Cheat sheet

MongoDB

MongoDB is a NoSQL, document-centric database. MongoDB is intended to be used small and enterprise grade projects that need a document-centric database solution that scales up on demand.

The following sections describe the command and tasks that are basic to using MongoDB.

The `$` symbol in the examples below represents the console prompt for a terminal window.

The `>` symbol in the examples below represents the Mongo shell prompt.

Accessing the Mongo shell

The following sections describe commands to access the Mongo shell from a terminal window.

mongo

```
mongo <optional URL> -u <username> -p <password>
```

Example:

Access a local instance

```
$ mongo -u root -p password

MongoDB shell version v4.4.14
connecting to: mongodb://127.0.0.1:27017/?
compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("2d62ffdb-a70d-40bf-
af6e-15de859ecc62") }
MongoDB server version: 5.0.8
```

Access a remote instance

```
$ mongo 192.168.86.93:27017/fruit -u root -p password  --
authenticationDatabase "admin"

MongoDB shell version v4.4.14
connecting to: mongodb://192.168.86.93:27017/fruit?
authSource=admin&compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" :
UUID("7c9b0ec4-51b3-4906-88bc-0b221cfcccf8") }
MongoDB server version: 4.4.2
Welcome to the MongoDB shell.
For interactive help, type "help".
.
.
.
```

Navigating around Mongo

The following sections show you how to list and select:

- A database running within a given Mongo server.
- Collections within a database.
- Roles in the given database.

Show all databases in a Mongo instance

```
show dbs
```

Example:

```
> show dbs

admin   0.000GB
config  0.000GB
fruit   0.000GB
local   0.000GB
```

Switch to a particular database

```
use <database_name>
```

Example:

```
> use fruit

switched to db fruit
```

show collections in database

```
show collections
```

Example:

```
> show collections

apples
oranges
countries
```

Show roles in a Mongo instance

```
show roles <database name [optional]>
```

Example:

```
> show roles

{
        "role" : "enableSharding",
        "db" : "test",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
{
        "role" : "dbOwner",
        "db" : "test",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
{
        "role" : "dbAdmin",
        "db" : "test",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
{
        "role" : "userAdmin",
        "db" : "test",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
{
        "role" : "read",
        "db" : "test",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
{
        "role" : "readWrite",
        "db" : "test",
        "isBuiltin" : true,
        "roles" : [ ],
        "inheritedRoles" : [ ]
}
```

Working with users

The following sections show you now to create a user, delete a user and list users in a given database.

Create user

```
db.runCommand(createUser <username> . . . . )
```

Example:

```
db.runCommand(
   {
     createUser: "cooluser",
     pwd: "newpassword",
     roles: [
       { role: "readWrite", db: "fruit" }
     ]
   })

{ "ok" : 1 }
```

Show users

```
show users
```

Example:

```
> show users

{
    "_id" : "fruit.cooluser",
    "userId" : UUID("78e368a7-dff0-45be-8633-f3d63802ca93"),
    "user" : "cooluser",
    "db" : "fruit",
    "roles" : [
            {
                "role" : "readWrite",
                "db" : "fruit"
            }
    ],
    "mechanisms" : [
            "SCRAM-SHA-1",
            "SCRAM-SHA-256"
    ]
}
```

Delete user

```
db.dropUser("<user_name>")
```

Example:

```
> use fruit
switched to db fruit

> db.dropUser("cooluser")
true
```

Or

```
> use fruit
switched to db fruit

> db.runCommand( { dropUser: "cooluser" } )
{ "ok" : 1 }
```

Working with a collection

A collection is an array of documents that exist within a given database. You can think of a document as a NoSQL record.

The following sections show you how to create and delete a collection in a given database as well as how to list collections in a given database.

Create a collection

```
db.createCollection(<collection_name>)
```

Example:

```
> db.createCollection("pears")

{ "ok" : 1 }
```

Show all collections

```
show collections
```

Example:

```
> show collections

apples
oranges
pears
```

Delete a collection

```
db.<collection_name>.drop()
```

Example:

```
> db.pears.drop()
true
```

Working with documents

The following sections show you how perform basic queries against a given collection in a database.

Show all documents in a collection

```
db.<collection_name>.find()
```

```
db.["<collection_name>"].find()
```

Example:

```
> db["apples"].find()

 "_id" : ObjectId("627d9053f7e6008a00844a81"), "type" : "granny smith",
"price" : 2.99, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627d9053f7e6008a00844a82"), "type" : "golden
delicious", "price" : 0.99, "countryOfOrigin" : "Ireland" }
{ "_id" : ObjectId("627d9053f7e6008a00844a83"), "type" : "gala", "price" :
1.29, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627d9053f7e6008a00844a84"), "type" : "empire",
"price" : 1.59, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627d9053f7e6008a00844a85"), "type" : "delicious",
"price" : 1.59, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627d9053f7e6008a00844a86"), "type" : "macintosh",
"price" : 0.99, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627d9053f7e6008a00844a87"), "type" : "fuji", "price" :
0.99, "countryOfOrigin" : "Chile" }
{ "_id" : ObjectId("627d9053f7e6008a00844a88"), "type" : "golden
delicious", "price" : 0.99, "countryOfOrigin" : "Mexico" }
{ "_id" : ObjectId("627d9053f7e6008a00844a89"), "type" : "crab", "price" :
0.09, "countryOfOrigin" : "Canada" }
```

Or

```
> db.apples.find()

 "_id" : ObjectId("627d9053f7e6008a00844a81"), "type" : "granny smith",
"price" : 2.99, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627d9053f7e6008a00844a82"), "type" : "golden
delicious", "price" : 0.99, "countryOfOrigin" : "Ireland" }
{ "_id" : ObjectId("627d9053f7e6008a00844a83"), "type" : "gala", "price" :
1.29, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627d9053f7e6008a00844a84"), "type" : "empire",
"price" : 1.59, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627d9053f7e6008a00844a85"), "type" : "delicious",
"price" : 1.59, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627d9053f7e6008a00844a86"), "type" : "macintosh",
"price" : 0.99, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627d9053f7e6008a00844a87"), "type" : "fuji", "price" :
0.99, "countryOfOrigin" : "Chile" }
{ "_id" : ObjectId("627d9053f7e6008a00844a88"), "type" : "golden
delicious", "price" : 0.99, "countryOfOrigin" : "Mexico" }
{ "_id" : ObjectId("627d9053f7e6008a00844a89"), "type" : "crab", "price" :
0.09, "countryOfOrigin" : "Canada" }
```

Sort all documents in a collection

```
db.apples.find().sort({<field_name_1> : <sort_order>, <field_name_2> :
<sort_order>, <field_name_n> : <sort_order>,})
```

Where `<sort_order>` is 1, the documents will be listed in ascending order; -1 indicates descending order.

Example:

The following example shows how to sort all documents in ascending order first sorting on `countryOfOrigin` and then sorting on `price`:

```
> db.apples.find().sort({countryOfOrigin : 1, price: 1})
{ "_id" : ObjectId("627e79917107db0de3aeb497"), "type" : "crab", "price" :
0.09, "countryOfOrigin" : "Canada" }
{ "_id" : ObjectId("627e79917107db0de3aeb495"), "type" : "fuji", "price" :
0.99, "countryOfOrigin" : "Chile" }
{ "_id" : ObjectId("627e79917107db0de3aeb490"), "type" : "golden
delicious", "price" : 0.99, "countryOfOrigin" : "Ireland" }
{ "_id" : ObjectId("627e79917107db0de3aeb496"), "type" : "golden
delicious", "price" : 0.99, "countryOfOrigin" : "Mexico" }
{ "_id" : ObjectId("627e79917107db0de3aeb494"), "type" : "macintosh",
"price" : 0.99, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627e79917107db0de3aeb491"), "type" : "gala", "price" :
1.29, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627e79917107db0de3aeb492"), "type" : "empire",
"price" : 1.59, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627e79917107db0de3aeb493"), "type" : "delicious",
"price" : 1.59, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627e79917107db0de3aeb48f"), "type" : "granny smith",
"price" : 2.99, "countryOfOrigin" : "USA" }
```

Find one of any document in a collection

```
db.<collection_name>.findOne()
```

Example:

```
> db.apples.findOne()

{
    "_id" : ObjectId("627d9053f7e6008a00844a81"),
    "type" : "granny smith",
    "price" : 2.99,
    "countryOfOrigin" : "USA"
}
```

Find a document in a collection according to the Mongo _id

When looking up a document by unique identifier, the unique `_id` needs to be cast to an ObjectId.

```
db.<collection_name>.findOne({ _id : ObjectId (“<object_identifier>) })
```

Example:

```
> db.apples.findOne({ _id : ObjectId ("627e79917107db0de3aeb496") })
{
        "_id" : ObjectId("627e79917107db0de3aeb496"),
        "type" : "golden delicious",
        "price" : 0.99,
        "countryOfOrigin" : "Mexico"
}
```

Find one of any document in a collection according to query criteria

```
db.<collection_name>.findOne({<query:criteria>})
```

Example:

```
>  db.apples.findOne({price: 0.99})
{
    "_id" : ObjectId("627e79917107db0de3aeb490"),
    "type" : "golden delicious",
    "price" : 0.99,
    "countryOfOrigin" : "Ireland"
}
```

Find all documents according to query criteria

```
db.<collection_name>.find({ <search_field>: <field_value>, <search_field>:
<field_value>})
```

Example:

Find all documents that have a `price` that is equal to `0.99`:

```
> db.apples.find({price: 0.99})
{ "_id" : ObjectId("627e79917107db0de3aeb490"), "type" : "golden
delicious", "price" : 0.99, "countryOfOrigin" : "Ireland" }
{ "_id" : ObjectId("627e79917107db0de3aeb494"), "type" : "macintosh",
"price" : 0.99, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627e79917107db0de3aeb495"), "type" : "fuji", "price" :
0.99, "countryOfOrigin" : "Chile" }
{ "_id" : ObjectId("627e79917107db0de3aeb496"), "type" : "golden
delicious", "price" : 0.99, "countryOfOrigin" : "Mexico" }
```

Find all documents that have a `price` that is equal to `0.99` and a `countryOfOrigin` of `USA`:

```
> db.apples.find({price: 0.99, countryOfOrigin: "USA" })
{ "_id" : ObjectId("627e79917107db0de3aeb494"), "type" : "macintosh",
"price" : 0.99, "countryOfOrigin" : "USA" }
```

Find all documents that have a `price` that is greater than `0.99`:

```
> db.apples.find({ price:{$gt: 0.99} })
{ "_id" : ObjectId("627e79917107db0de3aeb48f"), "type" : "granny smith",
"price" : 2.99, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627e79917107db0de3aeb491"), "type" : "gala", "price" :
1.29, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627e79917107db0de3aeb492"), "type" : "empire",
"price" : 1.59, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627e79917107db0de3aeb493"), "type" : "delicious",
"price" : 1.59, "countryOfOrigin" : "USA" }
```

Find all documents that have a `price` that is less than `1.29`:

```
> db.apples.find({ price:{$lt: 1.29} })
{ "_id" : ObjectId("627e79917107db0de3aeb490"), "type" : "golden
delicious", "price" : 0.99, "countryOfOrigin" : "Ireland" }
{ "_id" : ObjectId("627e79917107db0de3aeb494"), "type" : "macintosh",
"price" : 0.99, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627e79917107db0de3aeb495"), "type" : "fuji", "price" :
0.99, "countryOfOrigin" : "Chile" }
{ "_id" : ObjectId("627e79917107db0de3aeb496"), "type" : "golden
delicious", "price" : 0.99, "countryOfOrigin" : "Mexico" }
{ "_id" : ObjectId("627e79917107db0de3aeb497"), "type" : "crab", "price" :
0.09, "countryOfOrigin" : "Canada" }
```

Add a field to all documents in a collection

```
> db.apples.update({},{$set : { "genus":"malus" }},false,true)
WriteResult({ "nMatched" : 9, "nUpserted" : 0, "nModified" : 9 })

> db.apples.find( {price: 0.99})
{ "_id" : ObjectId("627e79917107db0de3aeb490"), "type" : "golden
delicious", "price" : 0.99, "countryOfOrigin" : "Ireland", "genus" :
"malus" }
{ "_id" : ObjectId("627e79917107db0de3aeb494"), "type" : "macintosh",
"price" : 0.99, "countryOfOrigin" : "USA", "genus" : "malus" }
{ "_id" : ObjectId("627e79917107db0de3aeb495"), "type" : "fuji", "price" :
0.99, "countryOfOrigin" : "Chile", "genus" : "malus" }
{ "_id" : ObjectId("627e79917107db0de3aeb496"), "type" : "golden
delicious", "price" : 0.99, "countryOfOrigin" : "Mexico", "genus" :
"malus" }
```

Adding, updating and removing documents

The following sections describe how to update and remove existing documents in a given database.

Adding a document to a collection

```
db.<collection_name>.insert({ <document_declaration_in_json>})
```

Example:

The following example inserts a new document into the `apples` collection

```
> db.apples.insert({ type : "honeycrisp", "price" : 1.79, countryOfOrigin:
"New Zealand" })
WriteResult({ "nInserted" : 1 })
```

Updating a document in a collection

```
> db.apples.update({ "_id" : ObjectId("627e9200be5baf249878171d") },{ $set:
{ "price":1.09} })
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.apples.find({ "_id" : ObjectId("627e9200be5baf249878171d") } )
{ "_id" : ObjectId("627e9200be5baf249878171d"), "type" : "golden
delicious", "price" : 1.09, "countryOfOrigin" : "Mexico" }
```

Removing a document in a collection

```
db.<collection_name>.remove(<deletion_criteria>)
```

Example:

```
> db.apples.remove( { "_id" : ObjectId("627e9200be5baf249878171d") })
WriteResult({ "nRemoved" : 1 })
```

Removing many documents in a collection

```
db.<collection_name>.remove(<deletion_criteria>)
```

Example:

```
> db.apples.remove({ "countryOfOrigin" : "USA" })
WriteResult({ "nRemoved" : 5 })
```

Setting query results to a variable

You can use a variable to store the results of a query. Once the result is stored, the variable can be used to modify data within the variable. A variable provides a shorthand for working with data.

Set the result set of a query to a variable

```
var <variable_name> = <mongo_statement>
```

Example:

```
> var crabapple = db.apples.findOne({ type : "crab" })

> crabapple
{
        "_id" : ObjectId("627ea4c10d8bd2fbf249eae7"),
        "type" : "crab",
        "price" : 0.09,
        "countryOfOrigin" : "Canada"
}
```

Set a cursor to a variable

A cursor is a pointer to documents in a database:

```
var <variable_name> = <mongo_statement>
```

```
> var usa = db.apples.find({ countryOfOrigin : "USA" })
Display the contents of the variable ùsà:
```

Example:

```
> var usa = db.apples.find({ countryOfOrigin : "USA" })

Display the contents of the variable ùsà:

> usa
{ "_id" : ObjectId("627ea4c10d8bd2fbf249eadf"), "type" : "granny smith",
"price" : 2.99, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627ea4c10d8bd2fbf249eae1"), "type" : "gala", "price" :
1.29, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627ea4c10d8bd2fbf249eae2"), "type" : "empire",
"price" : 1.59, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627ea4c10d8bd2fbf249eae3"), "type" : "delicious",
"price" : 1.59, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627ea4c10d8bd2fbf249eae4"), "type" : "macintosh",
"price" : 0.99, "countryOfOrigin" : "USA" }
>
```

Manipulating data using a variable

You can make changes in data assigned to a variable and then persist that data by saving the variable.

Example:

```
> var crabapple = db.apples.findOne({ type : "crab" })

> crabapple.price = 0.29
0.29

>  db.apples.save(crabapple)
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.apples.findOne({ type : "crab" })
{
        "_id" : ObjectId("627ea4c10d8bd2fbf249eae7"),
        "type" : "crab",
        "price" : 0.29,
        "countryOfOrigin" : "Canada"
}
```

Working with indexes

An index is a digest of the data in a Mongo database. If an index does not exist in the database, then Mongo scans every document in the given collection in order to select those documents that match the query statement. The efficiency of using an index is apparent.

A database can have any number of indexes. Indexes are created and removed according to a particular property in the documents.

Get indexes

```
db.<collection_name>.getIndexes()
```

Example:

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

Add an index

```
db.<collection_name>.createIndex(<field_name>, <sort_order>)
```

When `<sort_order>` is 1, order is ascending. A `<sort_order>` of -1 is descending order.

Example:

```
> db.apples.createIndex( { countryOfOrigin: 1 } )

{
    "numIndexesBefore" : 1,
    "numIndexesAfter" : 2,
    "createdCollectionAutomatically" : false,
    "ok" : 1
}
```

Drop an index

Drop an index according to `index_name`:

```
> db.apples.dropIndex ("countryOfOrigin_1")
{ "nIndexesWas" : 2, "ok" : 1 }
```

Drop an index according to the field name upon which index as created:

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

Aggregation

Aggregation is the capability to combine two collections together using a single query to create a single result set.

The following query combines the `apples` and `countries` collections together. The `apple` collection is combined with the `countries` collection by using the `apple.countryOfOrigin` field and the `countries.country` field as the common join fields.

The statement `{$match:{ countryOfOrigin: "Mexico" }}` indicates the the query will return only those documents in which the field `apples.countryOfOrigin` equals `Mexico`:

```
> db.apples.aggregate([
    {$match:{ countryOfOrigin: "Mexico" }},
    { $lookup:
        {
           from: "countries",
           localField: "countryOfOrigin",
           foreignField: "country",
           as: "regional_info"
        }
    }
]).pretty()

{
    "_id" : ObjectId("627efc53a96c699c93564740"),
    "type" : "golden delicious",
    "price" : 0.99,
    "countryOfOrigin" : "Mexico",
    "regional_info" : [
        {
            "_id" : ObjectId("627efc54bb142679f4604979"),
            "country" : "Mexico",
            "continent" : "North America"
        }
    ]
}
```

Dangerous tasks

The following sections describe tasks that need to be executed with care.

Dropping an entire collection

```
db.<collection_name>.drop()
```

Example:

```
> db.apples.drop()
```

Dropping an entire database

```
<database>.dropDatabase()
```

Example:

```
> use fruit

> db.dropDatabase()
{ "ok" : 1 }
```

Unintended document modification when a document is updated

Don't do this:

The following replaces the entire document, removing all fields except `price`:

```
db.apples.update({ type : "granny smith" }, { price : 2.49 })
```

Thus, when a document with the field `type` is found all fields in that document except the field `price` will be deleted from the identified document.

Example:

```
> db.apples.update({ type : "granny smith" }, { price : 2.49 })

> db.apples.find()

{ "_id" : ObjectId("627ec6019145af3d0eca46b9"), "price" : 2.49 }
{ "_id" : ObjectId("627ec6019145af3d0eca46ba"), "type" : "golden
delicious", "price" : 0.99, "countryOfOrigin" : "Ireland" }
{ "_id" : ObjectId("627ec6019145af3d0eca46bb"), "type" : "gala", "price" :
1.29, "countryOfOrigin" : "USA" }
```

Do this:

Using the `$set` keyword in the following only update the `price` field in the document:

```
db.apples.update({ type : "granny smith" }, {$set : { price : 2.49 }})
```

```
> db.apples.update({ type : "granny smith" }, {$set : { price : 2.49 }})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.apples.find()
{ "_id" : ObjectId("627ec9c81a9f54fbff86f145"), "type" : "granny smith",
"price" : 2.49, "countryOfOrigin" : "USA" }
{ "_id" : ObjectId("627ec9c81a9f54fbff86f146"), "type" : "golden
delicious", "price" : 0.99, "countryOfOrigin" : "Ireland" }
{ "_id" : ObjectId("627ec9c81a9f54fbff86f147"), "type" : "gala", "price" :
1.29, "countryOfOrigin" : "USA" }
```

The collection data used in the examples

Apples

```
[
    { "type": "granny smith", "price": 2.99, "countryOfOrigin": "USA" },
    { "type": "golden delicious", "price": 0.99, "countryOfOrigin":
"Ireland" },
    { "type": "gala", "price": 1.29, "countryOfOrigin": "USA" },
    { "type": "empire", "price": 1.59, "countryOfOrigin": "USA" },
    { "type": "delicious", "price": 1.59, "countryOfOrigin": "USA" },
    { "type": "macintosh", "price": 0.99, "countryOfOrigin": "USA" },
    { "type": "fuji", "price": 0.99, "countryOfOrigin": "Chile" },
    { "type": "golden delicious", "price": 0.99, "countryOfOrigin":
"Mexico" },
    { "type": "crab", "price": 0.09, "countryOfOrigin": "Canada"
    }
]
```

Oranges

```
[
    { "type": "navel", "price": 2.99, "countryOfOrigin": "USA" },
    { "type": "seville", "price": 0.99, "countryOfOrigin": "Spain" },
    { "type": "blood", "price": 1.69, "countryOfOrigin": "USA" },
    { "type": "mandarin", "price": 1.59, "countryOfOrigin": "USA" },
    { "type": "jaffa", "price": 1.59, "countryOfOrigin": "Israel" },
    { "type": "lima", "price": 0.99, "countryOfOrigin": "Brazil" },
    { "type": "cara cara", "price": 0.99, "countryOfOrigin": "Venezuela" },
    { "type": "cara cara", "price": 1.29, "countryOfOrigin": "USA" },
    { "type": "cherry", "price": 1.09, "countryOfOrigin": "Japan" },
    { "type": "queen", "price": 1.09, "countryOfOrigin": "South Africa" }
]
```

Countries

```
[
    { "country": "USA", "continent": "North America" },
    { "country": "Spain", "continent": "Europe" },
    { "country": "Brazil", "continent": "South America" },
    { "country": "Venezuela", "continent": "South America" },
    { "country": "Japan", "continent": "Asia" },
    { "country": "South Africa", "continent": "Africa" },
    { "country": "Chile", "continent": "South America" },
    { "country": "Ireland", "continent": "Europe" },
    { "country": "Mexico", "continent": "North America" },
    { "country": "Canada", "continent": "North America" },
    { "country": "China", "continent": "Asia" }
]
```