Intro to MongoDB

CMPSC 305 – Database Systems

ALLEGHENY COLLEGE

Database Language Guide

SQL systems versus NoSQL

RDBMS		MongoDB
Database	⇨	Database
Table, View	⇨	Collection
Row	⇨	Document (BSON)
Column	⇨	Field
Index	⇨	Index
Join	⇨	Embedded Document
Foreign Key	⇨	Reference
Partition	⇨	Shard

- The terms are different, but their meanings are similar
- Schema-less, collections (like tables) are populated by any data
- Documents are similar to the tuples of Sqlite3 programming

Let's code

Schools Data

Populate the MongoDB database

```
db.schools.insertMany([
        { 'school': "Washington", name: 'Ryan', gender: 'M', 'Job':'Teacher' },
        { 'school': "Edison", name: 'Joanna', gender: 'F', 'Job':'Professor'},
        { 'school': "Eaton", name: 'Roger', gender: 'M', 'Job':'Instructor'},
        { 'school': "Lewis", name: 'Presilla', gender: 'F', 'Job':'Instructor'}
]);
```

```
db.schools.find({ SEARCH-SPACE }, {CONSTRAINTS} )
```

- EARCH-SPACE → Scan this search space
- CONSTRAINTS → Find constraint(s) within the search space

- **Querying "Jobs" == "Instructor", show name, gender and school**

```
db.schools.find({'Job':'Instructor'}, {'name': 1, 'gender': 1, 'school':1}).pretty()
```

Let's code

output

Back to mycompiler.io ...

MongoDB

```
1 db.school.insertMany([
2   { 'school': "Washington", name: 'Ryan', gender: 'M', 'Job':'Teacher' },
3   { 'school': "Edison", name: 'Joanna', gender: 'F', 'Job':'Professor'},
4   { 'school': "Eaton", name: 'Roger', gender: 'M', 'Job':'Instructor'},
5   { 'school': "Lewis", name: 'Presilla', gender: 'F', 'Job':'Instructor'}
6 ]);
7
8
9 db.school.find({'Job':'Instructor'},{'name': 1, 'gender': 1, 'school':1 }).pretty()
```

https://www.mycompiler.io/new/mongodb

Let's code

output

```
mycompiler_mongodb> [
  {
    _id: ObjectId('690cc4935c08043b526b128e'),
    school: 'Eaton',
    name: 'Roger',
    gender: 'M'
  },
  {
    _id: ObjectId('690cc4935c08043b526b128f'),
    school: 'Lewis',
    name: 'Presilla',
    gender: 'F'
  }
]
mycompiler_mongodb>
```

Find all lamps: db.Furniture.find({},{lamp:1})

Query Challenge

What do these commands do?

- db.school.find({'school':"Washington"},{})
- db.school.find({'school':"Eaton"}, {_id:0, 'name':1})
- db.school.find({'school':"Washington"},{'Job':1, _id:0})
- db.school.find({}, {'name':1, 'Job':1, _id:0})

Query Challenge

- What Job does Presilla have?
- What Job does Presilla have AND where does she work?
- Who works at Eaton?
- List the schools for all Instructor positions.

Restaurant data

Populate the MongoDB database

```
db.restaurants.drop()

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

Inserting New Data Using $set{}

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 }
        });
```

Inserting New Data Using $set{}

Check document, again to see updates!

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

```
{
  _id: 3,
  name: 'Empire State Pub',
  Borough: 'Brooklyn',
  violations: 0,
  avgCost: 300,
  lastModified: ISODate('2025-11-09T02:51:55.023Z'),
  rating: 'Thumbs-Up',
  status: 'Loved it!'
}
```

Inserting New Data Using $set{}

Check document

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

Add to document

```
db.restaurants.updateOne(
        {"name":"Empire State Pub"},
        { $set: {
                chef: "The Swedish Chef",
                status: "Dum-Di-Dum!" },
        $currentDate: { lastModified: true }
        });
```

Check document, again to see updates!

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

Inserting New Data Using $set{}

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 }
        });
```

Check document, again to see updates!

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

Inserting New Data Using $set{}

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 }
});
```

Check document, again to see updates!

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

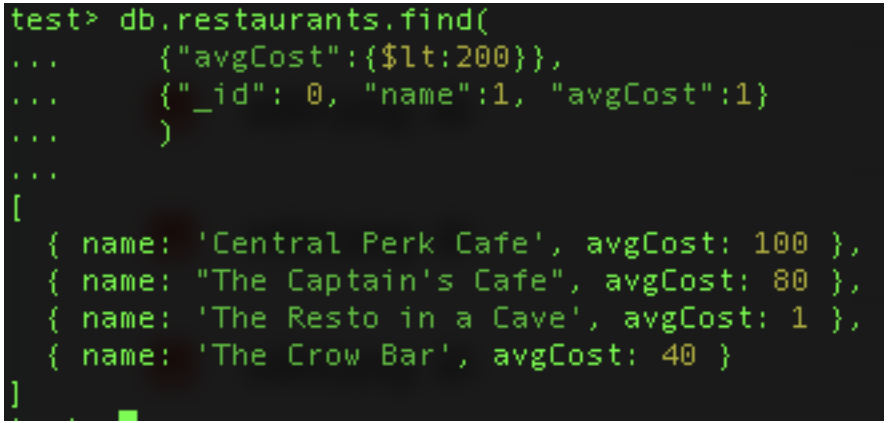
How Do I Compare Values?

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

How Do I Compare Values?

RestaurantsDB: Details where avgCost is less than 200

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



How Do I Compare Values?

RestaurantsDB: Details where avgCost is greater than 20 and less than 80 and

```
db.restaurants.find(
        {"avgCost":{$gt:20}, "avgCost":{$lt:80}},
        {}
        )
```

```
db.restaurants.find(
        { "avgCost": { $gt: 20, $lt: 80 } }
)
```

How Do I Compare Values?

```
test> db.restaurants.find(
...     { "avgCost": { $gt: 20, $lt: 80 } }
... )
...
[
  {
    _id: 6,
    name: 'The Crow Bar',
    Borough: 'Paris',
    violations: 98,
    note: 'gross',
    avgCost: 40
  }
]
```

Advanced Data and Queries

File: sandbox/syntheticData small.json

Let's horse around with different data ...

Synthetic data available from:
https://json-generator.com/

More Comparing Values. Yey!!

Commands for the employee database

age == 27

```
db.employee.find({"age":27},{"age":1, "name.first":1})
```

age > 28

```
db.employee.find({"age":{$gt:28}, },{"age":1, "name.first":1})
```

age < 28

```
db.employee.find({"age":{$lt:28}, },{"age":1, "name.first":1})
```

26 < age < 40

```
db.employee.find({"age":{$gt:26}, "age":{$lt:32}},{"age":1, "name":1,"_id":0})
```

Query Challenge

Commands for the employee database

What do the following queries do?

```
db.employee.find({},{"name.last":1, "name.first":1})
db.employee.find({},{company:1})
db.employee.find({},{"name.last":1,address:1})
db.employee.find({},{company:1,registered:1,_id:0})
db.employee.find({},{company:1,"friends.name":1,_id:0})
db.employee.find({company:"RADIANTIX"},
                {"friends.name":1,_id:0})
```

Query Challenge

Commands for the employee database

Gimme the following queries

- List the lat and long for all company entries
- List all firstname entries assoc with each company entry
- List the friends of each first name entry
- Give the company for which Nadia Soto is one of the friends

New Data: Cats and Dogs

File: sandbox/catsdogs.json

Syntax: OR

```
db.catsdogs.find({$or:[{expr},{expr}]})
```

Syntax: AND

```
db.catsdogs.find({$and:[{expr},{expr}]})
```

What do the following queries do?

```
db.catsdogs.find({$or:[{'age':4}, {age:5}]})

db.catsdogs.find({$and:[{'owner.name':'Charlie'}, {'type':'Cat'}]})

db.catsdogs.find({$and:[{'owner.name':'Charlie'}, {'type':'dog'}]})
```

Query Challenge

File: sandbox/catsdogs.json

Gimme the following queries?

- Give a list breed information for only dogs
- Give a list characteristics information for only cats
- Give the addresses of all owners of dogs
- Give all dogs who are at least age 2
- Give all cats who are at least a year old

Query Challenge

File: sandbox/catsdogs.json

Gimme the following queries?

- What kind of dog does Charlie have?
- What are the characteristics of Labrador Retrievers?
- What cats have color s of Seal Point and Black and Tan?

Consider This ...

Gimme the following queries?

- 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?