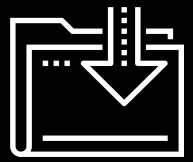
Introduction to NoSQL and MongoDB

Data Boot Camp

Lesson 12.1



Class Objectives

By the end of today's class you will be able to:

Identify the key differences between SQL and NoSQL databases to aid decisions around what kind of database to use in different situations.

Create and connect to local MongoDB databases.

Create, read, update, and delete MongoDB documents by using the mongo shell.

Import data from CSV and JSON files into a local MongoDB database.

Welcome & Intro to MongoDB

What is MongoDB?

Welcome & Intro to MongoDB

MongoDB is a popular noSQL database.

It uses a document-oriented model as opposed to a table-based relational model (SQL).

MongoDB stores data in a binary format, which allows it to be parsed much more quickly.

MongoDB has many drivers and packages for connecting to Node, C++, Java, etc.

Welcome & Intro to MongoDB

Relational Databases (SQL)

ID	Title	Author	Published
1	Relational Databases (SQL)	Johnson Hargraves	2010
2	Interstellar Journey	Cho Gyeong	2011
3	Fortuitous Events in History	Gabriel Garcia Hernandez	2013

SQL relies on **joins** to combine relevant data.

Author	Email	Phone Number
Johnson Hargrave	jhargraves42@gmail.com	911-546-5454
Cho Gyeong	chogyeong@gmail.com	911-544-5112
Gabriel Garcia Hernandez	gghernandez400@gmail.com	125-215-5645

Welcome & Intro to MongoDB

Document Database (noSQL)

- NoSQL databases are effectively JSONs.
- They excel at heterogeneous data formats and are easy to implement.


```
{
    "id": 1,
    "Title": "The Floating Winter Island",
    "Author": {
        "name": "Johnson Hargraves",
        "email": "jhargraves42@gmail.com",
        "phone": "911-546-5454"
    },
    "Published": 2010
},

{
    "id": 2,
    "Title": "Interstellar Journey",
    "Author": {
        "name": "Cho Gyeong",
        "email": "chogyeong@gmail.com",
        "phone": "911-544-5112"
    },
    "Published": 2011
}
```

Welcome & Intro to MongoDB

Terms are slightly different in the NoSQL context.

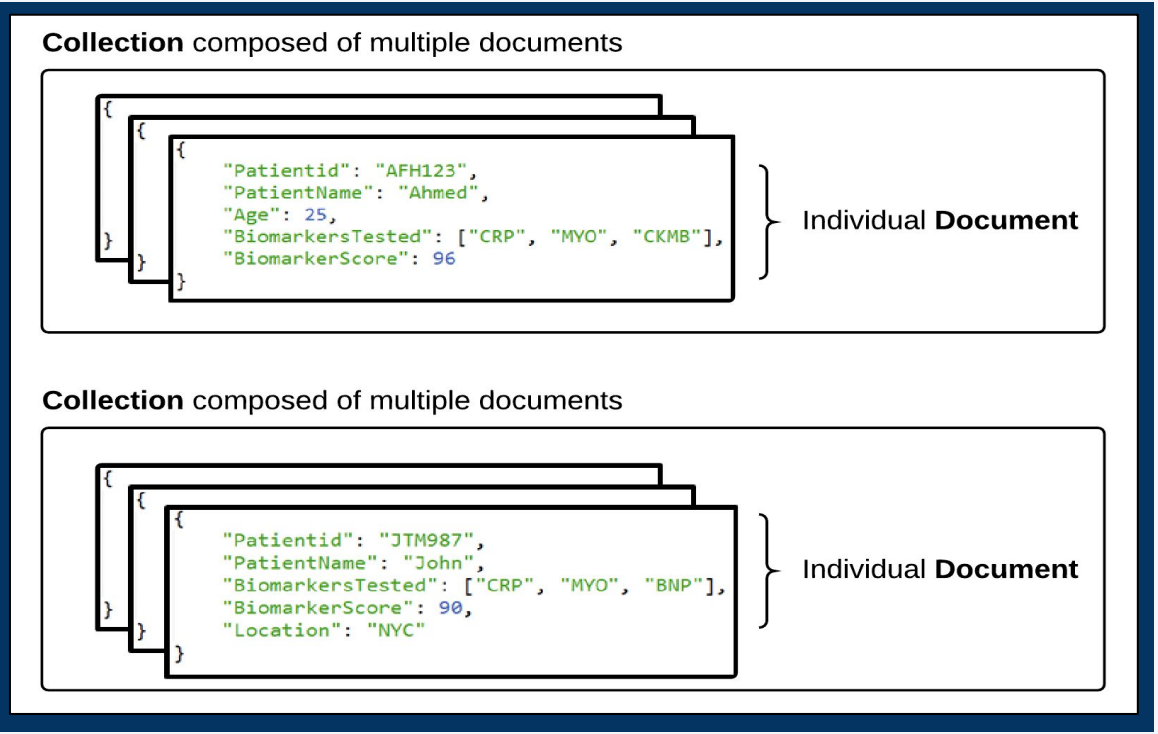
MongoDB Storage

SQL (RDBMS)	MongoDB
Database	Database
Table	Collection
Row	Document
Column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by MongoDB)

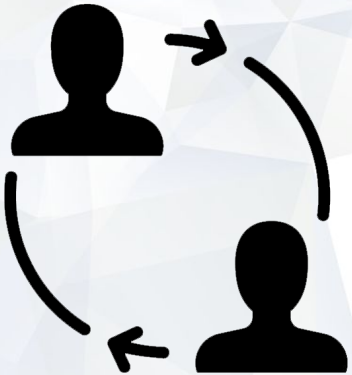
Welcome & Intro to MongoDB

MongoDB Storage

Database composed of multiple collections



Questions?



Activity: Quick MongoDB Research

In this activity, you will search the web to answer four questions about MongoDB.

Suggested Time:

10 Minutes

Activity: Quick Mongo Research

Answer the following questions:

According to the MongoDB website, what are the advantages of NoSQL databases like MongoDB?

According to reliable sources on the web (Quora, professional forums, etc.), what are the advantages of using MongoDB?

What are the disadvantages of NoSQL databases like MongoDB?

Find some examples of MongoDB databases used in the real world. What kinds of companies use them and what kind of data do they store?

Time’s Up! Let’s Review.

Activity: Quick MongoDB Research Review

In this activity, you will search the web to answer three questions about MongoDB.

Suggested Time:

15 Minutes

Everyone Do: Quick Mongo Research Review

Advantages

- MongoDB uses a document data model
- MongoDB supports dynamic queries on documents using a document-based query language that's nearly as powerful as SQL
- MongoDB stores data in RAM memory (binary format) which allows for quicker access.
- It has a flexible schema, or schema-free. Your code defines your schema.
- The syntax is easier to read and understand
- You can horizontally scale which helps increase storage capacity, whereas RDBM databases scale vertically and use a lot of memory.

Disadvantages

- You need a lot of RAM if you have large Mongo databases.
- It's difficult to create joins.
- Relationships are not well-defined
- Data can be duplicated
- Limited size is 16MB for a document

Time's Up! Let's Review.

Instructor Demonstration

Basic MongoDB Queries

⚠️ Important ⚠️

If you are a Mac user and have installed MongoDB by using Homebrew, run the following command:

```
brew services start mongodb-community@6.0
```

Basic MongoDB Queries

01 Typing and running `use travel_db` will create a database and switch to that database.

02 To show the current database you are using type and run: `db`

03 To show the current databases that exist type and run: `show dbs`

04 To create a collection named “destinations” type and run: `db.createCollection("destinations")`

05 To see all the collections in a database type and run: `show collections`

06 We use `db.collectionName.insertOne({key:value})` to insert a document (row).

Basic MongoDB Queries (Continued)

07 To insert data in the destinations collection of the travel_db, type and run:
`db.destinations.insertOne({"continent": "Africa", "country": "Morocco", "major_cities": ["Casablanca", "Fez", "Marrakech"]})`

08 We use `db.collectionName.insertMany([{key:value},{key:value}])` to insert multiple documents.

09 To show all the data in a collection type and run: `db.collectionName.find()`

10 To find specific documents in a collection type and run: `db.collectionName.find({key:value})`

Activity: Mongo Class

In this activity, you will learn basic query operations in MongoDB. Specifically, you'll practice inserting and finding documents.

Suggested Time:

15 minutes

Activity: Mongo Class

Instructions

Use the command line to create a `classDB` database.

Insert entries into this database for yourself and other students within a collection called `classroom`.

Each document should have:

- A `name` field for the person's name
- A `favorite_python_library` field for the person’s favorite Python library
- An `age` field for the person's age
- A `hobbies` field for a list of that person’s hobbies

Use the `find()` commands to get a list of everyone of a specific age, then use the `name` key to collect the entry for a single person.

Time’s Up! Let’s Review.

Instructor Demonstration

Removing, Updating, and Dropping in MongoDB

Removing, Updating, and Dropping in MongoDB

Update

The `updateOne()` method takes in two objects as its parameters and will only update the first entry that matches.

```
db.destinations.updateOne({"country": "Egypt"}, {$set: {"continent": "Antarctica"}})
```

To update more than one document, we can use the `updateMany()` method, which will update all the records that meet the given criterion.

```
db.destinations.updateMany({"country": "Egypt"}, {$set: {"continent": "Antarctica"}})
```

Removing, Updating, and Dropping in MongoDB

Inserting with Update

In a given scenario where the field `{"capital": "Rome"}` does not exist, what will happen when we run the following command?

```
db.destinations.update({'country': 'Egypt'}, {$set: {'capital': 'Rome'}})
```

- The document doesn't exist so nothing happens.
- We need to use `{upsert: true}` to create the new document.

To add elements to the collection, we use `$push`

```
db.destinations.update({"country": "Morocco"}, {$push: {"major_cities": "Agadir"}})
```

Removing, Updating, and Dropping in MongoDB

Delete

To delete *ALL* the documents from a collection, we pass an empty object with the `remove()` method; `db.destinations.remove()`

To delete one object from a collection we add the key:value pair;
`db.destinations.remove({"country": "USA"}, {justOne: true})`

To delete a collection from a database we use the `drop() method`;
`db.destinations.drop()`

To delete a database we use the `db.dropDatabase() method`.

Questions?

Activity: GardenDB

In this activity, you will gain further practice with CRUD operations in MongoDB by creating a database centered around building a garden.

Suggested Time:

15 minutes

Activity: GardenDB

Instructions	Create a new database called `gardenDB` using the mongo shell.
	Create a collection called `plants` which contains the following: • A string field for `plantName` • An integer field for `yearsGrowing` • A boolean field for `stillAlive` • An array of strings called `plantNutrition` to store information about how best to keep the plant alive.
Insert	Insert three new documents into the collection. You can be creative with what you put in here and have some fun with it.
Update	Update the `yearsGrowing` fields for your documents so that they are one greater than their original values.
	Update the `stillAlive` value for one of the documents so that it is now false.
	Add a new value into the `plantNutrition` array for one of the documents.
	Find the plant in the collection that isn't alive and remove it from the collection.

Time’s Up! Let’s Review.


Break

Instructor Demonstration

Importing Data, Accessing Nested Data, and Modifying Data Types

Importing Data

```
mongoimport --type json -d dbName -c collectionName --drop --jsonArray filename.json
```

JSON Files

Argument	Reference
`mongoimport`	The Mongo command to import data from a file into MongoDB.
`--type json`	Specifies that we want to import json data.
`-d dbName`	Specifies the name of the database to import the data to.
`-c collectionName`	Specifies the name of the collection to import the data to.
`--drop`	Asks MongoDB to drop the collection if it already exists in the database. If this is not specified and the collection exists, the data will be added to the existing collection.
`--jsonArray`	Specifies that the data that will be imported is contained within a json array.
`filename.json`	The file to use that contains the data we want to import.

Importing Data

```
mongoimport --type csv -d dbName -c collectionName --headerline --drop filename.csv
```

CSV Files

Argument	Reference
`mongoimport`	The Mongo command to import data from a file into MongoDB.
`--type csv`	Specifies that we want to import CSV data.
`-d dbName`	Specifies the name of the database to import the data to.
`-c collectionName`	Specifies the name of the collection to import the data to.
`--headerline`	Tells MongoDB to use the first row of the CSV as field names.
`--drop`	Asks MongoDB to drop the collection if it already exists in the database. If this is not specified and the collection exists, the data will be added to the existing collection.
`filename.json`	The file to use that contains the data we want to import.

To access nested data, we use **Dot Notation**.

Accessing Nested Data

Instructions

Switch to the database with `use autosaurus`.

Find a mechanic in the `mechanics` collection who specializes in “Acura” cars.

`db.mechanics.find({"car_specialties": "Acura"})`

```
{
  _id: ObjectId("63488310652701b16673d68a"),
  mechanic_name: 'Quenti Yupanqui',
  wages: { hourly_rate: '46.75', weekly_hours: 40 },
  contact: { phone: '555-876-8759', email: 'yupanquiq@autosaurus.com' },
  hours: {
    Monday: '8am-4pm',
    Tuesday: '11am-7pm',
    Wednesday: '10am-6pm',
    Thursday: '8am-4pm',
    Friday: '8am-4pm'
  },
  car_specialties: [
    'Ram', 'Bentley', 'Land Rover', 'Lamborghini', 'Acura', 'Suzuki',
    'Alfa Romeo', 'Scion'
  ]
}
```

Accessing Nested Data

Instructions

Find the mechanics who work 40 hour weeks.

`db.mechanics.find({"wages.weekly_hours": 40})`.

Find the mechanics whose email address is “yupanquiq@autosaurus.com.”

```
db.mechanics.find({"contact.email": "yupanquiq@autosaurus.com"})
```

Find the mechanic who are paid $50 per hour.

```
db.mechanics.find({"wages.hourly_rate": "50"})
```

Note

The `wages.hourly_rate` field is stored as a string, but the values are numeric. To use `.find()` with a numeric value, we would have to modify the data type in Mongo.

Modifying Data Types

Instructions

Convert the `wages.hourly_rate` field to numeric (Double).

```
db.mechanics.updateMany({}, [
                                {'$set': {
                                     "wages.hourly_rate" : {
                                                    '$toDouble': "$wages.hourly_rate"
                                                }
                                            }
                                }
                            ]
                        )
```

`$toDouble`	The operator that tells Mongo to convert the field that follows to data type Double.
`$wages.hourly_rate`	The `$` symbol specifies that we want to use the value inside the `wages.hourly_rate` field.

Modifying Data Types

Data Type Modification Operators

Operator	Data Type
`$toDouble`	Double (floating point value)
`$toInt`	Integer
`$toString`	String
`$toBool`	Boolean
`$toDate`	Date

Questions?

Activity: Import, Update, and Explore

In this activity, you will practice importing data from JSON and CSV files, then updating data types, and exploring data in the Mongo Shell.

Suggested Time:

25 minutes

Activity: Import, Update, and Explore

Instructions (Import)	Write a command that imports the data from annual_aqi_by_county_2022.csv to a database called `epa` and a collection called `annual_aqi_by_county`. If the collection already exists, drop the collection.
	Write a command that imports the data from ohio_daily_records_2022.json to a database called `epa` and a collection called `ohio_daily_records`. If the collection already exists, drop the collection.
	Write a command that imports the data from ohio_feb_2022.json to a database called `epa` and a collection called `ohio_air`. If the collection already exists, drop the collection.
	Write a command that imports the data from ohio_jan_2022.json to a database called `epa` and a collection called `ohio_air`.
Note	ohio_feb_2022.json and ohio_jan_2022.json are imported into the same collection, so be sure not to drop the collection when importing the second file.

Activity: Import, Update, and Explore

Instructions (Verify)	Run `mongo` or `mongosh` from your Terminal and verify that your files imported correctly by writing commands to do the following: • Use the `epa` database. • Show all the collections in the `epa` database. • Verify that there is data in each of the 3 collections (`ohio_daily_records`, `annual_aqi_by_county`, and `ohio_air`) by using `findOne()` to display an entry. • Verify that there is data from January 2022 in the `ohio_air` collection by searching on the `date_local` field. • Verify that there is data from February 2022 in the `ohio_air` collection by searching on the `date_local` field.
Hint	Dates for the `ohio_air` collection use the YYYY-MM-DD format.

Activity: Import, Update, and Explore

Instructions (Update and Explore)	Update the `DAILY_AQI_VALUE` nested under `CO` in the `ohio_daily_records` collection to change the string values to integers.
	Update the `PERCENT_COMPLETE` nested under `CO` in the `ohio_daily_records` collection to change the string values to doubles.
	Find a record in the `ohio_daily_records` collection where `CO.UNITS` matches "ppm" and `NO2.UNITS` matches "ppb". This will help us find sites that recorded values for carbon monoxide and nitrogen dioxide.
Hints	Nested records use dot notation.
	Remember to use square brackets as part of your query when changing the data type of a field. Your `updateMany()` query should use the following convention: `updateMany({}, [ {'$set': "field_name": {'$toDataType': "$field_name"}} ])`

Time’s Up! Let’s Review.

Questions?

Instructor Demonstration

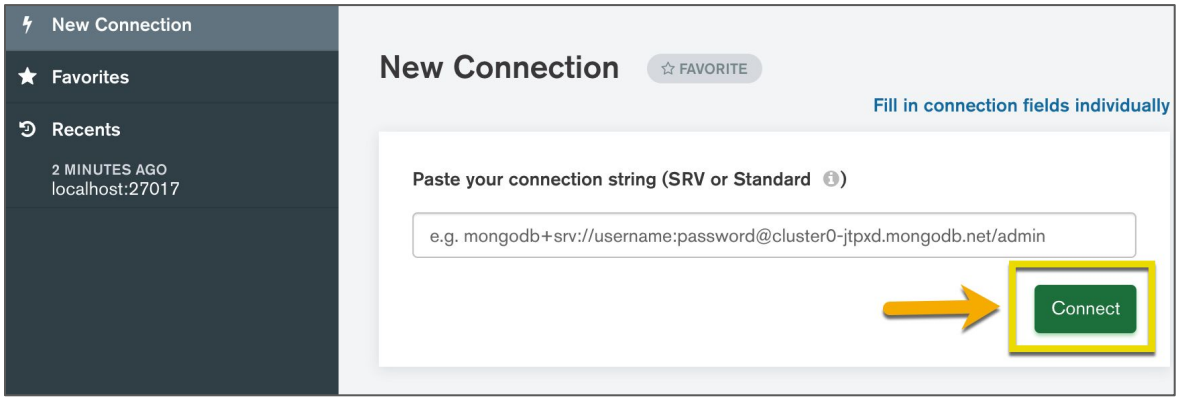
Mongo Compass

MongoDB Compass

01 Start the MongoDB server.

02 Open MongoDB Compass, if it's not already open.

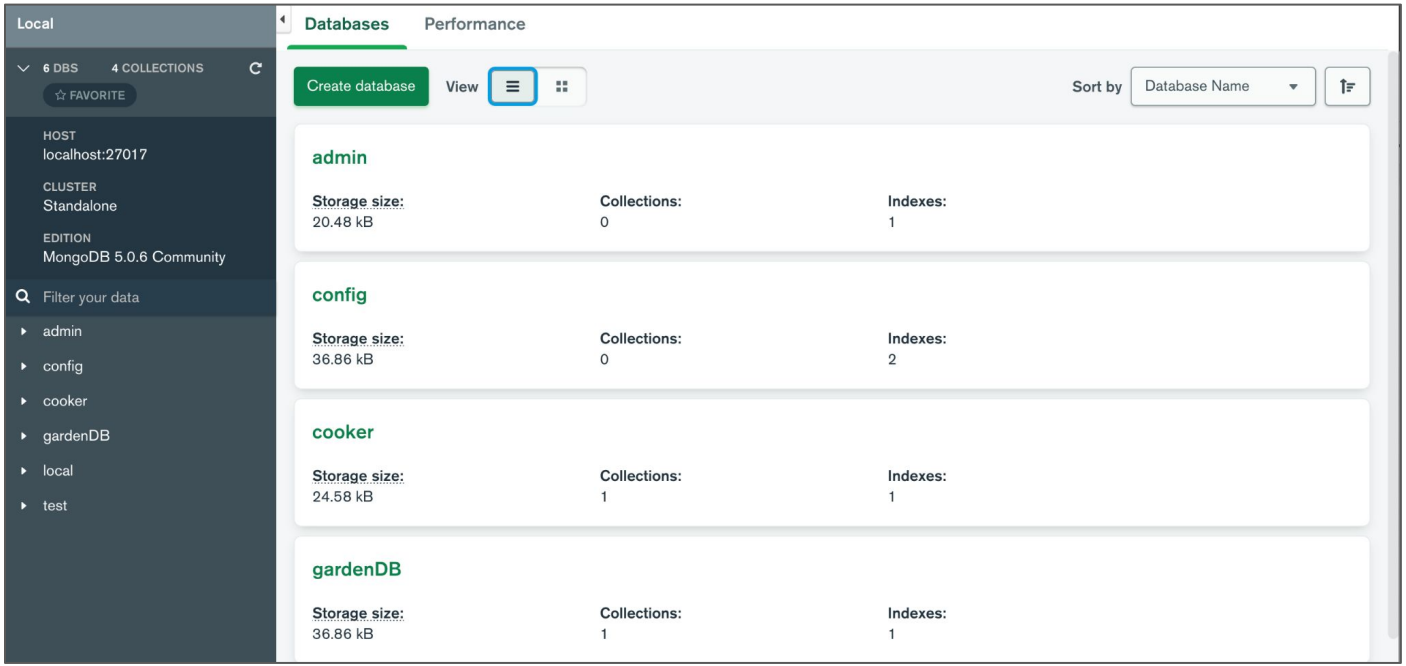
03 Click the "Connect" button to make the connection to your MongoDB.



MongoDB Compass

04

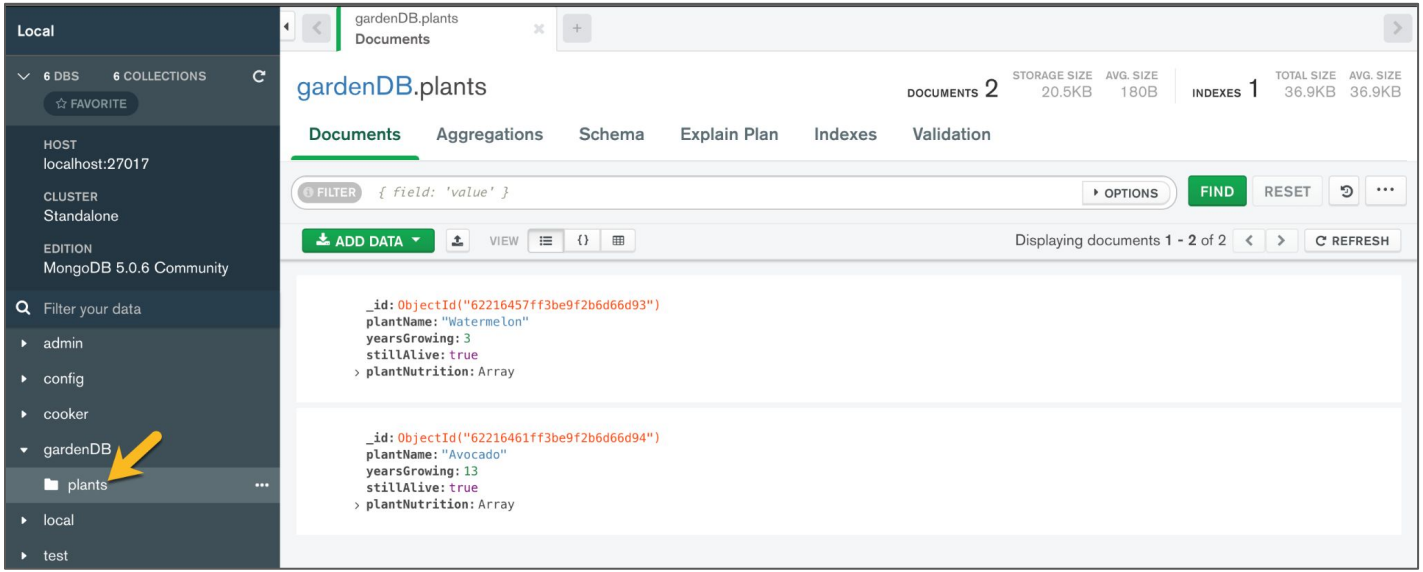
After clicking the "Connect" button, we can view a list of all of the MongoDB databases hosted on their localhost server.



MongoDB Compass

05 We can click on a database's name to see a list of all of the collections stored on that database.

06 Click on the `plants` a collection in the `gardenDB` to see all documents in the collection.



Activity: Compass Playground

In this activity, you will explore MongoDB Compass.

Suggested Time:

5 minutes

Questions?


The End.