Getting Started with MongoDB

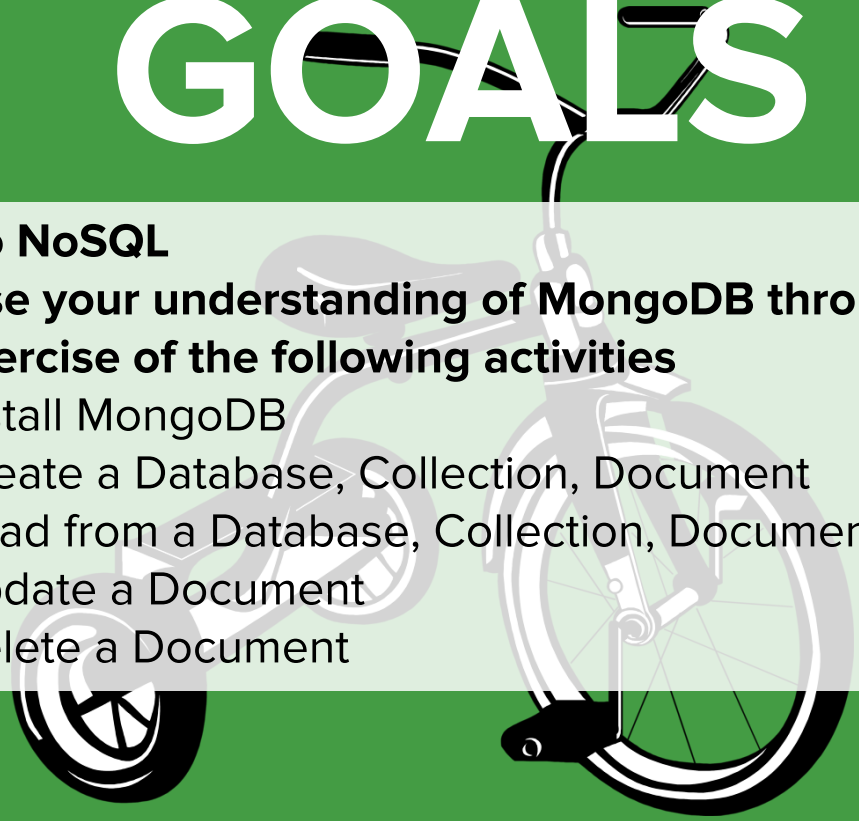
A Beginner's Introduction to MongoDB

Michael Lynn, February 2017

Big Thanks to SideCar!

http://getsidecar.com

GOALS

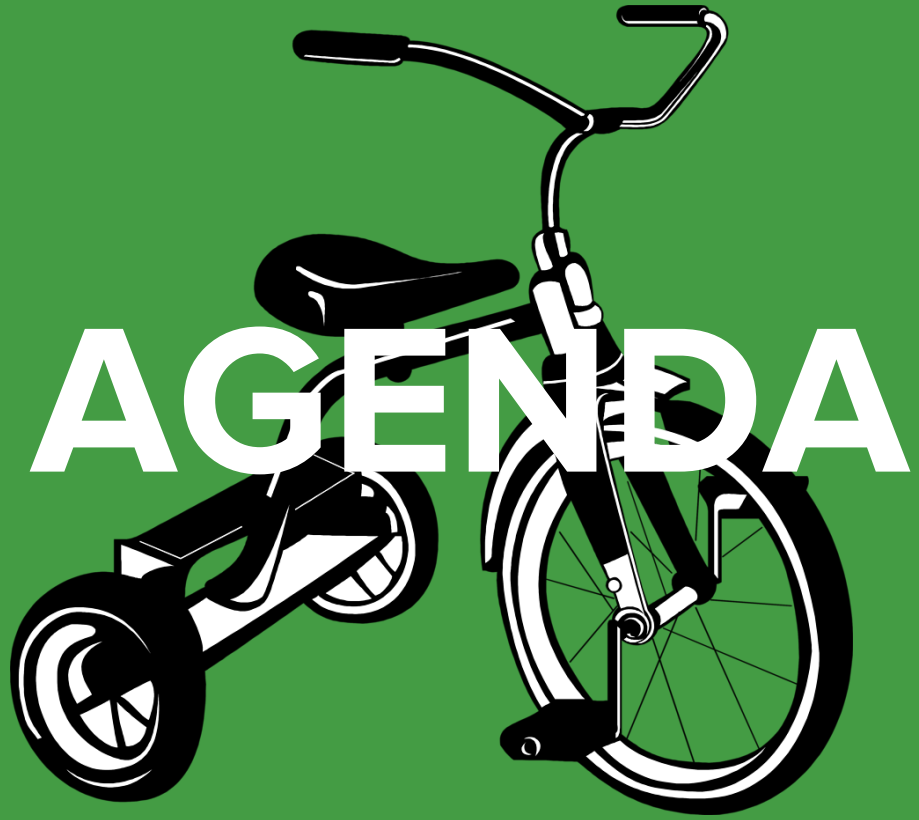


1. **Intro to NoSQL**
2. **Increase your understanding of MongoDB through exposure and exercise of the following activities**
   a. Install MongoDB
   b. Create a Database, Collection, Document
   c. Read from a Database, Collection, Document
   d. Update a Document
   e. Delete a Document

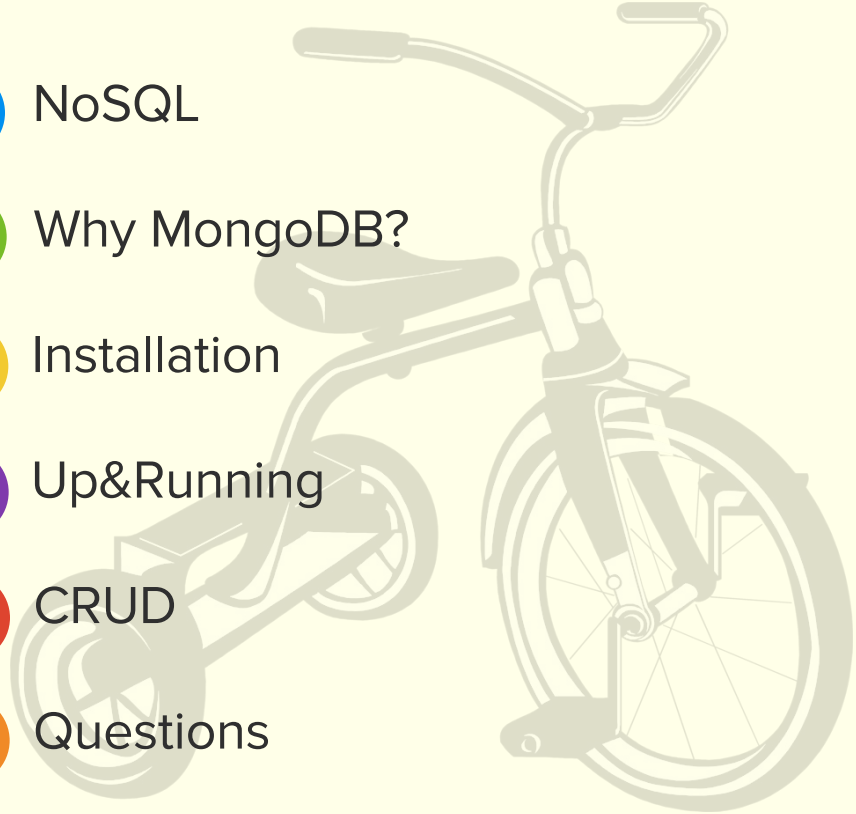
NOT GOALS

1. Techni… deep dive.
2. Debat… out the best way… oose a shard … deploy a cluste… convert a repli… to a sharded clu… or any other 3… evel topic…
3. Teach y… about v… erating system.
4. Sell you… n… e to sell you on Mongo…
5. Make you… d… there are no stupid… stions.

AGENDA



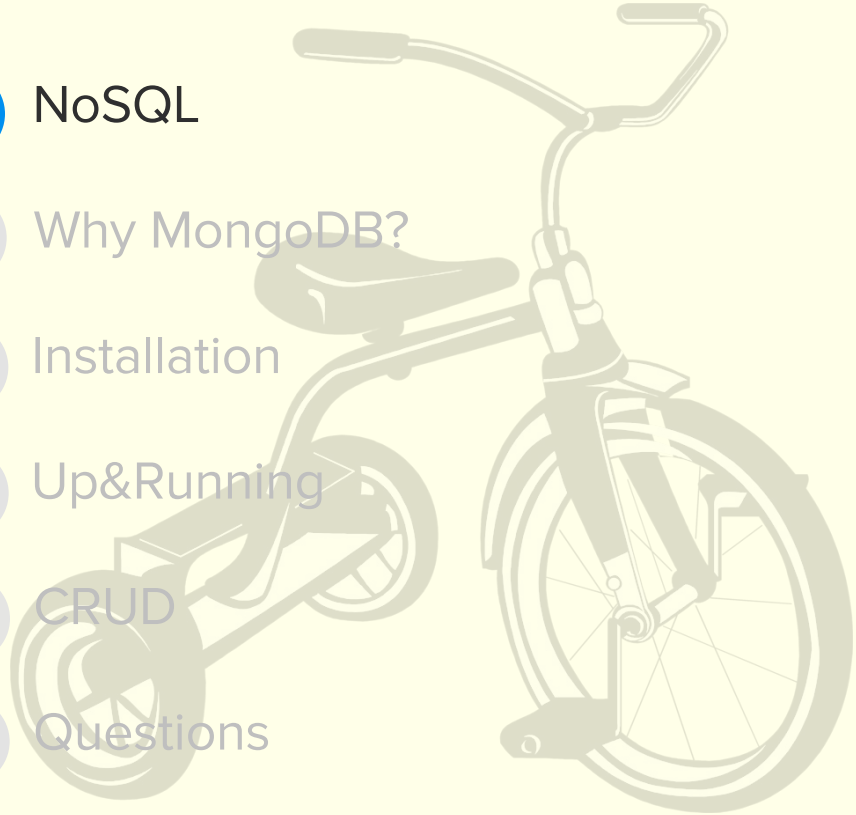
AGENDA



1. NoSQL
2. Why MongoDB?
3. Installation
4. Up&Running
5. CRUD
6. Questions

AGENDA

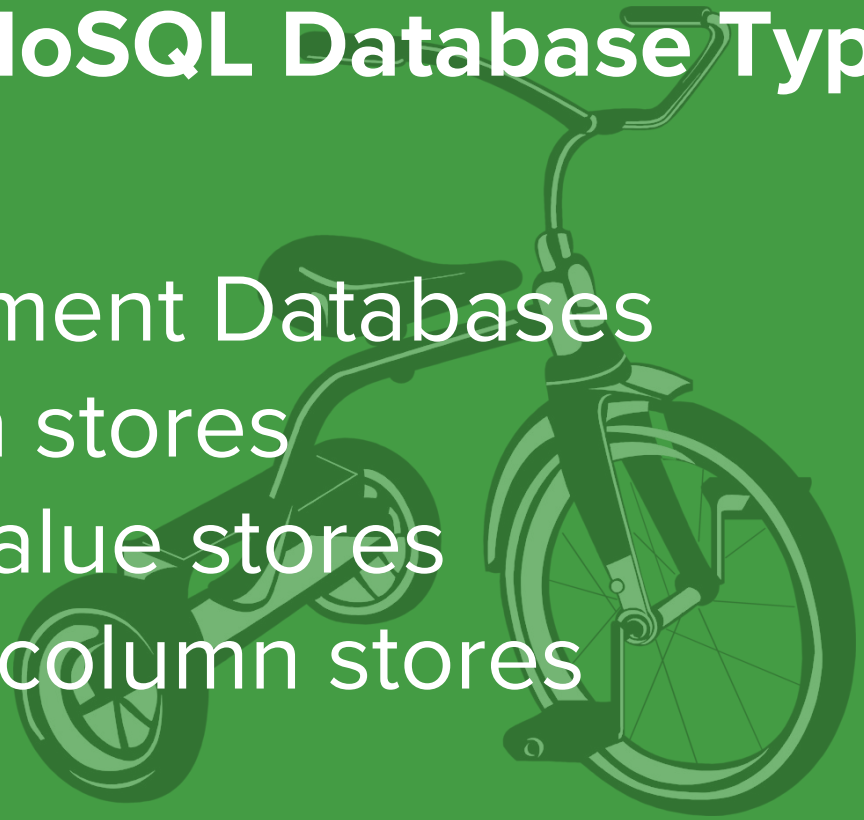
1. NoSQL
2. Why MongoDB?
3. Installation
4. Up&Running
5. CRUD
6. Questions



NoSQL Databases

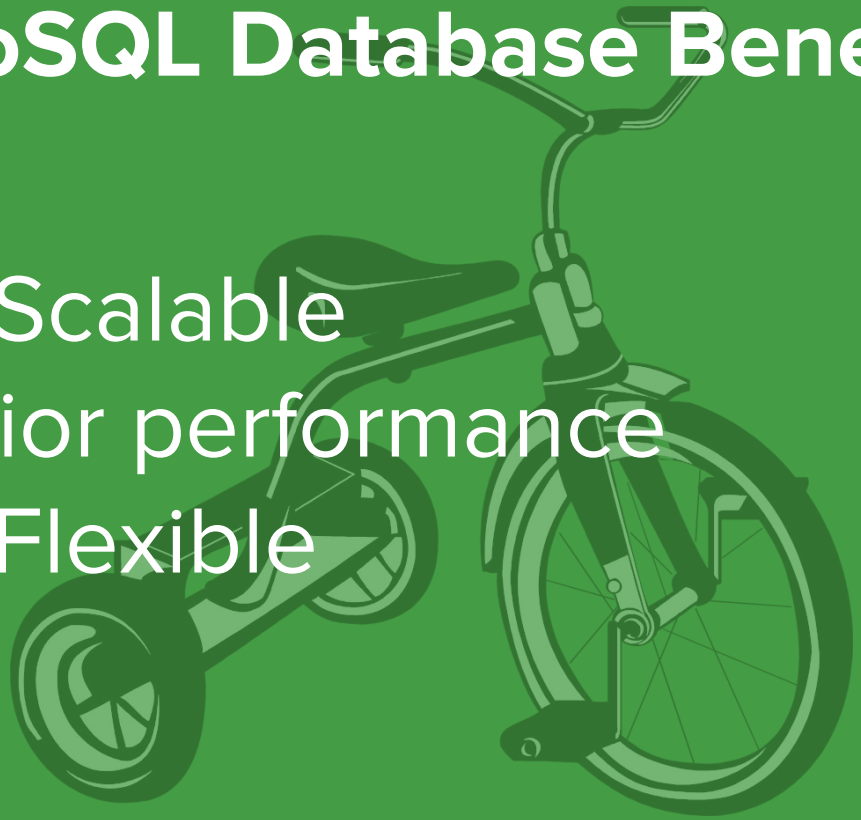
- A NoSQL (often interpreted as Not Only SQL) database provides a mechanism for storage and retrieval of data that is modeled in means other than the tabular relations used in relational databases. Motivations for this approach include simplicity of design, horizontal scaling and finer control over availability.

NoSQL Database Types



- Document Databases
- Graph stores
- Key-value stores
- Wide-column stores

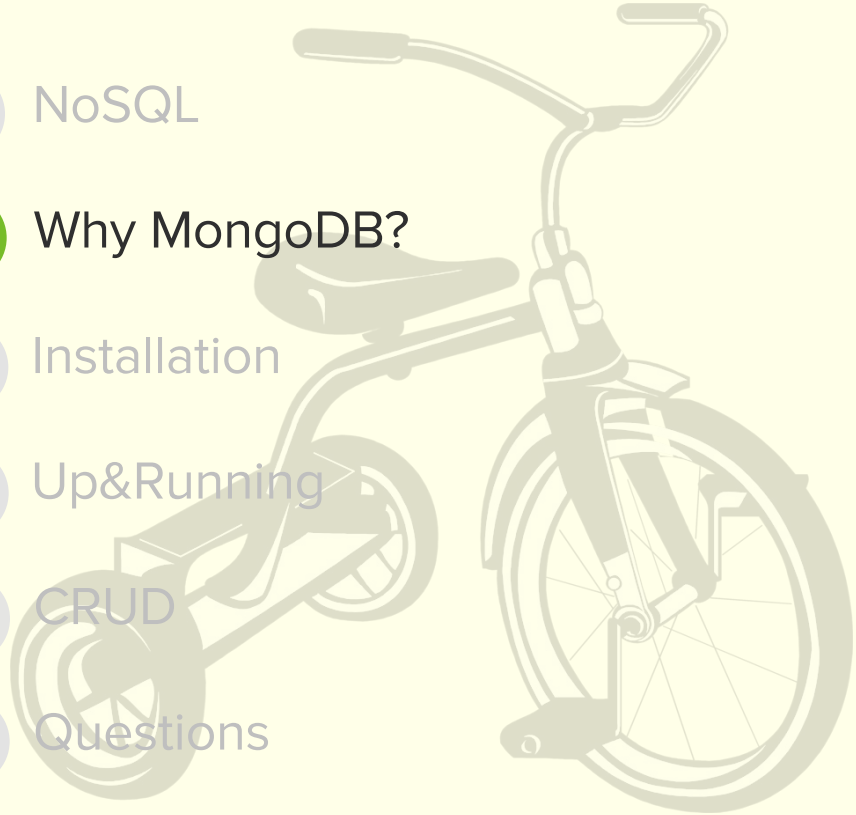
NoSQL Database Benefits



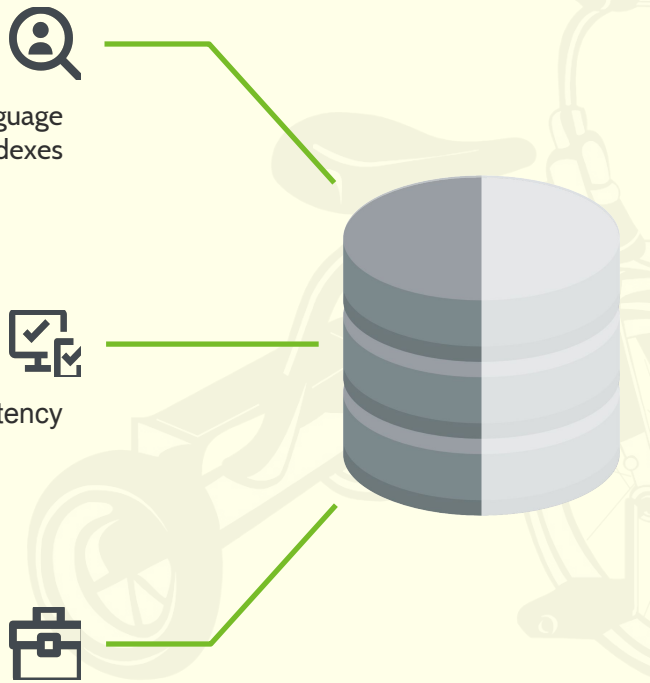
- More Scalable
- Superior performance
- More Flexible
- Agile

AGENDA

1. NoSQL
2. Why MongoDB?
3. Installation
4. Up&Running
5. CRUD
6. Questions



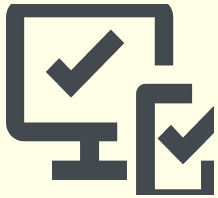
Relational


Expressive Query Language
& Secondary Indexes
Strong Consistency
Enterprise Management
& Integrations

Relational / Expressive Query Language & Secondary Indexes

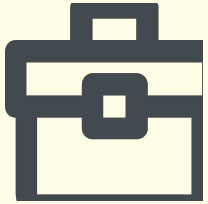
- Queries to access data in sophisticated ways
- Indexes for efficient access to data for reads/writes
- Table stakes for a database

Relational / Strong Consistency



- Provide the most up-to-date copy of the data
- Complicated to build an application around eventually consistent model, even for the most sophisticated teams

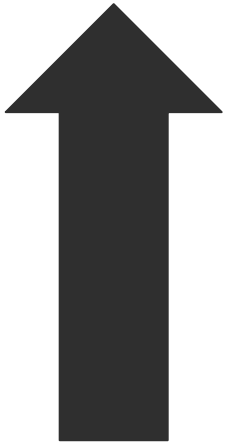
Relational / Enterprise Management & Integrations



Database is just one piece of the puzzle and it needs to fit into the enterprise IT stack:

- Streamline ops according to best practices
- Visualize data / schema to make informed decisions and ensure data integrity
- Encrypt data & restrict access to database in accordance with existing security infrastructure
- Employ standard BI and visualization tools

The World Has Changed

Data	Risk
	

Time	Budget
	

NoSQL

Expressive Query Language
& Secondary Indexes

Strong Consistency

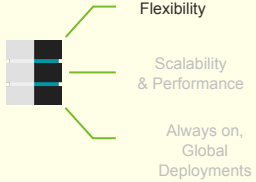
Enterprise Management
& Integrations

Flexibility

Scalability
& Performance

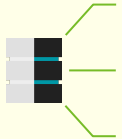
Always On,
Global Deployments

NoSQL / Flexibility



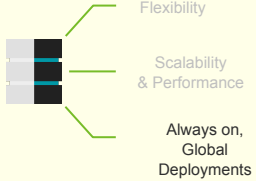
- Allows faster iteration
- Accommodates various data types
- Data models include document, key-value, wide column

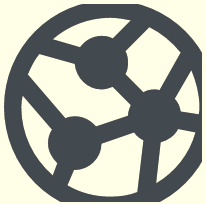
NoSQL / Scalability & Performance



- Scalability through partitioning / sharding
- NoSQL databases designed to deliver better performance than a relational database

NoSQL / Always On, Global Deployments

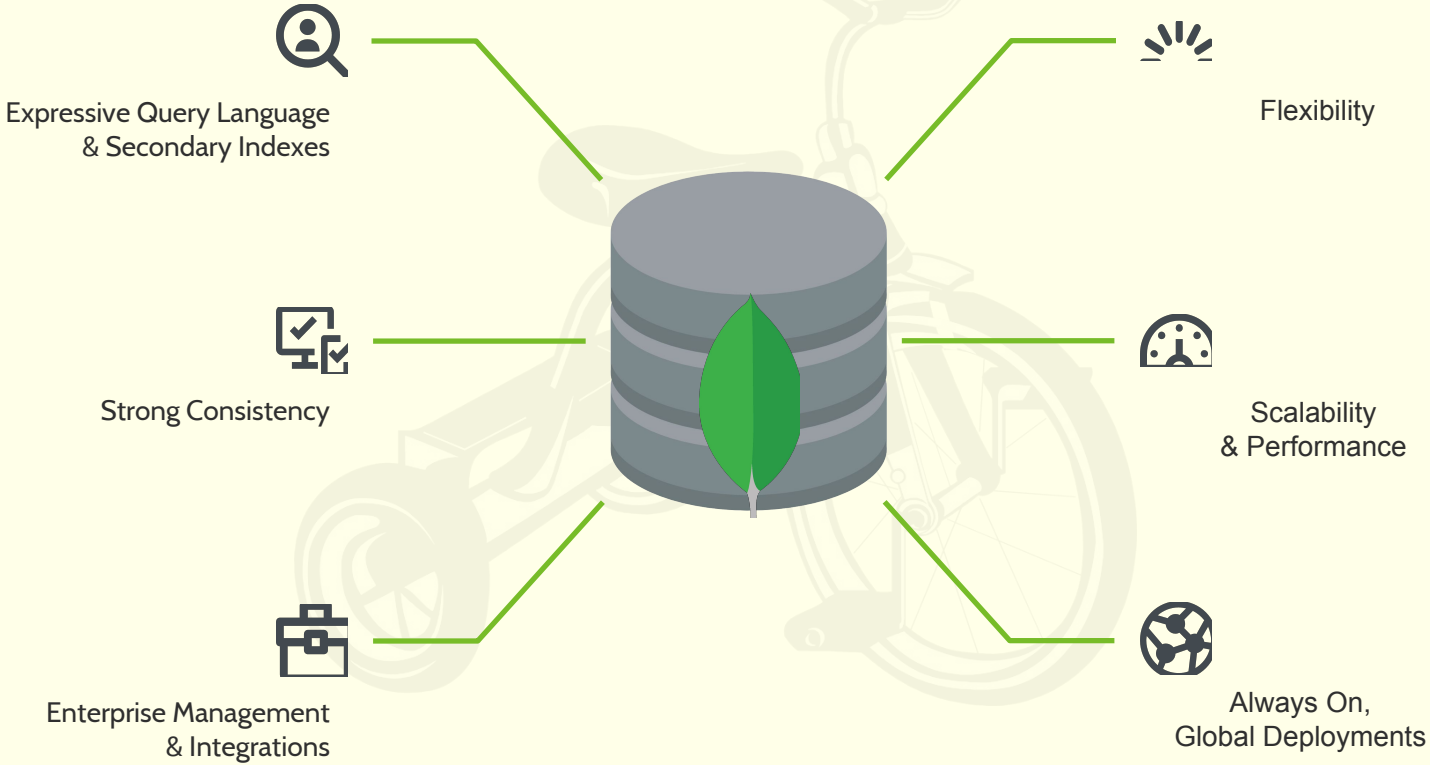




Support highly available systems that provide a consistent, high-quality experience to end users around the world with:

- Run across many computers
- Automatically synchronize data across servers, racks, data centers

Nexus Architecture



The Largest Ecosystem

15,000,000+ MongoDB Downloads	**35,000+** MongoDB User Group Members
350,000+ Online Education Registrants	**1,000+** Technology and Services Partners
40,000+ MongoDB Cloud Manager Users	**2,000+** Customers Across All Industries

Fortune 500 & Global 500

20 of the Top Financial Services Institutions

10 of the Top Electronics Companies

10 of the Top Media and Entertainment Companies

10 of the Top Retailers

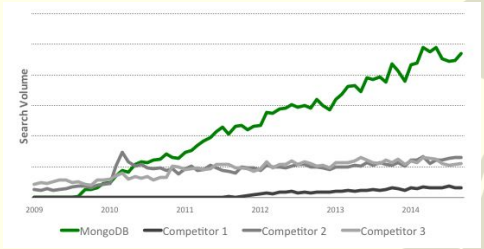
10 of the Top Telcos

10 of the Top Technology Companies

10 of the Top Healthcare Companies

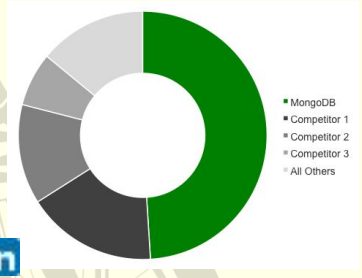
Fastest-Growing Database

Google



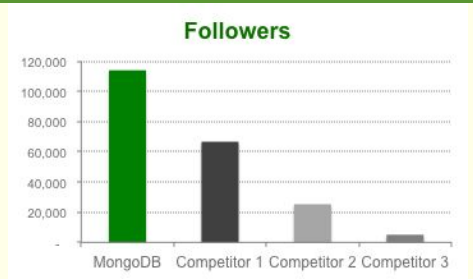
Google

LinkedIn

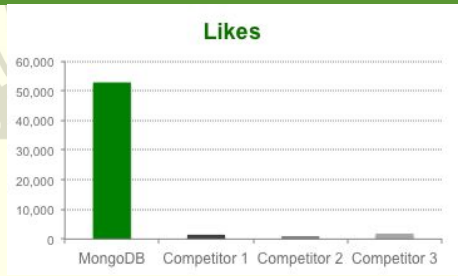


LinkedIn

Twitter



Facebook



4th Most Popular Database

Only non-relational in the top 5; 2.5x ahead of nearest NoSQL Competitor				
RANK	DBMS	MODEL	SCORE	GROWTH (20 MO)
1.	Oracle	Relational DBMS	1,442	-5%
2.	MySQL	Relational DBMS	1,294	2%
3.	Microsoft SQL Server	Relational DBMS	1,131	-10%
4.	**MongoDB**	**Document Store**	**277**	**172%**
5.	PostgreSQL	Relational DBMS	273	40%
6.	DB2	Relational DBMS	201	11%
7.	Microsoft Access	Relational DBMS	146	-26%
8.	Cassandra	Wide Column	107	87%
9.	SQLite	Relational DBMS	105	19%

Source: [DB-engines database popularity rankings: May 2015]

Terminology

RDBMS	MongoDB
Database →	Database
Table →	Collection
Index →	Index
Row →	Document
Column →	Field
Join →	Embedding & Linking

Document Model with Flexible Schema

Relational

PERSON

Pers_ID	Surname	First_Name	City
0	Miller	Paul	London
1	Ortega	Alvaro	Valencia
2	Huber	Urs	Zurich
3	Blanc	Gaston	Paris
4	Bertolini	Fabrizio	Rome

NO RELATION

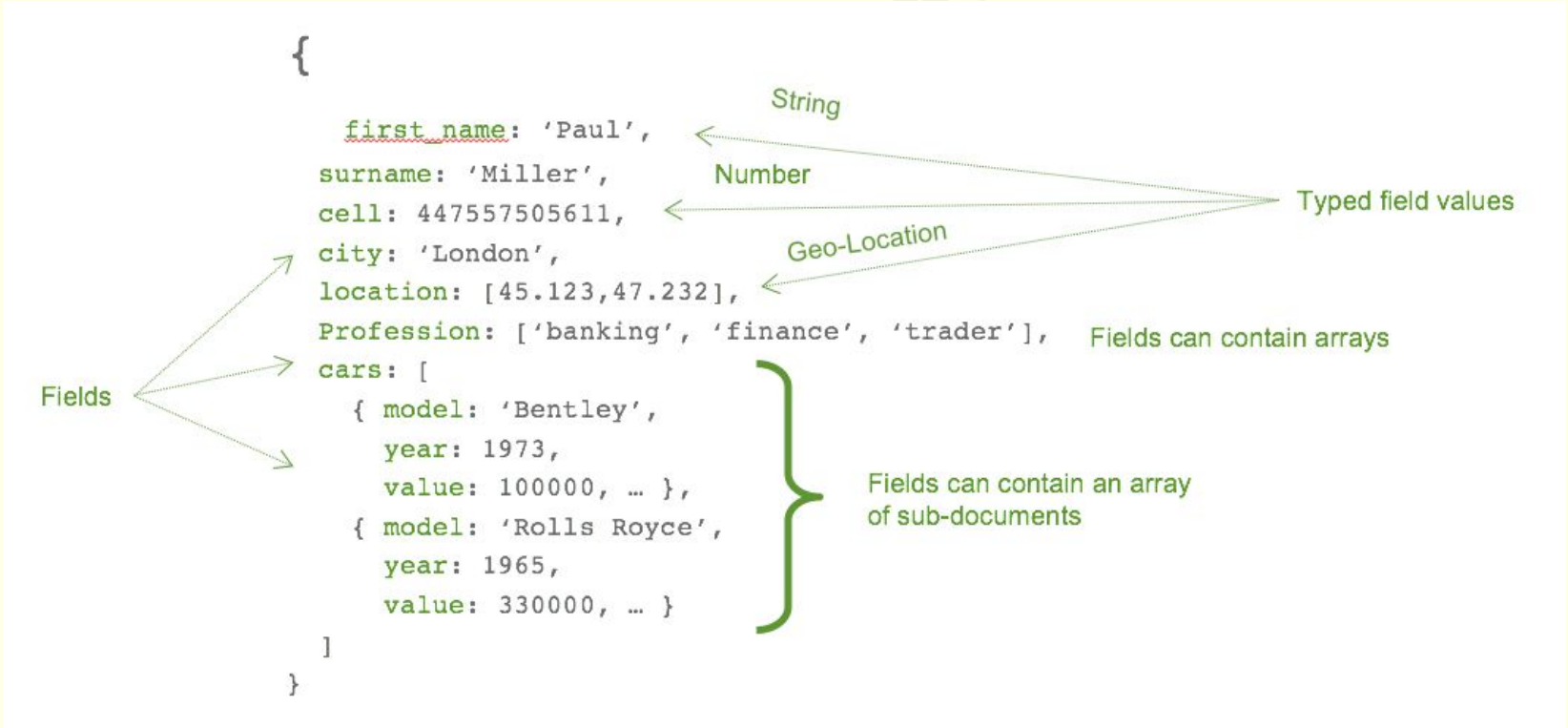
CAR

Car_ID	Model	Year	Value	Pers_ID
101	Bently	1973	100000	0
102	Rolls Royce	1965	330000	0
103	Peugeot	1993	500	3
104	Ferrari	2005	150000	4
105	Renault	1998	2000	3
106	Renault	2001	7000	3
107	Smart	1999	2000	2

MongoDB

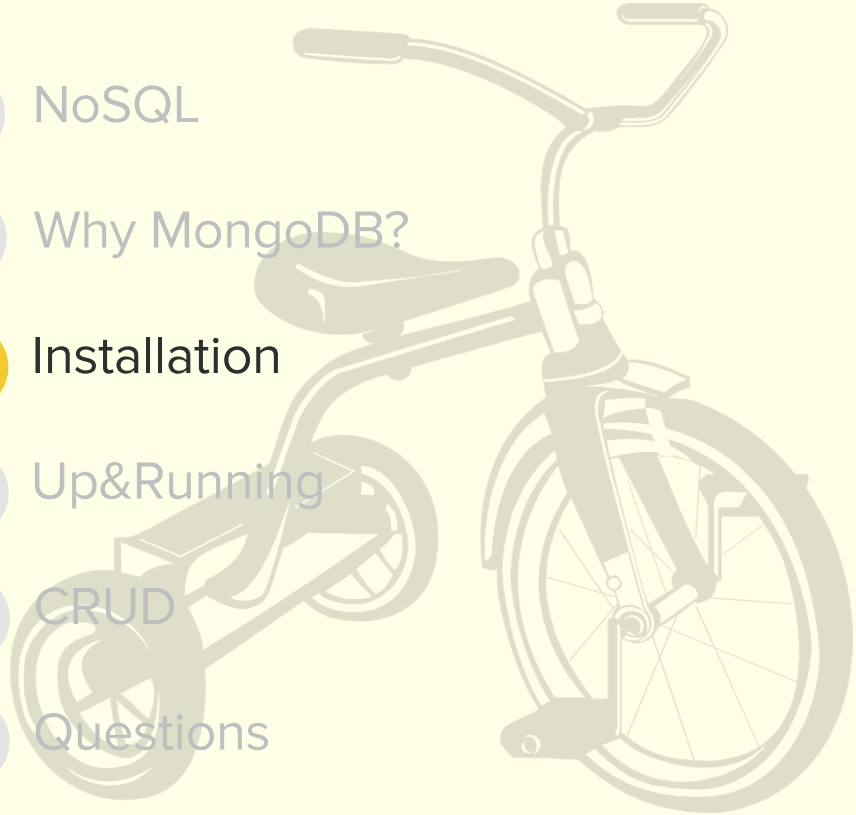
```
{
  first_name: 'Paul',
  surname: 'Miller',
  city: 'London',
  location: [45.123,47.232],
  cars: [
    { model: 'Bentley',
      year: 1973,
      value: 100000, … },
    { model: 'Rolls Royce',
      year: 1965,
      value: 330000, … }
  ]
}
```

Documents are Rich Data Structures



AGENDA

1. NoSQL
2. Why MongoDB?
3. Installation
4. Up&Running
5. CRUD
6. Questions



Installing MongoDB

Windows

- Change to the directory containing the .msi installation binary of your choice and invoke:

```
msiexec.exe /q /i
mongodb-win32-x86_64-2008plus-ssl-3.2.6-signed.msi ^
INSTALLLOCATION="C:\mongodb" ^ ADDLOCAL="all"
```

- You can specify the installation location for the executable by modifying the INSTALLLOCATION value.
- By default, this method installs all MongoDB binaries.

https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/

Installing MongoDB

OS X

- Change to the directory containing the .msi installation binary of your choice and invoke:

```
curl -O
https://fastdl.mongodb.org/osx/mongodb-osx-x86_64-3.2.6.tgz
tar -zxvf mongodb-osx-x86_64-3.2.6.tgz
mkdir -p mongodbcp -R -n mongodb-osx-x86_64-3.2.6/ mongodb
export PATH=<mongodb-install-directory>/bin:$PATH
```

- You can specify the installation location for the executable by modifying the INSTALLLOCATION value.
- By default, this method installs all MongoDB binaries.

https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/

Installing MongoDB

RedHat Variants

- Create the file: /etc/yum.repos.d/mongodb-org-2.6.repo and populate it with:

```
[mongodb-org-3.2]
name=MongoDB Repository
baseurl=https://repo.mongodb.org/yum/redhat/$releasever/mongodb-org/3.2/x86_64/
gpgcheck=1
enabled=1
gpgkey=https://www.mongodb.org/static/pgp/server-3.2.asc
```

- sudo yum install -y mongodb-org

https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/

Installing MongoDB

Manual Install

1. Download MongoDB
   a. http://mongodb.com/download
2. Open Archive
   a. Untar, Unzip, etc.
3. Create data directory
   a. mkdir, md
4. Start mongod server
   a. mongod --dbPath [dir] --logPath [dir]

Installing MongoDB

Components

- mongod – server daemon
- mongo - client

OS X

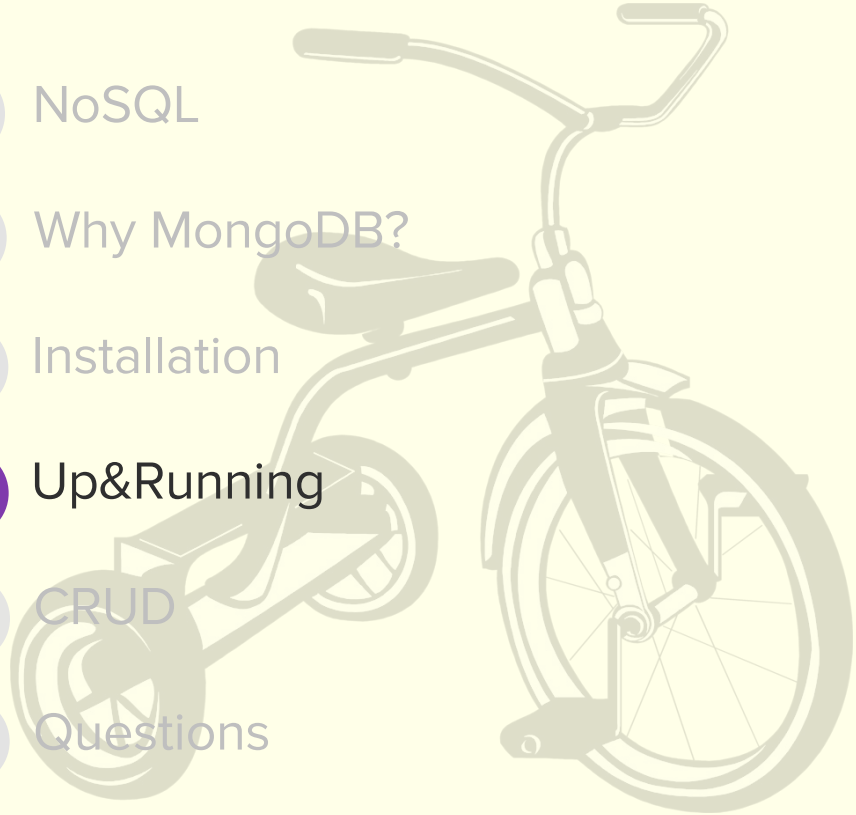
```
sudo mkdir -p /data/db
mongod
```

Windows

```
md \data\db
C:\mongodb\bin\mongod.exe
```

AGENDA

1. NoSQL
2. Why MongoDB?
3. Installation
4. Up&Running
5. CRUD
6. Questions



Take it for a spin

mongo - command line interface

Accepts javascript… example:

```
use foo

bar =
{ name: "baz",
  a_date: new Date(2016, 06, 02),
  categories: [“this”,”that”,”the other”],
  really: true
}

db.foo.save(bar)
```

Take it for a spin

Execute Javascript Code, too:

```
print('===== My Demo Script =====');

print(db.getCollectionNames()); // prints all collection names in respective database

print(db.events.count()); // prints the count of myCollection collection.
```

Load Some Data

https://raw.githubusercontent.com/mongodb/docs-assets/primer-dataset/primer-dataset.json

Shortlink:

http://bit.ly/28QFnCi

Import JSON

wget http://bit.ly/28QFnCi

mv 28????? restaurants.json

mongoimport restaurants.json

mongo localhost:27017/test

show dbs

show collections

View Imported Data

use test

db.restaurants.find()

db.restaurants.find().pretty()

db.restaurants.find({“borough”: “Queens”})

Find - Advanced

```
db.restaurants.find({
$or: [
        { ‘borough’: ‘Queens’ },
        { ‘address.zipcode’: ‘11369’ },
        { ‘address.zipcode’: ‘11368’ },
        { ‘address.zipcode’: ‘11367’ }
]}
```

Find - Conditional

```
db.restaurants.find( { "grades.score": { $gt: 30 } } )

db.restaurants.find( { "grades.score": { $lt: 10 } } )
```

Logical AND

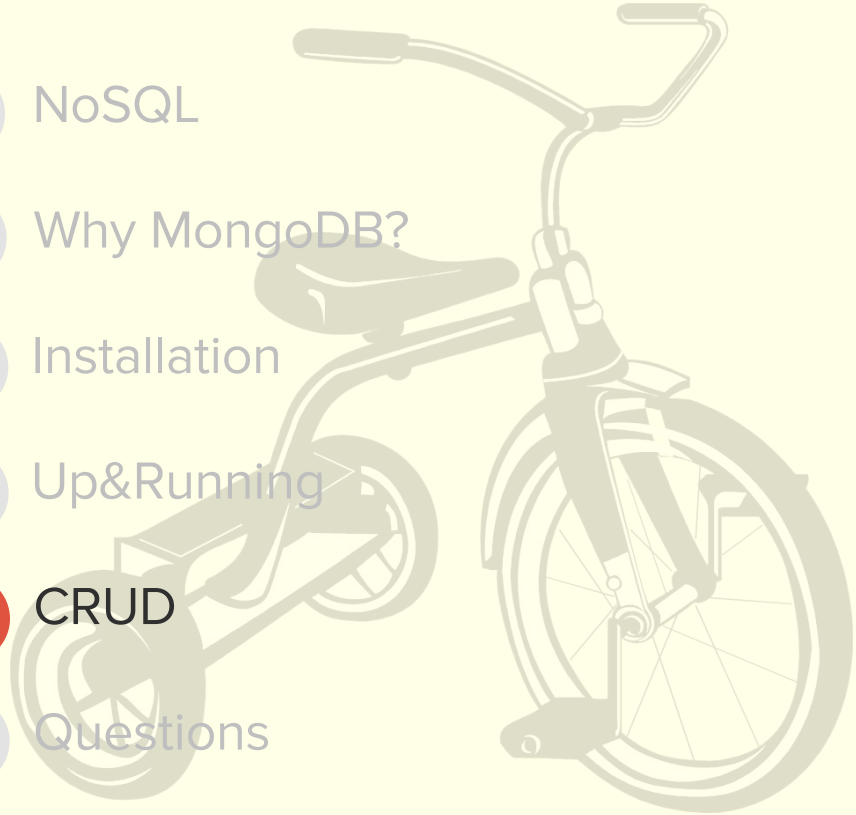
```
db.restaurants.find( { "cuisine": "Italian", "address.zipcode": "10075" } )
```

Logical OR

```
db.restaurants.find(
   { $or: [ { "cuisine": "Italian" }, { "address.zipcode": "10075" } ] }
)
```

AGENDA

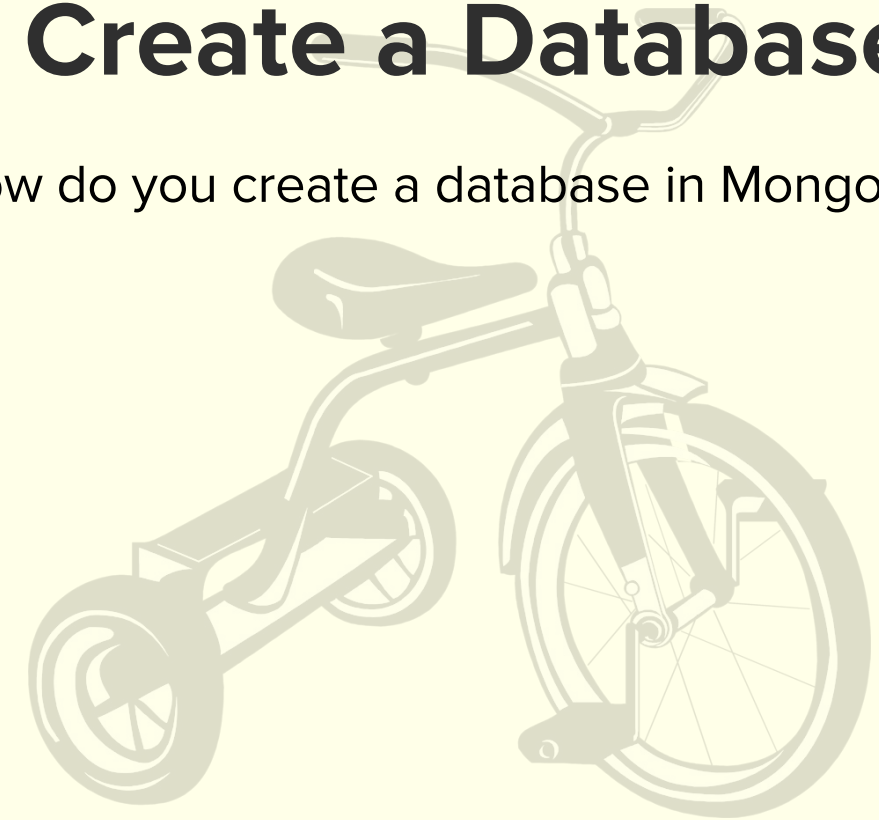
1. NoSQL
2. Why MongoDB?
3. Installation
4. Up&Running
5. CRUD
6. Questions




CRUD
Let’s get our hands dirty

Create a Database

How do you create a database in MongoDB???



Create a Database

WRONG! YOU DON'T CREATE THEM – YOU USE THEM!!

use dbname

Create a Database

How do I show what databases exist?

```
Michaels-MBP-3(mongod-3.2.1) test> show dbs
bulktest          → 0.078GB
cs5610            → 0.078GB
enferno           → 0.078GB
enron             → 0.953GB
```

Create a Database

How do we create and add data to MongoDB?

```
Michaels-MBP-3(mongod-3.2.1)> use test
switched to db test
Michaels-MBP-3(mongod-3.2.1) test> show collections
categories      → 0.000MB / 0.008MB
developers      → 0.000MB / 0.008MB
form            → 0.000MB / 0.008MB
forms           → 0.000MB / 0.008MB
log             → 0.001MB / 0.008MB
products        → 0.001MB / 0.008MB
system.indexes → 0.001MB / 0.008MB
users           → 0.000MB / 0.008MB
```

Create

How do we create and add data to MongoDB?

```
Michaels-MBP-3(mongod-3.2.1)> use mug
switched to db mug
Michaels-MBP-3(mongod-3.2.1) mug> show collections
```

Sets the database context

None exist yet!

Create

How do we create and add data to MongoDB?

```
Michaels-MBP-3(mongod-3.2.1) mug> db.fruit.insert({name: 'banana',
origin: 'south america'})
Inserted 1 record(s) in 239ms
WriteResult({  "nInserted": 1})
Michaels-MBP-3(mongod-3.2.1) mug> show collections
fruit          → 0.000MB / 0.008MB
system.indexes → 0.000MB / 0.008MB
Michaels-MBP-3(mongod-3.2.1) mug> db.fruit.find()
{  "_id": ObjectId("574ca9b307a43751574794d9"),  "name": "banana",
"origin": "south america"}
Fetched 1 record(s) in 4ms
```

_id created automatically

Create

How do we create and add data to MongoDB?

```
Michaels-MBP-3(mongod-3.2.1) mug> db.fruit.insert({name: 'banana',
origin: 'south america'})
Inserted 1 record(s) in 239ms
WriteResult({  "nInserted": 1})
Michaels-MBP-3(mongod-3.2.1) mug> show collections
fruit          → 0.000MB / 0.008MB
system.indexes → 0.000MB / 0.008MB
Michaels-MBP-3(mongod-3.2.1) mug> db.fruit.find()
{  "_id": ObjectId("574ca9b307a4375f574794d9"),  "name": "banana",
"origin": "south america"}
Fetched 1 record(s) in 4ms
```

_id created automatically

```
{
        "address" : {
                "building" : "82",
                "coord" : [
                        -73.856077,
                        40.848447
                ],
                "street" : "Mike Lynn Ave",
                "zipcode" : "19112"
        },
        "borough" : "Fooville",
        "cuisine" : "Seasnake",
        "grades" : [
                {
                        "date" : ISODate("2016-03-03T00:00:00Z"),
                        "grade" : "A",
                        "score" : 2
                },
                {
                        "date" : ISODate("2016-09-11T00:00:00Z"),
                        "grade" : "A",
                        "score" : 6
                },
                {
                        "date" : ISODate("2016-01-24T00:00:00Z"),
                        "grade" : "A",
                        "score" : 10
                },
                {
                        "date" : ISODate("2016-11-23T00:00:00Z"),
                        "grade" : "A",
                        "score" : 9
                },
                {
                        "date" : ISODate("2016-03-10T00:00:00Z"),
                        "grade" : "B",
                        "score" : 14
                }
        ],
        "name" : "Mike Lynn Park Seasnake Shop",
        "restaurant_id" : "3003992"
}
```


Read

To read a database we must first know the collection.

```
Michaels-MBP-3(mongod-3.2.1)> use test
switched to db test
Michaels-MBP-3(mongod-3.2.1) test> show collections
categories     → 0.000MB / 0.008MB
developers     → 0.000MB / 0.008MB
form           → 0.000MB / 0.008MB
forms          → 0.000MB / 0.008MB
log            → 0.001MB / 0.008MB
products       → 0.001MB / 0.008MB
system.indexes → 0.001MB / 0.008MB
users          → 0.000MB / 0.008MB
```

Read

To read a database we must first know the collection.

```
Michaels-MBP-3(mongod-3.2.1)> use test
> db.restaurants.findOne()
{
        "_id" : ObjectId("576ade9108b3e2f7775c674a"),
        "address" : {
                "building" : "1007",
                "coord" : [
                        -73.856077,
                        40.848447
                ],
                "street" : "Morris Park Ave",
                "zipcode" : "10462"
        },
        "borough" : "Bronx",
        "cuisine" : "Bakery",
        "grades" : [
                {
                        "date" : ISODate("2014-03-03T00:00:00Z"),
                        "grade" : "A",
                        "score" : 2
                },
                {
                        "date" : ISODate("2013-09-11T00:00:00Z"),
                        "grade" : "A",
                        "score" : 6
                },
                {
                        "date" : ISODate("2013-01-24T00:00:00Z"),
                        "grade" : "A",
                        "score" : 10
                },
                {
                        "date" : ISODate("2011-11-23T00:00:00Z"),
```

Read

Restricting - or “Projecting” fields we want to see

```
Michaels-MBP-3(mongod-3.2.1)> > db.restaurants.find({},{'_id':0,'address':1,'cuisine':1})
{ "address" : { "building" : "1007", "coord" : [ -73.856077, 40.848447 ], "street" : "Morris Park Ave", "zipcode" : "10462" },
"cuisine" : "Bakery" }
{ "address" : { "building" : "469", "coord" : [ -73.961704, 40.662942 ], "street" : "Flatbush Avenue", "zipcode" : "11225" },
"cuisine" : "Hamburgers" }
{ "address" : { "building" : "2780", "coord" : [ -73.98241999999999, 40.579505 ], "street" : "Stillwell Avenue", "zipcode" : "11224"
}, "cuisine" : "American " }
{ "address" : { "building" : "1269", "coord" : [ -73.871194, 40.6730975 ], "street" : "Sutter Avenue", "zipcode" : "11208" },
"cuisine" : "Chinese" }
{ "address" : { "building" : "705", "coord" : [ -73.9653967, 40.6064339 ], "street" : "Kings Highway", "zipcode" : "11223" },
"cuisine" : "Jewish/Kosher" }
{ "address" : { "building" : "1", "coord" : [ -73.96926909999999, 40.7685235 ], "street" : "East   66 Street", "zipcode" : "10065" },
"cuisine" : "American " }
{ "address" : { "building" : "203", "coord" : [ -73.97822040000001, 40.6435254 ], "street" : "Church Avenue", "zipcode" : "11218" },
"cuisine" : "Ice Cream, Gelato, Yogurt, Ices" }
{ "address" : { "building" : "265-15", "coord" : [ -73.7032601, 40.7386417 ], "street" : "Hillside Avenue", "zipcode" : "11004" },
"cuisine" : "Ice Cream, Gelato, Yogurt, Ices" }
{ "address" : { "building" : "6909", "coord" : [ -74.0259567, 40.6353674 ], "street" : "3 Avenue", "zipcode" : "11209" }, "cuisine" :
"Delicatessen" }
{ "address" : { "building" : "284", "coord" : [ -73.9829239, 40.6580753 ], "street" : "Prospect Park West", "zipcode" : "11215" },
"cuisine" : "American " }
{ "address" : { "building" : "129-08", "coord" : [ -73.839297, 40.78147 ], "street" : "20 Avenue", "zipcode" : "11356" }, "cuisine" :
"Delicatessen" }
```

Update

Modify Information in a Document

```
db.collection.update(
   <query>,
   <update>,
   {
     upsert: <boolean>,
     multi: <boolean>,
     writeConcern: <document>
   }
)
```

Update

Modify Information in a Document

```
db.fruit.update(
    {'name':'banana'},
    {'origin': 'botswana'
)
```

Delete

Two Primary Ways

```
db.collection.deleteOne(
    {filter},
    {write concern}
)

db.collection.deleteMany(
    {filter},
    {write concern}
)
```

Delete

Two Primary Ways

```
> db.restaurants.deleteOne({'cuisine': 'Seasnake'})
{ "acknowledged" : true, "deletedCount" : 1 }
```

Extra Credit

Application Development

```
python get-pip.py


#--
from pymongo import MongoClient

client = MongoClient()
db = client.test
coll = db.restaurants

for c in coll.find():
    print c
#---

$ python extra.py
{u'cuisine': u'Other', u'borough': u'Brooklyn', u'name': u'Luigis Pizzeria', u'restaurant_id': u'50018950', u'grades': [], u'address': {u'building': u'326', u'street': u'Dekalb
Ave', u'zipcode': u'11205', u'coord': [-73.9653622, 40.6897478]}, u'_id': ObjectId('576ade9208b3e2f7775cca47')}
{u'cuisine': u'Other', u'borough': u'Brooklyn', u'name': u'Northside Bakery', u'restaurant_id': u'50018952', u'grades': [], u'address': {u'building': u'190', u'street': u'Nassau
Ave', u'zipcode': u'11222', u'coord': [-73.9454721, 40.7253377]}, u'_id': ObjectId('576ade9208b3e2f7775cca48')}
{u'cuisine': u'Other', u'borough': u'Staten Island', u'name': u"Christine'S Restaurant", u'restaurant_id': u'50018953', u'grades': [], u'address': {u'building': u'307', u'street':
u'Nelson Ave', u'zipcode': u'10308', u'coord': [-74.1453806, 40.5422782]}, u'_id': ObjectId('576ade9208b3e2f7775cca49')}
```

Installing MongoDB

```
$ curl -O https://fastdl.mongodb.org/osx/mongodb-osx-x86_64-3.2.6.tgz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 60.9M  100 60.9M    0     0  2730k      0  0:00:22  0:00:22 --:--:-- 1589k
$ tar xzvf mongodb-osx-x86_64-3.2.6.tgz
x mongodb-osx-x86_64-3.2.6/README
x mongodb-osx-x86_64-3.2.6/THIRD-PARTY-NOTICES
x mongodb-osx-x86_64-3.2.6/MPL-2
x mongodb-osx-x86_64-3.2.6/GNU-AGPL-3.0
x mongodb-osx-x86_64-3.2.6/bin/mongodump
x mongodb-osx-x86_64-3.2.6/bin/mongorestore
x mongodb-osx-x86_64-3.2.6/bin/mongoexport
x mongodb-osx-x86_64-3.2.6/bin/mongoimport
x mongodb-osx-x86_64-3.2.6/bin/mongostat
x mongodb-osx-x86_64-3.2.6/bin/mongotop
x mongodb-osx-x86_64-3.2.6/bin/bsondump
x mongodb-osx-x86_64-3.2.6/bin/mongofiles
x mongodb-osx-x86_64-3.2.6/bin/mongooplog
x mongodb-osx-x86_64-3.2.6/bin/mongoperf
x mongodb-osx-x86_64-3.2.6/bin/mongosniff
x mongodb-osx-x86_64-3.2.6/bin/mongod
x mongodb-osx-x86_64-3.2.6/bin/mongos
x mongodb-osx-x86_64-3.2.6/bin/mongo
$ ln -s mongodb-osx-x86_64-3.2.6 mongodb
```

Running Mongod

```
JD10Gen:mongodb jdrumgoole$ ./bin/mongod --dbpath /data/b2b
2016-05-23T19:21:07.767+0100 I CONTROL  [initandlisten] MongoDB starting : pid=49209 port=27017 dbpath=/data/b2b
64-bit host=JD10Gen.local
2016-05-23T19:21:07.768+0100 I CONTROL  [initandlisten] db version v3.2.6
2016-05-23T19:21:07.768+0100 I CONTROL  [initandlisten] git version: 05552b562c7a0b3143a729aaa0838e558dc49b25
2016-05-23T19:21:07.768+0100 I CONTROL  [initandlisten] allocator: system
2016-05-23T19:21:07.768+0100 I CONTROL  [initandlisten] modules: none
2016-05-23T19:21:07.768+0100 I CONTROL  [initandlisten] build environment:
2016-05-23T19:21:07.768+0100 I CONTROL  [initandlisten]     distarch: x86_64
2016-05-23T19:21:07.768+0100 I CONTROL  [initandlisten]     target_arch: x86_64
2016-05-23T19:21:07.768+0100 I CONTROL  [initandlisten] options: { storage: { dbPath: "/data/b2b" } }
2016-05-23T19:21:07.769+0100 I -        [initandlisten] Detected data files in /data/b2b created by the 'wiredTiger'
storage engine, so setting the active storage engine to 'wiredTiger'.
2016-05-23T19:21:07.769+0100 I STORAGE  [initandlisten] wiredtiger_open config:
create,cache_size=4G,session_max=20000,eviction=(threads_max=4),config_base=false,statistics=(fast),log=(enabled=true,
archive=true,path=journal,compressor=snappy),file_manager=(close_idle_time=100000),checkpoint=(wait=60,log_size=2GB),s
tatistics_log=(wait=0),
2016-05-23T19:21:08.837+0100 I CONTROL  [initandlisten]
2016-05-23T19:21:08.838+0100 I CONTROL  [initandlisten] ** WARNING: soft rlimits too low. Number of files is 256,
should be at least 1000
2016-05-23T19:21:08.840+0100 I NETWORK  [HostnameCanonicalizationWorker] Starting hostname canonicalization worker
2016-05-23T19:21:08.840+0100 I FTDC     [initandlisten] Initializing full-time diagnostic data capture with directory
'/data/b2b/diagnostic.data'
2016-05-23T19:21:08.841+0100 I NETWORK  [initandlisten] waiting for connections on port 27017
2016-05-23T19:21:09.148+0100 I NETWORK  [initandlisten] connection accepted from 127.0.0.1:59213 #1 (1 connection now
open)
```

Connecting Via The Shell

```
$ ./bin/mongo
MongoDB shell version: 3.2.6
connecting to: test
Server has startup warnings:
2016-05-17T11:46:03.516+0100 I CONTROL  [initandlisten]
2016-05-17T11:46:03.516+0100 I CONTROL  [initandlisten] ** WARNING: soft rlimits too low.
Number of files is 256, should be at least 1000
>
```

Inserting your first record

```
> show databases
local  0.000GB
> use test
switched to db test
> show databases
local  0.000GB
> db.demo.insert( { "key" : "value" } )
WriteResult({ "nInserted" : 1 })
> show databases
local  0.000GB
test   0.000GB
> show collections
demo
> db.demo.findOne()
{ "_id" : ObjectId("573af7085ee4be80385332a6"), "key" : "value" }
>
```

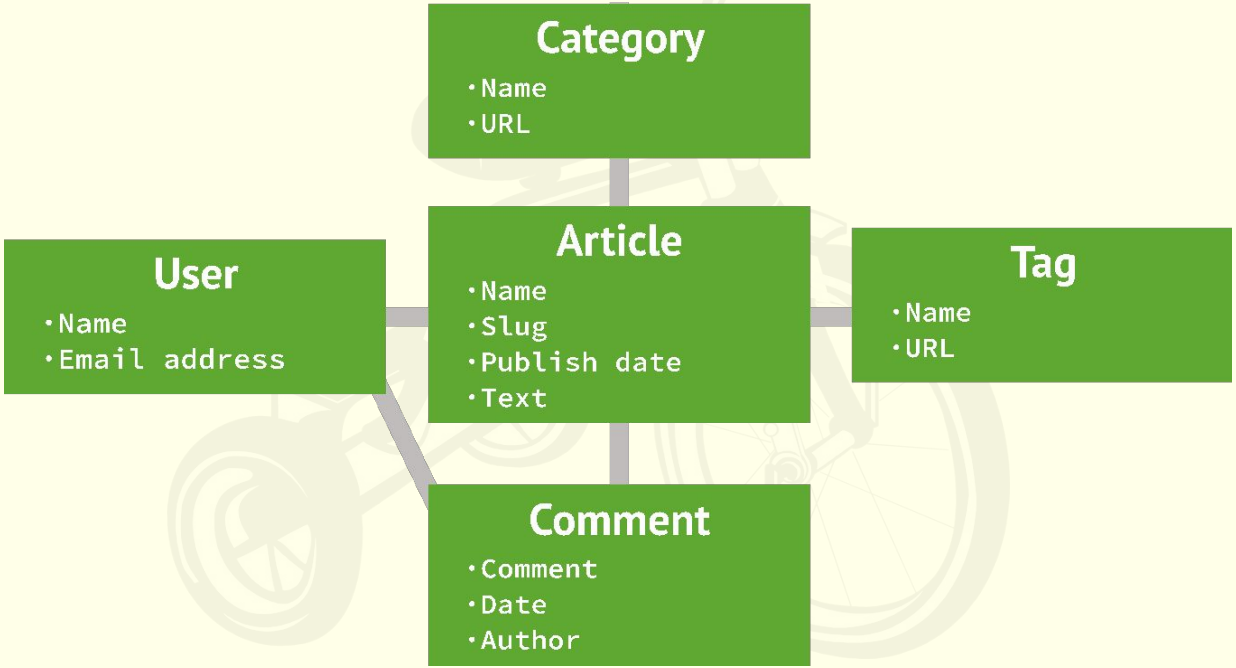
Object ID

```
573af7085ee4be80385332a6
TS-------ID----PID-Count-
```

A Simple Blog Application

- Lets create a blogging application with:
  - Articles
  - Users
  - Comments

Typical Entity Relation Diagram



In MongoDB we can build organically

```
> use blog
switched to db blog
> db.users.insert( { "username" : "jdrumgoole", "password" : "top secret", "lang" : "EN"
} )
WriteResult({ "nInserted" : 1 })
> db.users.findOne()
{
     "_id" : ObjectId("573afff65ee4be80385332a7"),
     "username" : "jdrumgoole",
     "password" : "top secret",
     "lang" : "EN"
}
```

How do we do this in a program?

```
'''
Created on 17 May 2016

@author: jdrumgoole
'''
import pymongo

#
# client defaults to localhost and port 27017. eg MongoClient('localhost', 27017)
client  = pymongo.MongoClient()
blogDatabase = client[ "blog" ]
usersCollection = blogDatabase[ "users" ]

usersCollection.insert_one( { "username" : "jdrumgoole",
                              "password" : "top secret",
                              "lang" : "EN" })

user = usersCollection.find_one()

print( user )
```

{ "name" : "Philadelphia MongoDB User Group" }

Next up Articles

```
...
articlesCollection = blogDatabase[ "articles" ]

author = "jdrumgoole"

article = { "title"  : "This is my first post",
            "body"   : "The is the longer body text for my blog post. We can add lots of text
here.",
            "author" : author,
            "tags"   : [ "joe", "general", "Ireland", "admin" ]
          }

#
# Lets check if our author exists
#

if usersCollection.find_one( { "username" : author }) :
    articlesCollection.insert_one( article )
else:
    raise ValueError( "Author %s does not exist" % author )
```

Create a new type of article

```
#
# Lets add a new type of article with a posting date and a section
#
author = "jdrumgoole"
title  = "This is a post on MongoDB"

newPost = { "title"    : title,
            "body"     : "MongoDB is the worlds most popular NoSQL database. It is a document
database",
            "author"   : author,
            "tags"     : [ "joe", "mongodb", "Ireland" ],
            "section"  : "technology",
            "postDate" : datetime.datetime.now(),
          }

#
# Lets check if our author exists
#

if usersCollection.find_one( { "username" : author }) :
    articlesCollection.insert_one( newPost )
```

{ "name" : "Philadelphia MongoDB User Group" }

Make a lot of articles 1

```
import pymongo
import string
import datetime
import random

def randomString( size, letters = string.letters ):
    return "".join( [random.choice( letters )  for _ in xrange( size )] )

client  = pymongo.MongoClient()

def makeArticle( count, author, timestamp ):

    return { "_id"      : count,
             "title"    : randomString( 20 ),
             "body"     : randomString( 80 ),
             "author"   : author,
             "postdate" : timestamp }


def makeUser( username ):
    return { "username" : username,
             "password" : randomString( 10 ) ,
             "karma" : random.randint( 0, 500 ),
             "lang" : "EN" }
```

Make a lot of articles 2

```
blogDatabase = client[ "blog" ]
usersCollection = blogDatabase[ "users" ]
articlesCollection = blogDatabase[ "articles" ]

bulkUsers = usersCollection.initialize_ordered_bulk_op()
bulkArticles = articlesCollection.initialize_ordered_bulk_op()

ts = datetime.datetime.now()

for i in range( 1000000 ) :
    #username = randomString( 10, string.ascii_uppercase ) + "_" + str( i )
    username = "USER_" + str( i )
    bulkUsers.insert( makeUser( username ) )

    ts = ts + datetime.timedelta( seconds =  1 )
    bulkArticles.insert( makeArticle( i, username, ts ))

    if ( i % 500 == 0 ) :
        bulkUsers.execute()
        bulkArticles.execute()
        bulkUsers = usersCollection.initialize_ordered_bulk_op()
        bulkArticles = articlesCollection.initialize_ordered_bulk_op()

bulkUsers.execute()
bulkArticles.execute()
```

Find a User

```
> db.users.findOne()
{
    "_id" : ObjectId("5742da5bb26a88bc00e941ac"),
    "username" : "FLFZQLSRWZ_0",
    "lang" : "EN",
    "password" : "vTlILbGWLt",
    "karma" : 448
}
> db.users.find( { "username" : "VHXDAUUFJW_45" } ).pretty()
{
    "_id" : ObjectId("5742da5bb26a88bc00e94206"),
    "username" : "VHXDAUUFJW_45",
    "lang" : "EN",
    "password" : "GmRLnCeKVp",
    "karma" : 284
}
```

Find Users with high Karma

```
> db.users.find( { "karma" : { $gte : 450 }} ).pretty()
{
     "_id" : ObjectId("5742da5bb26a88bc00e941ae"),
     "username" : "JALLFRKBWD_1",
     "lang" : "EN",
     "password" : "bCSKSKvUeb",
     "karma" : 487
}
{
     "_id" : ObjectId("5742da5bb26a88bc00e941e4"),
     "username" : "OTKWJJBNBU_28",
     "lang" : "EN",
     "password" : "HAWpiATCBN",
     "karma" : 473
}
{

...
```

Using projection

```
> db.users.find( { "karma" : { $gte : 450 }}, { "_id" : 0, username : 1, karma : 1 } )
{ "username" : "JALLFRKBWD_1", "karma" : 487 }
{ "username" : "OTKWJJBNBU_28", "karma" : 473 }
{ "username" : "RVVHLKTWHU_31", "karma" : 493 }
{ "username" : "JBNESEOOEP_48", "karma" : 464 }
{ "username" : "VSTBDZLKQQ_51", "karma" : 487 }
{ "username" : "UKYDTQJCLO_61", "karma" : 493 }
{ "username" : "HZFZZMZHYB_106", "karma" : 493 }
{ "username" : "AAYLPJJNHO_113", "karma" : 455 }
{ "username" : "CXZZMHLBXE_128", "karma" : 460 }
{ "username" : "KKJXBACBVN_134", "karma" : 460 }
{ "username" : "PTNTIBGAJV_165", "karma" : 461 }
{ "username" : "PVLCQJIGDY_169", "karma" : 463 }
```

Update an Article to Add Comments 1

```
> db.articles.find( { "_id" : 19 } ).pretty()
{
    "_id" : 19,
    "body" :
"nTzOofOcnHKkJxpjKAyqTTnKZMFzzkWFeXtBRuEKsctuGBgWIrEBrYdvFIVHJWaXLUTVUXblO
ZZgUqWu",
    "postdate" : ISODate("2016-05-23T12:02:46.830Z"),
    "author" : "ASWTOMMABN_19",
    "title" : "CPMaqHtAdRwLXhlUvsej"
}
> db.articles.update( { _id : 18 }, { $set : { comments : [] }} )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Update an article to add Comments 2

```
> db.articles.find( { _id :18 } ).pretty()
{
    "_id" : 18,
    "body" :
"KmwFSIMQGcIsRNTDBFPuclwcVJkoMcrIPwTiSZDYyatoKzeQiKvJkiVSrndXqrALVIYZxGpaM
jucgXUV",
    "postdate" : ISODate("2016-05-23T16:04:39.497Z"),
    "author" : "USER_18",
    "title" : "wTLreIEyPfovEkBhJZZe",
    "comments" : [ ]
}
>
```

Update an Article to Add Comments 3

```
> db.articles.update( { _id : 18 }, { $push : { comments : { username : "joe",
comment : "hey first post" }}} )
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })

> db.articles.find( { _id :18 } ).pretty()
{
     "_id" : 18,
     "body" :
"KmwFSIMQGcIsRNTDBFPuclwcVJkoMcrIPwTiSZDYyatoKzeQiKvJkiVSrndXqrALVIYZxGpaMjucgXUV",
     "postdate" : ISODate("2016-05-23T16:04:39.497Z"),
     "author" : "USER_18",
     "title" : "wTLreIEyPfovEkBhJZZe",
     "comments" : [
          {
               "username" : "joe",
               "comment" : "hey first post"
          }
     ]
>
```

Delete an Article

```
> db.articles.remove( { "_id" : 25 } )
WriteResult({ "nRemoved" : 1 })
> db.articles.remove( { "_id" : 25 } )
WriteResult({ "nRemoved" : 0 })
> db.articles.remove( { "_id" : { $lte : 5  }} )
WriteResult({ "nRemoved" : 6 })
```

- Deletion leaves holes
- Dropping a collection is cheaper than deleting a large collection element by element

A Quick Look at Users and Articles Again

```
> db.users.findOne()
{
     "_id" : ObjectId("57431c07b26a88bf060e10cb"),
     "username" : "USER_0",
     "lang" : "EN",
     "password" : "kGIxPxqKGJ",
     "karma" : 266
}
> db.articles.findOne()
{
     "_id" : 0,
     "body" :
"hvJLnrrfZQurmtjPfUWbMhaQWbNjXLzjpuGLZjsxHXbUycmJVZTeOZesTnZtojThrebRcUoiYwivjpwG",
     "postdate" : ISODate("2016-05-23T16:04:39.246Z"),
     "author" : "USER_0",
     "title" : "gpNIoPxpfTAxWjzAVoTJ"
>
```

Find a User

```
> db.users.find( { "username" : "ABOXHWKBYS_199" } ).explain()
{
        "queryPlanner" : {
                "plannerVersion" : 1,
                "namespace" : "blog.users",
                "indexFilterSet" : false,
                "parsedQuery" : {
                        "username" : {
                                "$eq" : "ABOXHWKBYS_199"
                        }
                },
                "winningPlan" : {
                        "stage" : "COLLSCAN",
                        "filter" : {
                                "username" : {
                                        "$eq" : "ABOXHWKBYS_199"
                                }
                        },
                        "direction" : "forward"
                },
                "rejectedPlans" : [ ]
        },
        "serverInfo" : {
                "host" : "JD10Gen.local",
                "port" : 27017,
                "version" : "3.2.6",
                "gitVersion" : "05552b562c7a0b3143a729aaa0838e558dc49b25"
        },
        "ok" : 1
}
```

Find a User – Execution Stats

```
> db.users.find( {"username" : "USER_999999" } ).explain( "executionStats" ).executionStats
{
        "executionSuccess" : true,
        "nReturned" : 1,
        "executionTimeMillis" : 433,
        "totalKeysExamined" : 0,
        "totalDocsExamined" : 1000000,
        "executionStages" : {
                "stage" : "COLLSCAN",
                "filter" : {
                        "username" : {
                                "$eq" : "USER_999999"
                        }
                },
                "nReturned" : 1,
                "executionTimeMillisEstimate" : 330,
                "works" : 1000002,
                "advanced" : 1,
                "needTime" : 1000000,
                "needYield" : 0,
                "saveState" : 7812,
                "restoreState" : 7812,
                "isEOF" : 1,
                "invalidates" : 0,
                "direction" : "forward",
                "docsExamined" : 1000000
```

We need an index

```
> db.users.createIndex( { username : 1 } )
{
    "createdCollectionAutomatically" : false,
    "numIndexesBefore" : 1,
    "numIndexesAfter" : 2,
    "ok" : 1
}
>
```

Indexes Overview

- **Parameters**
  - **Background** : Create an index in the background as opposed to locking the database
  - **Unique** : All keys in the collection must be unique. Duplicate key insertions will be rejected with an error.
  - **Name** : explicitly name an index. Otherwise the index name is autogenerated from the index field.
- Deleting an Index
  - `db.users.dropIndex({ "username" : 1 })`
- Get All the Indexes on a collection
  - `db.users.getIndexes()`

Query Plan Execution Stages

- **COLLSCAN** : for a collection scan
- **IXSCAN** : for scanning index keys
- **FETCH** : for retrieving documents
- **SHARD_MERGE** : for merging results from shards

Add an Index

```
> db.users.find( {"username" : "USER_999999"} ).explain("executionStats").executionStats
{
      "executionSuccess" : true,
      "nReturned" : 1,
      "executionTimeMillis" : 0,
      "totalKeysExamined" : 1,
      "totalDocsExamined" : 1,


...
```

Execution Stage

```
"executionStages" : {
        "stage" : "FETCH",
        "nReturned" : 1,
        "executionTimeMillisEstimate" : 0,
        "docsExamined" : 1,,
        "inputStage" : {
                "stage" : "IXSCAN",
                "nReturned" : 1,
                "executionTimeMillisEstimate" : 0,
                "keyPattern" : {
                        "username" : 1
                },
                "indexName" : "username_1",
                "isMultiKey" : false,
                "isUnique" : false,
                "isSparse" : false,
                "isPartial" : false,
                "indexVersion" : 1,
                "direction" : "forward",
                "indexBounds" : {
                        "username" : [
                                "[\"USER_999999\", \"USER_999999\"]"
                        ]
                },
                "keysExamined" : 1,
                "seenInvalidated" : 0
        }
}
```

What we have learned

- How to create a database and a collection
- How to insert content into that collection
- How to query the collection
- How to update a document in place
- How to delete a document
- How to check the efficiency of an operation
- How to add an index
- How to check an index is being used in an operation

Next Webinar : Thinking in Documents

- Rather than normalisation we look at a hybrid approach to schema that provides a coherent mapping between objects in your application and objects in your database.
- Then we will optimise that schema for query purposes based on the pattern of queries we expect to see.
- Finally we will show how dynamic schema and schema validation allow you to extend your schema in a controlled fashion

14-June-2016, 14:00 GMT.

AGENDA

1. NoSQL
2. Why MongoDB?
3. Installation
4. Up&Running
5. CRUD
6. Questions