MongoDB

rortegag

Connect MongoDB Shell

Conectar a mongodb://127.0.0.1:27017 - por defecto.
```
mongo
```
Omitir la contraseña si desea un prompt.
```
mongo --host <host> --port <puerto> -u <usuario> -p <contraseña>
```
MongoDB Atlas.
```
mongo "mongodb://192.168.1.1:27017"
mongo "mongodb+srv://cluster-name.abcde.mongodb.net/<nombre-bd>" --username <usuario>
```

Para usar con el nuevo mongosh, remplaza por mongo por mongosh.

Helpers

Mostrar todas las bases de datos.
```
show dbs
```
Muestra la base de datos actual.
```
db
```
Cambiar base de datos.
```
use <nombre_bd>
```
Mostrar colecciones.
```
show collections
```
Ejecutar archivo JavaScript.
```
load("myScript.js")
```

Índices

Listar índices.
```
db.coll.getIndexes()
db.coll.getIndexKeys()
```

Crear índices.

Tipos de índices.

Índice de campo único.
```
db.coll.createIndex({"name": 1})
```
Índice compuesto.
```
db.coll.createIndex(
  {"name": 1, "date": 1})
```
Índice del texto.
```
db.coll.createIndex(
  {foo: "text", bar: "text"})
```
Índice de texto comodín.
```
db.coll.createIndex({"$**": "text"})
```
Índice comodín.
```
db.coll.createIndex(
  {"userMetadata.$**": 1})
```
Índice 2D.
```
db.coll.createIndex({"loc": "2d"})
```
Índice 2dsphere.
```
db.coll.createIndex(
  {"loc": "2dsphere"})
```
Índice hash.
```
db.coll.createIndex(
  {"_id": "hashed"})
```

MÁS ↓

CRUD

Create (Crear).
```
db.coll.insertOne({name: "Max"})
```
Inserción masiva ordenada.
```
db.coll.insert([{name: "Max"}, {name:"Alex"}])
```
Inserción masiva desordenada.
```
db.coll.insert([{name: "Max"}, {name:"Alex"}], {ordered: false})
```
Uso de función.
```
db.coll.insert({date: ISODate()})
```
Con cometido de escritura.
```
db.coll.insert({name: "Max"},
  {"writeConcern": {"w": "majority", "wtimeout": 5000}})
```

Read (Lectura).

Devuelve un solo documento.
```
db.coll.findOne()
```
Devuelve un cursor - muestra 20 resultados - "it" para mostrar más.
```
db.coll.find()
```
Devuelve un cursor formateado para mejor lectura.
```
db.coll.find().pretty()
```
Lógica implícita "AND".
```
db.coll.find({name: "Max", age: 32})
db.coll.find({date: ISODate("2020-09-25T13:57:17.180Z")})
```
O bien "queryPlanner" o "allPlansExecution".
```
db.coll.find({name: "Max", age: 32}).explain("executionStats")

db.coll.distinct("name")
```

Contar.
```
db.coll.count({age: 32})
```
Estimación basada en los metadatos de la colección.
```
db.coll.estimatedDocumentCount()
```
Alias para una canalización de agregación - recuento exacto.
```
db.coll.countDocuments({age: 32})
```

Comparación.
```
db.coll.find({"year": {$gt: 1970}})
db.coll.find({"year": {$gte: 1970}})
db.coll.find({"year": {$lt: 1970}})
db.coll.find({"year": {$lte: 1970}})
db.coll.find({"year": {$ne: 1970}})
db.coll.find({"year": {$in: [1958, 1959]}})
db.coll.find({"year": {$nin: [1958, 1959]}})
```

MÁS ↓

Para más información: https://www.mongodb.com/docs/

MongoDB

rortegag

Índices

Opciones de índice.
Índice TTL.

```
db.coll.createIndex(
  {"lastModifiedDate": 1},
  {expireAfterSeconds: 3600})
db.coll.createIndex({"name": 1},
  {unique: true})
```

Índice parcial.

```
db.coll.createIndex({"name": 1},
  {partialFilterExpression:
  {age: {$gt: 18}}})
```

Índice insensible a mayúsculas y minúsculas con fuerza = 1 o 2.

```
db.coll.createIndex({"name": 1},
  {collation: {locale: 'en',
    strength: 1}})
db.coll.createIndex({"name": 1 },
  {sparse: true})
```

Quitar índices.

```
db.coll.dropIndex("name_1")
```

Ocultar/desocultar índices.

```
db.coll.hideIndex("name_1")
db.coll.unhideIndex("name_1")
```

Cambiar Streams

```
watchCursor = db.coll.watch( [
  { $match :
    {"operationType" : "insert" }
  }
] )

while (!watchCursor.isExhausted()) {
  if (watchCursor.hasNext()){
    print(tojson(watchCursor.next()));
  }
}
```

Replica Set

```
rs.status()
rs.initiate({"_id": "replicaTest",
 members: [
  { _id: 0, host: "127.0.0.1:27017" },
  { _id: 1, host: "127.0.0.1:27018" },
  { _id: 2, host: "127.0.0.1:27019",
    arbiterOnly:true }]
})
```

MÁS ↓

CRUD

Lógica.

```
db.coll.find({name:{$not: {$eq: "Max"}}})
db.coll.find({$or: [{"year" : 1958}, {"year" : 1959}]})
db.coll.find({$nor: [{price: 1.99}, {sale: true}]})
db.coll.find({
 $and: [
  {$or: [{qty: {$lt :10}}, {qty :{$gt: 50}}]},
  {$or: [{sale: true}, {price: {$lt: 5 }}]}
 ]
})
```

Elemento.

```
db.coll.find({name: {$exists: true}})
db.coll.find({"zipCode": {$type: 2 }})
db.coll.find({"zipCode": {$type: "string"}})
```

Tubería de agregación.

```
db.coll.aggregate([
 {$match: {status: "A"}},
 {$group: {_id: "$cust_id", total: {$sum: "$amount"}}},
 {$sort: {total: -1}}
])
```

Búsqueda de texto con un índice “texto”.

```
db.coll.find({$text: {$search: "cake"}}, {score: {$meta: "textScore"}})
   .sort({score: {$meta: "textScore"}})
```

Regex.
Comienza por la letra "M".

```
db.coll.find({name: /^Max/})
```

Sin distinguir mayúsculas de minúsculas.

```
db.coll.find({name: /^Max$/i})
```

Array.

```
db.coll.find({tags: {$all: ["Realm", "Charts"]}})
```

Imposible de indexar - es mejor almacenar el tamaño del array y actualizarlo.

```
db.coll.find({field: {$size: 2}})
db.coll.find({results: {$elemMatch: {product: "xyz", score: {$gte: 8}}}})
```

Proyecciones.
actores + _id.

```
db.coll.find({"x": 1}, {"actors": 1})
```

actores.

```
db.coll.find({"x": 1}, {"actors": 1, "_id": 0})
```

Todo, menos "actores" y "resumen".

```
db.coll.find({"x": 1}, {"actors": 0, "summary": 0})
```

Ordenar, omitir, limitar.

```
db.coll.find({}).sort({"year": 1, "rating": -1}).skip(10).limit(3)
```

Con cometido de lectura.

```
db.coll.find().readConcern("majority")
```

MÁS ↓

Para más información: https://www.mongodb.com/docs/

MongoDB

rortegag

Replica Set

```
rs.add
  ("mongodbd1.example.net:27017")
rs.addArb
  ("mongodbd2.example.net:27017")
rs.remove
  ("mongodbd1.example.net:27017")
rs.conf()
rs.isMaster()
rs.printReplicationInfo()
rs.printSlaveReplicationInfo()
rs.reconfig(<valid_conf>)
rs.slaveOk()
(stepDownSecs,
secondaryCatchUpPeriodSecs)
rs.stepDown(20, 5)
```

Sharded Cluster

```
sh.status()
sh.addShard
  ("rs1/mongodbd1.example.net:27017")
sh.shardCollection("mydb.coll",
  {zipcode: 1})

sh.moveChunk("mydb.coll",
  { zipcode: "53187" }, "shard0019")
sh.splitAt("mydb.coll", {x: 70})
sh.splitFind("mydb.coll", {x: 70})
sh.disableAutoSplit()
sh.enableAutoSplit()

sh.startBalancer()
sh.stopBalancer()
sh.disableBalancing("mydb.coll")
sh.enableBalancing("mydb.coll")
sh.getBalancerState()
sh.setBalancerState(true/false)
sh.isBalancerRunning()

sh.addTagRange("mydb.coll",
  {state: "NY", zip: MinKey },
  { state: "NY", zip: MaxKey }, "NY")
sh.removeTagRange("mydb.coll",
  {state: "NY", zip: MinKey },
  { state: "NY", zip: MaxKey }, "NY")
sh.addShardTag("shard0000", "NYC")
sh.removeShardTag("shard0000", "NYC")

sh.addShardToZone
  ("shard0000", "JFK")
sh.removeShardFromZone
  ("shard0000", "NYC")
sh.removeRangeFromZone
  ("mydb.coll", {a: 1, b: 1}, {a: 10, b: 10})
```

CRUD

Update (Actualizar).

(ATENCIÓN) Sustituye a todo el documento.

```
db.coll.update({"_id": 1}, {"year": 2016})
db.coll.update({"_id": 1}, {$set: {"year": 2016, name: "Max"}})
db.coll.update({"_id": 1}, {$unset: {"year": 1}})
db.coll.update({"_id": 1}, {$rename: {"year": "date"} })
db.coll.update({"_id": 1}, {$inc: {"year": 5}})
db.coll.update({"_id": 1}, {$mul: {price: NumberDecimal("1.25"), qty: 2}})
db.coll.update({"_id": 1}, {$min: {"imdb": 5}})
db.coll.update({"_id": 1}, {$max: {"imdb": 8}})
db.coll.update({"_id": 1}, {$currentDate: {"lastModified": true}})
db.coll.update({"_id": 1}, {$currentDate: {"lastModified": {$type: "timestamp"}}})
```

Array.

```
db.coll.update({"_id": 1}, {$push :{"array": 1}})
db.coll.update({"_id": 1}, {$pull :{"array": 1}})
db.coll.update({"_id": 1}, {$addToSet :{"array": 2}})
```

Primer elemento.

```
db.coll.update({"_id": 1}, {$pop: {"array": -1}})
```

Último elemento.

```
db.coll.update({"_id": 1}, {$pop: {"array": 1}})

db.coll.update({"_id": 1}, {$pullAll: {"array" :[3, 4, 5]}})
db.coll.update({"_id": 1}, {$push: {scores: {$each: [90, 92, 85]}}})
db.coll.updateOne({"_id": 1, "grades": 80}, {$set: {"grades.$": 82}})
db.coll.updateMany({}, {$inc: {"grades.$[]": 10}})
db.coll.update({}, {$set: {"grades.$[element]": 100}},
  {multi: true, arrayFilters: [{"element": {$gte: 100}}]})
```

Actualizar muchos.

```
db.coll.update({"year": 1999}, {$set: {"decade": "90's"}}, {"multi":true})
db.coll.updateMany({"year": 1999}, {$set: {"decade": "90's"}})
```

FindOneAndUpdate.

```
db.coll.findOneAndUpdate({"name": "Max"}, {$inc: {"points": 5}},
  {returnNewDocument: true})
```

Upsert.

```
db.coll.update({"_id": 1}, {$set: {item: "apple"},
  $setOnInsert: {defaultQty: 100}}, {upsert: true})
```

Reemplazar.

```
db.coll.replaceOne({"name": "Max"},
  {"firstname": "Maxime", "surname": "Beugnet"})
```

Guarda.

```
db.coll.save({"item": "book", "qty": 40})
```

Con cometido con escritura.

```
db.coll.update({}, {$set: {"x": 1}},
  {"writeConcern": {"w": "majority", "wtimeout": 5000}})
```

MÁS

MongoDB

rortegag

CRUD

Delete (Borrar).

```
db.coll.remove({name: "Max"})
db.coll.remove({name: "Max"}, {justOne: true})
```

ATENCIÓN Elimina todos los documentos, pero no la colección en sí ni sus definiciones de índice.

```
db.coll.remove({})
```

Elimina con contenación de escritura.

```
db.coll.remove({name: "Max"}, {"writeConcern": {"w": "majority", "wtimeout": 5000}})
```

Busca uno y lo elimina.

```
db.coll.findOneAndDelete({"name": "Max"})
```

Bases de datos y colecciones

Crear una colección.

Crear una colección con un $jsonschema.

```
db.createCollection("contacts", {
  validator: {$jsonSchema: {
    bsonType: "object",
    required: ["phone"],
    properties: {
      phone: {
        bsonType: "string",
        description: "debe ser una cadena de texto y
          es obligatorio"
      },
      email: {
        bsonType: "string",
        pattern: "@mongodb\.com$",
        description: "debe ser una cadena de texto y
          coincidir con el patrón de expresión regular"
      },
      status: {
        enum: [ "Desconocido", "Incompleto" ],
        description: "sólo puede ser uno de los valores
          del enum"
      }
    }
  }}
})
```

Borrar.

Elimina la colección y sus definiciones de índice.

```
db.coll.drop()
```

Compruebe que *NO* está en el clúster de producción...

```
db.dropDatabase()
```

Otras funciones de colección.

```
db.coll.stats()
db.coll.storageSize()
db.coll.totalIndexSize()
db.coll.totalSize()
db.coll.validate({full: true})
```

2º parámetro para eliminar la colección de destino si existe.

```
db.coll.renameCollection("nueva_coleccion", true)
```

Comandos útiles

```
use admin
db.createUser({
  "user": "root",
  "pwd": passwordPrompt(),
  "roles": ["root"]
})
db.dropUser("root")
db.auth( "user", passwordPrompt() )

use test
db.getSiblingDB("dbname")
db.currentOp()
db.killOp(123) // opid

db.fsyncLock()
db.fsyncUnlock()
```

```
db.getCollectionNames()
db.getCollectionInfos()
db.printCollectionStats()
db.stats()

db.getReplicationInfo()
db.printReplicationInfo()
db.isMaster()
db.hostInfo()
db.printShardingStatus()
db.shutdownServer()
db.serverStatus()

db.setSlaveOk()
db.getSlaveOk()
```

```
db.getProfilingLevel()
db.getProfilingStatus()
```

0 == OFF, 1 == ON con lentitud, 2 == ON

```
db.setProfilingLevel(1, 200)

db.enableFreeMonitoring()
db.disableFreeMonitoring()
db.getFreeMonitoringStatus()

db.createView("viewName", "sourceColl",
  [{$project:{department: 1}}])
```

Para más información: https://www.mongodb.com/docs/