NYUPoly

Enterprise Data Systems
2016

Alice in Unstructured World
(modified for 10/11/2018 Juan Rodriguez)

Raman Kannan

rk1750@nyu.edu
Thursdays 6PM

UnStructured

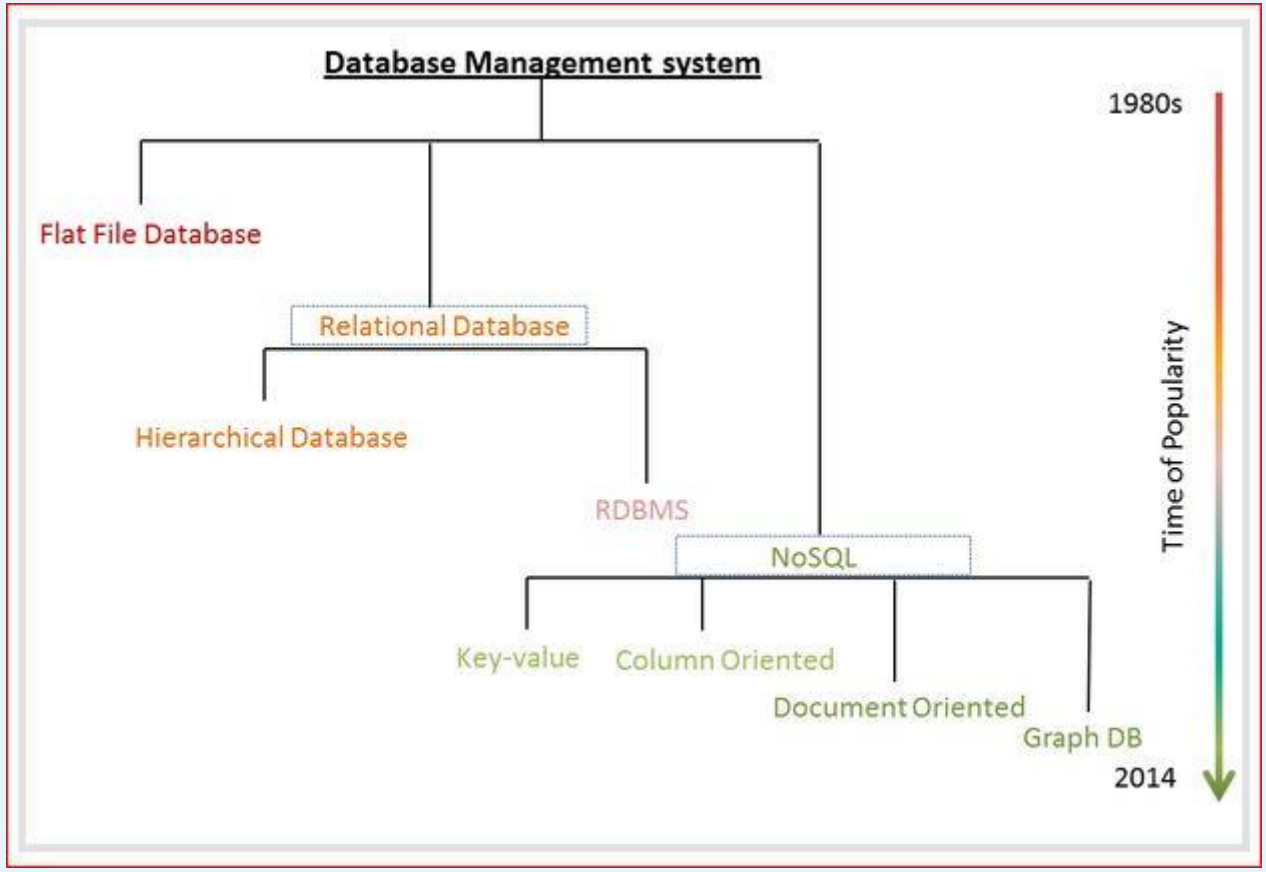
Putting them all together –
– Threading a narrative/story

Introduction to Unstructured data

NOSQL – Not Only SQL Engines

Introduction to mongo

History



Pre-WWW

Data is transactional
ACID/RDBMS rules

Advent of WWW
Email+FTP+Image
+MP3+A/V
Mostly p2p comm
Usenet Groups,
broadcast,list

Social
Communication
CSCW
Youtube
Facebook
twitter

Have we seen it all? Is this the end? Cognitive and contextual computing

Siesmic Shift

Transactional data used to be the bulk of the data
– impersonal, not people oriented, structured, text

Dynamic structure
Structure is not static
Changes
Unmanageable volumes
At speeds never
Anticipated
Interactions are long lived

→

RDBMS cannot handle.
Give up Schema
Give up Structure
Give up Transaction
trade ACID for CAP

Now, it is ALL unstructured, mostly human-oriented,
Unstructured, and variety multimedia stream

And Moore's Law effect → faster computer, faster network
Netflix is streaming movies on par with theater experience

Hello (n silent for hell no)

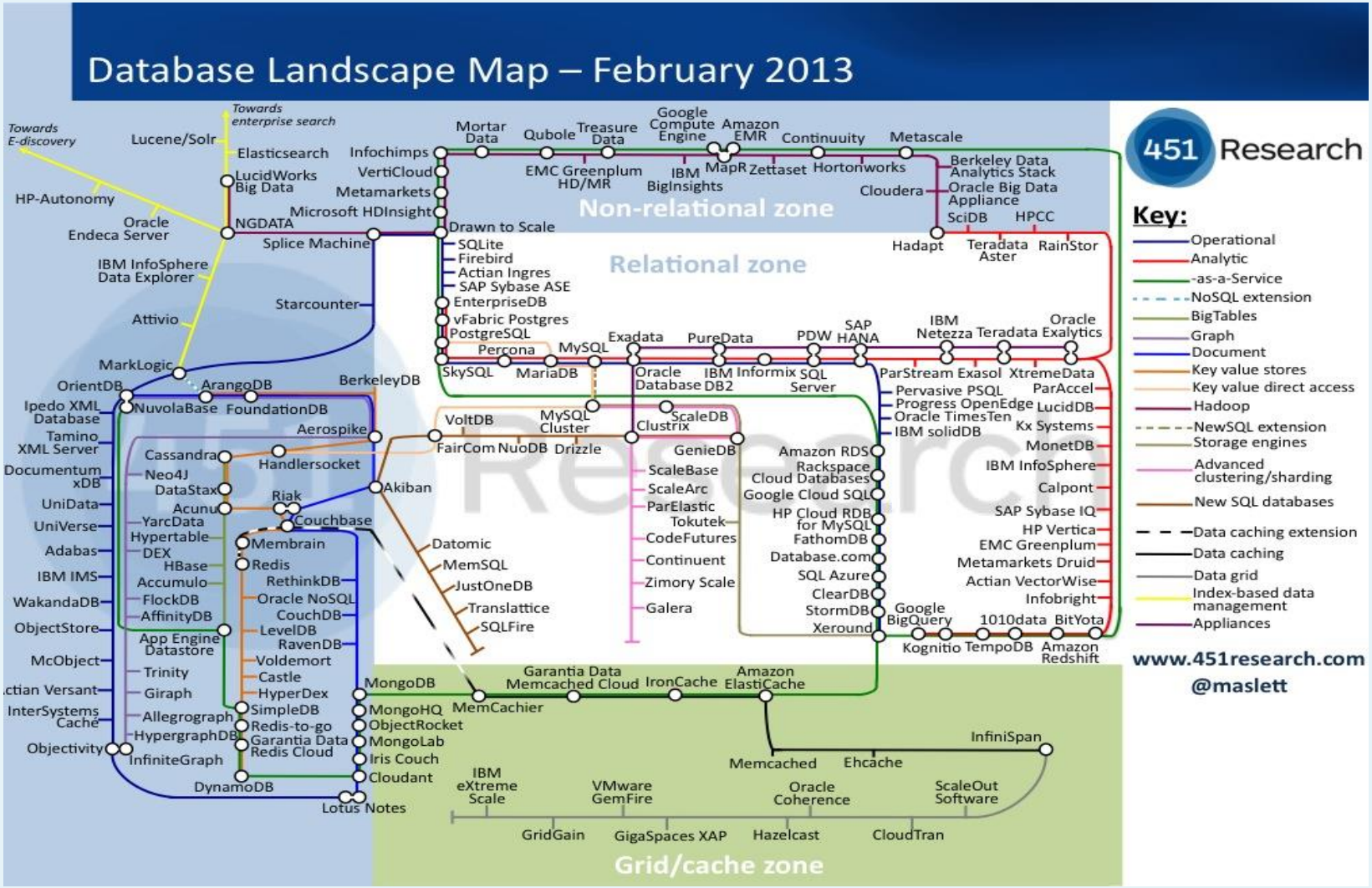
It just started – it will never be over –
pace of new technological innovation is accelerating

Imagine which
car said what to
which car
after a pile up
Insurance
companies
Where is the
black box?

Drones, geolocation, smart delivery
Skies will be crowded
Man made eclipse when there is a snarl
IoT internet of things
Talking humans → talking devices, cars
Nothing is listening everything is talking
Cognitive Computing
Intelligent human compatible computing

Paste is out of the tube..cannot start over.. it will never be the same

World of DBMS



Enter noSQL

- Not only SQL
- Many variant
  - Document oriented
  - Graph database
  - Key/value
  - Wide column
- Mongo – document oriented
  - Hu-mongo-us
  - No transaction / no acid / no schema / no relation /not tabular

Resources

http://docs.mongodb.org/manual/tutorial/write-scripts-for-the-mongo-shell/
http://howtodoinjava.com/2014/05/26/introduction-to-mongodb-why-mongodb/
http://docs.mongodb.org/manual/core/introduction/
http://howtodoinjava.com/2014/05/29/mongodb-selectqueryfind-documents-examples/
http://jandiandme.blogspot.com/search/label/CAP%20Theorem
http://jandiandme.blogspot.de/2013/06/mongodb-and-cap-theorem.html
http://info.mongodb.com/rs/mongodb/images/10gen_Top_5_NoSQL_Considerations.pdf
http://css.dzone.com/articles/how-acid-mongodb

http://docs.mongodb.org/manual/tutorial/getting-started/

http://www.mkyong.com/mongodb/how-to-install-mongodb-on-mac-os-x/
http://docs.mongodb.org/manual/reference/operator/#comparison

http://www.mkyong.com/mongodb/mongodb-authentication-example/

ACID Gone with the Wind!

CAP

The CAP theorem by Brewer basically says that a distributed systems can only have two of the following three properties:

- Consistency i.e. each node has the same data
- Availability i.e. a node will always answer queries if possible
- Partition tolerance i.e. work despite a network failure so nodes cannot communicate with one another

No relations → no joins

Object embedding

Object ID Reference

This needs to be resolved.

Lingo: JSON

Javascript object notation

http://json.org/ & http://stackoverflow.com
for doubts

http://www.freeformatter.com/json-formatter.html#json-explained

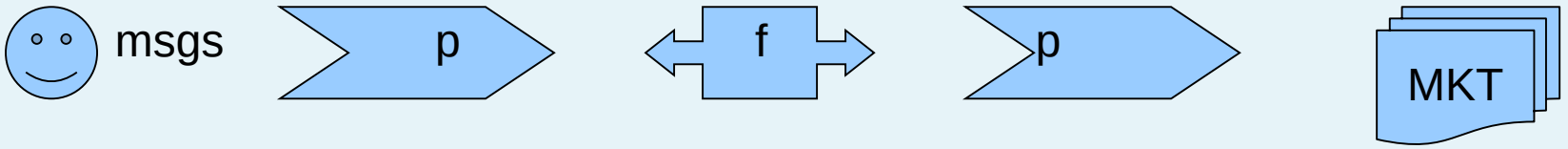
http://www.freeformatter.com/json-validator.html#json-explained

useless example

```
{
   "anObject": {
      "numericProperty": -122,
      "stringProperty": "An offensive \" is
problematic",
      "nullProperty": null,
      "booleanProperty": true,
      "dateProperty": "2011-09-23"
   },
   "arrayOfObjects": [
      {
         "item": 1
      },
      {
         "item": 2
      },
      {
         "item": 3
      }
   ],
   "arrayOfIntegers": [
      1,
      2,
      3,
      4,
      5
   ]
}
```

{example:"value"} is invalid but {"example":"value"} is valid.

Useful examples




```
db.filters.insert({ class:"stateless", type:"BasicFilter", name:"BF01",
...  predicate: [ {T:"35", op:"EQ", value: [ "D" ]} ,
 {T:"55",op:"EQ", value: ["IBM","ABB","CSCO"]} ] } );

book1 = {“name”: "Understanding JAVA", “pages” : 100}

db.books.save(book1)
```

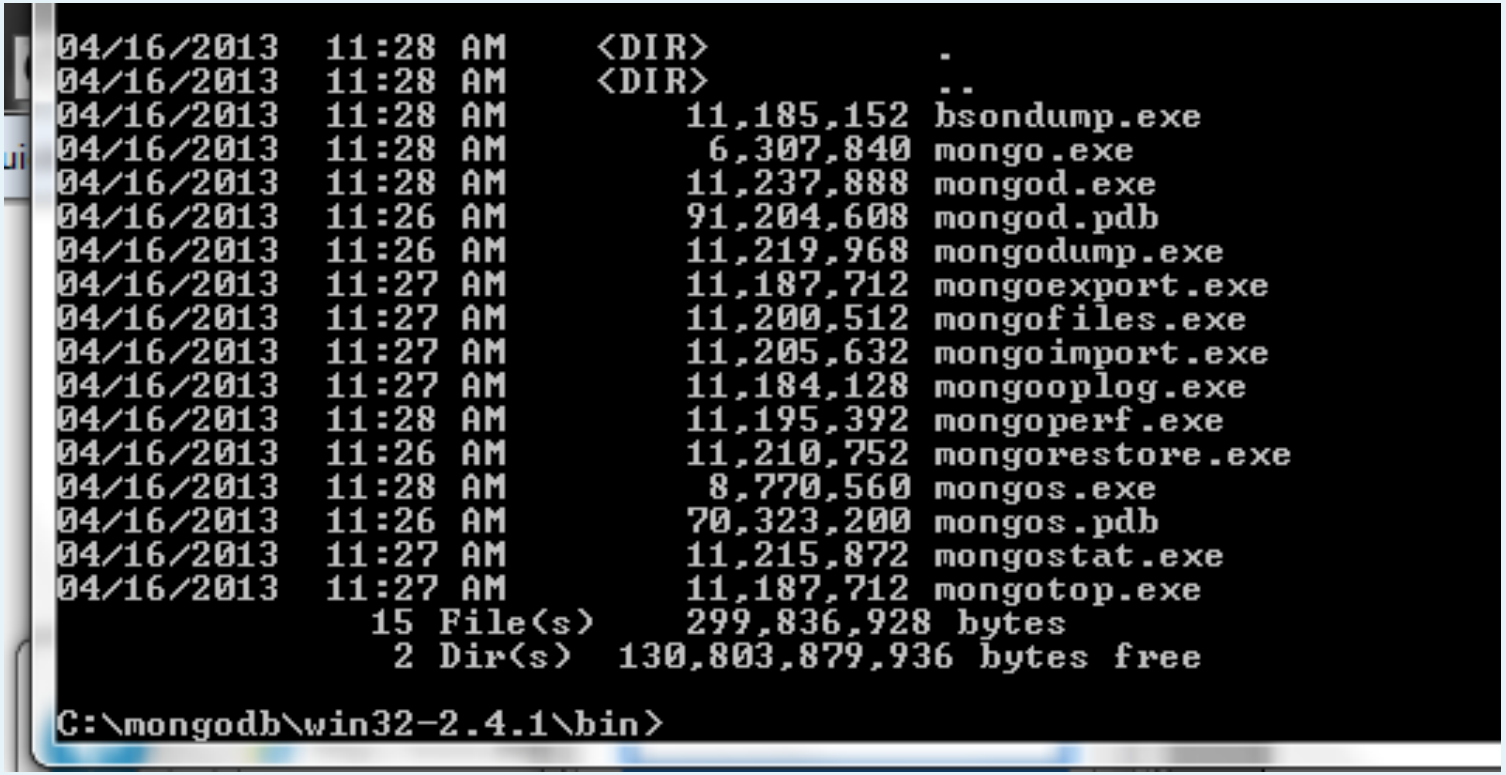
steps

http://docs.mongodb.org/manual/

Then (appraise yourself with install etc)
Minimal configuration
Starting and Stopping and
A few basic commands

http://www.mongodb.org/downloads

Setup: install and verify



Here mongod is the server and mongo is the shell client

Starting the server

Data is mine I dont want to loose if my windoooz goes down on me.

Data is in E drive and I can reinstall mongodb in C as many times as I want to without loosing my data

The default port on which mongod serves is data is 27017.
You can configure both these using a conf file...

Start mongod with the dbpath option

Stopping and admin db

```
C:\mongodb\win32-2.4.1\bin>mkdir e:\MONGODB\

C:\mongodb\win32-2.4.1\bin>mkdir e:\MONGODB\data

C:\mongodb\win32-2.4.1\bin>mkdir e:\MONGODB\data\db

C:\mongodb\win32-2.4.1\bin>mongod --dbpath e:\mongodo\data\db
Tue Apr 16 11:34:34.432
Tue Apr 16 11:34:34.437 warning: 32-bit servers don't have journaling enabled by
 default. Please use --journal if you want durability.
Tue Apr 16 11:34:34.439
Tue Apr 16 11:34:34.472 [initandlisten] MongoDB starting : pid=10464 port=27017
dbpath=e:\mongodo\data\db 32-bit host=vram
```

Create a db directory
Start mongod with the dbpath option

Stopping the server

```
> use admin
switched to db admin
> db.shutdownServer({timeoutSecs: 60});
Tue Apr 16 12:53:01.788 DBClientCursor::init ca
server should be down...
Tue Apr 16 12:53:01.808 trying reconnect to 127
```

use admin

db.shutdownServer({timeoutSecs: 60});

config

fork = true

bind_ip = 127.0.0.1

port = 27017

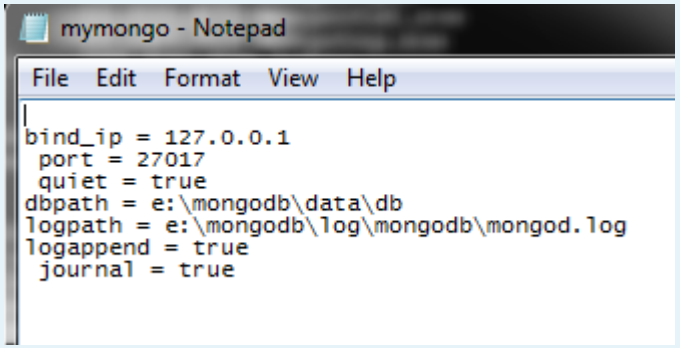
quiet = true

dbpath = e:\mongodb\data\db

logpath = e:\mongodb\log\mongodb\mongod.log

logappend = true

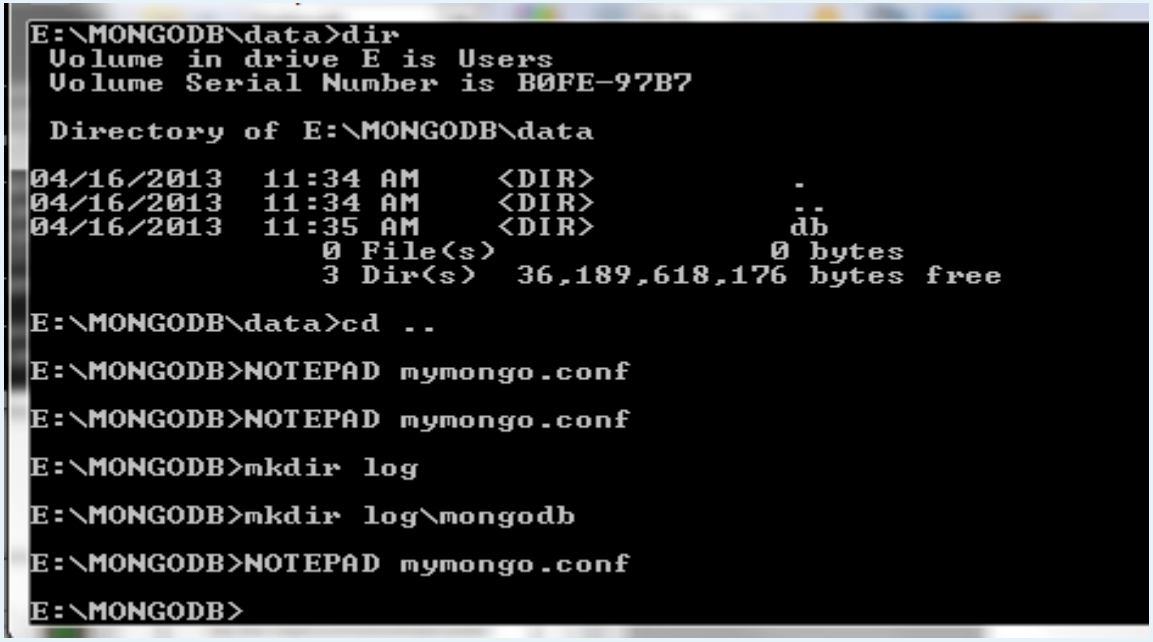
journal = true




```
fork = true
bind_ip = 127.0.0.1
port = 27017
quiet = true
dbpath = /srv/mongodb
logpath = /var/log/mongodb/mongod.log
logappend = true
journal = true
```

Working with PD Software is about patience and T/E.
You cannot give up.
Fork option did not work. My config does not have it

Preparing your DB



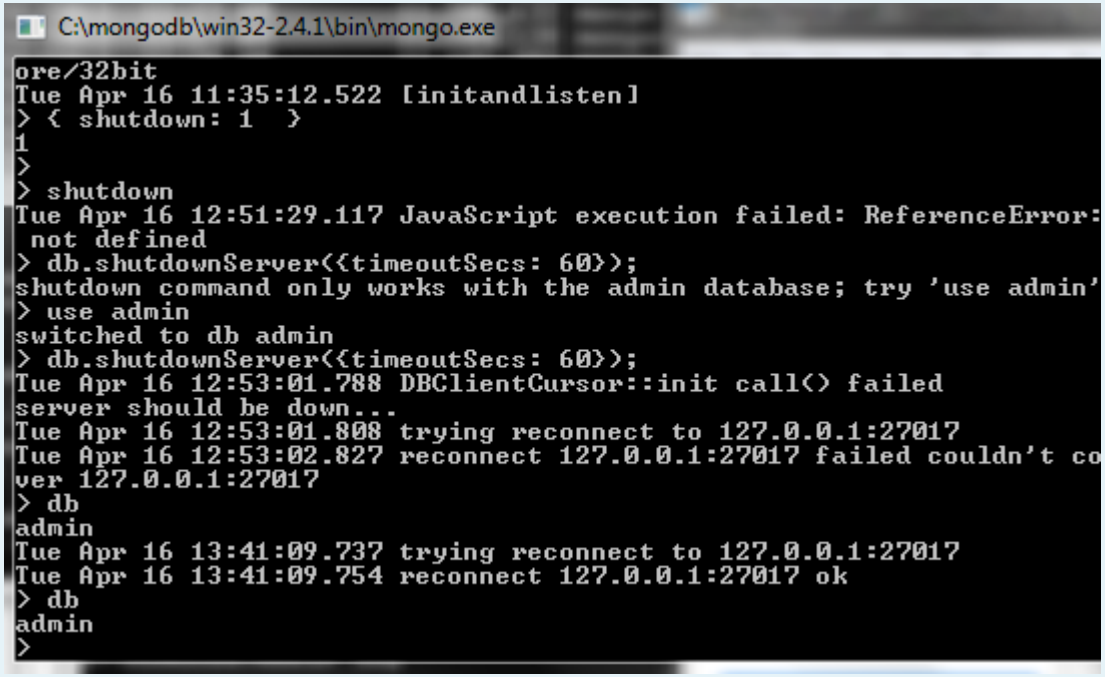
I have stored the config file in e:\mongodb – can be anywhere

I am directing all logs as specfied in the config file

reconnect

Start the server with the config option

```
C:\mongodb\win32-2.4.1\bin>mongod -f e:\MONGODB\mymongo.conf
all output going to: e:\mongodb\log\mongodb\mongod.log
```



Changing DB

```
E:\MONGODB>c:\mongodb\win32-2.4.1\bin\mongo
MongoDB shell version: 2.4.1
connecting to: test
Server has startup warnings:
Tue Apr 16 13:37:30.589 [initandlisten]
Tue Apr 16 13:37:30.590 [initandlisten] ** NOTE: This is a 32 bit MongoDB binary
.
Tue Apr 16 13:37:30.590 [initandlisten] **       32 bit builds are limited to le
ss than 2GB of data (or less with --journal).
Tue Apr 16 13:37:30.590 [initandlisten] **       See http://dochub.mongodb.org/c
ore/32bit
Tue Apr 16 13:37:30.590 [initandlisten]
> db
test
>
```

```
> use mydb
switched to db mydb
> use admin
switched to db admin
> use mydb
switched to db mydb
>
```

Same as mysql!!

C in CRUD

http://docs.mongodb.org/manual/reference/operators/#comparison
http://docs.mongodb.org/manual/core/shell-types/

{symbolname:"IPFF", date:"2013-04-12",open:"27.05",
high:"27.05",low:"26.81",close:"26.86",volume:"35300",
adjClose:"26.86"}

db.ticks.save({symbolname:"IPFF"})
db.ticks.save({symbolname:"IPFF", date:"2013-04-12",
open:"27.05",high:"27.05",low:"26.81",close:"26.86",
volume:"35300",adjClose:"26.86"})

More  Steps

```
db.ticks.find()
db.ticks.find({"high:{$gt:2}"}) -- WILL NOT WORK
db.ticks.find({high:"27.05"})
db.ticks.find({symbolname:"IPFF"})
db.ticks.save({symbolname:"IPFF", date:"2013-04-
12",open:27.05,high:27.05,low:26.81,close:26.86,
volume:35300,adjClose:26.86})
db.ticks.find({high:27.05})
db.ticks.find({high:"27.05"})
db.ticks.find({high:{$gt:27.05}})
db.ticks.find({high:{$gt:27.0}})
```

```
db.ticks.distinct("symbolname")
db.ticks.update([“symbolname”:"IPFF"],[$set:[“high”:2
8.02]])
db.ticks.remove(“symbolname”:"IPFF")
```

CRUD

```
db.execs.find({},{_id:0})

{ "ticker" : "IBM", "qty" : "100", "px" : "202" }

{ "ticker" : "IBM", "qty" : "100", "px" : "203" }

{ "ticker" : "IBM", "qty" : "200", "px" : "198.50"
}

{ "ticker" : "IBM", "qty" : "200", "px" : "199.50"
}

{ "ticker" : "IBM", "qty" : "200", "px" : "199.90"
}
```

```
db.execs.find({},{_id:0},{$sort:{px:1}})

{ "ticker" : "IBM", "qty" : "100", "px" : "202" }

{ "ticker" : "IBM", "qty" : "100", "px" : "203" }

{ "ticker" : "IBM", "qty" : "200", "px" : "198.50"
}

{ "ticker" : "IBM", "qty" : "200", "px" : "199.50"
}

{ "ticker" : "IBM", "qty" : "200", "px" : "199.90"
}
```

Loading

mongoimport --db mydb --collection rktrades --type csv --headerline --file trades.js

Sales collection

```
 db.sales.insert({ Year:2000, Region:"USA",Sales:400})
> db.sales.insert({ Year:2001, Region:"USA",Sales:200})
> db.sales.insert({ Year:2002, Region:"USA",Sales:400})

> db.sales.insert({ Year:2002, Region:"Europe",Sales:300})

> db.sales.insert({ Year:2003, Region:"Europe",Sales:100})

> db.sales.insert({ Year:2008, Region:"USA",Sales:500})
> db.sales.insert({ Year:2003, Region:"Asia",Sales:400})

> db.sales.count({Region:"UDS"})
0

> db.sales.count({Region:"USA"})
4
```

aggregate

```
db.sales.group( {

key: { Region: 1 },

cond: { Year: { $lt: 2005 } },

reduce: function(cur, result) { result.total += cur.Sales },

initial: { total: 0 }

} )
```

Group By

```
db.rktrades.group({key:{TKR:1},reduce:function(cur,result){result.total+=
cur.QTY * cur.PX,result.totalQty+=cur.QTY},
initial:{total:0,totalQty:0}})

db.rktrades.group({key:{TKR:1,SIDE:1},reduce:function(cur,result)
{result.total+=cur.QTY * cur.PX,result.totalQty+=cur.QTY},
initial:{total:0,totalQty:0}})


db.runCommand( { distinct: "sales", key:"Region", query:{Sales: {$gt:200}}})
```