48 comandos y consultas de MongoDB para conocer como desarrollador y administrador de bases de datos

Clip source: 48 comandos y consultas de MongoDB para conocer como desarrollador y administrador de bases de datos

48 comandos y consultas de MongoDB para conocer como desarrollador y administrador de bases de datos

Este artículo habla sobre las consultas y comandos de uso frecuente de MongoDB utilizado por desarrolladores y DBA en su desarrollo diario y vida operativa.

Quick Intro

En esta nueva era de tecnología inteligente, los datos se generan en un gran volumen y cada dato es igualmente importante para las industrias en crecimiento. Los usuarios están generando datos estructurados, semiestructurados y no estructurados en una cantidad ilimitada. Los datos estructurados comprenden el almacenamiento de datos en tablas y filas, mientras que los datos no estructurados consisten en imágenes, videos y clips de voz. Debido al creciente volumen de datos de datos estructurados y no estructurados, es necesario Base de datos NoSQL entra en la imagen.

Proporciona a los desarrolladores facilidad para diseñar esquemas y plataformas escalables a los administradores de bases de datos, proporciona una plataforma segura y rápida.

What is MongoDB?

MongoDB es una base de datos NoSQL de código abierto, multiplataforma y orientada a documentos que se utiliza para almacenar datos semiestructurados escritos en C ++. En lugar de tablas y filas, MongoDB almacena datos en pares clave-valor. Para que el aprendizaje sea fácil y sin complicaciones para los desarrolladores y administradores, estos son algunos de los comandos de MongoDB de uso frecuente.

Vamos a ponerlo en marcha.

Basic Commands

1. Verificación de la versión

El comando más importante es verificar la versión instalada del servidor MongoDB y Mongo Shell. Ejecute este comando en el terminal en Linux o en el indicador CMD en Windows.

`mongod --version`

```
C:WindowsSystem32>mongod --version
db version v4.2.7
git version: 51d9fe12b5d19720e72dcd7db0f2f17dd9a19212
allocator: tcmalloc
modules: none
build environment:
distmod: 2012plus
distarch: x86_64
target_arch: x86_64
```

También podemos usar `mongod` comando para verificar la versión, de la siguiente manera.

`mongo –version`

```
C:WindowsSystem32>mongo --version
MongoDB shell version v4.2.7
git version: 51d9fe12b5d19720e72dcd7db0f2f17dd9a19212
```

```
allocator: tcmalloc
modules: none
build environment:
distmod: 2012plus
distarch: x86_64
target_arch: x86_64
```

2. Listado de comandos de MongoDB

Este comando ayudará a los usuarios a descubrir todos los comandos que se pueden usar en MongoDB. Ejecute el comando en Mongo Shell.

`help()`

```
mongo> db.help()
DB methods:
db.adminCommand(nameOrDocument) - switches to 'admin' db, and runs command [just calls
db.runCommand(...)]
db.aggregate([pipeline], {options}) - performs a collectionless aggregation on this
database; returns a cursor
db.auth(username, password)
db.cloneDatabase(fromhost) - will only function with MongoDB 4.0 and below
db.commandHelp(name) returns the help for the command
db.copyDatabase(fromdb, todb, fromhost) - will only function with MongoDB 4.0 and below
db.createCollection(name, {size: ..., capped: ..., max: ...})
db.createUser(userDocument)
db.createView(name, viewOn, [{$operator: {...}}, ...], {viewOptions})
db.currentOp() displays currently executing operations in the db
db.dropDatabase(writeConcern)
db.dropUser(username)
db.eval() - deprecated
db.fsyncLock() flush data to disk and lock server for backups
db.fsyncUnlock() unlocks server following a db.fsyncLock()
db.getCollection(cname) same as db['cname'] or db.cname
db.getCollectionInfos([filter]) - returns a list that contains the names and options of
the db's collections
db.getCollectionNames()
db.getLastError() - just returns the err msg string
db.getLastErrorObj() - return full status object
db.getLogComponents()
db.getMongo() get the server connection object
db.getMongo().setSlaveOk() allow queries on a replication slave server
db.getName()
db.getProfilingLevel() - deprecated
db.getProfilingStatus() - returns if profiling is on and slow threshold
db.getReplicationInfo()
db.getSiblingDB(name) get the db at the same server as this one
```

```
db.getWriteConcern() - returns the write concern used for any operations on this db,
inherited from server object if set
db.hostInfo() get details about the server's host
db.isMaster() check replica primary status
db.killOp(opid) kills the current operation in the db
db.listCommands() lists all the db commands
db.loadServerScripts() loads all the scripts in db.system.js
db.logout()
db.printCollectionStats()
db.printReplicationInfo()
db.printShardingStatus()
db.printSlaveReplicationInfo()
db.resetError()
db.runCommand(cmdObj) run a database command. if cmdObj is a string, turns it into
{cmdObj: 1}
db.serverStatus()
db.setLogLevel(level,<component>)
db.setProfilingLevel(level,slowms) 0=off 1=slow 2=all
db.setVerboseShell(flag) display extra information in shell output
db.setWriteConcern(<write concern doc>) - sets the write concern for writes to the db
db.shutdownServer()
db.stats()
db.unsetWriteConcern(<write concern doc>) - unsets the write concern for writes to the
db
db.version() current version of the server
db.watch() - opens a change stream cursor for a database to report on all changes to its
non-system collections.
```

3. Estadísticas de base de datos

El siguiente comando proporcionará detalles de las bases de datos junto con varias colecciones y parámetros relacionados de esa base de datos.

`db.stats()`

```
> db.stats()
{
"db" : "test",
"collections" : 0,
"views" : 0,
"objects" : 0,
"avgObjSize" : 0,
"dataSize" : 0,
"storageSize" : 0,
"numExtents" : 0,
"indexes" : 0,
"indexSize" : 0,
"scaleFactor" : 1,
```

```
"fileSize" : 0,
"fsUsedSize" : 0,
"fsTotalSize" : 0,
"ok" : 1
}
```

4. Cree una nueva base de datos o cambie a una base de datos existente

Este comando simple ayuda a crear una nueva base de datos si no existe o ayuda a cambiar a la base de datos existente. En MongoDB, "prueba" es la base de datos predeterminada, por lo que los usuarios usan "**test**”DB una vez que se inicie sesión en Mongo Shell.

`use DB_Name`

```
mongos> use geekFlareDB
switched to db geekFlareDB
```

5. Listado de todas las bases de datos

El comando mencionado se utiliza para enumerar todas las bases de datos.

`show dbs`

```
mongo> show dbs
admin 0.000GB
config 0.002GB
geekFlareDB 0.000GB
test 0.000GB
```

6. Verifique la base de datos actualmente en uso

Ejecute el siguiente comando en Mongo Shell para ver la base de datos actualmente en uso.

`db`

```
> db
```

```
GeekFlare
```

7. Eliminar base de datos

El comando dado ayuda al usuario a eliminar la base de datos requerida. Ejecute el comando en el cliente MongoDB. Asegúrese de seleccionar la base de datos antes de ejecutar el comando drop. De lo contrario, eliminará el valor predeterminado "**test**”Base de datos.

`db.dropDatabase()`

Primero enumeremos toda la base de datos, cambiemos a una de ellas y luego suéltela

```
> show dbs
admin 0.000GB
config 0.001GB
local 0.000GB
test 0.000GB
training 0.000GB
>
> use training
switched to db training
>
> db.dropDatabase()
{ "dropped" : "training", "ok" : 1 }
```

8. Crear colección

Las colecciones son similares a las tablas en RDBMS.

El comando Crear una colección consta de dos parámetros. La colección consta de cero o más documentos. Por lo tanto, para crear una colección, el parámetro obligatorio para usar en el comando es su nombre y parámetro opcional puede incluir el nombre de los documentos, su tamaño y su índice.

- Creando una colección sencilla.

Sintaxis: `db.createCollection(Name,Options)`

Ejemplo:

```
> use geekFlare
switched to db geekFlare
```

```
>
> db.createCollection("geekFlareCollection")
{ "ok" : 1 }
>
> show collections
geekFlareCollection
```

- Creación de una colección limitada

En esto, restrinja el tamaño y la cantidad de documentos que se insertarán en la colección. La colección limitada tiene la propiedad de eliminar los documentos más antiguos para dejar espacio para los documentos nuevos.

Sintaxis:

`db.createCollection(Name,{capped : true, size : sizeLimit , max : documentLimit })`

Ejemplo: Creemos una colección limitada, insertemos un registro y recuperémoslo

```
> db.createCollection("Login",{capped:true,max:1,size:200})
{ "ok" : 1 }
>
> db.Login.insertMany([{"id":1,status:"Active"},{"id":2,status:"Hold"},
{"id":3,status:"Pending"}])
{
"acknowledged" : true,
"insertedIds" : [
ObjectId("5edc5f4f411247725e75e52e"),
ObjectId("5edc5f4f411247725e75e52f"),
ObjectId("5edc5f4f411247725e75e530")
]
}
>
> db.Login.find()
{ "_id" : ObjectId("5edc5f4f411247725e75e530"), "id" : 3, "status" : "Pending" }
```

9. Recogida de gotas

El comando Drop Collection es similar a DDL en RDBMS. Adquiere bloqueos en la colección requerida hasta la ejecución del comando. Drop collection elimina la colección de la base de datos junto con todos los índices asociados con esa colección. Para eliminar la colección, se requiere el método drop ().

Devuelve verdadero para una caída exitosa y falso en caso de cualquier error o si la base de datos no existe.

Sintaxis: `collectionName.drop()`

Ejemplo:

```
> use geekFlare
switched to db geekFlare
>
> show collections
geekFlareCollection
>
> db.geekFlareCollection.drop()
true
>
> db.geekFlareCollection.drop()
false
```

CRUD Operations related

10. Inserte el documento en la colección

En MongoDB, el documento es similar a una tupla en RDBMS.

Para crear un documento, el `insert()` se utiliza el método. El método insert () crea uno o varios documentos en la colección existente. También crea una colección si no está presente en la base de datos. En MongoDB, Document no tiene esquema, lo que significa que no hay restricciones para insertar cualquier número de claves en un documento.

- **Insertar un solo registro**

Para insertar un registro `insert()` or `insertOne()` se puede utilizar el método.

Sintaxis: `collectionName.insertOne({document})`

Ejemplo:

```
> db.geekFlareCollection.insertOne( {
code: "P123", Qty: 200, status: "Active"
});
{
"acknowledged" : true,
```

```
"insertedId" : ObjectId("5ed309725429283aee2e134d")
}
```

- **Insertar varios registros**

Para insertar muchos registros, se pasará una lista de registros a `insert()` or `insertMany()` método.

Sintaxis:

`collectionName.insertMany([{document1},{document2},{ document3}….{documentn}])`

Ejemplo:

```
db.geekFlareCollection.insertMany([
... { code: "P1", Qty: 100, status: "Active"},
... { code: "P2", Qty: 200, status: "Active"},
... { code: "P3", Qty: 0, status: "Dective"}
... ]);
{
"acknowledged" : true,
"insertedIds" : [
ObjectId("5edf7b4e18b2c26b9dfe8cac"),
ObjectId("5edf7b4e18b2c26b9dfe8cad"),
ObjectId("5edf7b4e18b2c26b9dfe8cae")
]
}
> db.geekFlareCollection.find()
{ "_id" : ObjectId("5edf546fdfa12b33b7cb75b8"), "product" : "bottles", "Qty" : 100 }
{ "_id" : ObjectId("5edf546fdfa12b33b7cb75b9"), "product" : "bread", "Qty" : 20 }
{ "_id" : ObjectId("5edf546fdfa12b33b7cb75ba"), "product" : "yogurt", "Qty" : 30 }
{ "_id" : ObjectId("5edf7b4e18b2c26b9dfe8cac"), "code" : "P1", "Qty" : 100, "status" :
"Active" }
{ "_id" : ObjectId("5edf7b4e18b2c26b9dfe8cad"), "code" : "P2", "Qty" : 200, "status" :
"Active" }
{ "_id" : ObjectId("5edf7b4e18b2c26b9dfe8cae"), "code" : "P3", "Qty" : 0, "status" :
"Dective" }
>
```

- **Insertar registro a granel**

También se puede insertar una gran cantidad de documentos de forma ordenada y desordenada ejecutando `initializeOrderedBulkOp()` y `initializeUnorderedBulkOp()` métodos.

Sintaxis:

```
var bulk = db.collectionName.initializeUnorderedBulkOp();

bulk.insert({document1} );

bulk.insert({document2} );

bulk.insert({documentn} );

bulk.execute();
```

Ejemplo:

```
> var bulk = db.geekFlareCollection.initializeUnorderedBulkOp();
> bulk.insert({ code: "P1", Qty: 100, status: "Active"});
> bulk.insert({ code: "P2", Qty: 200, status: "Active"});
> bulk.insert({ code: "P3", Qty: 0, status: "Dective"});
> bulk.execute();
BulkWriteResult({
"writeErrors" : [ ],
"writeConcernErrors" : [ ],
"nInserted" : 3,
"nUpserted" : 0,
"nMatched" : 0,
"nModified" : 0,
"nRemoved" : 0,
"upserted" : [ ]
})
> db.geekFlareCollection.find()
{ "_id" : ObjectId("5edf7be318b2c26b9dfe8caf"), "code" : "P1", "Qty" : 100, "status" :
"Active" }
{ "_id" : ObjectId("5edf7be318b2c26b9dfe8cb0"), "code" : "P2", "Qty" : 200, "status" :
"Active" }
{ "_id" : ObjectId("5edf7be318b2c26b9dfe8cb1"), "code" : "P3", "Qty" : 0, "status" :
"Dective" }
>
```

11. Recuperar documento de una colección

Para buscar el documento almacenado en una colección, se puede utilizar el método find (). El siguiente comando se utilizará para recuperar todos los documentos de la colección.

- `find()` El método se puede utilizar para recuperar todos los documentos almacenados en una colección.

Sintaxis: `collectionName.find()`

Ejemplo:

```
> db.geekFlareCollection.find()
{ "_id" : ObjectId("5ed31186b6f2c2bb1edb86ce"), "code" : "P1", "Qty" : 200, "status" :
"Active" }
{ "_id" : ObjectId("5ed31186b6f2c2bb1edb86cf"), "code" : "P2", "Qty" : 200, "status" :
"Active" }
{ "_id" : ObjectId("5ed31186b6f2c2bb1edb86d0"), "code" : "P3", "Qty" : 200, "status" :
"Active" }
{ "_id" : ObjectId("5ed3159eb6f2c2bb1edb86d1"), "code" : "P4", "Qty" : 100, "status" :
"Inactive" }
```

- `find({condition})` El método se puede usar para recuperar solo los documentos requeridos en función de algunas condiciones de la colección. MongoDB proporciona una lista de operadores de proyección y consulta para recuperar el valor del tipo BSON.

Sintaxis: `collectionName.find({ condition })`

Ejemplo:

```
> db.geekFlareCollection.find({ Qty: { $eq: 100 }});
{ "_id" : ObjectId("5ed3159eb6f2c2bb1edb86d1"), "code" : "P4", "Qty" : 100, "status" :
"Inactive" }
```

- Para recuperar solo un documento, MongoDB proporciona la `findOne` () método. Da una salida formateada.

Sintaxis: `collectionName.findOne()`

Ejemplo:

```
> db.geekFlareCollection.findOne();
{
"_id" : ObjectId("5ed31186b6f2c2bb1edb86ce"),
"code" : "P1",
"Qty" : 200,
"status" : "Inactive"
}
```

12. Embellecer la salida de recuperación

La `find()` El método da una salida desorganizada. MongoDB proporciona `pretty()` comandos para obtener la salida formateada.

Sintaxis: `collectionName.find().pretty()`

Ejemplo:

```
> db.geekFlareCollection.find({ Qty: { $eq: 100 }}).pretty();
{
"_id" : ObjectId("5ed3159eb6f2c2bb1edb86d1"),
"code" : "P4",
"Qty" : 100,
"status" : "Inactive"
}
```

13. Actualizar documento en una colección

MongoDB proporciona `update()` método para establecer nuevos valores para claves existentes en documentos. El comando de actualización brinda detalles de documentos modificados y coincidentes. La sintaxis del comando de actualización es:

Sintaxis: `collectionName.update({KeyToUpdate},{Set Command})`

Ejemplo:

```
> db.geekFlareCollection.find()
{ "_id" : ObjectId("5edf3f67d6bfbd8125f58cfb"), "product" : "bottles", "Qty" : 100 }
{ "_id" : ObjectId("5edf3f67d6bfbd8125f58cfc"), "product" : "bread", "Qty" : 20 }
{ "_id" : ObjectId("5edf3f67d6bfbd8125f58cfd"), "product" : "yogurt", "Qty" : 30 }
>
> db.geekFlareCollection.update({"product" : "bottles"},{$set : {"Qty": 10}} )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
>
> db.geekFlareCollection.find()
{ "_id" : ObjectId("5edf3f67d6bfbd8125f58cfb"), "product" : "bottles", "Qty" : 10 }
{ "_id" : ObjectId("5edf3f67d6bfbd8125f58cfc"), "product" : "bread", "Qty" : 20 }
{ "_id" : ObjectId("5edf3f67d6bfbd8125f58cfd"), "product" : "yogurt", "Qty" : 30 }
>
> db.geekFlareCollection.find()
{ "_id" : ObjectId("5edf3f67d6bfbd8125f58cfb"), "product" : "bottles", "Qty" : 10 }
{ "_id" : ObjectId("5edf3f67d6bfbd8125f58cfc"), "product" : "bread", "Qty" : 20 }
{ "_id" : ObjectId("5edf3f67d6bfbd8125f58cfd"), "product" : "yogurt", "Qty" : 30 }
```

- **`updateOne()`** **:** Para actualizar un solo documento hay `updateOne()` método. `updateOne()` Dar el recuento de documentos coincidentes y modificados.

Sintaxis: `collectionName.updateOne({SingleKeyToUpdate},{Set Command})`

Ejemplo:

```
> db.geekFlareCollection.updateOne({"product" : "bottles"},{$set : {"Qty": 40}} )
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

- `<strong>updateMany()</strong>` : Para actualizar varios documentos en alguna condición, MongoDB tiene `updateMany()` método.

Sintaxis: `collectionName.updateMany({filter},{Set Command})`

Ejemplo:

```
> db.geekFlareCollection.updateMany( { "Qty" : { $lt: 30 } },{ $set: { "Qty":
"Inactive"} } )
{ "acknowledged" : true, "matchedCount" : 1, "modifiedCount" : 1 }
```

14. Eliminar documento de una colección

Para eliminar el documento, MongoDB consta de `deleteOne()` y `deleteMany()` métodos. La sintaxis de los métodos de eliminación es:

- `<strong>deleteOne({condition}) </strong>` elimina el documento único que cumple los criterios de eliminación.

Sintaxis: `collectionName.deleteOne({DeletionCondition})`

Ejemplo:

```
> db.geekFlareCollection.deleteOne({"product" : "bread"})
{ "acknowledged" : true, "deletedCount" : 1 }
```

- `<strong>deleteMany() </strong>` elimina todos los documentos que coinciden con los criterios de eliminación. Sin los criterios de eliminación `deleteMany({condition})` elimina todos los documentos.

Sintaxis: `collectionName.deleteMany({DeletionCondition})`

Ejemplo:

```
> db.geekFlareCollection.deleteMany({"product" : "bottles"})
{ "acknowledged" : true, "deletedCount" : 2 }
```

- `<strong>remove() </strong>` Existe otro método para eliminar todos los documentos que coinciden con los criterios de eliminación. El método remove () toma dos argumentos, uno es la condición de eliminación y el otro es solo una bandera.

Nota: El método de eliminación está obsoleto en las próximas versiones.

Sintaxis: `collectionName.remove({DeletionCondition},1)`

Ejemplo:

```
> db.geekFlareCollection.remove({"product" : "bottles"})
WriteResult({ "nRemoved" : 1 })
```

15. Recuperar distinto

La `distinct()` El método se utiliza para obtener registros únicos.

- Para obtener registros distintos de un campo.

Sintaxis: `collectionName.distinct(field)`

Ejemplo:

```
> db.geekFlareCollection.distinct("product")
[ "Cheese", "Snacks2", "Snacks3", "bread", "ketchup" ]
```

- Para obtener registros distintos de un campo mientras se especifica la consulta.

Sintaxis: `collectionName.distinct(field,query)`

Ejemplo:

```
> db.geekFlareCollection.distinct('product',{"Qty":20})
[ "Snacks3", "bread" ]
```

16. Cambiar el nombre de la colección

MongoDB proporciona `renameCollection ()` método para cambiar el nombre de la colección.

Sintaxis: `collectionName.renameCollection(newCollectionName)`

Ejemplo:

```
>db.geekFlareCollection.renameCollection('geekFlareCol')
{ "ok" : 1 }
> show collections
geekFlareCol
```

Indexing

17. Crear índice en documento

Los índices son una estructura de datos especial que almacena una pequeña parte del conjunto de datos de la colección de forma fácil de recorrer. Los índices admiten el orden ascendente y descendente de los valores de los campos y, por lo tanto, facilitan un mejor rendimiento durante la recuperación.

MongoDB proporciona `default_id` índice. Además, MongoDB admite la creación de índices definidos por el usuario. Los índices de MongoDB se definen a nivel de colecciones y brindan soporte en el campo o subcampo de un documento. La sintaxis de crear el índice es:

- Crea un índice en un solo campo.

Sintaxis: `collectionName.createIndex({Key:1})`

En este, la clave indica el campo en el que se crea el índice y 1 significa orden ascendente. Para crear un índice en orden descendente se puede utilizar -1.

Ejemplo:

```
> db.geekFlareCollection.createIndex({"product" : 1})
{
"createdCollectionAutomatically" : false,
"numIndexesBefore" : 1,
"numIndexesAfter" : 2,
"ok" : 1
}
```

- Cree un índice en varios campos.

Sintaxis: `collectionName.createIndex({Key1:1,key2:1…keyn:1})`

Ejemplo:

```
> db.geekFlareCollection.createIndex({"product" : 1,"Qty":-1})
{
"createdCollectionAutomatically" : false,
"numIndexesBefore" : 1,
"numIndexesAfter" : 2,
"ok" : 1
}
```

18. Mostrar índice en el documento

MongoDB proporciona `getIndexes()` método para enumerar todos los índices creados en un documento.

Sintaxis: `collectionName.getIndexes()`

Ejemplo:

```
> db.geekFlareCollection.getIndexes()
[
{
"v" : 2,
"key" : {
"_id" : 1
},
"name" : "_id_",
"ns" : "geekFlareCollection.geekFlareCollection"
}
]
```

19. Eliminar índice del documento

`dropIndex()` El método se usa para eliminar el índice único y el método dropIndexes () se usa para eliminar múltiples índices.

- Quitar índice único

Sintaxis: `collectionName.dropIndex({key})`

Ejemplo:

```
> db.geekFlareCollection.dropIndex({"product" : 1})
{ "nIndexesWas" : 3, "ok" : 1 }
```

- Eliminar varios índices.

Sintaxis: `collectionName.dropIndexes({key1,key2…,keyN})`

Ejemplo:

```
> db.geekFlareCollection.dropIndexes({"product" : 1,"Qty":-1})
{ "nIndexesWas" : 3, "ok" : 1 }
```

Retrieval related

20. Limitar la recuperación de documentos

`limit()` El método ayuda a limitar el número de documentos devueltos. El método limit () acepta argumentos numéricos.

Sintaxis: `collectionName.find().limit(number)`

Ejemplo:

```
> db.geekFlareCollection.find().limit(2)
{ "_id" : ObjectId("5ed3c9e7b6f2c2bb1edb8702"), "product" : "bottles", "Qty" : 100 }
{ "_id" : ObjectId("5ed3c9e7b6f2c2bb1edb8703"), "product" : "bread", "Qty" : 20 }
```

21. Omitir la recuperación de documentos

Soporta MongoDB `skip()` método. Este método omite el número requerido de documentos. Acepta un argumento numérico.

Sintaxis: `collectionName.find().skip(number)`

Ejemplo:

```
> db.geekFlareCollection.find().skip(2)
{ "_id" : 3, "product" : "yogurt", "Qty" : 30 }
> db.geekFlareCollection.find().skip(3)
```

22. Clasificar recuperación de documentos

MongoDB `sort()` Este método clasifica los documentos de salida en orden ascendente o descendente. Este método acepta el nombre de las claves con el número para

especificar el orden de clasificación. 1 se usa para el orden ascendente, mientras que -1 se usa para especificar el orden descendente.

Sintaxis: `collectionName.find().sort({key:1})`

Ejemplo:

```
> db.geekFlareCollection.find().sort({"Qty":1})
{ "_id" : 2, "product" : "bread", "Qty" : 20 }
{ "_id" : 3, "product" : "yogurt", "Qty" : 30 }
{ "_id" : 1, "product" : "bottles", "Qty" : 100 }
```

Validation related

23. Validación de documentos

Los validadores ayudan a restringir el tipo de datos que se insertan en los documentos. Los validadores se definen en la recopilación. Se requiere la creación de un validador para usar la palabra clave **validador** y opcional **nivel de validación** y **acción de validación** para especificar el modo de validación. La validación del documento no restringe la inserción del nuevo campo en el documento.

Sintaxis: `createCollection(“collectionName”,{validator:{ fields condition }})`

Ejemplo:

```
> db.createCollection( "Login",
... { validator: { $and:
... [
... { phone: { $type: "string" } },
... { email: { $regex: /@flares.com$/ } },
... { status: { $in: [ "Registered", "Unknown" ] } }
... ]
... }
... } )
{ "ok" : 1 }
>
> db.Login.insert({phone:1234})
WriteResult({
"nInserted" : 0,
"writeError" : {
"code" : 121,
```

```
"errmsg" : "Document failed validation"
}
})
>
> db.Login.insert({phone:"1234",email:"abc@flares.com",status:"Unknown",mode:"limited"})
WriteResult({ "nInserted" : 1 })
```

24. Validadores de esquema en una nueva colección

Palabra clave adicional **$ jsonSchema** para cada año fiscal junto con la **propiedades adicionales** valor como **Falso** es necesario para poner restricción a nivel de esquema. Evita que se agreguen nuevos campos en el documento.

Sintaxis: `createCollection("collectionName",{validator: { $jsonSchema { schema condition } }})`

Ejemplo:

```
> db.createCollection( "Login", {
... validator: { $jsonSchema: {
... bsonType: "object",
... "additionalProperties": false,
... required: [ "email" ],
... properties: {
... email: {
... bsonType : "string",
... pattern : "@flares.com$",
... description: "string meets the given expression"
... },
... status: {
... enum: [ "registered", "Invalid" ],
... description: "status must be within enum values"
... }
... }
... } },
... } )
{ "ok" : 1 }
>
> db.Login.insert({email:"abc@flares.com"})
WriteResult({
"nInserted" : 0,
"writeError" : {
"code" : 121,
"errmsg" : "Document failed validation"
}
```

```
})
```

25. Actualizar o crear validadores de esquema en una colección existente

Se puede crear un validador en una colección existente usando `collMod`

Sintaxis: `runCommand({collMod:"collectionName",validator:{schema condition}})`

Ejemplo:

```
> db.runCommand( {
collMod: "Login",
validator: { $jsonSchema: {
bsonType: "object",
"additionalProperties": false,
required: [ "email","status" ],
properties: {
email: {
bsonType : "string",
pattern : "@flares.com$",
description: "string meets the given expression"
},
status: {
enum: [ "registered", "Invalid" ],
description: "status must be within enum values"
}
}
} },
validationAction: "error",
validationLevel: "strict"
} )
{ "ok" : 1 }
```

26. Eliminar validadores de esquema en una colección existente

Para eliminar validadores de esquema, es necesario configurar `<strong>validationLevel</strong>` como apagado.

Sintaxis: `runCommand({collMod:"collectionName",validator:{ },validationLevel:off})`

Ejemplo:

```
> db.runCommand({
collMod:"Login",
validator:{},
validationLevel:"off"
})
{ "ok" : 1 }
>
> db.Login.insert({"email":"abc"})
WriteResult({ "nInserted" : 1 })
```

27. Verificar validadores en una colección existente

Para verificar si la colección existente tiene validadores de esquema que se ejecutan debajo del comando. Sin especificar el nombre de la colección `db.getCollectionInfos()` El método proporciona detalles de validadores en todas las colecciones que residen dentro de una base de datos.

Sintaxis: `getCollectionInfos({name : “collectionName”})`

Ejemplo:

```
> db.getCollectionInfos({name: "Login"})
[
{
"name" : "Login",
"type" : "collection",
"options" : {
"validator" : {
"email" : {
"$regex" : /@flares.com$/
}
}
},
"info" : {
"readOnly" : false,
"uuid" : UUID("646674f6-4b06-466d-93b0-393b5f5cb4ff")
},
"idIndex" : {
"v" : 2,
"key" : {
```

```
"_id" : 1
},
"name" : "_id_",
"ns" : "geekFlareDB.Login"
}
}
]
```

Cursors related

28. Cursor en MongoDB

El cursor es un puntero para iterar sobre el conjunto de resultados. Usos de MongoDB `<strong>hasNext()</strong>` y `<strong>forEach()</strong>` método de iteración. Una lista de métodos de cursor se ha proporcionado.

Ejemplos:

```
> var newCursor=db.geekFlareCollection.find()
> newCursor.forEach(printjson)
{ "_id" : 1, "product" : "bottles", "Qty" : 100 }
{ "_id" : 2, "product" : "bread", "Qty" : 20 }
{ "_id" : 3, "product" : "yogurt", "Qty" : 30 }
>
> var newCursor1=db.geekFlareCollection.find()
> while(newCursor1.hasNext()){ printjson(newCursor1.next())}
{ "_id" : 1, "product" : "bottles", "Qty" : 100 }
{ "_id" : 2, "product" : "bread", "Qty" : 20 }
{ "_id" : 3, "product" : "yogurt", "Qty" : 30
```

Utility

29. Realización de una copia de seguridad de la base de datos

`mongodump` La utilidad se utiliza para exportar el contenido de la base de datos MongoDB como copia de seguridad. Este comando se ejecuta desde la consola del sistema y no desde mongo shell. Generará una copia de seguridad binaria junto con la información de metadatos.

Sintaxis:

`mongodump --db dbName --out outFile --host "IP:PORT"  --username <user> --password <pass>`

Ejemplo:

```
C:mongodbdump>mongodump --db geekFlareDB --out "C:mongodbdump" --host "127.0.0.1:27017"
2020-06-02T12:26:34.428+0530 writing geekFlareDB.myTable to
2020-06-02T12:26:34.430+0530 writing geekFlareDB.geekFlareCollection to
2020-06-02T12:26:34.430+0530 writing geekFlareDB.mCollection to
2020-06-02T12:26:34.433+0530 writing geekFlareDB.users to
2020-06-02T12:26:34.434+0530 done dumping geekFlareDB.myTable (2 documents)
2020-06-02T12:26:34.434+0530 done dumping geekFlareDB.geekFlareCollection (4 documents)
2020-06-02T12:26:34.435+0530 writing geekFlareDB.contacts2 to
2020-06-02T12:26:34.435+0530 writing geekFlareDB.Login to
2020-06-02T12:26:34.436+0530 done dumping geekFlareDB.mCollection (2 documents)
2020-06-02T12:26:34.437+0530 done dumping geekFlareDB.users (1 document)
2020-06-02T12:26:34.437+0530 done dumping geekFlareDB.Login (0 documents)
2020-06-02T12:26:34.438+0530 done dumping geekFlareDB.contacts2 (0 documents)
```

30. Restaurar la base de datos desde la copia de seguridad

La utilidad `<strong>mongorestore</strong>` se utiliza para restaurar datos binarios generados por `mongodump`.

Sintaxis: `mongorestore --db newDB "pathOfOldBackup"`

Ejemplo:

```
C:Usersasad.ali>mongorestore --db geekFlareNew "C:mongodbdumpgeekFlare" --host
"127.0.0.1:27017"
2020-06-09T15:49:35.147+0530 the --db and --collection args should only be used when
restoring from a BSON file. Other uses are deprecated and will not exist in the future;
use --nsInclude instead
2020-06-09T15:49:35.148+0530 building a list of collections to restore from
C:mongodbdumpgeekFlare dir
2020-06-09T15:49:35.152+0530 reading metadata for geekFlareNew.geekFlareCollection from
C:mongodbdumpgeekFlaregeekFlareCollection.metadata.json
2020-06-09T15:49:35.321+0530 restoring geekFlareNew.geekFlareCollection from
C:mongodbdumpgeekFlaregeekFlareCollection.bson
2020-06-09T15:49:35.461+0530 no indexes to restore
2020-06-09T15:49:35.462+0530 finished restoring geekFlareNew.geekFlareCollection (3
documents, 0 failures)
```

```
2020-06-09T15:49:35.467+0530 3 document(s) restored successfully. 0 document(s) failed
to restore.
```

31. Exportación de colecciones

Para exportar el contenido de la colección a un archivo (JSON o CSV) `mongoexport` Se ha proporcionado utilidad. Para ejecutar este comando, use el terminal del sistema.

- Exporta una sola colección a un archivo.

Sintaxis: `mongoexport --db dbName --collection collectionName --out outputFile`

Ejemplo:

```
C:mongodbNew folder>mongoexport --db geekFlareDB --collection geekFlareCol --out
outFile.json
2020-06-06T19:02:29.994+0530 connected to: mongodb://localhost/
2020-06-06T19:02:30.004+0530 exported 6 records
```

- Exporta un campo específico de la colección a un archivo.

Sintaxis: `mongoexport --db dbName --collection collectionName --out outputFile --fields fieldname`

Ejemplo:

```
C:mongodbNew folder>mongoexport --db geekFlareDB --collection geekFlareCol --out
outFile.json --fields product
2020-06-06T19:05:22.994+0530 connected to: mongodb://localhost/
2020-06-06T19:05:23.004+0530 exported 6 records
```

32. Importación de colecciones

Para importar datos de un archivo (CSV o JSON) `mongoimport` Se puede utilizar la herramienta de línea de comandos.

Sintaxis: `mongoimport --db dbName --collection collectionName --file inputFile`

```
C:Usersasad.ali>mongoimport --db geekFlareDB --collection geekFlareNew --file
outFile.json
2020-06-09T14:52:53.655+0530 connected to: mongodb://localhost/
```

```
2020-06-09T14:52:53.924+0530 6 document(s) imported successfully. 0 document(s) failed
to import.
```

Replica related

La replicación es diferente a la fragmentación, consulte esta guía para [implementar fragmentación](#).

33. Replicación de MongoDB

La replicación es el proceso para sincronizar datos en varios servidores. Evita la pérdida de datos debido a un mal funcionamiento del hardware o software. MongoDB logra la replicación utilizando conjuntos de réplicas. El conjunto de réplicas consta de conjuntos de datos de Mongo primarios y secundarios en el clúster.

El conjunto de datos primarios acepta todas las operaciones de escritura y las lecturas del conjunto de datos secundarios del primario. Se requieren 3 conjuntos de datos como mínimo en el conjunto de réplicas de Mongo. Se requiere el siguiente proceso para configurar un conjunto de réplicas:

- Comienzo `mongod` servidor con `replset` opción en un mínimo de 3 nodos.

`mongod --port 27017 --dbpath C:datadata1 --replSet rs0 --oplogSize 128`

`mongod --port 27018 --dbpath C:datadata1 --replSet rs0 --oplogSize 128`

`mongod --port 27019 --dbpath C:datadata1 --replSet rs0 --oplogSize 128`

- Inicialice el conjunto de réplicas.

`rs.initiate( {    _id : "rs0",    members: [   { _id: 0, host: "IP:27017" },   { _id: 1, host: "IP:27018" },   { _id: 2, host: "IP:27019" }    ] })`

```
> rs.initiate( {
... _id : "rs0",
... members: [
... { _id: 0, host: "localhost:27017" },
... { _id: 1, host: "localhost:27018" },
... { _id: 2, host: "localhost:27019" }
... ]
... })
{
```

```
"ok" : 1,
"$clusterTime" : {
"clusterTime" : Timestamp(1591089166, 1),
"signature" : {
"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
"keyId" : NumberLong(0)
}
},
"operationTime" : Timestamp(1591089166, 1)
}
```

34. Compruebe el estado de la replicación

Ejecute el siguiente comando desde el nodo de réplica principal para obtener información completa del conjunto de réplicas.

`rs.conf()`

`rs.status()`

```
rs0:PRIMARY> rs.conf()
{
"_id" : "rs0",
"version" : 2,
"protocolVersion" : NumberLong(1),
"writeConcernMajorityJournalDefault" : true,
"members" : [
{
"_id" : 0,
"host" : "localhost:27017",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 1,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 1,
"host" : "localhost:27018",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 1,
"tags" : {
```

```
},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 2,
"host" : "localhost:27019",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 1,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
},
{
"_id" : 3,
"host" : "localhost:27016",
"arbiterOnly" : false,
"buildIndexes" : true,
"hidden" : false,
"priority" : 1,
"tags" : {

},
"slaveDelay" : NumberLong(0),
"votes" : 1
}
],
"settings" : {
"chainingAllowed" : true,
"heartbeatIntervalMillis" : 2000,
"heartbeatTimeoutSecs" : 10,
"electionTimeoutMillis" : 10000,
"catchUpTimeoutMillis" : -1,
"catchUpTakeoverDelayMillis" : 30000,
"getLastErrorModes" : {

},
"getLastErrorDefaults" : {
"w" : 1,
"wtimeout" : 0
},
"replicaSetId" : ObjectId("5ed6180d01a39f2162335de5")
}
}
```

35. Agregar una nueva instancia de MongoDB a un conjunto de réplicas

Inicie el cliente principal de MongoDB y ejecute el siguiente comando

Sintaxis: `rs.add(“hostname:port”)`

Ejemplo:

```
rs0:PRIMARY> rs.add("localhost:27016")
{
"ok" : 1,
"$clusterTime" : {
"clusterTime" : Timestamp(1591094195, 1),
"signature" : {
"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
"keyId" : NumberLong(0)
}
},
"operationTime" : Timestamp(1591094195, 1)
}
```

36. Elimina la instancia de MongoDB existente del conjunto de réplicas

El siguiente comando eliminará el host secundario requerido del conjunto de réplicas.

Sintaxis: `rs.remove("localhost:27017")`

Ejemplo:

```
rs0:PRIMARY> rs.remove("localhost:27016")
{
"ok" : 1,
"$clusterTime" : {
"clusterTime" : Timestamp(1591095681, 1),
"signature" : {
"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
"keyId" : NumberLong(0)
}
},
"operationTime" : Timestamp(1591095681, 1)
```

```
}
rs0:PRIMARY>
```

37. Establecer como primario como conjunto de réplicas secundario

MongoDB proporciona un comando para indicar a la réplica principal que se convierta en un conjunto de réplicas secundario.

Sintaxis: `rs.stepDown( stepDownSecs , secondaryCatchupSecs )`

Ejemplo:

```
rs0:PRIMARY> rs.stepDown(12)
{
"ok" : 1,
"$clusterTime" : {
"clusterTime" : Timestamp(1591096055, 1),
"signature" : {
"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
"keyId" : NumberLong(0)
}
},
"operationTime" : Timestamp(1591096055, 1)
}
rs0:SECONDARY>
```

38. Compruebe el retraso de la réplica entre primaria y secundaria

El siguiente comando se utilizará para verificar el retraso de replicación entre todos los conjuntos de réplicas del primario.

Sintaxis: `rs.printSlaveReplicationInfo()`

Ejemplo:

```
rs0:PRIMARY> rs.printSlaveReplicationInfo()
source: localhost:27018
syncedTo: Tue Jun 02 2020 16:14:04 GMT+0530 (India Standard Time)
0 secs (0 hrs) behind the primary
source: localhost:27019
```

```
syncedTo: Thu Jan 01 1970 05:30:00 GMT+0530 (India Standard Time)
1591094644 secs (441970.73 hrs) behind the primary
source: localhost:27016
syncedTo: Tue Jun 02 2020 16:14:04 GMT+0530 (India Standard Time)
0 secs (0 hrs) behind the primary
rs0:PRIMARY>
```

Transactions related

39. Transacciones en MongoDB

MongoDB admite propiedades ACID para transacciones en documentos.

Para iniciar una transacción, se requiere una sesión para comenzar y se requiere un compromiso para guardar los cambios en la base de datos. Las transacciones se admiten en juegos de réplicas o mangos. Una vez que la sesión se comprometió con éxito, las operaciones realizadas dentro de la sesión serán visibles en el exterior.

- Iniciar sesión

Sintaxis: `session =db.getMongo().startSession()`

- Iniciar transacción,

Sintaxis: `session.startTransaction()`

- Confirmar transacción

Sintaxis: `session.commitTransaction()`

Ejemplo:

Creemos una sesión, iniciemos la transacción, realicemos alguna inserción / actualización y luego confirmemos la transacción.

```
rs0:PRIMARY> session = db.getMongo().startSession()
session { "id" : UUID("f255a40d-81bd-49e7-b96c-9f1083cb4a29") }
rs0:PRIMARY> session.startTransaction()
rs0:PRIMARY> session.getDatabase("geekFlareDB").geekFlareCollection.insert([
... {_id: 4 , product: "ketchup"},
... {_id: 5, product: "Cheese"}
... ]);
BulkWriteResult({
"writeErrors" : [ ],
"writeConcernErrors" : [ ],
```

```
"nInserted" : 2,
"nUpserted" : 0,
"nMatched" : 0,
"nModified" : 0,
"nRemoved" : 0,
"upserted" : [ ]
})
rs0:PRIMARY> session.getDatabase("geekFlareDB").geekFlareCollection.find()
{ "_id" : 1, "product" : "bread", "Qty" : 20 }
{ "_id" : 2, "product" : "bottles", "Qty" : 100 }
{ "_id" : 3, "product" : "bread", "Qty" : 20 }
{ "_id" : 4, "product" : "ketchup" }
{ "_id" : 5, "product" : "Cheese" }
rs0:PRIMARY> db.geekFlareCollection.find()
{ "_id" : 1, "product" : "bread", "Qty" : 20 }
{ "_id" : 2, "product" : "bottles", "Qty" : 100 }
{ "_id" : 3, "product" : "bread", "Qty" : 20 }
rs0:PRIMARY> session.commitTransaction()
rs0:PRIMARY> db.geekFlareCollection.find()
{ "_id" : 1, "product" : "bread", "Qty" : 20 }
{ "_id" : 2, "product" : "bottles", "Qty" : 100 }
{ "_id" : 3, "product" : "bread", "Qty" : 20 }
{ "_id" : 4, "product" : "ketchup" }
{ "_id" : 5, "product" : "Cheese" }
rs0:PRIMARY> session.getDatabase("geekFlareDB").geekFlareCollection.find()
{ "_id" : 1, "product" : "bread", "Qty" : 20 }
{ "_id" : 2, "product" : "bottles", "Qty" : 100 }
{ "_id" : 3, "product" : "bread", "Qty" : 20 }
{ "_id" : 4, "product" : "ketchup" }
{ "_id" : 5, "product" : "Cheese" }
```

40. Conflicto de transacciones de un solo documento

Si dos transacciones intentaron actualizar el mismo documento, MongoDB arroja un error de conflicto de escritura.

- `session1.startTransaction()`
- `session2.startTransaction()`

Realice alguna inserción / actualización en Session1 seguida de en Session2. Ahora observe el error en el siguiente ejemplo

Ejemplo:

```
rs0:PRIMARY> session1.startTransaction()
```

```
rs0:PRIMARY> session1.getDatabase("geekFlareDB").geekFlareCollection.update({_id:3},
{$set:{ product: "Bread" }})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
rs0:PRIMARY> session2.getDatabase("geekFlareDB").geekFlareCollection.update({_id:3},
{$set:{ product: "Snacks" }})
WriteCommandError({
"errorLabels" : [
"TransientTransactionError"
],
"operationTime" : Timestamp(1591174593, 1),
"ok" : 0,
"errmsg" : "WriteConflict",
"code" : 112,
"codeName" : "WriteConflict",
"$clusterTime" : {
"clusterTime" : Timestamp(1591174593, 1),
"signature" : {
"hash" : BinData(0,"AAAAAAAAAAAAAAAAAAAAAAAAAAA="),
"keyId" : NumberLong(0)
}
}
})
rs0:PRIMARY>
```

41. Transacciones de varios documentos

MongoDB admite transacciones de varios documentos en una sola sesión.

- db.getMongo ().startTransaction ()

Realice alguna inserción / actualización en varios documentos

- de preguntas y respuestas.commitTransaction ()

Ejemplo:

```
rs0:PRIMARY> var session = db.getMongo().startSession()
rs0:PRIMARY> session.startTransaction()
rs0:PRIMARY> session.getDatabase("geekFlareDB").geekFlareCollection.update({_id:3},
{$set:{ product: "Snacks3" }})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })
rs0:PRIMARY> session.getDatabase("geekFlareDB").geekFlareCollection.update({_id:2},
{$set:{ product: "Snacks2" }})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 0 })
rs0:PRIMARY> session.commitTransaction()
rs0:PRIMARY> db.geekFlareCollection.find()
{ "_id" : 1, "product" : "bread", "Qty" : 20 }
{ "_id" : 2, "product" : "Snacks2", "Qty" : 100 }
{ "_id" : 3, "product" : "Snacks3", "Qty" : 20 }
```

```
{ "_id" : 4, "product" : "ketchup" }
{ "_id" : 5, "product" : "Cheese" }
rs0:PRIMARY>
```

42. Perfilado en MongoDB

La creación de perfiles ayuda a registrar consultas lentas en el `system.profile` colección. Nivel de perfilador y la frecuencia de muestreo define el porcentaje de consultas que se iniciarán sesión `system.profile` colección.

- Establecer / obtener nivel de perfil

Sintaxis:

`db.setProfilingLevel(profilingLevel, {"slowms":time,"sampleRate":LoggingPercentage})`

```
> db.setProfilingLevel(2,{"slowms":1,"sampleRate":1})
{ "was" : 1, "slowms" : 1, "sampleRate" : 0.5, "ok" : 1 }
>
> db.getProfilingLevel()
2
```

- Obtener estado de perfil

Sintaxis: db.getProfilingStatus ()

```
> db.getProfilingStatus()
{ "was" : 2, "slowms" : 1, "sampleRate" : 1 }
```

- Para habilitar la generación de perfiles en el nivel de instancia de MongoDB, inicie la instancia con la información del generador de perfiles o agregue los detalles del generador de perfiles en el archivo de configuración.

Sintaxis:

`mongod --profile <Level> --slowms <time> --slowOpSampleRate <%Logging>`

Ejemplo:

```
C:WindowsSystem32>mongod --port 27017 --dbpath C:datadata1 --profile 1 --slowms 25 --
slowOpSampleRate 0.5
2020-06-09T02:34:41.110-0700 I CONTROL [main] Automatically disabling TLS 1.0, to force-
enable TLS 1.0 specify --sslDisabledProtocols 'none'
```

```
2020-06-09T02:34:41.113-0700 W ASIO [main] No TransportLayer configured during
NetworkInterface startup
2020-06-09T02:34:41.113-0700 I CONTROL [initandlisten] MongoDB starting : pid=22604
port=27017 dbpath=C:datadata1 64-bit host=MCGL-4499
2020-06-09T02:34:41.114-0700 I CONTROL [initandlisten] targetMinOS: Windows 7/Windows
Server 2008 R2
2020-06-09T02:34:41.116-0700 I CONTROL [initandlisten] db version v4.2.7
2020-06-09T02:34:41.116-0700 I CONTROL [initandlisten] git version:
51d9fe12b5d19720e72dcd7db0f2f17dd9a19212
```

43. MongoDB Explain ()

MongoDB `explains()` El método devuelve estadísticas y proporciona información para seleccionar un plan ganador y ejecutarlo hasta su finalización. Devuelve resultados según el plan de verbosidad.

Sintaxis: `collectionName.explain("verbosityName")`

Para ejecutar el método / comandoexplica (), creemos un **verbosidad** y luego ejecute el método explica (), eche un vistazo al siguiente ejemplo, donde se han ejecutado estos pasos.

Ejemplo:

```
> db.runCommand(
... {
... explain: { count: "product", query: { Qty: { $gt: 10 } } },
... verbosity: "executionStats"
... }
... )
{
"queryPlanner" : {
"plannerVersion" : 1,
"namespace" : "test.product",
"indexFilterSet" : false,
"winningPlan" : {
"stage" : "COUNT",
"inputStage" : {
"stage" : "EOF"
}
},
"rejectedPlans" : [ ]
},
"executionStats" : {
"executionSuccess" : true,
"nReturned" : 0,
```

```
"executionTimeMillis" : 47,
"totalKeysExamined" : 0,
"totalDocsExamined" : 0,
"executionStages" : {
"stage" : "COUNT",
"nReturned" : 0,
"executionTimeMillisEstimate" : 0,
"works" : 1,
"advanced" : 0,
"needTime" : 0,
"needYield" : 0,
"saveState" : 0,
"restoreState" : 0,
"isEOF" : 1,
"nCounted" : 0,
"nSkipped" : 0,
"inputStage" : {
"stage" : "EOF",
"nReturned" : 0,
"executionTimeMillisEstimate" : 0,
"works" : 0,
"advanced" : 0,
"needTime" : 0,
"needYield" : 0,
"saveState" : 0,
"restoreState" : 0,
"isEOF" : 1
}
}
},
"serverInfo" : {
"host" : "MCGL-4499",
"port" : 27017,
"version" : "4.2.7",
"gitVersion" : "51d9fe12b5d19720e72dcd7db0f2f17dd9a19212"
},
"ok" : 1
}
>
```

```
> var expv = db.geekFlareCol.explain("executionStats")
> expv.find( { product: "bread"} )
{
"queryPlanner" : {
"plannerVersion" : 1,
"namespace" : "geekFlareDB.geekFlareCol",
"indexFilterSet" : false,
"parsedQuery" : {
"product" : {
```

```
"$eq" : "bread"
}
},
"winningPlan" : {
"stage" : "COLLSCAN",
"filter" : {
"product" : {
"$eq" : "bread"
}
},
"direction" : "forward"
},
"rejectedPlans" : [ ]
},
"executionStats" : {
"executionSuccess" : true,
"nReturned" : 2,
"executionTimeMillis" : 0,
"totalKeysExamined" : 0,
"totalDocsExamined" : 6,
"executionStages" : {
"stage" : "COLLSCAN",
"filter" : {
"product" : {
"$eq" : "bread"
}
},
"nReturned" : 2,
"executionTimeMillisEstimate" : 0,
"works" : 8,
"advanced" : 2,
"needTime" : 5,
"needYield" : 0,
"saveState" : 0,
"restoreState" : 0,
"isEOF" : 1,
"direction" : "forward",
"docsExamined" : 6
}
},
"serverInfo" : {
"host" : "MCGL-4499",
"port" : 27017,
"version" : "4.2.7",
"gitVersion" : "51d9fe12b5d19720e72dcd7db0f2f17dd9a19212"
},
"ok" : 1
}
>
```

Access control related

44. Control de acceso en MongoDB

Las funciones de control de acceso permiten el acceso de autenticación a los usuarios existentes. Para el control de acceso, la base de datos habilitada para garantizar la creación de un rol de administrador de usuario en la base de datos de administración.

- Conecte la base de datos sin autenticación.
- Cambiar a la base de datos

`use admin`

- Crear usuario como a continuación

`db.createUser( {user: “UserAdmin”, pwd: “password” ,role: [adminRole])`

Ejemplo:

```
db.createUser(
... {
... user: "AdminUser",
... pwd: passwordPrompt(),
... roles: [ { role: "userAdminAnyDatabase", db: "admin" }, "readWriteAnyDatabase" ]
... }
... )
Enter password:
Successfully added user: {
"user" : "AdminUser",
"roles" : [
{
"role" : "userAdminAnyDatabase",
"db" : "admin"
},
"readWriteAnyDatabase"
]
}
```

- Reanudar `mongod` ejemplo
- Acceda nuevamente con el usuario y la contraseña creados.

```
C:Users>mongo --port 27017 --authenticationDatabase "admin" -u "AdminUser" -p
MongoDB shell version v4.2.7
Enter password:
connecting to: mongodb://127.0.0.1:27017/?
authSource=admin&compressors=disabled&gssapiServiceName=mongodb
```

45. Recuperar y eliminar usuario de control de acceso

El siguiente comando se puede usar para verificar la información del usuario y eliminarla.

- `db.getUser("AdminUser")`
- `db.dropUser("AdminUser")`

Ejemplo:

```
> db.getUser("AdminUser")
{
"_id" : "admin.AdminUser",
"userId" : UUID("78d2d5bb-0464-405e-b27e-643908a411ce"),
"user" : "AdminUser",
"db" : "admin",
"roles" : [
{
"role" : "userAdminAnyDatabase",
"db" : "admin"
},
{
"role" : "readWriteAnyDatabase",
"db" : "admin"
}
],
"mechanisms" : [
"SCRAM-SHA-1",
"SCRAM-SHA-256"
]
}
> db.dropUser("AdminUser")
true
> db.getUser("AdminUser")
null
>
```

46. Otorgar funciones definidas por el usuario

MongoDB proporciona `db.createRole()` método para especificar los privilegios de un usuario y una serie de roles heredados.

- Conecte la instancia de MongoDB con el usuario administrador.
- Ejecute el siguiente comando para generar un nuevo rol.

Sintaxis:

`db.createRole({role:”roleName”,privileges:[{privilegeName}],roles:[InheritedArray]})`

Ejemplo:

```
use admin
> db.createRole(
... {
... role: "abc",
... privileges: [ { resource: { db: "geekFlareDB", collection: "geekFlareCol" },
actions: [ "killop" ,"inprog"] } ],
... roles: []
... }
... )
{
"role" : "abc",
"privileges" : [
{
"resource" : {
"db" : "geekFlareDB",
"collection" : "geekFlareCol"
},
"actions" : [
"killop",
"inprog"
]
}
],
"roles" : [ ]
}
```

47. Revocar funciones definidas por el usuario

Para modificar los roles existentes, use el siguiente comando.

Sintaxis: `db.revokeRolesFromUser( userName, [{ "role" : roleName , db:dbName} ] )`

Ejemplo:

```
> db.getUser("AdminUser")
{
"_id" : "admin.AdminUser",
"userId" : UUID("fe716ed1-6771-459e-be13-0df869c91ab3"),
```

```
"user" : "AdminUser",
"db" : "admin",
"roles" : [
{
"role" : "userAdminAnyDatabase",
"db" : "admin"
},
{
"role" : "readWriteAnyDatabase",
"db" : "admin"
}
],
"mechanisms" : [
"SCRAM-SHA-1",
"SCRAM-SHA-256"
]
}
> db.revokeRolesFromUser( "AdminUser", [{ "role" : "userAdminAnyDatabase" , db:"admin"}
] )
> db.getUser("AdminUser")
{
"_id" : "admin.AdminUser",
"userId" : UUID("fe716ed1-6771-459e-be13-0df869c91ab3"),
"user" : "AdminUser",
"db" : "admin",
"roles" : [
{
"role" : "readWriteAnyDatabase",
"db" : "admin"
}
],
"mechanisms" : [
"SCRAM-SHA-1",
"SCRAM-SHA-256"
]
}
>
```

48. Conectividad de MongoDB con Python

Se requiere el paquete pymongo para conectar MongoDB desde la consola de Python.

```
>>> from pymongo import MongoClient
>>> mClient=MongoClient("mongodb://127.0.0.1:27017/")
>>> mDB=mClient.geekFlareDB
>>> mRecord={"_id":4,"name":"XYZ"}
>>> mDB.geekFlareCollection.insert_one(mRecord)
<pymongo.results.InsertOneResult object at 0x000002CC44256F48>
>>> for i in mDB.geekFlareCollection.find({"_id":4}):
```

```
... print(i)
...
{'_id': 4, 'name': 'XYZ'}
>>>
```

¿Qué es lo siguiente?

Mira esta lista de Clientes NoSQL para administrar MongoDB y otras bases de datos NoSQL. Si su trabajo implica trabajar con frecuencia en MongoDB, es posible que desee obtener más información de este Curso Udemy.