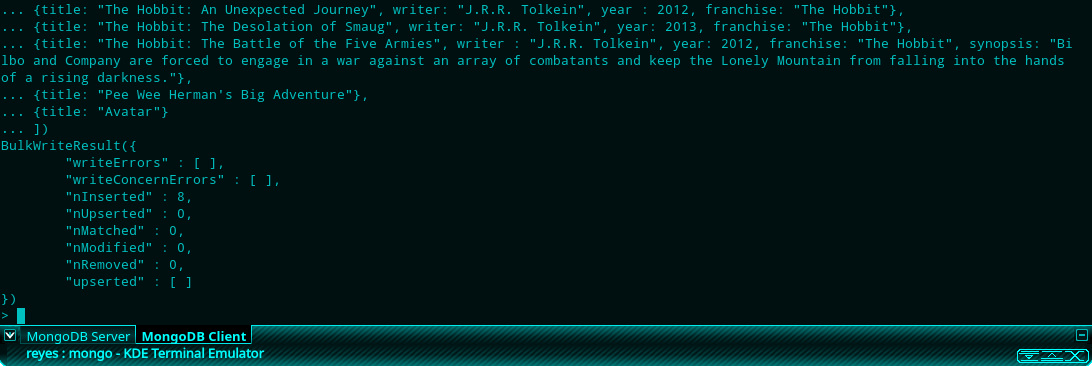
De inmediato obtenemos la retroalimentación respectiva que indica que ambos comandos fueron procesados y finalizados sin errores



Ahora que la colección *movies* existe podemos agregar los registros solicitados, para lo cual se usará la función *insert([])* con datos de los registros a insertar



La base de datos retorna la cantidad de errores, en este caso, nula



**Sección III: Consultas**

A continuación, se realizarán ciertas consultas solicitadas sobre los documentos añadidos:

1. Obtener todos los documentos



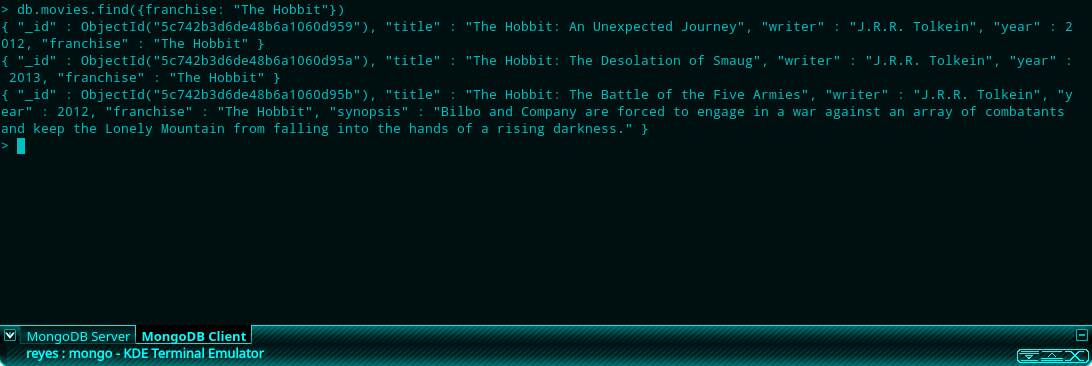
2. Obtener documentos con *writer* igual a *"Quentin Tarantino"*



3. Obtener documentos con *actors* que incluyan a *"Brad Pitt"*



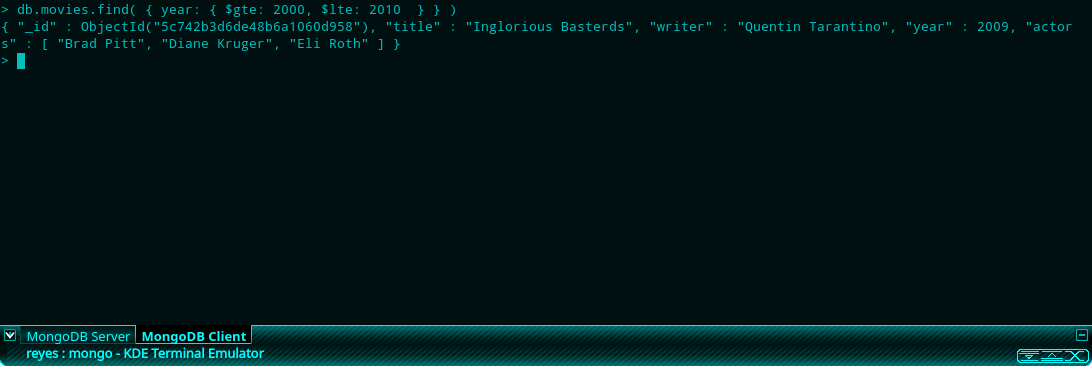
4. Obtener documentos con franchise igual a "The Hobbit"



5. Obtener todas las películas de los 90s.



6. Obtener las películas estrenadas entre el año 2000 y 2010.



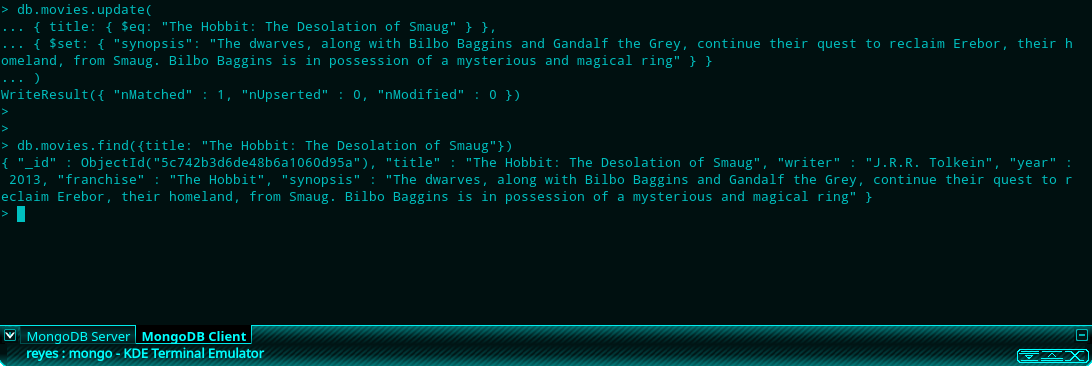
**Sección IV: Actualizaciones**

A continuación, se realizarán modificaciones a los registros existentes, para ello será útil la función *update()* con dos valores especificados: las cláusulas que el registro debe de cumplir para ser susceptible de modificación y el nuevo valor a asignar en el atributo especificado

1. Agregar sinopsis a "The Hobbit: An Unexpected Journey" : "A reluctant hobbit, Bilbo Baggins, sets out to the Lonely Mountain with a spirited group of dwarves to reclaim their mountain home - and the gold within it - from the dragon Smaug."



2. Agregar sinopsis a "The Hobbit: The Desolation of Smaug" : "The dwarves, along with Bilbo Baggins and Gandalf the Grey, continue their quest to reclaim Erebor, their homeland, from Smaug. Bilbo Baggins is in possession of a mysterious and magical ring."



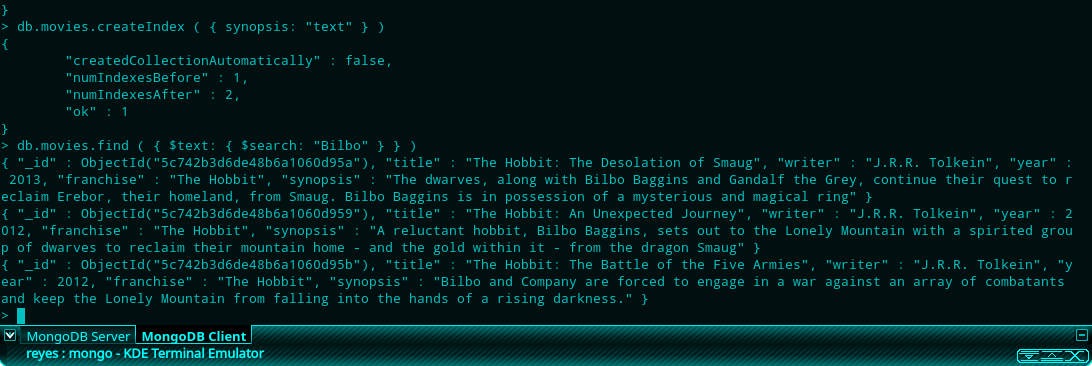
3. Agregar un actor llamado "Samuel L. Jackson" a la película "Pulp Fiction"



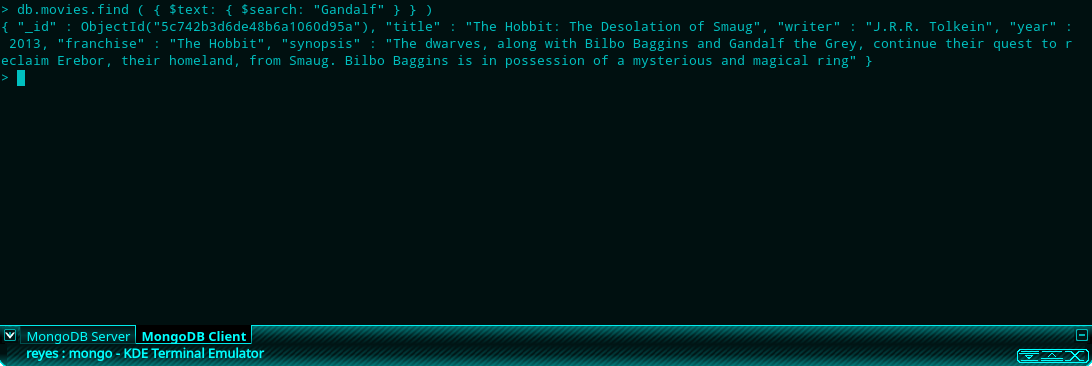
**Sección V: Búsquedas por texto**

Para conseguir una búsqueda por texto exitosa, es necesario agregar lo que en Mongo se conocen como *índices*, mediante los cuales se permite realizar búsquedas para atributos indexados de tipo string

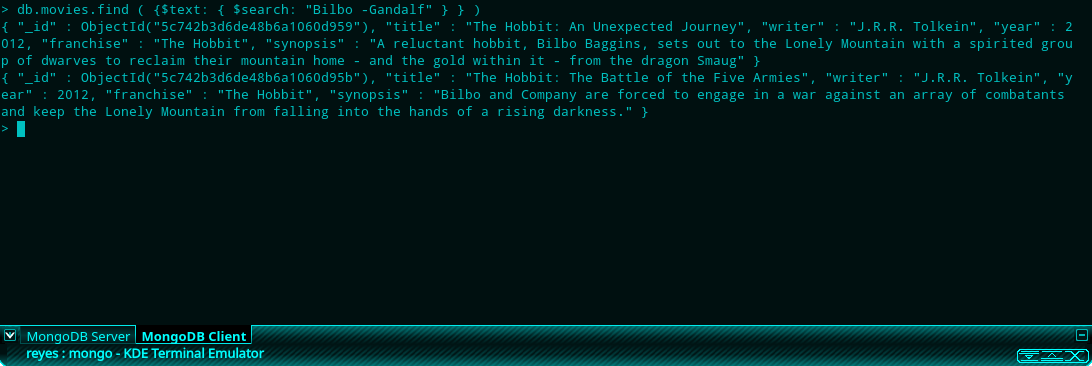
1. Encontrar las películas que en la sinopsis contengan la palabra "Bilbo"



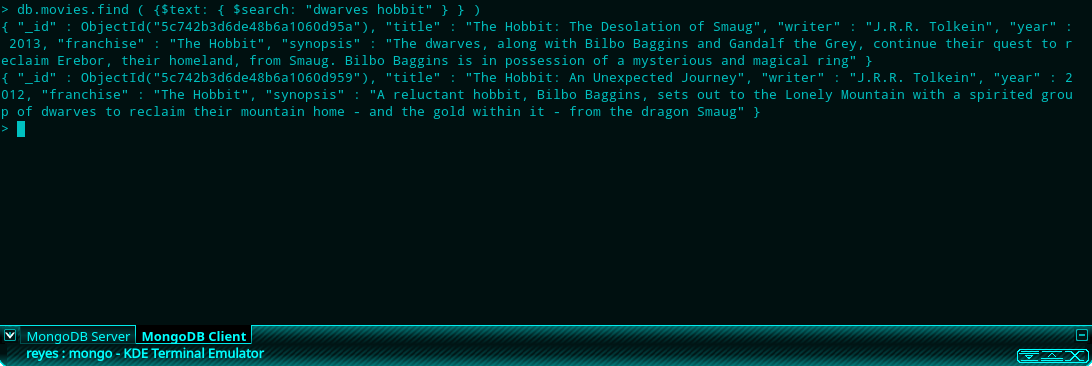
2. Encontrar las películas que en la sinopsis contengan la palabra "Gandalf"



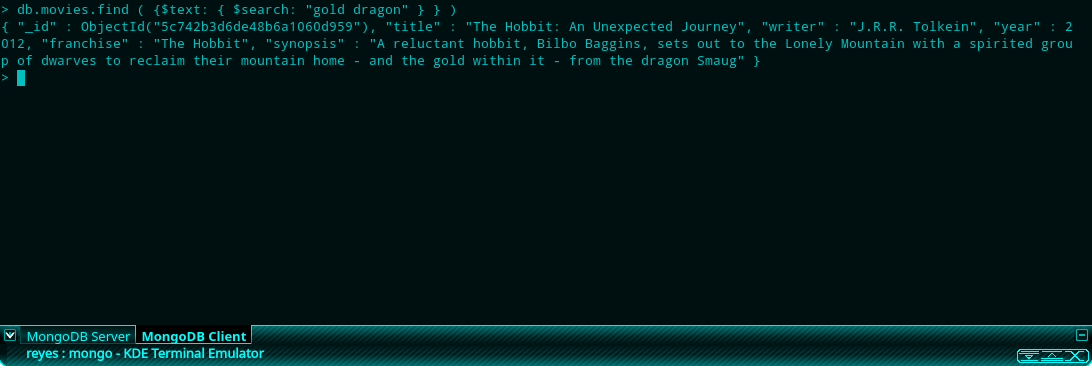
3. Encontrar las películas que en la sinopsis contengan la palabra "Bilbo" y no la palabra "Gandalf"



4. Encontrar las películas que en la sinopsis contengan la palabra "dwarves" ó "hobbit"



5. Encontrar las películas que en la sinopsis contengan la palabra "gold" y "dragon"



**Sección VI: Eliminaciones**

1. Eliminar la película "Pee Wee Herman's Big Adventure"



2. Eliminar la película "Avatar"



## Discusión

A comparación con los conocimientos previos que los autores poseían sobre Sistemas de Gestión de Bases de Datos Relacionales, el contacto con este nuevo tipo de SGBD representó un giro a la conciencia sobre los mismos, permitiendo a estos percibir diferencias notorias que contrastan a unos de otros, decayendo en ciertos aspectos, como el hecho de no seguir un esquema predefinido que acontece causalmente en potenciales fallas de consistencia de datos, pero que compensa con aspectos de flexibilidad sobre todo en cuestiones referentes a la escalabilidad horizontal que estos son capaces de alcanzar debido a la inclusión de técnicas como sharding o Indexación.

# Práctica 2

## Objetivos

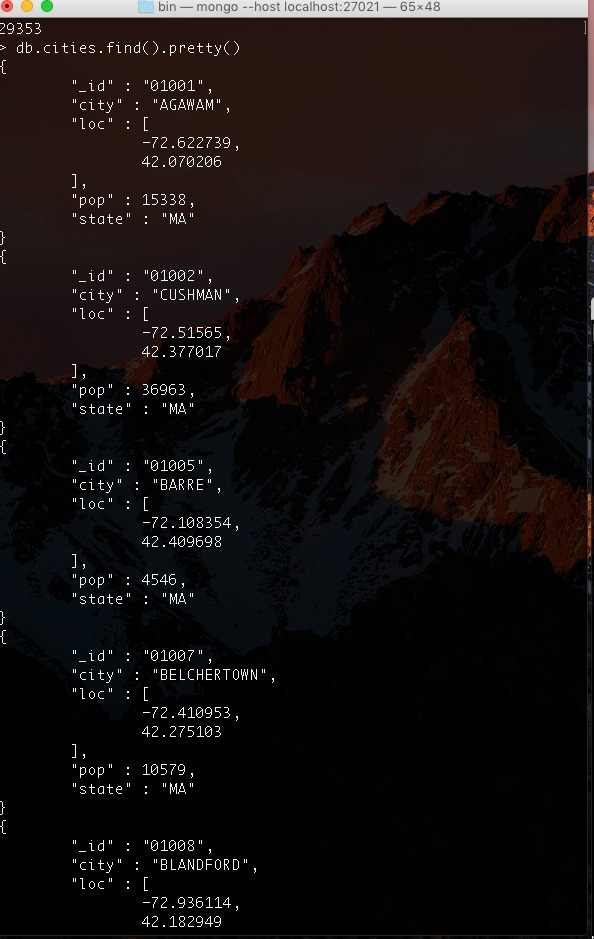
Ilustrar la técnica de partición de una base de datos llamada sharding. Para esto, se usará una base de datos NoSQL creada con ayuda del DBMS Mongo. Al final de la práctica, se tendrán 3 shards, 1 query router y 1 config server.

## Desarrollo

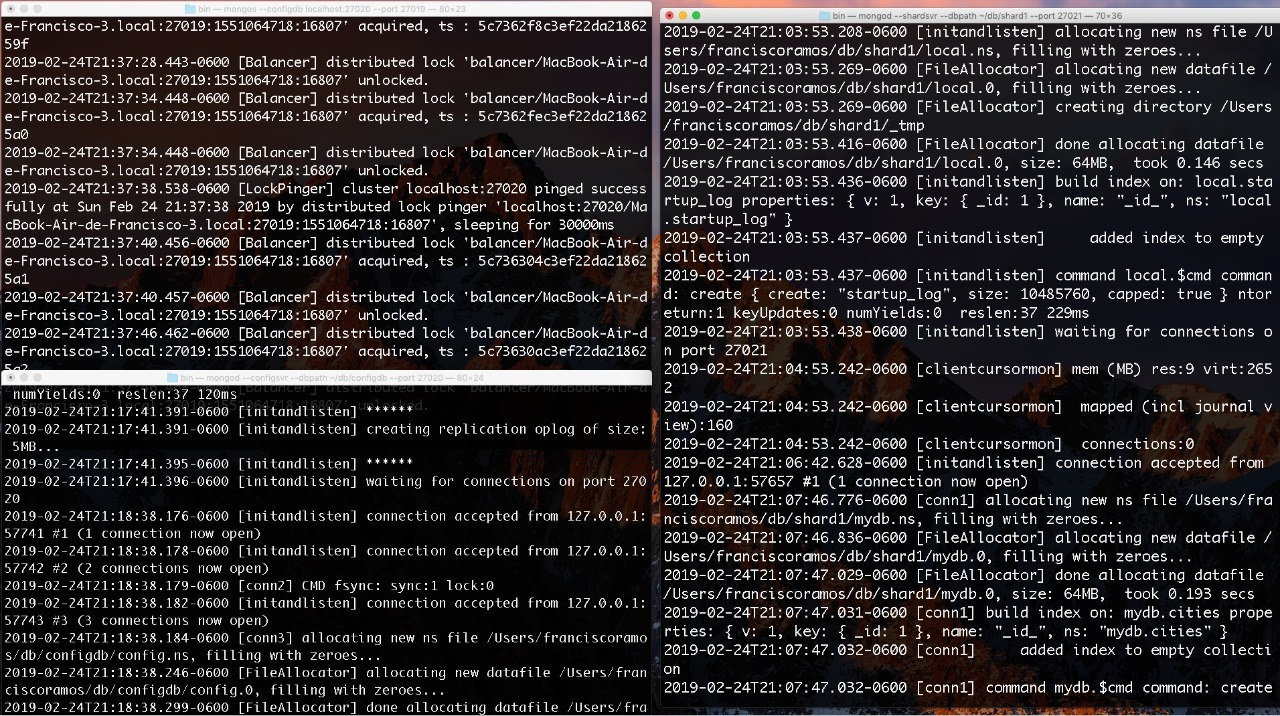
### Parte 1

En esta parte de la práctica 2, se crearán los componentes de una base de datos que implementa la técnica de sharding.

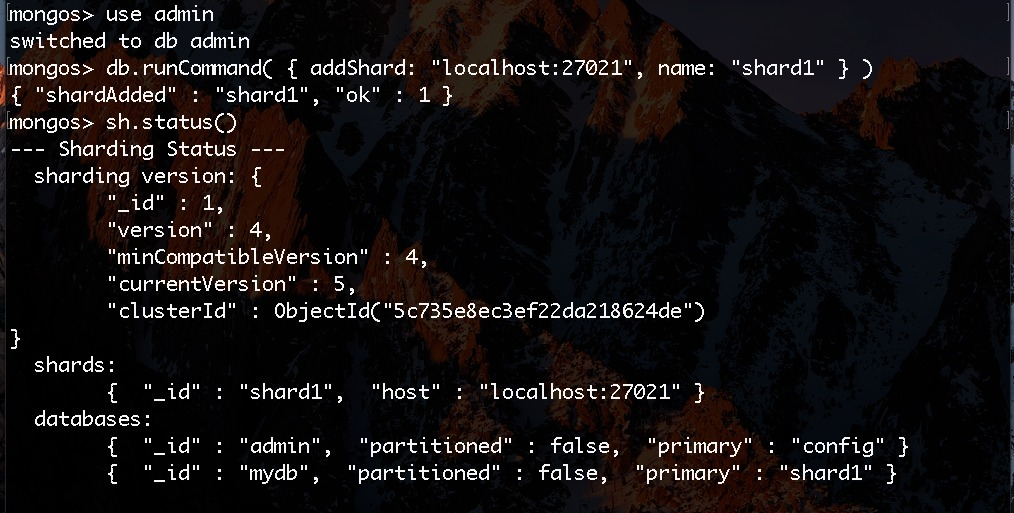
Primero, se creó un directorio para almacenar los archivos de la base de datos creada por MongoDB. Posteriormente, se inició un servidor sobre ese directorio en el puerto 27021 con el comando *mongod --shardsvr --dbpath ~/db/shard1 --port 27021*. Para agregar datos a esta base, se usó un archivo txt, el cual, con ayuda de un cliente conectado al servidor, se agregaron datos de ciudades a la base de datos. Al momento de realizar la consulta *db.cities.find().pretty()*, Mongo nos devuelve una lista con los valores de todos los objetos almacenados. Igualmente, la consulta *db.cities.count()* nos devuelve el número de ciudades almacenadas (29353).



Después, se creó un *config server* en el puerto 27020 con el comando *mongod --configsvr --dbpath ~/db/configdb --port 27020.* Igualmente, con ayuda de *mongos --configdb localhost:27020 --port 27019* y *db.runCommand( { addShard: "localhost:27021", name: "shard1" } )* se creó un *query router* y se agregó un *shard* respectivamente.



Al momento de realizar el comando *sh.status()*, obtenemos información sobre el número de shards, las bases de datos almacenadas, y el shard en donde estas se encuentran.

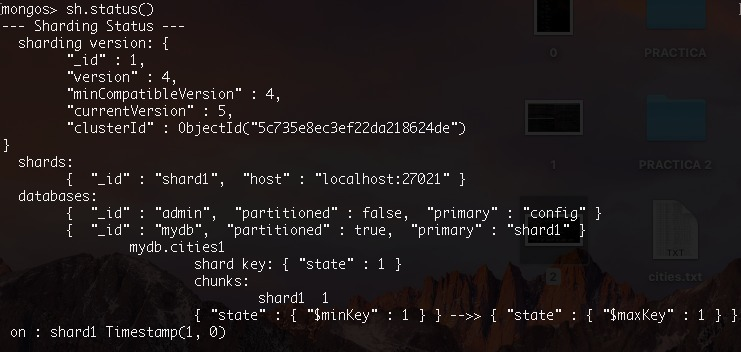


### Parte 2

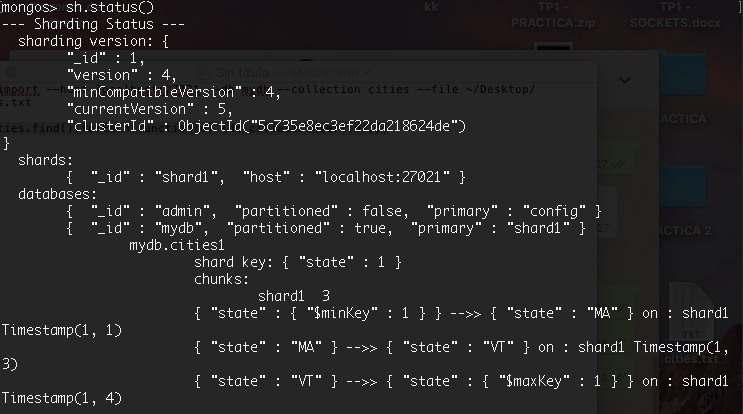
En esta parte, se llevará a cabo la distribución de los datos contenidos en la base a través de 3 shards por medio de 2 técnicas: usando rangos o con una función de hash.

Primero, se creó una nueva colección llamada “cities 1” en la base de datos con el comando *db.createCollection("cities1")*. Posteriormente, con la ayuda de los comandos *sh.enableSharding("mydb")* y *sh.shardCollection("mydb.cities1", { "state": 1} )*, se indicó que la base de datos, y la colección implementarán la técnica de sharding usando el atributo “estado” como llave para el sharding.

Al momento de realizar el comando *sh.status()*, se puede notar que se creó 1 chunk que tiene como llave el atributo state. Éste, tiene el rango de 1 a 1, ya que es el único chunk, y la colección que contiene no tiene datos.

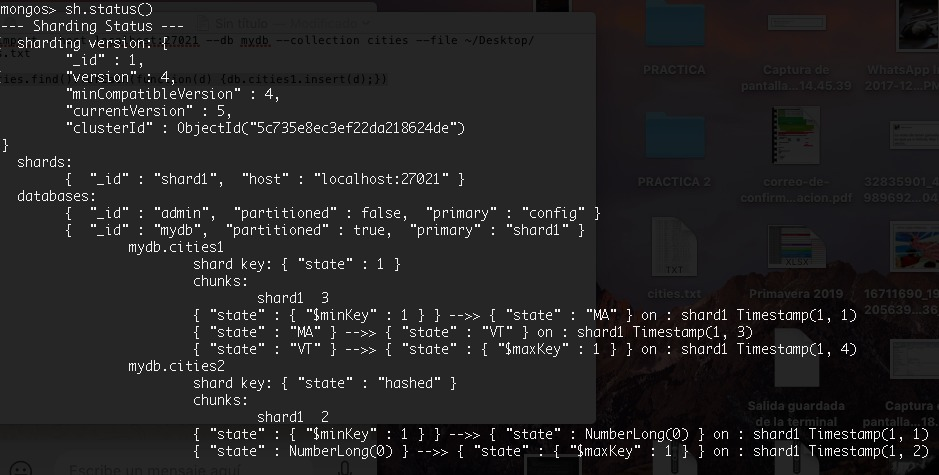


Posteriormente, se agregaron los datos de la colección “cities” a la colección “cities1”, y se verificó el estado de la base de datos. Se obtuvo lo siguiente:



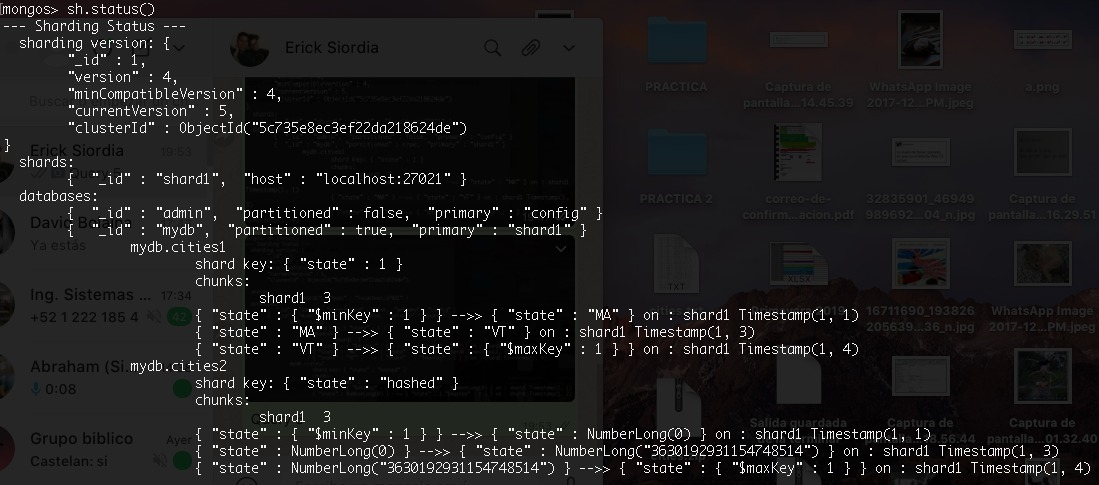
Como se puede notar, ahora existen 3 chunks en el shard con diferentes rangos. Por ejemplo, el primer chunk tiene un rango de 1 a “MA”, y el segundo tiene un rango de “MA” a “VT”.

Después, se seguirá el mismo proceso seguido para la creación de una colección nueva llamada “cities2”. Ésta, será particionada en base a una función de hash y no en base a rangos. Para especificar que el sharding se realizará en base a hashing, se usa el comando *sh.shardCollection("mydb.cities2", { "state": "hashed"} )*. Al momento de obtener el estado de la base de datos, se obtiene lo siguiente:



En la imagen se pueden observar las dos colecciones “cities1” y “cities2” que están particionadas en chuncks en el shard creado. Como se especificó, la colección “cities2” usa una partición en base a hash, por lo que los rangos que este chunk tiene asociados no son los nombres de las ciudades, sino valores numéricos (por el momento son 0 porque la colección está vacía). Se puede notar también que se crearon 2 chunks para almacenar la colección “cities2”.

Posteriormente, se copiaron los datos de la colección “cities” a la colección “cities2” y se verificó de nuevo el estado de la base de datos.

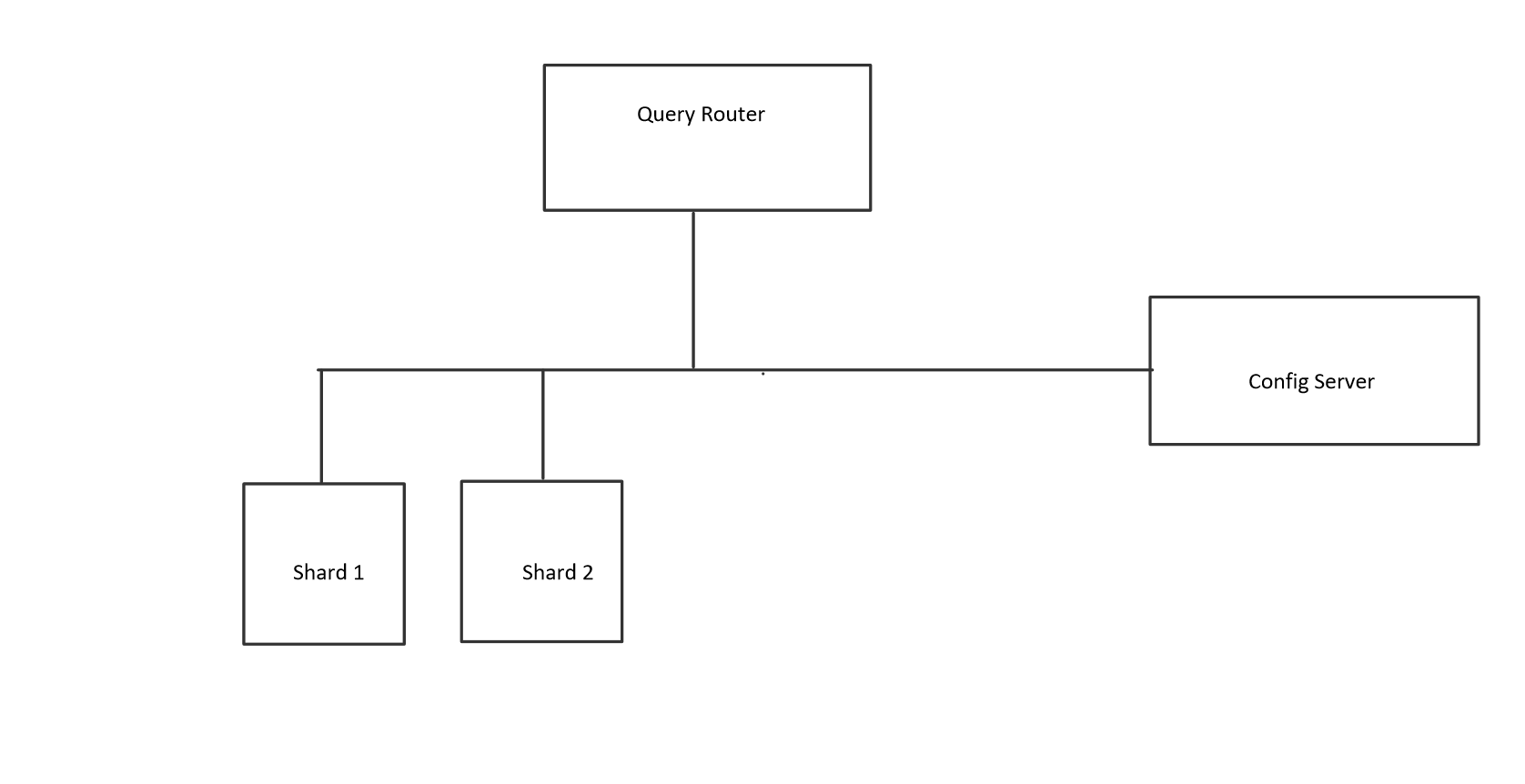


Después de agregar los datos a la colección, se muestra que el shard tiene 3 chunks asociados. De igual forma, se observa que los valores que se usan para el hashing han cambiado, ya que toman valores diferentes de 0.

Para la parte final de esta práctica, se ilustrará el balanceo que realiza el DBMS Mongo para que la distribución de los datos dentro de una base de datos esté normalizada. De igual forma, el concepto de tagging se ilustrará, ya que este es útil para asociar un shard a datos específicos.

Para esto, se creó 1 shard adicional y se añadió a la base de datos con ayuda del query router.

Ahora, la estructura de nuestra base de datos se ve de la siguiente manera:



27019

27020

27021

27022

Luego, se creó otro shard adicional a la base de datos, y se asociarán *tags* a cada uno de los shards. Se asoció “CA” al shard1, “NY” al shard2 y “Others” al shard3 con los siguientes comandos:

*sh.addShardTag("shard1", "CA")*

*sh.addShardTag("shard2", "NY")*

*sh.addShardTag("shard3", "Others")*

A continuación, se creó una nueva colección llamada “cities3”, y se copiaron los datos de la colección “cities” a ella.

Finalmente, se asociaron rangos basados en los *tags*, para poder especificar dónde se guardarán los registros dependiendo del valor que contengan en el atributo “state”.

Como se muestra en la imagen siguiente, el estado de la base de datos muestra que hay 3 chunks en el shard1, 3 en el shard2, y 7 en el shard3. También se pueden notar los *tags* asociados a los rangos que se especificaron, y en los shards indicados. Por ejemplo, el chunk con el tag “CA” se encuentra en el shard1 y el chunk con el tag “NY” se encuentra en el shard2.



## Discusión

En esta práctica se pudo observar la técnica de partición de bases de datos llamada sharding. Esta puede ser útil cuando se tiene que administrar una gran cantidad de datos en una base, ya que separa la administración de estos en partes para que trabajen de una manera más eficaz. De igual forma, el uso de *tags* en la práctica nos demuestra que se pueden asociar rangos de datos a un shard específico. Esto puede ser útil en caso de que un cliente necesite datos específicos de manera continua, y necesite una respuesta rápida a un shard que esté geográficamente más cercano que otros. Igualmente, los *tags* pueden ser usados para organizar los datos y mejorar el acceso que se tiene a ellos.

En conclusión, se puede decir que MongoDB implementa técnicas que pueden hacer el acceso y administración de una base de datos NoSQL más fácil, más eficaz, y más adaptable al caso de uso para el que se desarrolla.