MongoDBSpain.com
Shell Cheat Sheet for mongoDB 2.4

César Trigo Esteban
info@mongodbspain.com

Command Line

mongo | Start the shell from the command line

Command line options:
--hostname localhost | hostname to connect
--port 27017 | connect to port 27017 (default)
-u foo | username foo
-p bar | password bar
--athenticationDatabase arg | database to authenticate

mongoimport | import a data from a file into MongoDB
> mongoimport --db <db> -c <coll> --file <filename> --type <type>
mongodump | Dumps contents of a db to a file
> mongodump --db <db> -c <coll>
mongorestore | restore from a dump to MongoDB
> mongorestore --db <db> -c <coll> <bson file>

Basic Shell commands

help| get help for the context you are in
exit | exit the shell

use <database> | select and use database
db | show selected database
show dbs | show databases on server
show collections (show tables) | show collections in current db
show users | show users in current database

Collection commands

help() | show a list of help commands for a collection

copyTo(name) | copy a collection to a new collection name
count() | get number of documents in a collection
drop() | remove the collection from the database
mapReduce() | performs map-reduce data aggregate
renameCollection(name) | Rename a collection
stats() | get stats about the collection

Cursors (db.collection.find().)

it | iterate on a cursor
count() | return a count of the documents on a cursor
explain() | get query execution plan for a cursor
hasNext() | true if cursor has more docs and can be iterated
hint({age: 1}) | force to use an specific index for a query
limit(5) | limits the size of the cursor result set
next() | return the next document in a cursor
skip() | skip through some documents and the return cursor
sort({age:-1}) | return results ordered ascending/descending
toArray() | return an array of all documents in the cursor

pretty() | pretty print the returned documents

DB Commands (db.)

help() | show a list of help commands for a dbIndexing

copyDatabase() | copies a db to another db
> db.copyDatabase('records', 'archive_records')
dropDatabase() | remove the current db
getLastError() | get status of last error
hostInfo() | getinfo about the host system
serverStatus() | get an overview of server status
shutdownServer() | shutdown current server
stats() | get stats on the current db selected
version() | get the current version of the server

Authentication (db.)

addUser() | add a user to system.users or admin collection
> db.addUser({user: "username", pwd: "password",roles: ["readWrite"] })
changeUserPassword() | change an existing users password
> db.changeUserPassword("reporting", "SOhSS3")
removeUser() | remove a user from a database
> db.changeUserPassword("reporting")

auth() | authenticates a user to a database
> db.auth("reporting", "SOhSS3")
logout() | logout from a database

Top & Stats System Commands

./mongotop | Shows time spent per operation per collection
./mongostat | Shows snapshot on the MongoDB System

Query commands (db.collection.)

Finding documents:
find().pretty() | Finds all documents using nice formatting
find({}, {name:true, _id:false}) | retrieve only *name* field
findOne() | Finds one arbitrary document
findOne({name:'abc'}) | Finds one document by attribute

Inserting document:
insert({name:'a'}) | Insert new document in the collection

Removing document:
remove() |Remove all collection documents
remove({name:'a'}) |Remove by criteria

Updating documents:
update({name:'a'}, {age:25}) | replaces the whole document
update({name:'a'}, {$set:{age:25}}) | change certain attribute
update({name:'b'}, {$unset:{age:1}}) | unset attribute
findAndModify({query:{...}, sort:{...},update:{...}}) | Automatically find and update

Query operators ($)

Comparison:
Ex: db.<collection>.find({ qty: { $gt: 20 } })
$gt | matches values greater than the value
$in | matches values supplied in an array
$gte | matches values greater than or equal the value
$lte | matches values less than or equal the value
$ne | matches all values that are not equal to given
$lt | matches values less than the value
$nin | matches values that do not exist in an array

Logical:
Ex: db.<coll>.find({ price:9, $or: [{ qty: { $lt:20 } },{ sale: true}]})
$or | joins query clauses with a logical OR
$and | joins query clauses with a logical AND
$not | returns documents that do not match
$nor | joins query clauses with a logical NOR

Evaluation:
$exists | matches documents that have a field
> db.inventory.find({ qty: { $exists: true, $nin: [5, 15] } })
$type | matches a field if it is of a given BSON type
> db.inventory.find({ price: { $type : 1 } })

Evaluation:
$mod | perform a modulo on a field and select if 0
> db.inventory.find({ qty: { $mod: [4, 0] } })
$regex | matches a regex expression on a field
> db.collection.find({ field : /acme.*corp/i })
$where | matches against a JavaScript expression
> db.myCollection.find({ $where: "this.credits - this.debits < 0" })

Geospatial:
$geoWithin | matches within a bounding GeoJSON geometry
> db.coll.find({ loc : { $geoWithin : { $geometry : { type : "Polygon" , coordinates : [[[0,0] , [3,6] ,[6,1] , [0,0]]] } } })
$geoIntersects | matches all docs that intersect with a GeoJSON object
> db.coll.find({ loc : { $geoIntersects: { $geometry : { type : "Polygon" , coordinates : [[[0,0] , [3,6] ,[6,1] , [0,0]]] } } })
$near | matches near a geospatial point
> db.place.find({ loc :{ $near : [40 , 5] ,$maxDistance : 10 } })
> db.place.find({ loc : { $near :{ $geometry :{ type : "Point" , coordinates: [40 , 5] } },$maxDistance : 500} })
$nearSphere | matches near a point on a sphere
> db.place.find({ loc :{ $nearSphere : [40 , 5] ,$maxDistance : 10 } })
> db.place.find({ loc : { $nearSphere :{ $geometry :{ type : "Point" , coordinates: [40 , 5] } },$maxDistance : 500} })

Geospatial Operators:
$geometry | specifies a GeoJSON for a geospatial query
$maxDistance | specifies the distance for $near & $nearSphere
$center | return documents within the circle center + radius
> db.place.find({ loc: { $geoWithin :{ $center : [[-74, 40], 10] } } })
$centerSphere | return documents within spherical geometry
> db.pl.find({loc:{$geoWithin :{ $centerSphere: [[88 , 30] , 10 /3959]} } })
$box | returns all documents that are within the box
> db.place.find({ loc: { $geoWithin : { $box :[[0 , 0] , [100 , 100]] } } })
$polygon | returns all documents that are within the polygon
> db.place.find({loc :{ $geoWithin :{ $polygon : [[0,0] ,[3,6] ,[6,0]] } } })
$uniqueDocs:true | returns a document only once

Array:
$all | matches arrays that contain all elements given
> db.inventory.find({ tags: { $all: ["appliance", "school", "book"] } })
$elemMatch | matches multiple conditions in array
> db.collection.find({array: {$elemMatch: {value1: 1, value2: { $gt: 1 } } } })
$size | matches if the array is of specified size
> db.collection.find({ field: { $size: 1 } })

Other:
$hint | force to use an specific index

Update operators ($)

Field operators:
$inc | Increment a value by a specified amount
$rename | rename a field
$setOnInsert | set a value only if inserting
$set | set the value of a field on an existing document
$unset | remove the field from an existing document

Array operators:
$ | update the first element in an array that matches
> db.coll.update({ _id: 1, grades: 80 }, { $set: { "grades.$" : 82 } })
> db.coll.update({ _id: 4,"grades.stack": 85 }, {$set:{"grades.$.std":6}})

Ex: db.coll.update({ name: "joe" }, { $push: { scores: 89 } })
$addToSet | add element to array if it doesn't exist
$push | adds an item to an array
$pop | update the first element in an array that matches
$pull | remove items which match a query statement
$pullAll | remove multiple values from an array

Array modifiers:
$each | modify $push and $addToSet to add many
> db.coll.update({ name: "joe" }, { $push: { scores: { $each: [90,85] } } })
$slice | modify $push to limit size of updated array
> db.coll.update({ _id: 2 }, {$push: { grades: {$each:[80,78], $slice: -5} } })
$sort | modify $push to reorder documents in array
> db.coll.update({ _id: 2 }, {$push: { grades: 81 , $sort: { grades: 1 } } })

Bitwise:
$bit | performs a bitwise update of a field
> db.coll.update({field:NumberInt(1) }, {$bit: {field:{and:NumberInt(5)}}})

Isolation:
$isolated | isolates a write operation for multiple documents
> db.coll.update({field1:1,$isolated:1}, {$inc:{field2 : 1 }} , {multi: true})

Indexes

Indexing - db.collection.
ensureIndex({a:1}) | Cretes an ascending Index
dropIndex({'a':1}) | Removes a index from a collection
getIndexes() | Get indexes details of a collections
reIndex() | Rebuild all indexes on a collections
compact() | Defragment a collection and rebuild indexes

Properties- db.collection.<IndexOp>({...},{<option>})
expireAfterSeconds:300 | delete docs after set time
unique:true | only unique data
sparse:true | only documents with the index field

Options - db.collection.<IndexOp>{{...},{<option>}}
background:true | create index in the background
dropDups:true | Drop duplicates on unique index creation

Info - db.collection.
totalIndexSize() | get index size

Projection (db.)

$ | project the first element in an array that matches
$elemMatch | project only the first element match
$slice | limit number of elements projected from array

Durability of writes

w | Tells the driver to wait for the write to be acknowledged. Ensures no indexes are violated. Neverless the data can still be lost as it is not necessarilly already persisted to disc.

j | Stands the journal-mode. Tells the driver to wait until the journal has been commited to disk. Once this has happened it is quite sure that the write be persistent unless there are any disc-failures.

Combinations
w=0 j=0 | Fire and forget
w=1 j=0 | Waits for an acknowledgement that the write was recived and no indexes have been violated. Data can be lost.
w=1 j=1 | Most safe configuration by waiting for the write to the journal to be completed.
w=0 j=1 | Wait for the for the journal. Indexes could be violated

MongoDBSpain
www.mongodbspain.com
info@mongodbspain.com