**Difference between RDBMS & NoSql DB:**

In Relational DB, we have impedance mismatch(In Java, we have the data as objects, but in DB, it will be stored as tables. This is called Impedance mismatch). In NoSql DB, there is no impedance mismatch(There is no table or schema, data will be stored as simple, complex, flat as your app requires.)

**Mongo DB Executables:**



mongod.exe is the daemon server. mongo.exe is the shell to connect to the server. Remaining exe’s are back & restore mongo, monitor mongo, scale the db etc..

**First run :**

Download the mongodb from mongodb.org and extract it in your local. Then create a directory using md “/data/db” for it to store the db files. Then go to the installation path and navigate till bin and just say mongod on the command prompt

Then give “mongo” to connect to the database through shell.

**Command Line Options:**

You can create your own configuration file with the updated path to store the database files, space to store log files etc

Mongod -f <Path to the configuration file>

**Install a service:**

mongod -f <Path to the configuration file> --install // To install mongo db as a service

net start mongodb // To start the mongodb service

net start | findstr Mongo // To check if the mongo db service is running

mongo// To connect to the mongodb using the mongo shell

ctrl + c or exit // to exit the shell

show dbs// To show the list of dbs. Local db is used by mongo for internal purpose. If you want to store anything, you have to use some other dbs

db// To tell which db you are connected to. By default, this would show test if you don’t specify the db name while connecting to the database.

Use foo// If you want to connect to a different db

Help// To get help on the different options in the shell

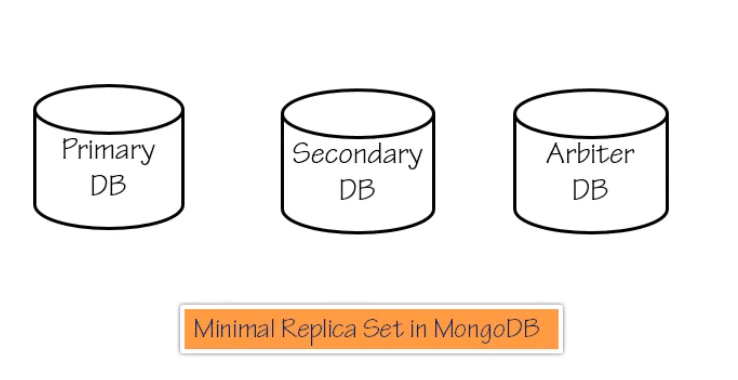
**Replica set:**

This will be useful if there is any single server failure.

Members of a replica set are

* Primary DB(Only one writable instance. That means all clients should be connected to the primary. An attempt to connect to the secondary will fail)
* Secondary DB(Read only instances. Any number of secondary dbs can be there. This means that you have scalability, you can attempt to read from more than 1 read instance. Data from primary db will be copied to the secondary db eventually. This is called eventual consistency. If in case of any primary server failure, one of the secondary servers will take over based on some elections. The secondary server which gets more than 50% of the vote will become the primary. )
* Arbiter DB(Arbiter doesn’t have any data on it. The primary purpose is to break the tie in the delection)

*Replica set up in single machine:*



The idea is to start the above three servers in 3 different ports.

1. Open up the command prompt as administrator, go to the mongo db path and issue the following command,

*mongod -f C:\PS\data1\config\mongod.conf --replSet "setone"*

1. Open up the command prompt as administrator, go to the mongo db path and try connecting to the first db instance(Primary) through mongo shell,

*mongo –port 60000*

1. Open up the command prompt as administrator, go to the mongo db path and issue the following command,

*mongod -f C:\PS\data2\config\mongod.conf --replSet "setone"*

1. Open up the command prompt as administrator, go to the mongo db path and try connecting to the first db instance(Secondary) through mongo shell,

*mongo –port 60001*

1. Open up the command prompt as administrator, go to the mongo db path and issue the following command,

*mongod -f C:\PS\data3\config\mongod.conf --replSet "setone"*

1. Open up the command prompt as administrator, go to the mongo db path and try connecting to the first db instance(Primary) through mongo shell,

*mongo –port 60003*

1. Now 3 servers are up and running. We have set up the replica as follows in the mongo shell connecting to the primary server,

*var setoneconfig = {"\_id" : "setone","members" :[{"\_id" : 0,"host" : "localhost:60000","priority" : 10},{"\_id" : 1,"host" : "localhost:60001"},{"\_id" : 2,"host" : "localhost:60003","arbiterOnly" : true}]}*

1. Then issue the command to initiate the replication set,

*rs.initiate(setoneconfig)*

1. Status of the replication set can be got from the following command,

*rs.status()*

*Verifying Replication*

1. Give the following command in the primary mongo shell,

*db.test.save({\_id:1, value:'hello world'})*

1. To check if the data is properly inserted in the primary server, use

*db.test.find()*

1. Then go to the secondary mongo shell and try to find the data inserted using

*db.test.find()*

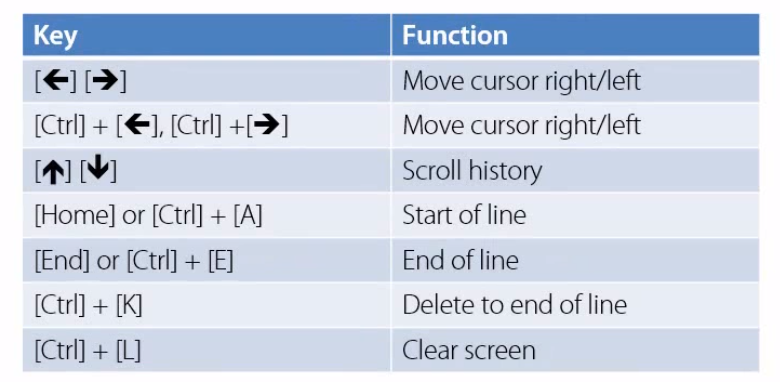
1. If you see any error in the secondary shell, set slaveok as true in the secondary and then give the find command. Now the data inserted in primary server will be available in the secondary server as well.

*Replica Set Failover*

1. Kill the primary server
2. Go to the secondary mongo shell and try to find the data in secondary, data will be available and the prompt will show as PRIMARY in the secondary shell.

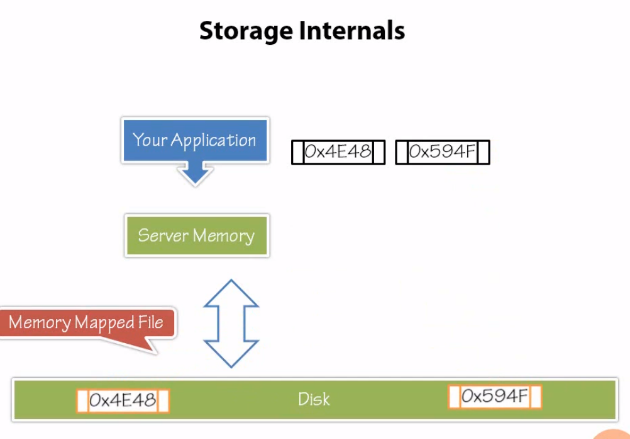
**Mongo Shell**

Shell is a javascript interpreter that helps you to interact with the mongodb.



**Saving Data**

***Storage***



Everything is stored in memory as memory mapped file in form of byte arrays. How is the data gets saved without any schema information.? BSON.( <http://bsonspec.org/>). Storing the data in the form of bson in memory mapped files makes it very fast data storage engine

Rules for storing data in Mongo DB : A document must have an \_id field.

Size of the document is limited to 16 MB. It may be get increased in future releases

***Collection*** : Collection in mongo defines the scope of interaction with documents so that you can issue specific commands against a collection. Since this is not relational, you can issue commands across multiple collections

Show collections // list all collections

Db. Foo. Save({ \_id:1, x:3}) // Here foo is a collection.

Datatype of the id column can not be an array

***ObjectId*** : This gets automatically inserted if there is no \_id column. You can generate timestamp with this objectId.

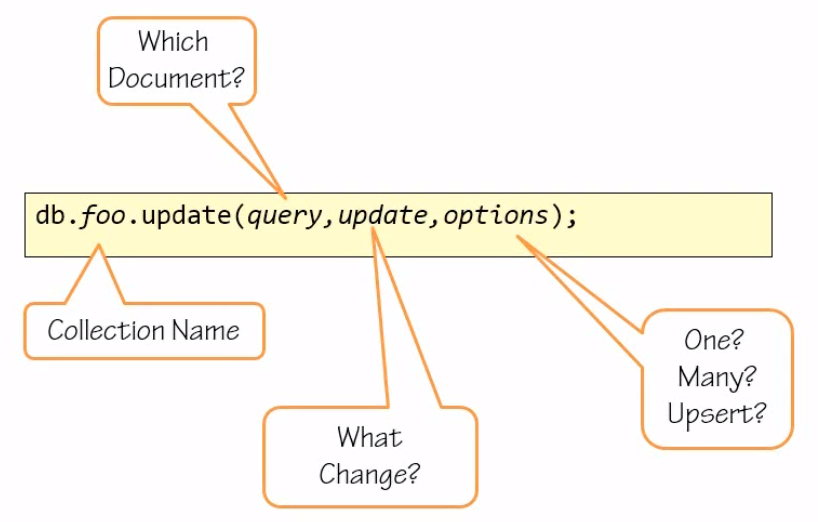
ObjectId().getTimestamp()

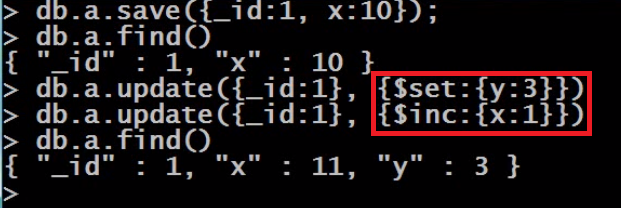
***Difference between save and insert*** : save will simply override the values if the id is the same for 2 records. Insert will not override the values if 2 records have the same id throwing an error

Why is update needed.? To avoid concurrency problem. Update is atomic in nature. Operations are sequential

***Update***

Options is an optional field. If you don’t specify the options field, only the first record will be updated. Upsert means if the record is not available, record will be inserted





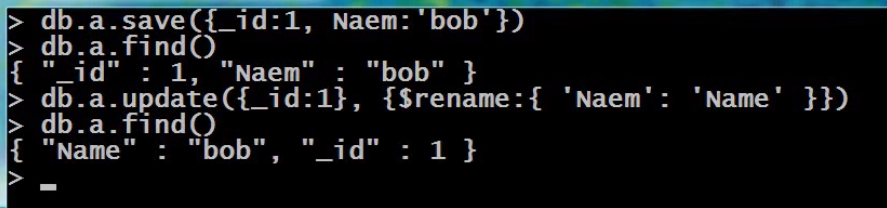
***Operators***:

Set operator // The $set operator replaces the value of a field with the specified value. If the field is not there, it will be inserted

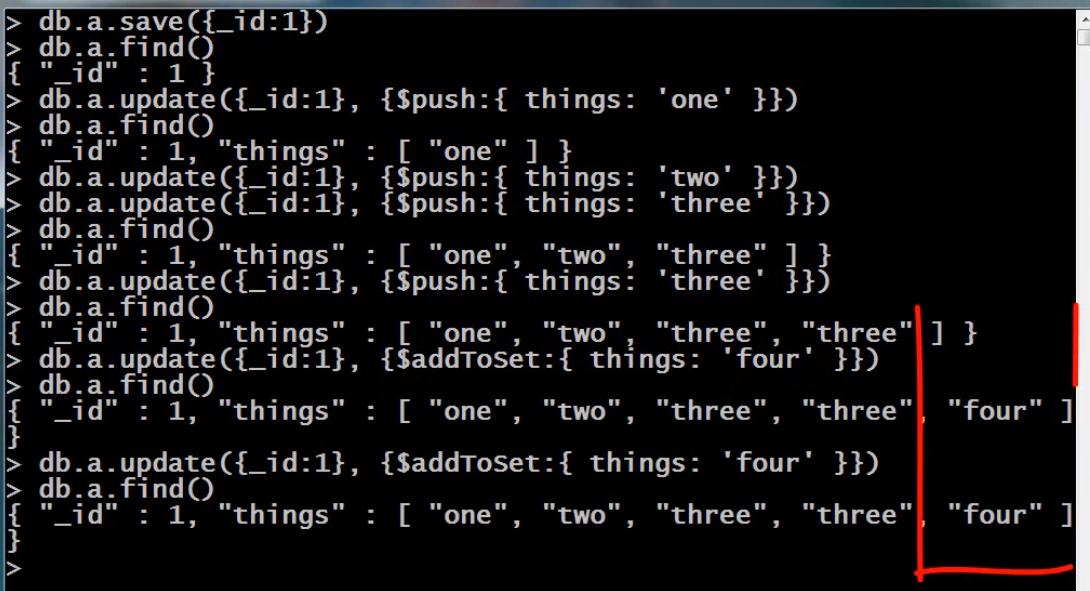
Inc operator// To increment the value

Unset operator// Deletes a particular field

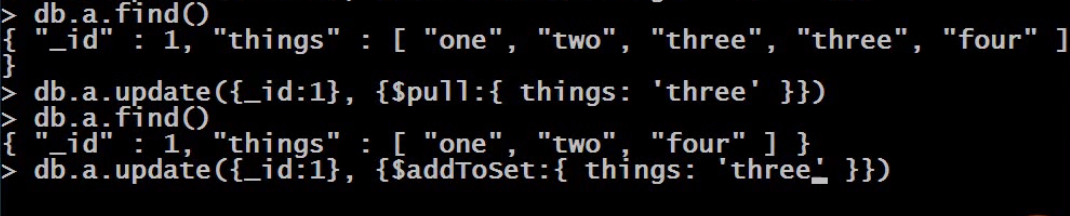
Rename operator// To rename a field



Push & AddtoSet Operator // Push will create an array field with duplicate elements. AddtoSet will not allow for duplicate elements in the array



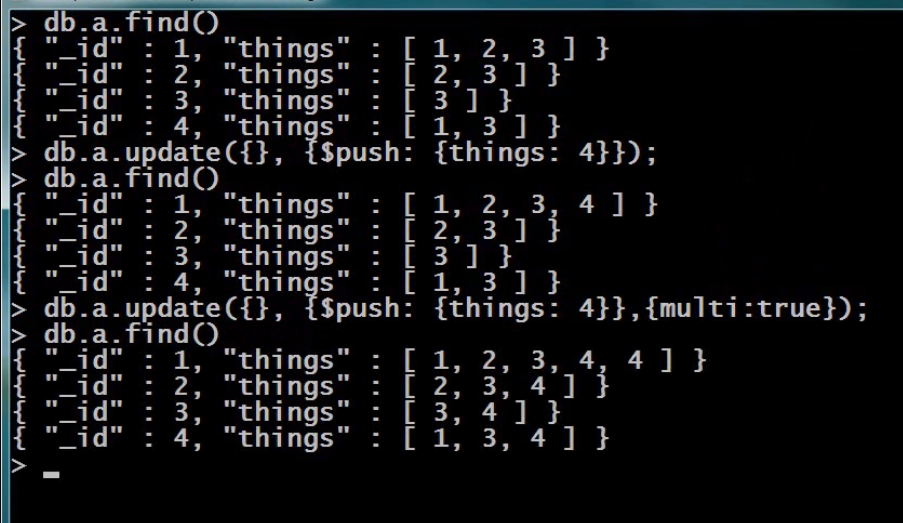
Pull operator// Pulls out all instances of the value from the array



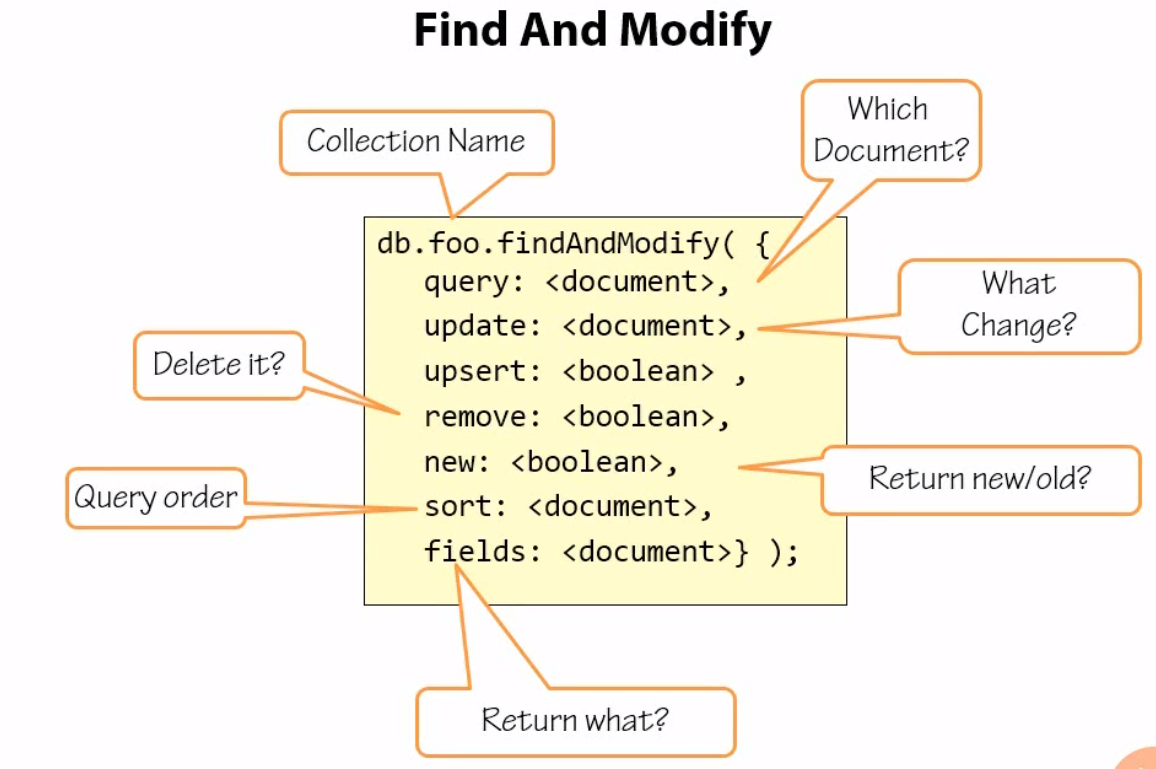
Pop operator//if we don’t know what element has to be pulled out exactly. You use pop operator.

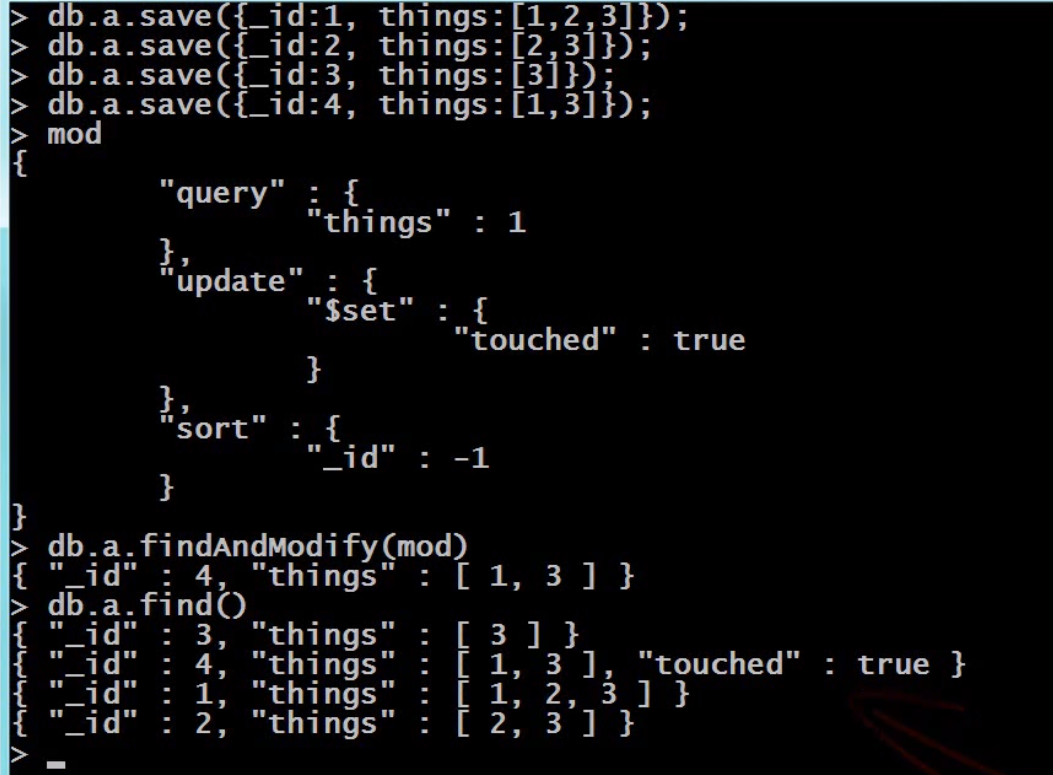
Pop, Push, Pull, AddtoSet // will work only on array types

Multi update // Allow for multiple updates



Find and Modify

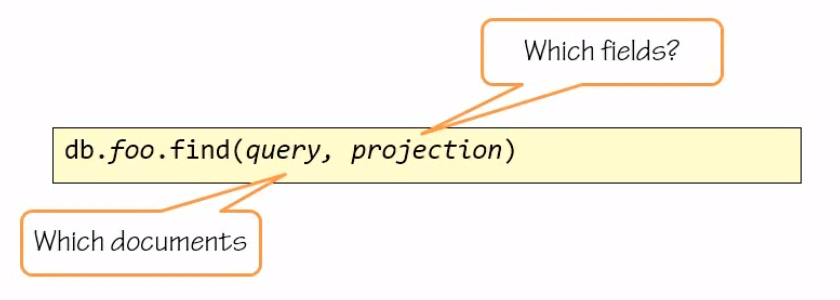




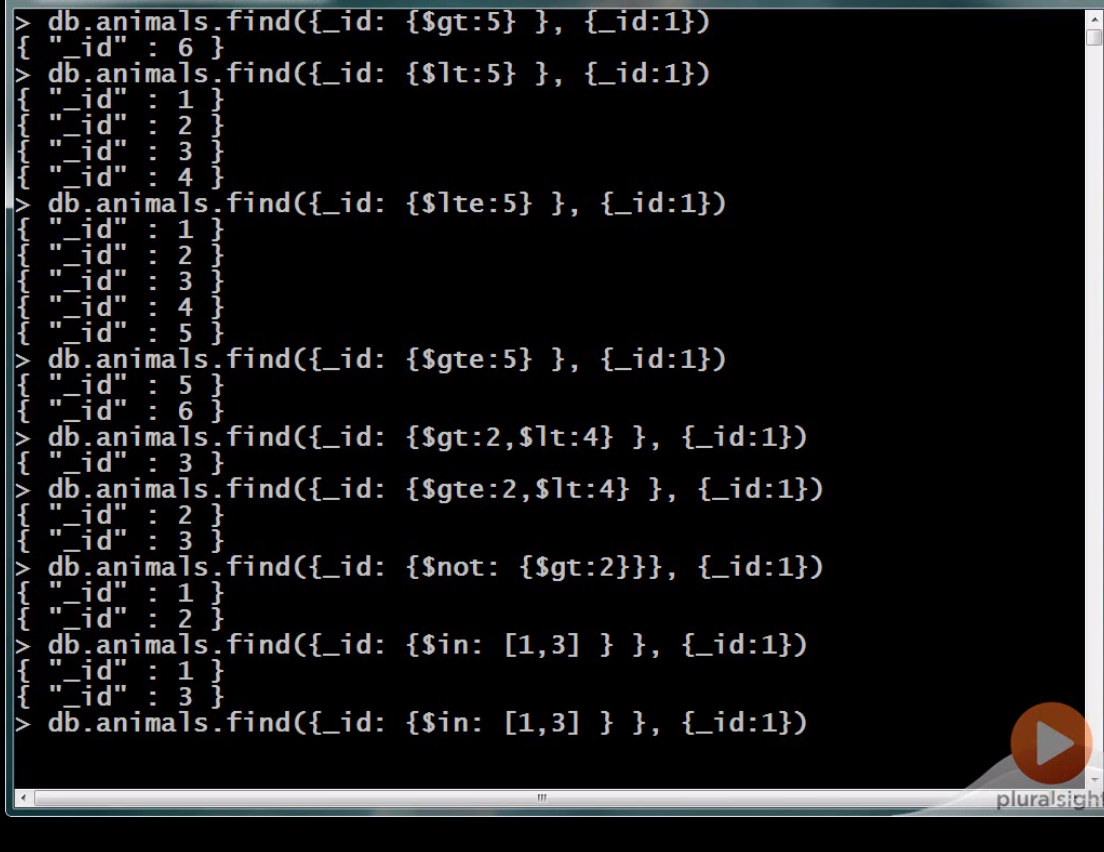
The above will update the only one record by first sorting it in descending order, then based on the query condition things should have 1 in it, then display the record before modification. If you say find(), it will fetch the current state of the record

For documentation, go to mongodb.org and search for documentation

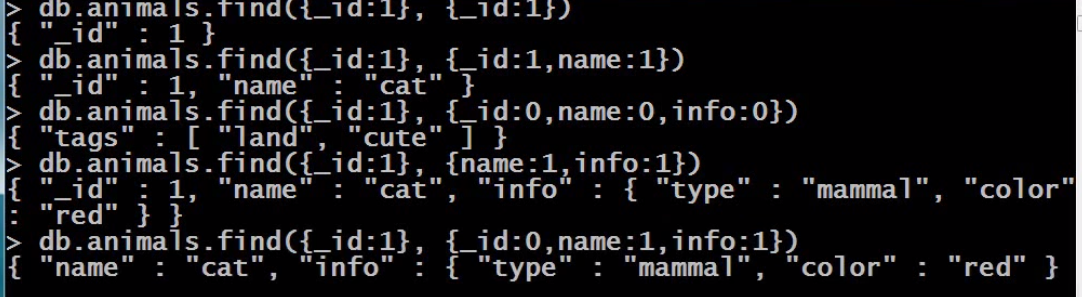
**Finding Documents**

****

Projection is optional field. If you don’t specify this field, all fields will be retrieved

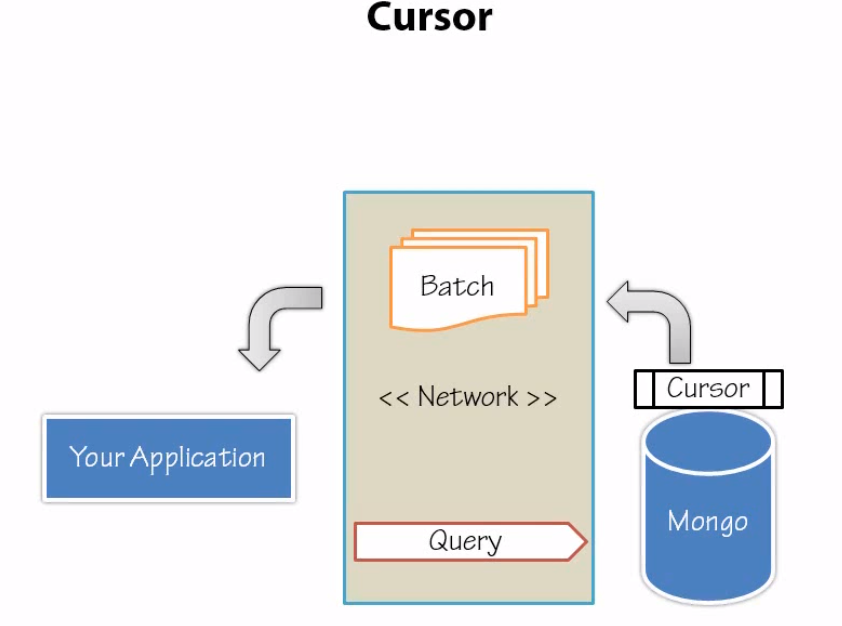


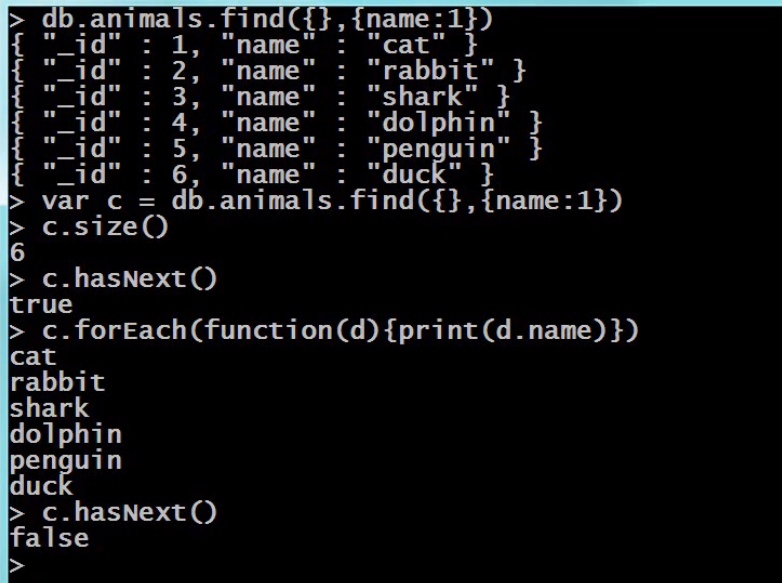
Projection



Projection with a field value zero will be excluded and 1 will be included

Cursor





Sort

