Objectives

- Backend: Data stores
  - MongoDB

Review

- Any questions about the final project and next steps?
- PostgreSQL
  - How does the Dockerfile work?
  - Not explicitly discussed: what is a database's *schema*?

Project Timeline

Objective	Deadline	% of Final Project
Tool preferences (individual)	March 21, 11:59 p.m.	1%
Preliminary Exploration	March 29, 11:59 p.m.	5%
Presentation	Last week of classes	49%
Web page(s)	Tues of finals week	16%
Analyses (individual) • Tool • Team • Individual reflection	End of Finals Period	29%

3

MONGODB

4

Overview

- An open source and document-oriented database
- Data is stored in BSON (binary JSON)
  - NoSQL: *Documents* rather than *records*
  - BSON Benefits: provide additional data types, ordered fields, efficient for encoding and decoding within different languages
- Designed for both scalability and developer agility

5

Terminology: SQL → MongoDB

- Database → Database
- Table → Collection
- Record / Row → Document
- Column → Field

6

MongoDB: Dynamic Schemas

- ***Collections*** hold a bunch of documents
- Number of fields, content and size of the document can differ from one document to another
- Same for Elasticsearch

mongodb

- To show all the databases: `show dbs`
- To show the collections in the database:
  - `show collections`
  - `db.getCollectionNames()`
- To switch database: `use database`

CRUD

- Create
  - db.*collection*.insertOne/Many(<document>)
- Read
  - db.*collection*.find(<query>, <projection>)
  - db.*collection*.findOne(<query>, <projection>)

Which fields you want

- Update
  - db.*collection*.updateOne/Many(<filter>, <update>, <options>)
- Delete
  - db.*collection*.deleteOne/Many(<filter>, <options>)

CRUD Examples

```
> db.user.insertOne({
    first: "John",
    last : "Doe",
    age: 39
})
```

```
> db.user.find ({
    "first" : "John",
    "last" : "Doe",
    "age" : 39
});
```

```
> db.user.updateOne(
    {"_id" : ObjectId("51…")},
    {
      $set: {
        age: 40,
        salary: 7000}
    }
)
```

```
> db.user.removeOne({
    "first": /^J/
})

   DELETE * FROM user
   WHERE first LIKE "J%";
```

Search: find()

- db.bank.findOne() – finds the first one
- db.bank.find()
- db.bank.find().pretty()
  - Deprecated in MongoDB 5
- db.bank.find({lastname: "Johnson"})

What Do These Queries Do?

1. db.products.find({ baseprice : { $gt : 4.99 }})
2. db.products.find({ baseprice : { $lte : 4.99 }})
3. db.products.find({ promotions : { $lte : 4.99 }})
4. db.products.find({ colors : { $in : ["red"] }})
5. db.products.find({ "promotions.coupon" : { $in : ["XY678"] }})
6. db.products.find({ $and : [{ category : "toys"},{ baseprice : { $gt : 4.99 }}]}

In-Class Exercise

Benefits

- Speed!
- Rich dynamic queries
- Lazy creation
- Schema-less
- Returns JSON
- Easy Replication and Failover
- Auto-Sharding
- MapReduce

Limitations

- No Transactions
  - Not good for purchases, banking, inventory
- RAM intensive
- Awkward (for lack of a better word) JOINs
  - Not what MongoDB is for
- No referential integrity
- Eventual Consistency

15

Schema Design

- SQL: Optimizing how data is stored
- MongoDB: Optimize how data is used

- SQL: What answers do I have?
- MongoDB: What questions do I have?

16

Looking Ahead

- Rinn Joireman on testing
- Patrick Reynolds on Github Actions
  - AMA: 2:30-3:30