# Introduction to Database Systems: CS312 Editing Documents

Oliver Bonham-Carter

2 Nov 2020

# Getting started with Mongo in Docker

Start MongoDB
Windows
MacOS and Linux
Modifying Documents
The Coolest Thing
Comparing Values
Parsing Text
Consider This ...

**Create a Docker container from DockerHub for Mongo: Windows.**

```
docker pull mongo
```

**Create a Docker container from DockerHub for Mongo: MacOS and Linux.**

```
sudo docker pull mongo
```

# Setup Mongo in Docker Container

## Windows, Powershell

Start MongoDB
Windows
MacOS and Linux
Modifying Documents
The Coolest Thing
Comparing Values
Parsing Text
Consider This ...

### Create a volume for data to persist

```
docker volume create --name=mongodata
```

### Start a Docker container running the Mongo DB server

```
docker run -it -v mongodata:/data/db --name mongodb -d mongo
```

### Check log to see that the server is operational

```
docker logs mongodb
docker ps
```

### Run instance of MongoDB, goes into root of container.

```
docker exec -it mongodb bash
```

# Setup Mongo in Docker Container

## Windows, Powershell

Start MongoDB
Windows
MacOS and Linux
Modifying Documents
The Coolest Thing
Comparing Values
Parsing Text
Consider This ...

**Start the MongoDB client**

```
mongo
```

You are now able to run MongoDB commands here.

Note: `exit` to quit.

# Setup Mongo in Docker Container

## Windows, Powershell

Start MongoDB
Windows
MacOS and Linux
Modifying Documents
The Coolest Thing
Comparing Values
Parsing Text
Consider This ...

**Leave the container**

```
exit
```

**Stop Mongo container**

```
docker stop mongodb
```

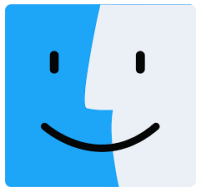
**Removing all stopped containers, if necessary due to errors in launching container**

```
docker rm $(docker ps -a -q)
```

# Setup Mongo in Docker Container

## MacOS and Linux, Terminal

Start MongoDB
Windows
MacOS and Linux
Modifying Documents
The Coolest Thing
Comparing Values
Parsing Text
Consider This ...



### Create a directory for data to persist.

```
mkdir -p ~/mongodata
```

### Start a Docker container running the Mongo DB server

```
sudo docker run -it -v ~/mongodata:/data/db --name mongodb -d mongo
```

### Check log to see that the server is operational

```
sudo docker logs mongodb
sudo docker ps
```

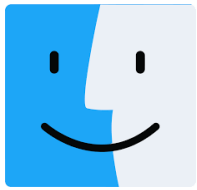
### Run instance of MongoDB, goes into `root` of container.

```
sudo docker exec -it mongodb bash
```

# Setup Mongo in Docker Container

## MacOS and Linux, Terminal

Start MongoDB
Windows
MacOS and Linux
Modifying Documents
The Coolest Thing
Comparing Values
Parsing Text
Consider This ...



**Start the MongoDB client**

```
mongo
```

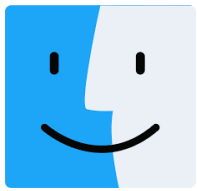
You are now able to run MongoDB commands here.

Note: `exit` to quit.

# Setup Mongo in Docker Container

## MacOS and Linux, Terminal

Start MongoDB
Windows
MacOS and Linux
Modifying Documents
The Coolest Thing
Comparing Values
Parsing Text
Consider This ...



**Leave the container**

```
exit
```

**Stop Mongo container**

```
sudo docker stop mongodb
```

**Removing all stopped containers, if necessary due to errors in launching container**

```
sudo docker rm $(docker ps -a -q)
```

# Adding to a Document in Mongo With a Date

Start MongoDB
Modifying Documents
The Coolest Thing
Comparing Values
Parsing Text
Consider This ...

## Insertion of several documents using `insertMany()`

```
 db.restaurants.insertMany( [
{ "_id" : 1, "name": "Central Perk Cafe", "Borough": "Manhattan", "avgCost": 100},
{ "_id" : 2, "name": "Rock A Feller Bar and Grill", "Borough": "Queens", "violations": 2,
"avgCost": 250},
{ "_id" : 3, "name": "Empire State Pub", "Borough": "Brooklyn", "violations": 0,
"avgCost": 300},
{ "_id" : 4, "name": "The Captain's Cafe", "Borough": "London", "avgCost": 80  },
{ "_id" : 5, "name": "The Resto in a Cave", "Borough": "Paris", "violations": 0,
"avgCost": 1 },
{ "_id" : 6, "name": "The Crow Bar", "Borough": "Paris", "violations": 98, "note": "gross",
 "avgCost": 40}
]);
```

- Note: we are able to override the `_id` settings from Mongo and implement our own values.

## Check that the Insertion worked correctly

```
db.restaurants.find().pretty()
```

# Inserting New Data Using $set{}

Start MongoDB
Modifying Documents
The Coolest Thing
Comparing Values
Parsing Text
Consider This ...

## Check document

```
db.restaurants.find({"name":"Empire State Pub"},{})
```

## Add to document

```
db.restaurants.updateOne(
   {"name":"Empire State Pub"},
   { $set: {
        rating: "Thumbs-Up",
        status: "Loved it!" },
     $currentDate: { lastModified: true }
   });
```

```
{
        "_id" : 3,
        "name" : "Empire State Pub",
        "Borough" : "Brooklyn",
        "violations" : 0,
        "lastModified" : ISODate("2019-04-03T01:06:48.584Z"),
        "rating" : "Thumbs-Up",
        "status" : "Love it!"
}
```

# Inserting New Data Using $set{}

Start MongoDB
Modifying Documents
The Coolest Thing
Comparing Values
Parsing Text
Consider This ...

## Check document

```
db.restaurants.find({"name":"Central Perk Cafe"},{})
```

```
db.restaurants.updateOne(
   {"name":"Central Perk Cafe"},
   { $set: {
        rating: "Good",
        status: "Loved it!",
        kitchenQuality: "ok" },
     $currentDate: { lastModified: true }
   });
```

```
> db.restaurants.find({name:"Central Perk Cafe"},{}).pretty()
{
        "_id" : 1,
        "name" : "Central Perk Cafe",
        "Borough" : "Manhattan",
        "kitchenQuality" : "ok",
        "lastModified" : ISODate("2019-04-03T01:51:11.879Z"),
        "rating" : "Good",
        "status" : "Loved it!"
}
```

# Inserting New Data Using $set{}

Start MongoDB
Modifying Documents
The Coolest Thing
Comparing Values
Parsing Text
Consider This ...

## Check document

```
db.restaurants.find({"name":"The Resto in a Cave"},{})
```

```
db.restaurants.updateOne(
{"name":"The Resto in a Cave"},
{
 $set: {
 rating: "Musty",
 status: "Dirty and full of rocks!!!",
 kitchenQuality: "A fire in the floor",
 Note: "The waiter was a bear who chased me."},
 $currentDate: { lastModified: true }
});
```

```
> db.restaurants.find({name:"The Resto in a Cave"},{}).pretty()
{
        "_id" : 5,
        "name" : "The Resto in a Cave",
        "Borough" : "Paris",
        "violations" : 0,
        "Note" : "The waiter was a bear who chased me.",
        "kitchenQuality" : "A fire in the floor",
        "lastModified" : ISODate("2019-04-03T02:03:59.055Z"),
        "rating" : "Musty",
        "status" : "Dirty and full of rocks!!!"
}
```

# MongoDB Is, *"The Coolest Thing"*!!

Start MongoDB
Modifying Documents
The Coolest Thing
Comparing Values
Parsing Text
Consider This ...

BTW: This graphic was *also* found as a result of searching, "The Coolest Thing" on the Internet. Apologies.

# How Do I Compare Values?

## Commands for the `restaurant` database

| Operation | MongoDB | RDBMS |
|---|---|---|
| Equality | db.employees.find({"salary":"5000"}) | where 'salary '= '5000' |
| Less Than | db.employees.find({"age":{**$lt**:30}}) | where age < 30 |
| Less Than Equals | db.employees.find({"age":{**$lte**:30}}) | where age <= 30 |
| Greater Than | db.employees.find({"age":{**$gt**:30}}) | where age > 30 |
| Greater Than Equals | db.employees.find({"age":{**$gte**:30}}) | where age >= 30 |
| Not Equals | db.employees.find({"age":{**$ne**:30}}) | where age != 30 |

## RestaurantsDB: Details where *avgCost* is `less than` 200

```
db.restaurants.find({"avgCost":{$lt:200}},{"_id": 0, "name":1, "avgCost":1})
```

```
> db.restaurants.find({"avgCost":{$lt:200}},{"_id": 0, "name":1, "avgCost":1})
{ "name" : "Central Perk Cafe", "avgCost" : 100 }
{ "name" : "The Captain's Cafe", "avgCost" : 80 }
{ "name" : "The Resto in a Cave", "avgCost" : 1 }
{ "name" : "The Crow Bar", "avgCost" : 40 }
```

# How Do I Compare Values?

## Commands for the restaurant database

Start MongoDB
Modifying Documents
The Coolest Thing
Comparing Values
Parsing Text
Consider This ...

**RestaurantsDB: Details where *avgCost* is greater than 20 and less than 100 and**

```
db.restaurants.find({"avgCost":{$gt:20}, "avgCost":{$lt:100}},{} )
```

```
> db.restaurants.find({"avgCost":{$gt:20}, "avgCost":{$lt:100}},
... {"_id":0, "name" : 1, "avgCost":1} )
{ "name" : "The Captain's Cafe", "avgCost" : 80 }
{ "name" : "The Resto in a Cave", "avgCost" : 1 }
{ "name" : "The Crow Bar", "avgCost" : 40 }
```

# How Do I Compare Values?

## Commands for the employee database

Start MongoDB
Modifying Documents
The Coolest Thing
Comparing Values
Parsing Text
Consider This ...

### age > 25

```
db.employee.find({"age":{$gt:25}, },{"age":1, "name.first":1})
```

```
> db.employee.find({"age":{$gt:25}, },{"age":1, "name.first":1})
{ "_id" : "5c89c73ea91e2b221611b950", "age" : 33, "name" : { "first" : "Pennington" } }
{ "_id" : "5c89c73ec4863a49668b3784", "age" : 27, "name" : { "first" : "Belinda" } }
{ "_id" : "5c89c73ea080dbf02bd40f0e", "age" : 31, "name" : { "first" : "Rosemary" } }
>
```

### age < 25

```
db.employee.find({"age":{$lt:25}, },{"age":1, "name.first":1})
```

```
> db.employee.find({"age":{$lt:25}, },{"age":1, "name.first":1})
{ "_id" : "5c89c73e5e7b4f23f0a32660", "age" : 23, "name" : { "first" : "Cora" } }
```

### 25 < age < 31

```
db.employee.find({"age":{$gt:25}, "age":{$lt:31}},{"age":1, "name":1,"_id":0})
```

```
> db.employee.find({"age":{$gt:25}, "age":{$lt:31}},{"age":1, "name":1, _id:0})
{ "age" : 27, "name" : { "first" : "Belinda", "last" : "Guzman" } }
{ "age" : 23, "name" : { "first" : "Cora", "last" : "Keith" } }
```

# Parsing Text Using $text{}

## Commands for the restaurants database

Start MongoDB
Modifying Documents
The Coolest Thing
Comparing Values
Parsing Text
Consider This ...

- Index the text field to allow for parsing
- The $text operator assigns a score to each document that contains the search term in the indexed fields.
- The score represents the relevance of a document to a given text search query.
- Return documents that have high scores from text searches.

### Create an index-able name field

```
db.restaurants.createIndex( { subject: "name" } )
```

### Search the name field for the word, "Cafe"

```
db.restaurants.find( { $text: { $search: "Cafe" } } )
```

# Consider This ...

Start MongoDB
Modifying Documents
The Coolest Thing
Comparing Values
Parsing Text
Consider This ...


- Can you create and populate a completely new MongoDB database?
- Can you edit your data in your database?
- Can you write sophisticated queries in your database to isolate meaningful information from the data?

# Give It a Try!


- Play with your code!
  1. Try adding new attributes to a document
  2. Try modifying existing attributes in a document
  3. Add some numerical data to try the encoded operators; $\{<, >, \leq, \geq, ==\}$