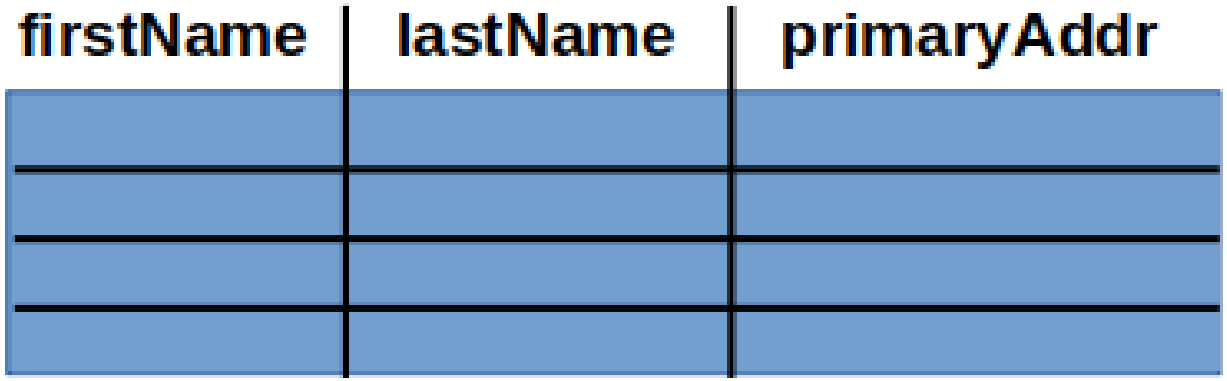
Introduction to Database Systems: CS312 MongoDB

Oliver BONHAM-CARTER

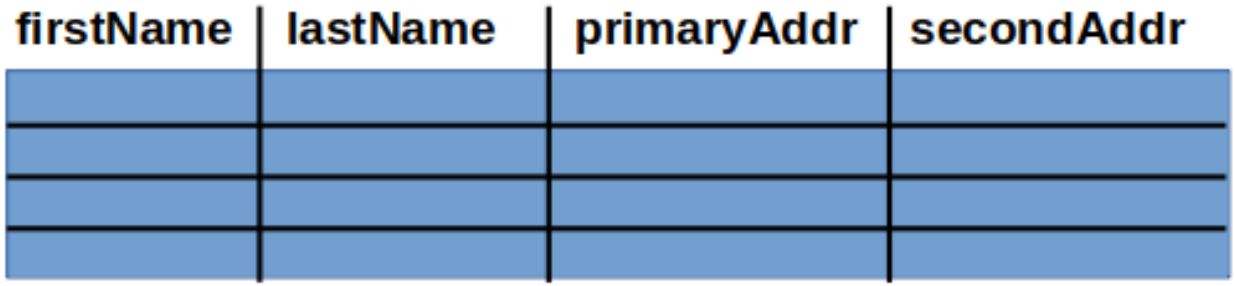
26 Oct 2020

The Problem with SQL



- Let's say that we have a (perfectly) working SQL table
- The schema has been designed and coded for *current data requirements*

Table Update (i)



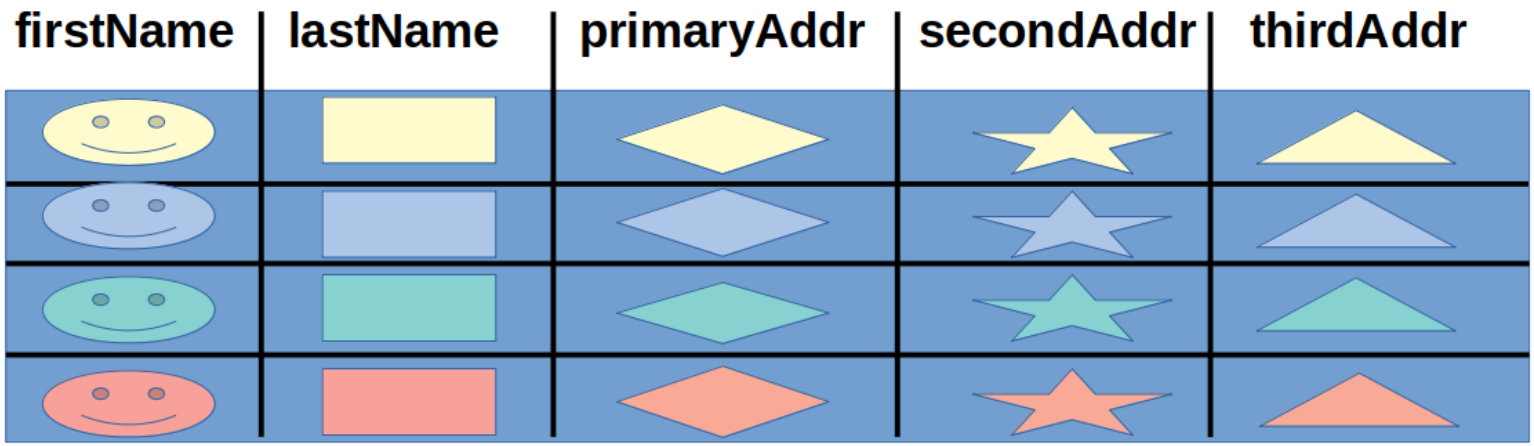
- The data we collect has changed.
- We need to update our schema for the *new data requirements*

Table Update (ii)

firstName	lastName	primaryAddr	secondAddr	thirdAddr

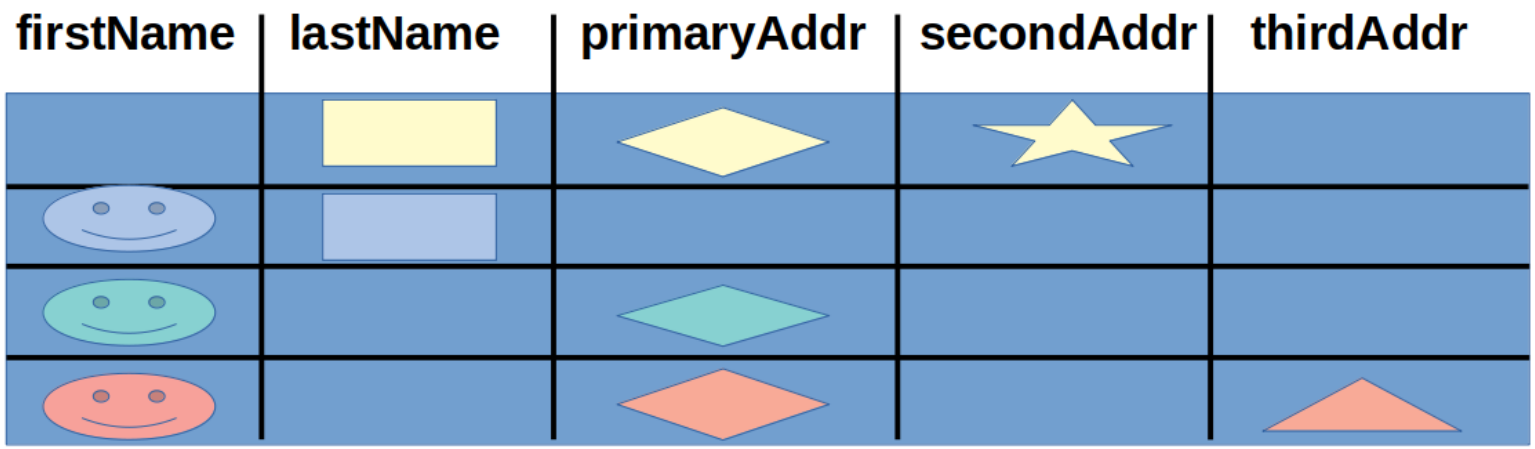
- Our needs have changed again and the SQL table must be updated.
- The schema is reprogrammed

Expectations



- We expect that the table will be full when in use
- Expectations are not always fulfilled...

In Reality, Much Data is Missing

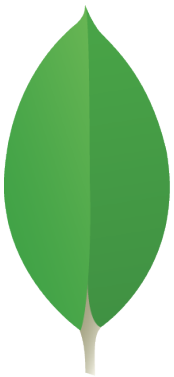


- But, in reality, much of the table is empty!
- The table can easily get huge and be hard to manage.

We Might Stop and Ask Ourselves...

- What can we do to stop having to redesign our database schema with our changing data?
- Is SQL the right type of database management system for our changing data requirements?

A NoSQL Database Management System



mongoDB®

- NoSQL: *Not Only SQL* database systems that support SQL-like query languages, but are used increasingly in big data applications and real-time web applications.
- The stored data is allowed to change

- `https://www.mongodb.com/`

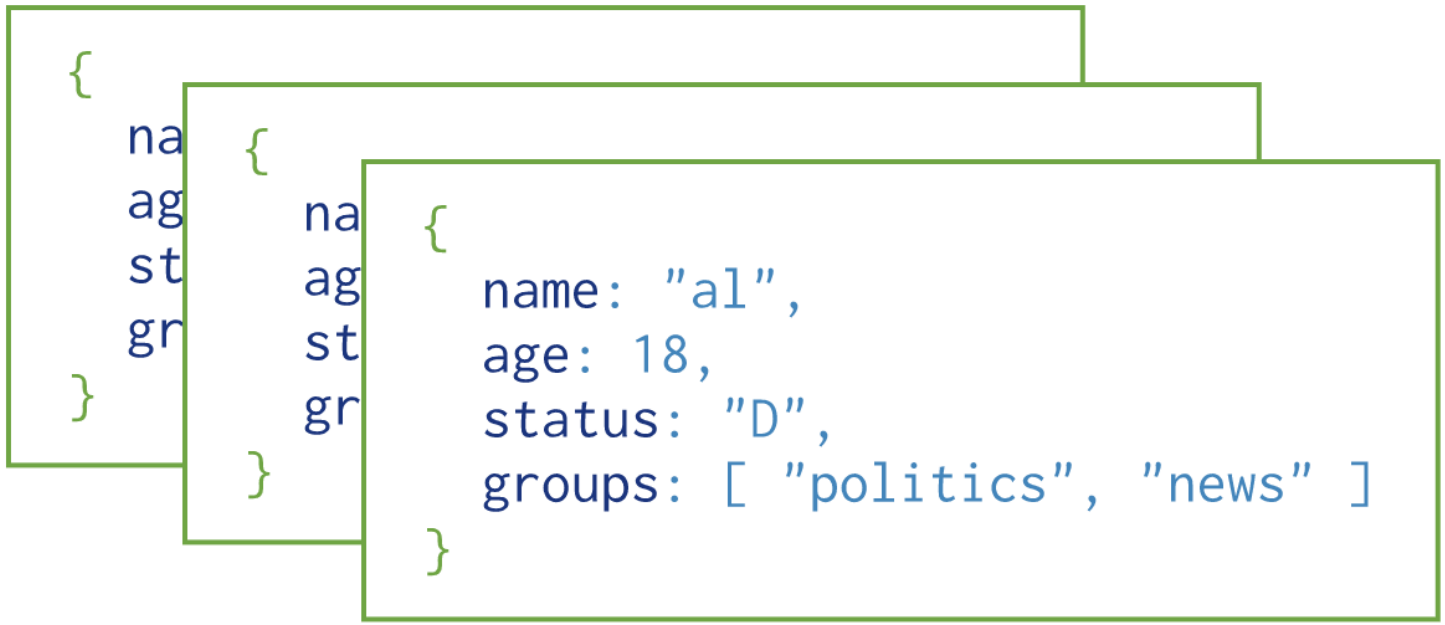
Database Language Guide

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

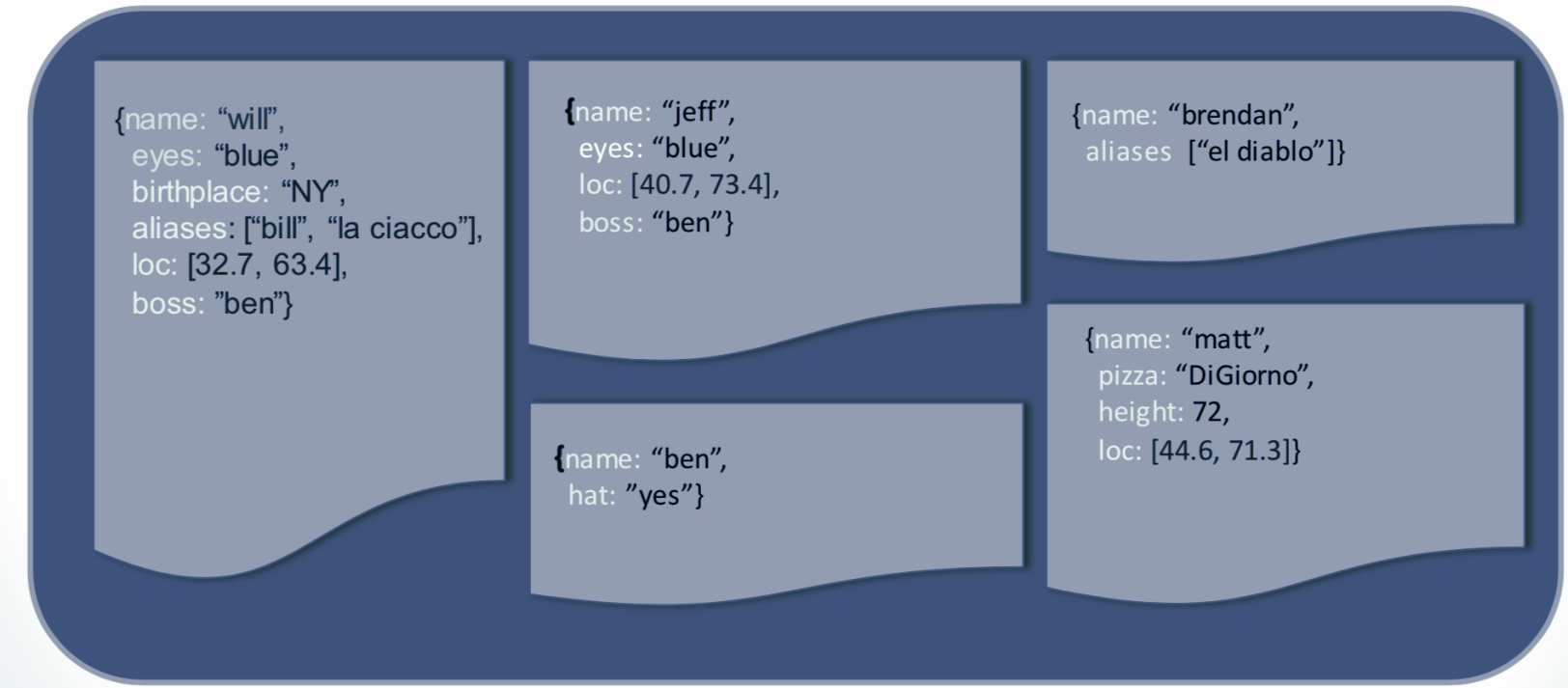
Schema Free



- No pre-defined data schema
  - Data may be entered in absence of a defined schema
- **Documents** (*rows*) of **collections** (*DB*'s) may have different types of data

Schema Free

Mostly similar documents



- Sometimes not all the data is available to create a document.
- The query *interprets* missing data as NULL entries

JSON and MongoDB Code

- **Data is in name / value pairs**
- A name/value pair consists of a field name followed by a colon, followed by a value:
  - Example: `{ "name": "R2-D2 "}`
- **Data is separated by commas**
  - Example: `{ "name": "R2-D2", race : "Droid"}`
- **Curly braces hold objects**
  - Example: `{"name": "R2-D2", race : "Droid", affiliation: "rebels"}`
- **An array is stored in brackets** `[]`
  - Example `[ { "name": "R2-D2", race : "Droid", affiliation: "rebels"}, { "name": "Yoda", affiliation: "rebels"} ]`

CRUD Operations

Create, Read, Update and Delete

We will see more of these commands later!

- **Db.collection** specifies the collection or the *table* in which to store the document (*tuple*)
- **Create**
  - `db.collection.insert()`
  - `db.collection.save()`
  - `db.collection.update()`
- **Read**
  - `db.collection.find()`
  - `db.collection.findOne()`
- **Update**
  - `db.collection.update()`
- **Delete**
  - `db.collection.remove()`

Getting started with Mongo in Docker

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

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 the MongoDB client

```
mongo
```

You are now able to run MongoDB commands here.

Note: `exit` to quit.

Setup Mongo in Docker Container

Windows, Powershell

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

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 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

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)
```

Run MongoDB Commands

Working in MongoDB

Find databases or collections, from Mongo's client

```
show dbs
show collections
```

Begin a new database, from Mongo's client

```
use myDB
```

Simple Collections
Simple Example of Queries

Simple Insert into the *Greetings* collection

```
db.Greetings.drop()

db.Greetings.insert({"Hello":"English greeting"})
db.Greetings.insert({"Goodbye":"English farewell "})

db.Greetings.insert({"Bonjour":"French greeting"})
db.Greetings.insert({"Au revoir":"French farewell"})

db.Greetings.insert({"Buongiorno":"Italian greeting"})
db.Greetings.insert({"Arrivederci":"Italian farewell"})
```

Find contents of *Greetings* collection: select * from Greetings;

```
db.Greetings.find({},{})
db.Greetings.find({},{}).pretty();
db.Greetings.find({},{_id:0}).pretty();
```

General Finds in a "Music" Collection

For example ... `true` or `1`

- The `find()` method with no parameters returns all documents from a collection and returns all fields for the documents.

The following operation returns all documents in the `music` collection

```
db.music.find({},{})
```

Returns all "bandName" in the `music` collection

```
db.music.find({ }, {"bandName":True})
db.music.find({ }, {"bandName":1})
```

General Finds in a "Music" Collection

For example ... `true` or `1`


```
db.collection.find({ Search field },{ What to filter in field })
```

- The `find()` method can also be used to specify particular fields to show from a collection-wide search.

The following operation returns all records with no filtration

```
db.music.find({},{})
```

The following operation filters-out only the bandName from the `music` collection

```
db.music.find({ },{"bandName":true})
```

Returns all `bandName` and `songs` in the `music` collection

```
db.music.find({ },{"bandName":true, "songs":true})
```

Insert into the *Country* collection

```
db.Country.drop()
db.Country.insert({
   language_en: "English",
   language_fr: "anglais",
   country_en: "England",
   country_fr :"l'angleterre"
   })
```

Adding to the Collection


```
db.Country.insert({
    language_en: "French",
    language_fr: "francais",
    country_en: "France",
    country_fr :"la France"
})

db.Country.insert({
    language_en: "Italian",
    language_fr: "italien",
    country_en: "Italy",
    country_fr :"l'Italie"
})
```

Simple Queries

Find contents of *Country* collection:
select * from Country;

```
db.Country.find()
db.Country.find().pretty()
```

```
> db.Country.find().pretty()
{
        "_id" : ObjectId("5c89c87211455eb94061b2f1"),
        "language_en" : "English",
        "language_fr" : "anglais",
        "country_en" : "England",
        "country_fr" : "l'angleterre"
}
{
        "_id" : ObjectId("5c89c87211455eb94061b2f2"),
        "language_en" : "French",
        "language_fr" : "francais",
        "country_en" : "France",
        "country_fr" : "la France"
}
{
        "_id" : ObjectId("5c89c87311455eb94061b2f3"),
        "language_en" : "Italian",
        "language_fr" : "italien",
        "country_en" : "Italy",
        "country_fr" : "l'Italie"
}
```

Simple Queries

Find specific contents of *Country* collection
select * from Country where country_en == "France";

```
db.Country.find({country_en : "France"}).pretty()
```

Find specific contents of *Country* collection
select * from Country where language_en == "English";

```
db.Country.find({language_en : "English"}).pretty()
```

Find specific contents of *Country* collection
select language_fr, language_en from Country where language_en == "English";

```
db.Country.find({language_en : "English"},
  {_id:0}).pretty()

db.Country.find({language_en:"English"},
{language_fr : 1, language_en : 1}).pretty()
```

How to shut down a session

Drop the collection (Greetings): Destroy the data, remove collection

```
db.Greetings.drop()
```

Drop the collection (Country): Destroy the data, remove collection

```
db.Country.drop()
```

Closing down

- Type `exit` in the client terminal

Advanced Data and Queries

Let's horse around with data and code ...

Locate your data file, open it in Atom and copy and past the contents (all at once) into the MondoDB client

`syntheticData_small.json`

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

Investigating Queries

```
db.collection.find({searchSpace},{showAttrib:1})
```

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})
```

Rewriting Queries


```
db.collection.find({searchSpace},{showAttrib:1})
```

Rewrite the queries to get...

- The last name, first name and balance information of each document
- The company name, latitude and longitude of each document
- The list of friends of record associated with the company PROSURE
- The phone number for the PROSURE company

GitHub submission

- Write queries in MongoDB
- File: `sandbox/03_practical/queries.md`
  - Move the directory (`03_practical/`) into your practicals repository (with files) to submit your work.
- GitHub Classroom repository link: `https://classroom.github.com/a/3FKw8tMD`
- Useful commands:
  - `git add -A`
  - `git commit -m "Your commit caption here"`
  - `git push`
- Your check-marked assignment is due by 2^{nd} Nov 2020, 11:30 EDT

Consider this...

- Can you create and populate a new Mongo database?
- Can you write sophisticated queries in your database?