MongoDB Cheat Sheet

David Brau | 2023

> Basics

>> Start / Stop MongoDB

```
$ sudo service mongod start          $ sudo service mongod stop
```

>> Mongosh

```
$ mongosh --port 27017
database> show databases ⇔ database> show dbs
database> show collections ⇔ database> show tables
database> use database
database> db.dropDatabase()  | Remove database
database> db.collection.drop()  | Remove collection from database
```

>> Main data types

>>> Object	>>> Decimal
>>> Array	>>> ObjectId
>>> String	>>> Date
>>> Boolean	>>> TimeStamp
>>> Integer	>>> Binary Data
>>> Double	>>> Null

>> Date

```
{myDate:new Date("YYYY-MM-DD")}
{myDate:ISODate("YYYY-MM-DDThh:mm:ssZ")}
```

>> Reference and Dot notation

>>> Fields and selectors may be used with or without quotation marks, whereas strings must always be enclosed in quotation marks.

>>> Dot notation for field reference:

```
Field.subfield.subsubfield...
Field[position].subfield.subsubfield[position]...
```

> Input - Output

>> Mongoimport

```
$ mongoimport -d database -c collection -f path/to/file.json
$ mongoimport --db database --collection collection --file path/to/file.json
```

>> Mongoexport

```
$ mongoexport --uri mongodb://localhost:27017/database
              --collection collection --out path/to/file.json
```

> Index

>> Get

```
database> db.collection.getIndexes()
```

>> Create / Delete

```
database> db.collection.createIndex({field1:1, field2:-1})
database> db.collection.createIndex({field1:1},{unique: true})
database> db.collection.dropIndex({field1:1})
```

> CRUD

>> Create: Insert

```
database> db.collection.insertOne({document})
database> db.collection.insertMany([documents])
```

>> Read: Find

```
database> db.collection.findOne()
database> db.collection.find({field:value},{projection}).resultProcessingMeth()
database> db.collection.find({field1:value1, field2:value2},      | Plain Search
                             {_id: false, field1: true})          | Projection
                            .option1().option2()                  | Options
database> db.collection.find({logicQuerySelector:[{field1:value1},
                                                   {field2:value2}]})
database> db.collection.find({field1:{querySelector1:value1},
                             {field2:{querySelector2:value2}})
```

>>> Comparison query selectors:

```
$eq:value
$ne:value
$gt:value
$gte:value
$lt:value
$lte:value
$in:[value1, value2]
$nin:[value1, value2]
```

>>> Logic query selectors:

```
$and:[]   $or:[]
$not:[]   $nor:[]
```

>>> Array query selectors:

```
$all:[]
$size:integer
$elemMatch:{queries}
```

>>> Element query selectors:

```
$exists:boolean
$type:bsontype
```

>>> Evaluation query selectors:

```
$mod:value        | Module
$text:options
$where:function(){...}
$regex:/regulareExpression/
```

>>> Result processing methods:

```
.countDocuments()  ⇔  .count()      .toArray()
.sort(0)  | Descendent              .sort(1) | Ascendent
.limit(n) | n first results         .skip(n) | skip n first results
.forEach()
.map(func(){...})  |Applies func to matches and returns [results]
```

>> Update: Update

```
database> db.collection.updateOne({search}, {updateOperator:
                                   {field1:change1, field2:change2}})
database> db.collection.updateMany({search}, {updateOperator:{field:change}})
database> db.collection.replaceOne({search}, {changes})
```

>>> Update operators:

```
$set:{field:newValue}                $unset:{field:1}
$inc:{field:incrementNumber}         $mul:{field:multiplyNumber}
$rename:{field1:newName1, field2: newName2}
$currentDate:{field1:true, field2:true}
$push:{array:{json}}
$pull:{array:{json}}                 $pullAll:{array:[val1, val2]}
```

>> Delete: Delete

```
database> db.collection.deleteOne({field1:value1, field2:value2})
database> db.collection.deleteMany()
```

> Aggregation pipeline

```
database> db.collection.aggregate([ {$pipelineOperator1:options1},
                                    {$pipelineOperator2:options2}])
```

>> Pipeline operators

```
$lookup:{ from:"database2", localField: "localfield"
          foreignField:"foreignField", as:"arrayDest"}}
          |Docs from db2 with foreign=local added to arrayDest
$unwind:$preArrayField
$match:{field:value}    | Like find()
$project:{postField2:{$projectOperator:value}}
$group:{_id:idField,    | _id:null → All in the same group
        postField2:{$accumulator:value}}
        postField3:{$accumulator:$preField3}}
$sort:{pre&postField: -1}          $sort:{pre&postField: 1}
$limit:numLimit
        |Match and sort don't use $Field
```

>> Accumulators:

$sum:	$first	$avg:	$max:
$multiply:	$last	$count	$min:

>> Project Operators:

$substr	$subtract:	$toUpper	$if

> Transactions

```
database> session = db.getMongo().startSession()
database> session.startTransaction()
database> db.collection.query1()      database> db.collection.query2()      ...
database> session.commitTransaction()
```

> Replication and Sharding

```
$ mongod --configsvr --replSet configName --port 27021 --dbpath path/to/folder
$ mongod --shardsvr --replSet shardName --port 27022 --dbpath path/to/folder
         -- oplogSize 50  | Max oplog size in MB
(27022) database> rs.initiate({_id : shardName, members: [
                  { _id:0, host:IP:27022, priority:100},
                  { _id:1, host:IP:27023, priority:50}]})
$ mongos --configdb configServerName/IP:27021 --port 27026
(27026) database> sh.addShard("shardServersName/IP:27022,IP:27023")
(27026) database> sh.status()
(27026) database> sh.enableSharding("database")
(27026 )database> sh.shardCollection("database.collection",
                                     {"partitionKey":"hashed"})
$ mongod  --shardsvr --replSet shardName --port 27027 --dbpath path/to/folder
          --oplogSize 50  | New server added after
```

> Regular expressions

/myword/	Like myword
/myword/i	Like myword no case sensitive
/myword.*/	Like myword + whatever
/^myword$/	Starts and ends with myword
/[A][B|C][D]/	ABD or ACD
/[A-Z|a-z|0-9]/	A letter or a number