**NoSQL v SQL**

**Consistency, Availbility, Partitionability (CAP)**

The CAP Theorem says you can have any two of those three.

**SQL: Consistency & Availability**

SQL doesn't scale well.

**NoSQL: Availability & Partitionability**

Scales well, however, it is eventual consistent.

FYI: NoSQL can also be thought of a \*\*schemaless\*\*. You do not define any schema. There is no defining that a table is this way or that way. There is no defining that tables are connected in different ways. With SQL, the schema is imposed by the RDBMS. With schemaless nosql, the schema needs to be imposed by the developer; the developer needs to make it all work.

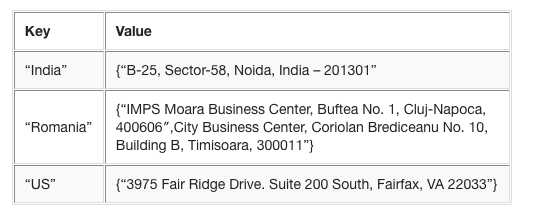
**Types**

**Key-Value**

Items are stored as key-value pairs.

KV dbs usually lack the query capabilities of other nosql dbs.

You can use KV dbs for memcache, tracking transient data, large object storage.



[source: [www.3pillarglobal.com](http://www.3pillarglobal.com/insights/exploring-the-different-types-of-nosql-databases)](http://www.3pillarglobal.com](http://www.3pillarglobal.com/insights/exploring-the-different-types-of-nosql-databases))

Examples: Redis, Riak

**Document**

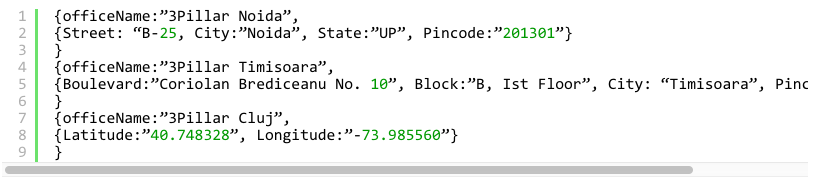
Like a key-value store but the value has greater structure, such as JSON or BSON (binary encoding of JSON).

\*\*Denormalizing data is common with document dbs\*\*: data that is frequently queried together should be stored together.

Example: a blog post would have the blog post, comments, and likes all stored together.

Document dbs have indexing and querying capability (you can look data up by more than just the key).

You can use document dbs for web apps, data with variable schema, JSON, denormalized data (structures embedded in structures).

 Examples: Mongodb, couchdb

**Column**

Designed for super huge scale; run on clusters.

More complicated than key-value and document dbs.

Data is stored in columns, rather than rows.

Up to this point, we have been thinking about data storage in terms of rows. RDBMS stores data in rows. Key-value and Document nosql store data in rows (for this key, here is the value).

With column nosql, data is stored in columns. This has performance gains.

"Relational databases store all the data in a particular table’s rows together on-disk, making retrieval of a particular row fast. Column-family databases generally serialize all the values of a particular column together on-disk, which makes retrieval of a large amount of a specific attribute fast. This approach lends itself well to aggregate queries and analytics scenarios where you might run range queries over a specific field." [source](<http://www.jamesserra.com/archive/2015/04/types-of-nosql-databases/>)

Examples: Big Table (google), Cassandra (facebook), HBase

Use this if you are creating the next facebook social network, the next google search engine, big science, stock market analysis, writing vast amounts of data quickly and being able to query it.

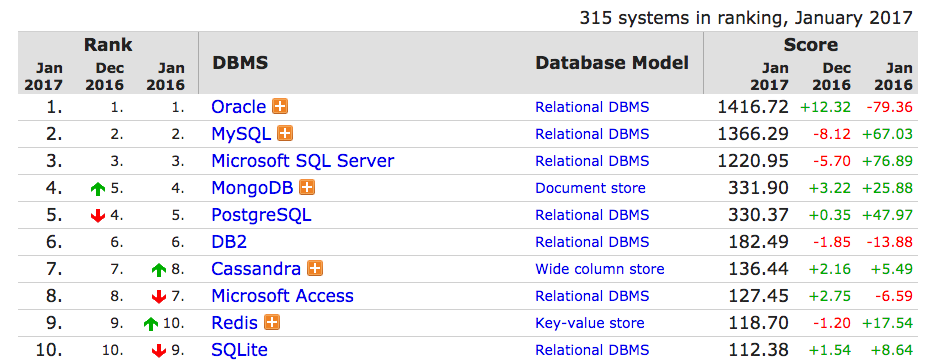
**Graph**

Great for interconnected data; relationships: who knows who.

Stores information about networks such as social relations.

**MongoDB**

The [world's leading NoSQL database](<http://db-engines.com/en/ranking>)



**Mongo --> Database --> Collection --> Document**

A nosql database has collections which have documents.

A rdbms database has tables which have records.

**Document**

A collection of key-value pairs.

The value is a dynamic schema - it can change from document to document. Documents can have different fields and structure; fields with the same name in different documents can hold different types of data.

|  |  |
| --- | --- |
| **RDBMS** | **MongoDB** |
| database | database |
| table | collection |
| row | document |
| field | field |
| join | embed |
| primary key | default key \_id provided by mongo |

**advantages over RDBMS**

1. documents can vary

2. store objects as they are in your program

3. you don't have to do join queries

4. scalability performance

**data modeling**

Denormalize your data (combine data that will be used together). Sometimes you hear this referred to as "join on write, not read."

Having duplicates of stored data can be fine!

**install mongo**

[installing mongodb](<https://www.mongodb.com/download-center#community>)

Path - Set a path environment variable pointing to the ```mongod``` program. (D:\Mo\MongoDB\Server\4.2)

data folder - Create a folder for data.

Make sure service is running.

**run mongo**

```

mongod

```

use terminal with mongo

```

mongo

```

stop

```

ctrl+c

```

```

exit

```

**db commands**

**create**

```

use <db name>

```

The above will use the database of the provided name if it exists, and create it if it doesn't

**use**

```

use <db name>

```

example

```

use temp

```

The above will use the database of the provided name if it exists, and create it if it doesn't

**see current db**

```

db

```

**see all db**

```

show dbs

```

You need to have at least one document in a db for it to be seen.

**insert document**

```

db.<collection name>.insert({"name":"McLeod"})

```

example

```

db.dogs.insert({"name":"toby"})

```

**view documents**

```

db.<collection name>.find()

```

example

```

db.cats.insert({"firstname":"coco"})

```

```

db.cats.find().pretty()

```

**view collections**

```

show collections

```

**drop db**

```

db.dropDatabase()

```

**collection commands**

**create implicitly**

```

db.<collection name>.insert({"name":"McLeod"})

```

**create explicitly**

```

db.createCollection(<name>, {<optional options>})

```

**optional options**

| option | type | description |

| --- | --- | --- |

| capped | bool | caps the size |

| size | number | sets size of cap in bytes |

| max | bool | maximum number of documents allowed in capped collection |

[other options including validation](https://docs.mongodb.com/manual/reference/method/db.createCollection/)

examples

```

db.createCollection("customers")

```

```

db.createCollection("crs",{capped:true, size:65536,max:1000000})

```

**view collections**

```

show collections

```

**drop**

```

db.<collection name>.drop()

```

**document commands**

**insert**

```

db.<collection name>.insert({document})

```

**insert multiple**

```

db.<collection name>.insert(< [{document}, {document}, ..., {document}] >)

```

pass in an array of documents

example

```

use playroom

```

```

db

```

```

show dbs

```

```

db.crayons.insert([

{

"hex": "#EFDECD",

"name": "Almond",

"rgb": "(239, 222, 205)"

},

{

"hex": "#CD9575",

"name": "Antique Brass",

"rgb": "(205, 149, 117)"

},

{

"hex": "#FDD9B5",

"name": "Apricot",

"rgb": "(253, 217, 181)"

},

{

"hex": "#78DBE2",

"name": "Aquamarine",

"rgb": "(120, 219, 226)"

},

{

"hex": "#87A96B",

"name": "Asparagus",

"rgb": "(135, 169, 107)"

},

{

"hex": "#FFA474",

"name": "Atomic Tangerine",

"rgb": "(255, 164, 116)"

},

{

"hex": "#FAE7B5",

"name": "Banana Mania",

"rgb": "(250, 231, 181)"

}

])

```

[source of crayon json] <https://gist.githubusercontent.com/jjdelc/1868136/raw/c9160b1e60bd8c10c03dbd1a61b704a8e977c46b/crayola.json>

```

show collections

```

```

db.crayons.find()

```

```

db.crayons.drop()

```

```

db.dropDatabase()

```

**find (aka, query)**

**setup**

```

use store

db

show dbs

db.customers.insert([{"role":"double-zero","name": "Bond","age": 32},{"role":"citizen","name": "Moneypenny","age":32},{"role":"citizen","name": "Q","age":67},{"role":"citizen","name": "M","age":57},{"role":"citizen","name": "Dr. No","age":52}])

```

**find**

```

db.<collection name>.find()

db.customers.find()

```

**find one**

```

db.<collection name>.findOne()

db.customers.findOne()

```

**find specific**

```

db.customers.find({"name":"Bond"})

db.customers.find({name:"Bond"})

```

You can do it either way: ```"name" or name```. JSON specification is to enclose name (object name-value pair) in double qoutes

**and**

```

db.customers.find({$and: [{name:"Bond"}, {age:32}]})

db.customers.find({$and: [{name:"Bond"}, {age:{$lt:20}}]})

db.customers.find({$and: [{name:"Bond"}, {age:{$gt:20}}]})

```

**or**

```

db.customers.find({$or: [{name:"Bond"}, {age:67}]})

db.customers.find({$or: [{name:"Bond"}, {age:{$lt:20}}]})

db.customers.find({$or: [{name:"Bond"}, {age:{$gt:32}}]})

```

**and or**

```

db.customers.find({role:"citizen"})

db.customers.find({age:52})

db.customers.find({$and: [{role:"citizen"}, {age:52}]})

db.customers.find({$or: [{role:"citizen"}, {age:52}]})

db.customers.find({$or: [{role:"citizen"}, {age:52}, {name:"Bond"}]})

```

```

db.customers.find({$or:[

{ $and : [ { role : "citizen" }, { age : 32 } ] },

{ $and : [ { role : "citizen" }, { age : 67 } ] }

]})

```

**regex**

```

db.customers.find({name: {$regex: '^M'}})

```

[regex cheatsheet](regex.pdf)

**pretty**

```

db.<collection name>.find().pretty()

```

pretty prints the results

**operators**

| operator | syntax | example |

| --- | --- | --- |

| equality | {key:value} | db.customers.find({"name":"Bond"}).pretty() |

| less than | {key:{$lt:value}} | db.customers.find({"age":{$lt:20}}).pretty() |

| less than equals | {key:{$lte:value}} | db.customers.find({"age":{$lte:20}}).pretty() |

| greater than | {key:{$gt:value}} | db.customers.find({"age":{$gt:20}}).pretty() |

| greater than equals | {key:{$gte:value}} | db.customers.find({"age":{$gte:20}}).pretty() |

| not equals | {key:{$ne:value}} | db.customers.find({"age":{$ne:20}}).pretty() |

**JSON reminder**

JavaScript Object Notation (JSON) is a text format for the serialization of structured data.

It is derived from the object literals of JavaScript.

JSON can represent four primitive types (strings, numbers, booleans, and null) and two structured types (objects and arrays)

**Primitive JSON**

Here are four small JSON texts containing only values:

```

"Hello world!"

42

true

null

```

**Object JSON**

An object structure is represented as a pair of curly brackets surrounding zero or more \*\*name-value\*\* pairs (or members).

An object is an unordered collection of zero or more \*\*name:value\*\* pairs

A \*\*name\*\* is a string

A \*\*value\*\* is a string, number, boolean, null, object, or array.

Declare properties using \*\*name:value\*\* pairings separated by commas

Enclose names in curly braces

There is no trailing comma

This is a JSON object:

```

{

"Image": {

"Width": 800,

"Height": 600,

"Title": "View from 15th Floor",

"Thumbnail": {

"Url": "http://www.example.com/image/481989943",

"Height": 125,

"Width": 100

},

"Animated" : false,

"IDs": [116, 943, 234, 38793]

}

}

```

**Array JSON**

An array structure is represented as square brackets surrounding zero or more values (or elements).

Elements are separated by commas.

A value must be an

```

object

array

number

string

three literal names

true

false

null

```

This is a JSON array containing two objects:

[

{

"precision": "zip",

"Latitude": 37.7668,

"Longitude": -122.3959,

"Address": "",

"City": "SAN FRANCISCO",

"State": "CA",

"Zip": "94107",

"Country": "US"

},

{

"precision": "zip",

"Latitude": 37.371991,

"Longitude": -122.026020,

"Address": "",

"City": "SUNNYVALE",

"State": "CA",

"Zip": "94085",

"Country": "US"

}

]

**Number**

The representation of numbers is similar to that used in most programming languages. A number is represented in base 10 using decimal digits. It contains an integer component that may be prefixed with an optional minus sign, which may be followed by a fraction part and/or an exponent part. Leading zeros are not allowed. A fraction part is a decimal point followed by one or more digits.

**String**

The representation of strings is similar to conventions used in the C family of programming languages. A string begins and ends with \*\*double quotation marks\*\*.

source: The Internet Engineering Task Force (IETF)

**update & save**

update will update a record

save will overwrite a record

**update**

```

db.<collection name>.update(<selection criteria>, <update data>, <optional options>)

```

example

```

db.customers.find()

```

gives this data

```

{ "\_id" : ObjectId("5891221756867ebff44cc885"), "role" : "double-zero", "name" : "Bond", "age" : 32 }

{ "\_id" : ObjectId("5891221756867ebff44cc886"), "role" : "citizen", "name" : "Moneypenny", "age" : 32 }

{ "\_id" : ObjectId("5891221756867ebff44cc887"), "role" : "citizen", "name" : "Q", "age" : 67 }

{ "\_id" : ObjectId("5891221756867ebff44cc888"), "role" : "citizen", "name" : "M", "age" : 57 }

{ "\_id" : ObjectId("5891221756867ebff44cc889"), "role" : "citizen", "name" : "Dr. No", "age" : 52 }

```

update like this

```

db.customers.update({\_id:ObjectId("5891221756867ebff44cc886")},{$set:{role:"double-zero"}})

db.customers.update({name:"Moneypenny"},{$set:{role:"double-zero"}})

db.customers.update({name:"Moneypenny"},{$set:{role:"citizen", name: "Miss Moneypenny"}})

db.customers.update({age:{$gt:35}},{$set:{role:"double-zero"}})

db.customers.update({age:{$gt:35}},{$set:{role:"double-zero"}}, {multi:true})

[see options](https://docs.mongodb.com/manual/reference/method/db.collection.update/)

db.customers.update({},{$set:{role:"citizen"}}, {multi:true})

[see query documentation](https://docs.mongodb.com/manual/tutorial/query-documents/)

("5893888012acb8ada532a8e4"

**save**

```

db.customers.save({"role":"villain","name":"Jaws","age":43})

db.customers.save({"\_id":ObjectId("5891221756867ebff44cc889"),"role":"villain","name":"Goldfinger","age":77})

db.customers.save({"\_id":ObjectId("5893888012acb8ada532a8e4"),"role":"villain","name":"PussyGalore","age":31})

**remove document**

```

db.<collection name>.remove(<selection criteria>)

db.customers.remove({role:"double-zero"})

db.customers.remove({role:"villain"})

```

removes all it matches

**remove only 1**

```

db.customers.remove({role:"citizen"},1)

```

**remove**

```

db.customers.remove({role:"citizen"})

```

**put documents back**

```

db.customers.insert([{"role":"double-zero","name": "Bond","age": 32},{"role":"citizen","name": "Moneypenny","age":32},{"role":"citizen","name": "Q","age":67},{"role":"citizen","name": "M","age":57},{"role":"citizen","name": "Dr. No","age":52}])

```

**remove all**

```

db.customers.remove({})

```

**put documents back**

```

db.customers.insert([{"role":"double-zero","name": "Bond","age": 32},{"role":"citizen","name": "Moneypenny","age":32},{"role":"citizen","name": "Q","age":67},{"role":"citizen","name": "M","age":57},{"role":"citizen","name": "Dr. No","age":52}])

```

**projection**

Retrieving part of a document; only some of the fields.

```

db.<collection name>.find(<selection criteria>,<list of fields with toggle 0 or 1>)

db.customers.find({},{\_id:0,name:1,})

\_id is displayed by default; turn off with 0

db.customers.find({},{\_id:0,name:1,age:1})

db.customers.find({age:{$gt:32}},{\_id:0,name:1,age:1})

**limit**

**setup**

db.crayons.insert([

{

"hex": "#EFDECD",

"name": "Almond",

"rgb": "(239, 222, 205)"

},

{

"hex": "#CD9575",

"name": "Antique Brass",

"rgb": "(205, 149, 117)"

},

{

"hex": "#FDD9B5",

"name": "Apricot",

"rgb": "(253, 217, 181)"

},

{

"hex": "#78DBE2",

"name": "Aquamarine",

"rgb": "(120, 219, 226)"

},

{

"hex": "#87A96B",

"name": "Asparagus",

"rgb": "(135, 169, 107)"

},

{

"hex": "#FFA474",

"name": "Atomic Tangerine",

"rgb": "(255, 164, 116)"

},

{

"hex": "#FAE7B5",

"name": "Banana Mania",

"rgb": "(250, 231, 181)"

}

])

**limit**

db.<collection name>.find(<selection criteria>).limit(n)

db.crayons.find().limit(3)

db.customers.find({age:{$gt:32}},{\_id:0,name:1,age:1}).limit(2)

**sort**

Run \*\*setup\*\* below first

db.<collection name>.find().sort(<field to sort on>:<1 for ascend, -1 descend>)

db.oscars.find().limit(10)

db.oscars.find({},{\_id:0,year:1,title:1}).limit(10)

db.oscars.find({},{\_id:0,year:1,title:1}).limit(10).sort({title:1})

db.oscars.find({},{\_id:0,year:1,title:1}).sort({title:1}).limit(10)

db.oscars.find({},{\_id:0,year:1,title:1}).limit(10).sort({title:-1})

db.oscars.find({releaseYear:{$gt:1970}},{\_id:0,year:1,title:1}).limit(10).sort({title:1})

db.oscars.find({releaseYear:{$gt:1980}},{\_id:0,year:1,title:1})

```

setup

```

db.oscars.insert([

{ "year": "1927",

"title": "Wings",

"imdbId": "tt0018578",

"releaseDate": "1927-05-19T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1927,

"releaseMonth": 4,

"releaseDay": 19

},

{ "year": "1929",

"title": "The Broadway Melody",

"imdbId": "tt0019729",

"releaseDate": "1929-02-01T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1929,

"releaseMonth": 1,

"releaseDay": 1

},

{ "year": "1930",

"title": "All Quiet on the Western Front",

"imdbId": "tt0020629",

"releaseDate": "1930-04-21T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1930,

"releaseMonth": 3,

"releaseDay": 21

},

{ "year": "1931",

"title": "Cimarron",

"imdbId": "tt0021746",

"releaseDate": "1931-01-26T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1931,

"releaseMonth": 0,

"releaseDay": 26

},

{ "year": "1932",

"title": "Grand Hotel",

"imdbId": "tt0022958",

"releaseDate": "1932-04-12T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1932,

"releaseMonth": 3,

"releaseDay": 12

},

{ "year": "1933",

"title": "Cavalcade",

"imdbId": "tt0023876",

"releaseDate": "1933-01-05T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1933,

"releaseMonth": 0,

"releaseDay": 5

},

{ "year": "1934",

"title": "It Happened One Night",

"imdbId": "tt0025316",

"releaseDate": "1934-02-22T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1934,

"releaseMonth": 1,

"releaseDay": 22

},

{ "year": "1935",

"title": "Mutiny on the Bounty",

"imdbId": "tt0026752",

"releaseDate": "1935-11-08T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1935,

"releaseMonth": 10,

"releaseDay": 8

},

{ "year": "1936",

"title": "The Great Ziegfeld",

"imdbId": "tt0027698",

"releaseDate": "1936-03-22T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1936,

"releaseMonth": 2,

"releaseDay": 22

},

{ "year": "1937",

"title": "The Life of Emile Zola",

"imdbId": "tt0029146",

"releaseDate": "1937-08-11T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1937,

"releaseMonth": 7,

"releaseDay": 11

},

{ "year": "1938",

"title": "You Can't Take It with You",

"imdbId": "tt0030993",

"releaseDate": "1938-08-23T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1938,

"releaseMonth": 7,

"releaseDay": 23

},

{ "year": "1939",

"title": "Gone with the Wind",

"imdbId": "tt0031381",

"releaseDate": "1939-12-28T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1939,

"releaseMonth": 11,

"releaseDay": 28

},

{ "year": "1940",

"title": "Rebecca",

"imdbId": "tt0032976",

"releaseDate": "1940-03-27T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1940,

"releaseMonth": 2,

"releaseDay": 27

},

{ "year": "1941",

"title": "How Green Was My Valley",

"imdbId": "tt0033729",

"releaseDate": "1941-10-28T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1941,

"releaseMonth": 9,

"releaseDay": 28

},

{ "year": "1942",

"title": "Mrs. Miniver",

"imdbId": "tt0035093",

"releaseDate": "1942-07-22T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1942,

"releaseMonth": 6,

"releaseDay": 22

},

{ "year": "1943",

"title": "Casablanca",

"imdbId": "tt0034583",

"releaseDate": "1942-11-26T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1942,

"releaseMonth": 10,

"releaseDay": 26

},

{ "year": "1944",

"title": "Going My Way",

"imdbId": "tt0036872",

"releaseDate": "1944-08-16T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1944,

"releaseMonth": 7,

"releaseDay": 16

},

{ "year": "1945",

"title": "The Lost Weekend",

"imdbId": "tt0037884",

"releaseDate": "1945-11-29T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1945,

"releaseMonth": 10,

"releaseDay": 29

},

{ "year": "1946",

"title": "The Best Years of Our Lives",

"imdbId": "tt0036868",

"releaseDate": "1946-12-25T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1946,

"releaseMonth": 11,

"releaseDay": 25

},

{ "year": "1947",

"title": "Gentleman's Agreement",

"imdbId": "tt0039416",

"releaseDate": "1947-11-11T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1947,

"releaseMonth": 10,

"releaseDay": 11

},

{ "year": "1948",

"title": "Hamlet",

"imdbId": "tt0040416",

"releaseDate": "1948-10-27T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1948,

"releaseMonth": 9,

"releaseDay": 27

},

{ "year": "1949",

"title": "All the Kings Men",

"imdbId": "tt0041113",

"releaseDate": "1949-11-08T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1949,

"releaseMonth": 10,

"releaseDay": 8

},

{ "year": "1950",

"title": "All About Eve",

"imdbId": "tt0042192",

"releaseDate": "1950-10-13T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1950,

"releaseMonth": 9,

"releaseDay": 13

},

{ "year": "1951",

"title": "An American in Paris",

"imdbId": "tt0043278",

"releaseDate": "1951-10-04T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1951,

"releaseMonth": 9,

"releaseDay": 4

},

{ "year": "1952",

"title": "The Greatest Show on Earth",

"imdbId": "tt0044672",

"releaseDate": "1952-01-10T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1952,

"releaseMonth": 0,

"releaseDay": 10

},

{ "year": "1953",

"title": "From Here to Eternity",

"imdbId": "tt0045793",

"releaseDate": "1953-09-30T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1953,

"releaseMonth": 8,

"releaseDay": 30

},

{ "year": "1954",

"title": "On the Waterfront",

"imdbId": "tt0047296",

"releaseDate": "1954-07-28T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1954,

"releaseMonth": 6,

"releaseDay": 28

},

{ "year": "1955",

"title": "Marty",

"imdbId": "tt0048356",

"releaseDate": "1955-07-15T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1955,

"releaseMonth": 6,

"releaseDay": 15

},

{ "year": "1956",

"title": "Around the World in 80 Days",

"imdbId": "tt0048960",

"releaseDate": "1956-12-22T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1956,

"releaseMonth": 11,

"releaseDay": 22

},

{ "year": "1957",

"title": "The Bridge on the River Kwai",

"imdbId": "tt0050212",

"releaseDate": "1957-12-19T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1957,

"releaseMonth": 11,

"releaseDay": 19

},

{ "year": "1958",

"title": "Gigi",

"imdbId": "tt0051658",

"releaseDate": "1958-07-10T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1958,

"releaseMonth": 6,

"releaseDay": 10

},

{ "year": "1959",

"title": "Ben-Hur",

"imdbId": "tt0052618",

"releaseDate": "1959-11-18T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1959,

"releaseMonth": 10,

"releaseDay": 18

},

{ "year": "1960",

"title": "The Apartment",

"imdbId": "tt0053604",

"releaseDate": "1960-06-21T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1960,

"releaseMonth": 5,

"releaseDay": 21

},

{ "year": "1961",

"title": "West Side Story",

"imdbId": "tt0055614",

"releaseDate": "1961-12-13T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1961,

"releaseMonth": 11,

"releaseDay": 13

},

{ "year": "1962",

"title": "Lawrence of Arabia",

"imdbId": "tt0056172",

"releaseDate": "1962-12-21T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1962,

"releaseMonth": 11,

"releaseDay": 21

},

{ "year": "1963",

"title": "Tom Jones",

"imdbId": "tt0057590",

"releaseDate": "1963-10-24T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1963,

"releaseMonth": 9,

"releaseDay": 24

},

{ "year": "1964",

"title": "My Fair Lady",

"imdbId": "tt0058385",

"releaseDate": "1964-10-28T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1964,

"releaseMonth": 9,

"releaseDay": 28

},

{ "year": "1965",

"title": "The Sound of Music",

"imdbId": "tt0059742",

"releaseDate": "1965-03-10T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1965,

"releaseMonth": 2,

"releaseDay": 10

},

{ "year": "1966",

"title": "A Man for All Seasons",

"imdbId": "tt0060665",

"releaseDate": "1966-12-14T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1966,

"releaseMonth": 11,

"releaseDay": 14

},

{ "year": "1967",

"title": "In the Heat of the Night",

"imdbId": "tt0061811",

"releaseDate": "1967-08-23T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1967,

"releaseMonth": 7,

"releaseDay": 23

},

{ "year": "1968",

"title": "Oliver!",

"imdbId": "tt0063385",

"releaseDate": "1968-12-20T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1968,

"releaseMonth": 11,

"releaseDay": 20

},

{ "year": "1969",

"title": "Midnight Cowboy",

"imdbId": "tt0064665",

"releaseDate": "1969-05-25T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1969,

"releaseMonth": 4,

"releaseDay": 25

},

{ "year": "1970",

"title": "Patton",

"imdbId": "tt0066206",

"releaseDate": "1970-02-18T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1970,

"releaseMonth": 1,

"releaseDay": 18

},

{ "year": "1971",

"title": "The French Connection",

"imdbId": "tt0067116",

"releaseDate": "1971-10-07T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1971,

"releaseMonth": 9,

"releaseDay": 7

},

{ "year": "1972",

"title": "The Godfather",

"imdbId": "tt0068646",

"releaseDate": "1972-03-22T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1972,

"releaseMonth": 2,

"releaseDay": 22

},

{ "year": "1973",

"title": "The Sting",

"imdbId": "tt0070735",

"releaseDate": "1973-12-25T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1973,

"releaseMonth": 11,

"releaseDay": 25

},

{ "year": "1974",

"title": "The Godfather Part II",

"imdbId": "tt0071562",

"releaseDate": "1974-12-18T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1974,

"releaseMonth": 11,

"releaseDay": 18

},

{ "year": "1975",

"title": "One Flew over the Cuckoo's Nest",

"imdbId": "tt0073486",

"releaseDate": "1975-11-19T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1975,

"releaseMonth": 10,

"releaseDay": 19

},

{ "year": "1976",

"title": "Rocky",

"imdbId": "tt0075148",

"releaseDate": "1976-11-21T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1976,

"releaseMonth": 10,

"releaseDay": 21

},

{ "year": "1977",

"title": "Annie Hall",

"imdbId": "tt0075686",

"releaseDate": "1977-04-20T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1977,

"releaseMonth": 3,

"releaseDay": 20

},

{ "year": "1978",

"title": "The Deer Hunter",

"imdbId": "tt0077416",

"releaseDate": "1978-12-08T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1978,

"releaseMonth": 11,

"releaseDay": 8

},

{ "year": "1979",

"title": "Kramer vs. Kramer",

"imdbId": "tt0079417",

"releaseDate": "1979-12-19T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1979,

"releaseMonth": 11,

"releaseDay": 19

},

{ "year": "1980",

"title": "Ordinary People",

"imdbId": "tt0081283",

"releaseDate": "1980-09-26T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1980,

"releaseMonth": 8,

"releaseDay": 26

},

{ "year": "1981",

"title": "Chariots of Fire",

"imdbId": "tt0082158",

"releaseDate": "1981-10-09T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1981,

"releaseMonth": 9,

"releaseDay": 9

},

{ "year": "1982",

"title": "Gandhi",

"imdbId": "tt0083987",

"releaseDate": "1982-12-07T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1982,

"releaseMonth": 11,

"releaseDay": 7

},

{ "year": "1983",

"title": "Terms of Endearment",

"imdbId": "tt0086425",

"releaseDate": "1983-11-20T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1983,

"releaseMonth": 10,

"releaseDay": 20

},

{ "year": "1984",

"title": "Amadeus",

"imdbId": "tt0086879",

"releaseDate": "1984-09-06T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1984,

"releaseMonth": 8,

"releaseDay": 6

},

{ "year": "1985",

"title": "Out of Africa",

"imdbId": "tt0089755",

"releaseDate": "1985-12-10T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1985,

"releaseMonth": 11,

"releaseDay": 10

},

{ "year": "1986",

"title": "Platoon",

"imdbId": "tt0091763",

"releaseDate": "1986-12-19T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1986,

"releaseMonth": 11,

"releaseDay": 19

},

{ "year": "1987",

"title": "The Last Emperor",

"imdbId": "tt0093389",

"releaseDate": "1987-11-19T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1987,

"releaseMonth": 10,

"releaseDay": 19

},

{ "year": "1988",

"title": "Rain Man",

"imdbId": "tt0095953",

"releaseDate": "1988-12-14T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1988,

"releaseMonth": 11,

"releaseDay": 14

},

{ "year": "1989",

"title": "Driving Miss Daisy",

"imdbId": "tt0097239",

"releaseDate": "1989-12-15T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1989,

"releaseMonth": 11,

"releaseDay": 15

},

{ "year": "1990",

"title": "Dances With Wolves",

"imdbId": "tt0099348",

"releaseDate": "1990-10-19T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1990,

"releaseMonth": 9,

"releaseDay": 19

},

{ "year": "1991",

"title": "The Silence of the Lambs",

"imdbId": "tt0102926",

"releaseDate": "1991-01-30T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1991,

"releaseMonth": 0,

"releaseDay": 30

},

{ "year": "1992",

"title": "Unforgiven",

"imdbId": "tt0105695",

"releaseDate": "1992-08-03T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1992,

"releaseMonth": 7,

"releaseDay": 3

},

{ "year": "1993",

"title": "Schindler's List",

"imdbId": "tt0108052",

"releaseDate": "1993-11-30T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1993,

"releaseMonth": 10,

"releaseDay": 30

},

{ "year": "1994",

"title": "Forrest Gump",

"imdbId": "tt0109830",

"releaseDate": "1994-06-23T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1994,

"releaseMonth": 5,

"releaseDay": 23

},

{ "year": "1995",

"title": "Braveheart",

"imdbId": "tt0112573",

"releaseDate": "1995-05-19T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1995,

"releaseMonth": 4,

"releaseDay": 19

},

{ "year": "1996",

"title": "The English Patient",

"imdbId": "tt0116209",

"releaseDate": "1996-11-12T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1996,

"releaseMonth": 10,

"releaseDay": 12

},

{ "year": "1997",

"title": "Titanic",

"imdbId": "tt0120338",

"releaseDate": "1997-12-14T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1997,

"releaseMonth": 11,

"releaseDay": 14

},

{ "year": "1998",

"title": "Shakespeare in Love",

"imdbId": "tt0138097",

"releaseDate": "1998-12-08T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1998,

"releaseMonth": 11,

"releaseDay": 8

},

{ "year": "1999",

"title": "American Beauty",

"imdbId": "tt0169547",

"releaseDate": "1999-09-08T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 1999,

"releaseMonth": 8,

"releaseDay": 8

},

{ "year": "2000",

"title": "Gladiator",

"imdbId": "tt0172495",

"releaseDate": "2000-05-01T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 2000,

"releaseMonth": 4,

"releaseDay": 1

},

{ "year": "2001",

"title": "A Beautiful Mind",

"imdbId": "tt0268978",

"releaseDate": "2001-12-13T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 2001,

"releaseMonth": 11,

"releaseDay": 13

},

{ "year": "2002",

"title": "Chicago",

"imdbId": "tt0299658",

"releaseDate": "2002-12-18T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 2002,

"releaseMonth": 11,

"releaseDay": 18

},

{ "year": "2003",

"title": "The Lord of the Rings: The Return of the King",

"imdbId": "tt0167260",

"releaseDate": "2003-12-17T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 2003,

"releaseMonth": 11,

"releaseDay": 17

},

{ "year": "2004",

"title": "Million Dollar Baby",

"imdbId": "tt0405159",

"releaseDate": "2004-12-15T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 2004,

"releaseMonth": 11,

"releaseDay": 15

},

{ "year": "2005",

"title": "Crash",

"imdbId": "tt0375679",

"releaseDate": "2005-04-26T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 2005,

"releaseMonth": 3,

"releaseDay": 26

},

{ "year": "2006",

"title": "The Departed",

"imdbId": "tt0407887",

"releaseDate": "2006-09-26T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 2006,

"releaseMonth": 8,

"releaseDay": 26

},

{ "year": "2007",

"title": "No Country for Old Men",

"imdbId": "tt0477348",

"releaseDate": "2007-11-04T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 2007,

"releaseMonth": 10,

"releaseDay": 4

},

{ "year": "2008",

"title": "Slumdog Millionaire",

"imdbId": "tt1010048",

"releaseDate": "2008-11-12T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 2008,

"releaseMonth": 10,

"releaseDay": 12

},

{ "year": "2009",

"title": "The Hurt Locker",

"imdbId": "tt1655246",

"releaseDate": "2009-01-29T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 2009,

"releaseMonth": 0,

"releaseDay": 29

},

{ "year": "2010",

"title": "The King's Speech",

"imdbId": "tt1504320",

"releaseDate": "2010-12-24T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 2010,

"releaseMonth": 11,

"releaseDay": 24

},

{ "year": "2011",

"title": "The Artist",

"imdbId": "tt1655442",

"releaseDate": "2011-11-23T05:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 2011,

"releaseMonth": 10,

"releaseDay": 23

},

{ "year": "2012",

"title": "Argo",

"imdbId": "tt1024648",

"releaseDate": "2012-10-04T04:00:00.000Z",

"releaseCountry": "USA",

"releaseYear": 2012,

"releaseMonth": 9,

"releaseDay": 4

}])

**create index**

db.<collection name>.createIndex({<field to index>:<1 for ascend, -1 descend>})

**create index**

db.oscars.createIndex({title:1})

**see indexes**

db.oscars.getIndexes()

[learn to create a unique index and more]

(<https://docs.mongodb.com/manual/reference/method/db.collection.createIndex/#db.collection.createIndex>)

**aggregate**

Aggregations operations process data records and return computed results. Aggregation operations group values from multiple documents together, and can perform a variety of operations on the grouped data to return a single result. MongoDB provides three ways to perform aggregation: the aggregation pipeline, the map-reduce function, and single purpose aggregation methods.

[documenation about aggregation](https://docs.mongodb.com/manual/aggregation/)

**single purpose aggregation**

[documenation about single purpose aggregation]

(<https://docs.mongodb.com/manual/aggregation/#single-purpose-agg-operations>)

There are two functions you can use:

**[db.collection.count()]**

(https://docs.mongodb.com/manual/reference/method/db.collection.count/#db.collection.count)

**[db.collection.distinct()]**

(https://docs.mongodb.com/manual/reference/method/db.collection.distinct/#db.collection.distinct)

db.collection.distinct(field, query, options)

| Parameter | Description |

| --- | --- |

| field | The field for which to return distinct values.

| query | A query that specifies the documents from which to retrieve the distinct values.

| options | Optional. A document that specifies the options. See Options.

examples - count()

db.oscars.count()

db.oscars.find().count()

db.customers.find({role:"citizen"}).count()

db.customers.find({$or: [{name:"Bond"}, {age:{$gt:32}}]}).count()

examples - distinct() - setup

db.inventory.insert([

{ "\_id": 1, "dept": "A", "item": { "sku": "111", "color": "red" }, "sizes": [ "S", "M" ] },

{ "\_id": 2, "dept": "A", "item": { "sku": "111", "color": "blue" }, "sizes": [ "M", "L" ] },

{ "\_id": 3, "dept": "B", "item": { "sku": "222", "color": "blue" }, "sizes": "S" },

{ "\_id": 4, "dept": "A", "item": { "sku": "333", "color": "black" }, "sizes": [ "S" ] }

])

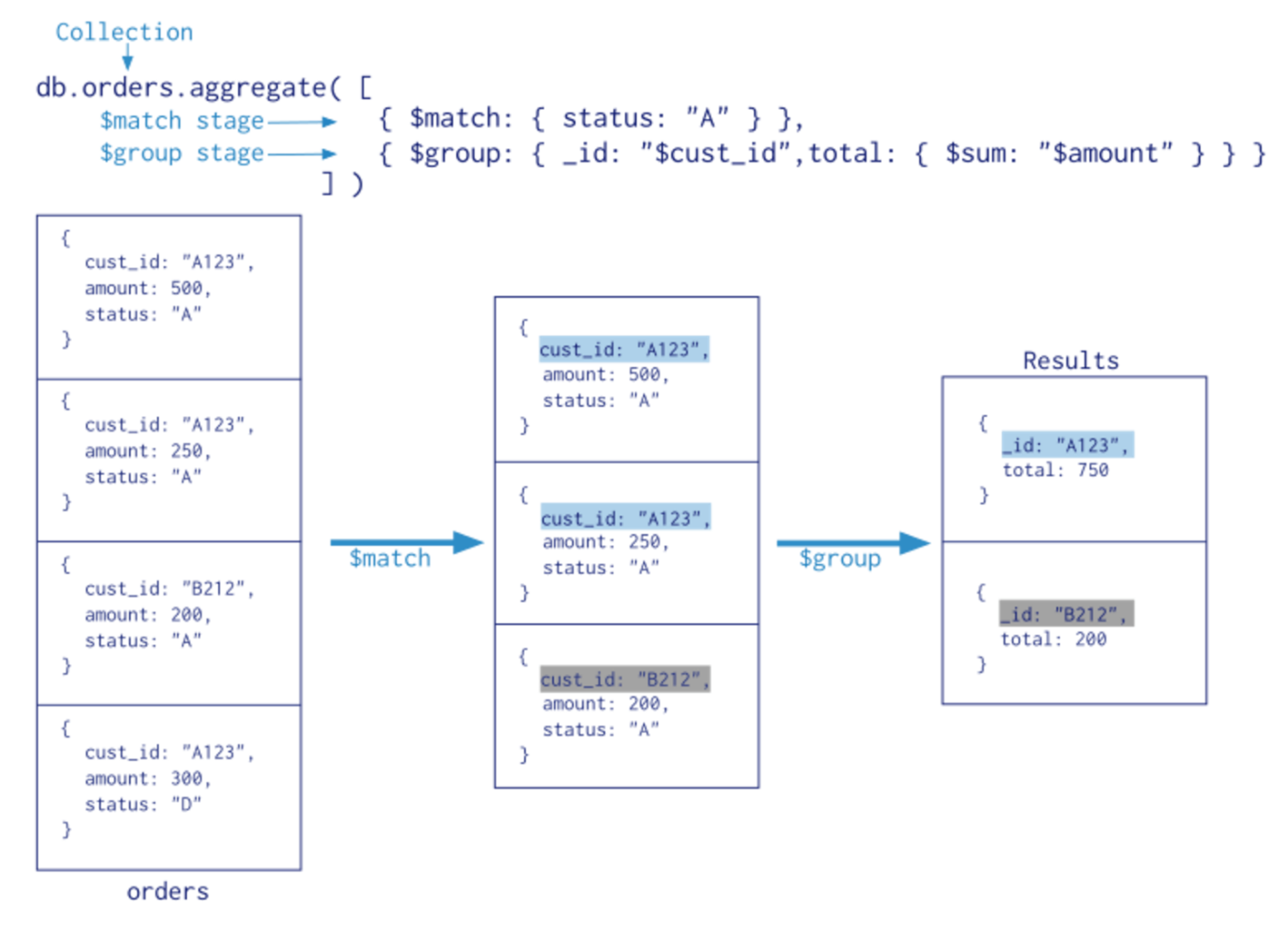
examples - distinct()

db.inventory.distinct( "dept" )

db.inventory.distinct( "item.sku" )

db.inventory.distinct( "sizes" )

**aggregation pipeline**



db.<collection name>.aggregate([{<match, sort, geoNear>},{<group>}])

MongoDB’s aggregation framework is modeled on the concept of data processing pipelines. Documents enter a multi-stage pipeline that transforms the documents into an aggregated result.

The most basic pipeline stages provide filters that operate like queries and document transformations that modify the form of the output document.

Other pipeline operations provide tools for grouping and sorting documents by specific field or fields as well as tools for aggregating the contents of arrays, including arrays of documents. In addition, pipeline stages can use operators for tasks such as calculating the average or concatenating a string.

The pipeline provides efficient data aggregation using native operations within MongoDB, and is the preferred method for data aggregation in MongoDB.

(<https://docs.mongodb.com/manual/aggregation/>)

example - setup

db.orders.insert([

{"cust\_id":"A123","amount":500,"status":"A"},

{"cust\_id":"A123","amount":250,"status":"A"},

{"cust\_id":"B212","amount":200,"status":"A"},

{"cust\_id":"A123","amount":300,"status":"D"}

])

example

db.orders.aggregate([

{$match:{status:"A"}},

{$group:{\_id: "$cust\_id",total: {$sum:"$amount"}}}

])

**locking down your database**

**create admin super user**

```

use admin

db.createUser(

{

user: "jamesbond",

pwd: "moneypennyrocks007sworld",

roles: [ { role: "userAdminAnyDatabase", db: "admin" } ]

}

)

```

[built in user roles](https://docs.mongodb.com/manual/reference/built-in-roles/)

**exit mongo & then start again**

```

mongod --auth

mongo -u "jamesbond" -p "moneypennyrocks007sworld" --authenticationDatabase "admin"

```

**see current user**

```

db.runCommand({connectionStatus : 1})

```

**create regular user**

Give this user readwrite permissions on the ```store``` db.

```

db.createUser(

{

user: "bond",

pwd: "moneypenny007",

roles: [ { role: "readWrite", db: "store" } ]

}

)

```

**exit mongo & then start again**

```

mongod --auth

mongo -u "bond" -p "moneypenny007" --authenticationDatabase "store"

```

**see current user**

```

db.runCommand({connectionStatus : 1})

```

**lock down the database**

[enable auth](https://docs.mongodb.com/master/tutorial/enable-authentication/)

[getting auth running on mongo](https://docs.mongodb.com/manual/tutorial/enable-authentication/)

**exit mongo & then start again with auth enabled**

```

mongod --auth

mongo -u "bond" -p "moneypenny007" --authenticationDatabase "store"

```

**test**

```

use store

show collections

db.customers.find()

db.customers.insert({"role" : "double-zero", "name" : "Elon Musk", "age" : 47 })

```

**test**

launch a new terminal window

```

mongo

```

should be unauthorized:

```

show collections

```

**drop user**

```

db.dropUser("<user name>")

```