CS5220 Advanced Topics in Web Programming

Introduction to MongoDB

Chengyu Sun
California State University, Los Angeles

NoSQL

- Not SQL, Not Only SQL, *Not Relational*
- A term that describes a class of data storage and manipulation technologies and products that do not follow the RDBMS principles and focus on large datasets, performance, scalability, and agility

Types of NoSQL Databases

- Key-Value Stores
- Column Family Stores
- Graph Databases
- Document Databases
  - A *document* in a document database consists of a loosely structured set of key-value pairs.

A "Document" Example

```
{
    "first_name": "John",
    "last_name": "Doe",
    "age": 20,
    "address": {
        "street": "123 Main"
        "city": "Los Angeles"
        "state": "CA"
    }
}
```

- It's basically JSON
- Why is it called a *document*, not a *object*??

DBMS Popularity

- DB-Engines.com
  - Ranking
  - Trend

MongoDB

- The most popular NoSQL database
  - Document database
  - BSON data types
- The "M" in MEAN/MERN stack

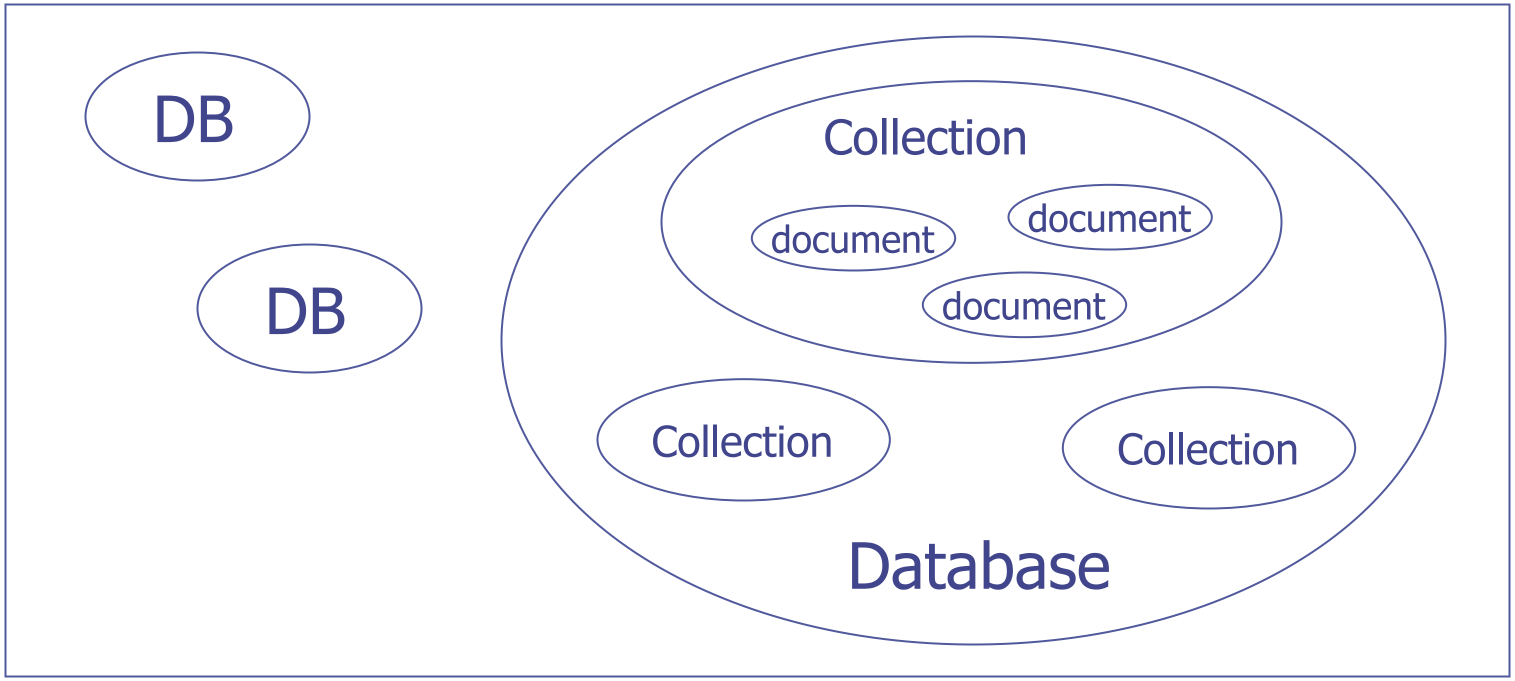
MongoDB Installation

- Install MongoDB Community Edition
  - No security by default
  - Localhost binding (default since 3.6)
- Install MongoDB Compass
  - MongoDB Windows Installer installs Compass by default

Basic Usage of MongoDB Compass

- Connection
- Create databases and collections
- Document CRUD

MongoDB Server



- A collection is the equivalent of a table in relational databases
- Collection does *not* enforce a schema

Blog Example

- Users
- Articles
- Comments
- Tags

Data Modeling

- Understand MongoDB data types
- Knowledge of relational modeling still applies
- To embed or not to embed: that is the question

Data Modeling (I): Data Types

- Numbers
  - Boolean, Integer, Double, Decimal
- Text and binary data
  - String, Regular Expression, Code, Binary Data
- Date and timestamp (for internal use)
- Special types
  - Null, Min/Max Key, ObjectId
- Object and Array

A MongoDB Document …

```
{
  _id: ObjectId("5a09e956df8d3a91d14628d4"),
  title: "My First Blog Post",
  publishDate: 2018-03-31 20:00:00:00,
  author: {
    name: "John"
    email: "john@localhost"
  },
  tags: ["web development", "mongodb"]
}
```

… A MongoDB Document

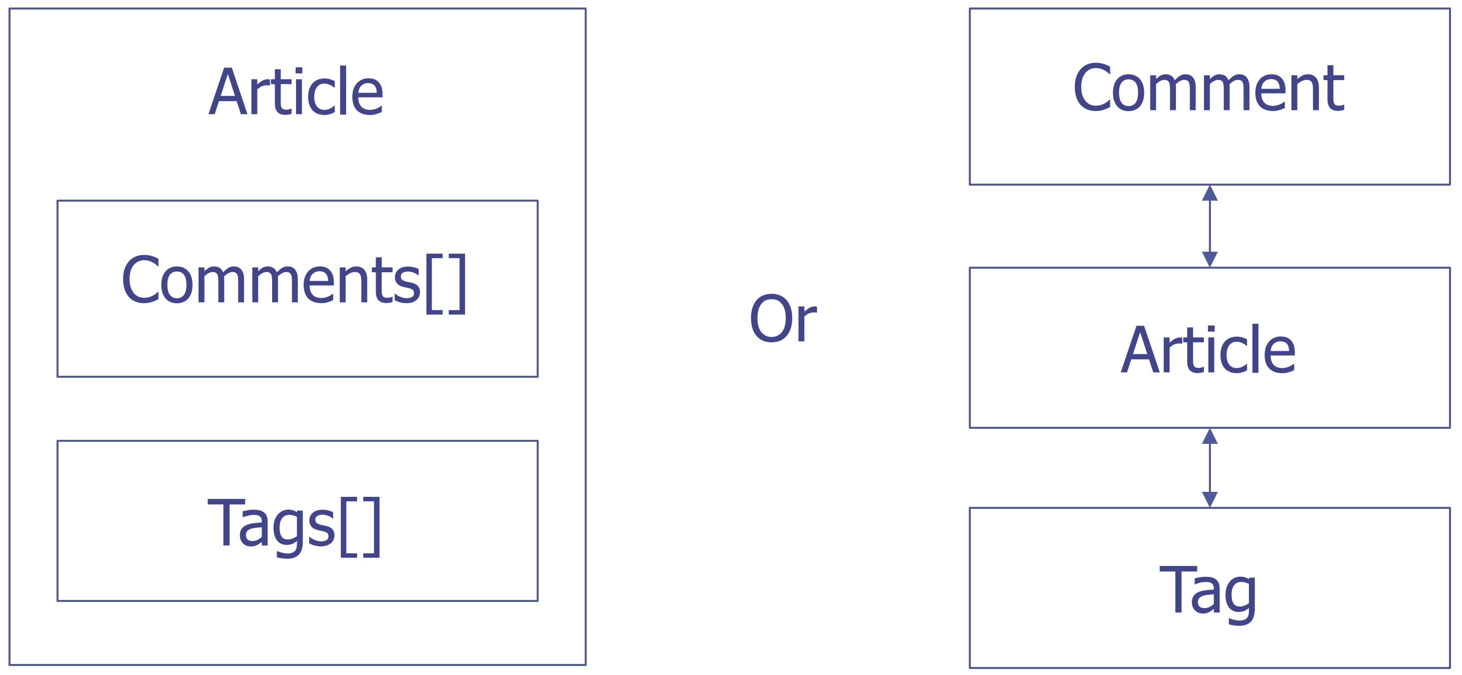
- Each document must have a unique `_id` field that serves as the primary key
- The value of `_id` can be user-assigned (of any type) or auto-generated (of ObjectId type)
- An `Object` value is also known as an *embedded document* or a *sub-document*

Data Modeling (II)

- All the relational design knowledge still applies
  - Entities and relationships
  - Tables → collections
- With that said, document databases are not relational databases
  - Doing JOIN is difficult
  - Does have much richer data types

Data Modeling (III)

- To embed or not to embed: that's the question



Embed Or Not Embed

- Read performance vs data redundancy
  - Basically the same arguments as *normalized* data vs *de-normalized* data in relational design
  - Examples:
    - Article and article author
    - Comment and comment author
- Atomicity requirements

Need for Transactions …

- Not all operations can be done with a single operation, e.g. transferring money from one bank account to anther:

```
-- 1. Check the balance of account #1
select balance from accounts where id = 1;

-- 2. Withdraw $100 from account #1
update accounts set balance = balance – 100
        where id = 1;

-- 3. Deposit $100 to account #2
update accounts set balance = balance + 100
        where id = 2;
```

… Need for Transactions …

- Bad things could happen due to concurrent access and/or system failure

```
-- 1. Check the balance of account #1
select balance from accounts where id = 1;

-- 2. Withdraw $100 from account #1
update accounts set balance = balance – 100
        where id = 1;

-- 3. Deposit $100 to account #2
update accounts set balance = balance + 100
        where id = 2;
```

The other owner of the joint account withdraws all the money in account #1

… Need for Transactions …

- Bad things could happen due to concurrent access and/or system failure

```
-- 1. Check the balance of account #1
select balance from accounts where id = 1;

-- 2. Withdraw $100 from account #1
update accounts set balance = balance – 100
          where id = 1;

-- 3. Deposit $100 to account #2
update accounts set balance = balance + 100
          where id = 2;
```

The other owner of the joint account checks the balances of both accounts and notices that $100 is missing

… Need for Transactions

- Bad things could happen due to concurrent access and/or system failure

```
-- 1. Check the balance of account #1
select balance from accounts where id = 1;

-- 2. Withdraw $100 from account #1
update accounts set balance = balance – 100
        where id = 1;

-- 3. Deposit $100 to account #2
update accounts set balance = balance + 100
        where id = 2;
```

$\xleftarrow{\textit{System crash}}$

Transaction and ACID

- A transaction in RDBMS is a group of SQL statements treated by the DBMS as a *single unit of work*
- Transactions in RDBMS are ACID
  - Atomic
  - Consistent
  - Isolated
  - Durable

Transactions in MongoDB

- Write operations are atomic on the level of a single document
- Multi-documents transaction support
  - For replica sets since 4.0 (6/2018)
  - For sharded clusters since 4.2 (8/2019)

MongoDB Shell

- `mongo`
- A command line client that provides an interactive JavaScript interface to MongoDB

Basic MongoDB Shell Commands

- `help`
- `show dbs`
- `use <db>`
  - Switch to database `<db>`
  - `<db>` won’t be created until some data is inserted into it
  - `show collections`
  - `db.dropDatabase()`

MongoDB's Query Language

- JavaScript with [MongoDB methods]
- Some collection methods:
  - db.<collection>.insert()
  - db.<collection>.find()
  - db.<collection>.update()
  - db.<collection>.remove()
  - db.<collection>.drop()

Example of Basic Operations

- Create a database `test1`
- Create two documents `John` and `Jane`
- Save the two documents to a collection `users`
- List the documents in the collection

Mongo Shell Script

- Example: `test2.js`
  - connect(<url>)
  - print()
  - printjson()
  - cursor
- Run Mongo shell script
  - `mongo <script>`, or
  - `load("<script>")` inside Mongo shell

About Mongo Shell Script

- Don't use shell helper commands like `show dbs` as they are not valid JavaScript
- Don't use fancy JavaScript syntax as the may not be supported by Mongo shell's JavaScript interpreter

Using find()

- `find( [query], [projection] )`
  - [Query tutorials]()
  - [Query operators]()
- Examples using the Blogs database

Queries (I) Basic Conditions and Projections

- List all users
- List the first name of all users
  - Without _id
- Find the users whose last name is Doe
  - Using comparison operators

Queries (II) Logical Operators

- Find the users whose first name is John and last name is Doe
  - Implicit and explicit `$and`
- Find the users whose first name is John or last name is Doe
- Find the users whose first name is John or (the first name is Jane and the last name is Doe)

Queries (III) Arrays and Sub-documents

- Find the articles whose tags contain "NoSQL"
  - Using `$all`
- Find the articles John Doe has commented on

Queries (IV) Join

- List the articles with their authors (i.e. not just author id)
  - db.<collection>.aggregate()
  - [$lookup]
- List the article authors
  - [$project]

Update and Delete

db.<collection>.update(query, update, options)
db.<collection>.remove(query, update, options)

- Change John Doe's name to Tom Smith
  - $set
  - Other update operators
- Delete the article "Using MongoDB"
- Add a tag "Tutorial" to the article "Programming Node.js"
  - $push
- Delete the comments made by John Doe
  - $pull

Indexing

- Indexes are crucial for performance just like in RDBMS
- db.<collection>.createIndex(<keys>, [options])
  - keys: {field: -1|1, field2: -1|1, …}
  - options: unique, name, …

Index Examples

Ascending order

```
db.users.createIndex({lastName: 1});

db.users.createIndex({email: 1}, {
  unique: true,
  name: 'EmailIndex'
});

db.articles.createIndex({tags: 1});
```

Index on array field

MongoDB with Node.js

- MongoDB Node.js driver
- [Mongoose]
  - Model classes
    - Validation
  - DAO methods

Using MongoDB Driver

- `npm install mongodb`
- Understand the API
  - Connection string format
  - MongoClient, Db, Collection, Cursor

About Using MongoDB Driver

- Difference between Node.js code and MongoDB shell script
- MongoClient maintains its own connection pool – reuse the same client/db/collection as much as possible before closing it

Support for Other Programming Languages

- Drivers for various server-side programming language – https://docs.mongodb.org/ecosystem/drivers/

NoSQL vs Relational …

- NoSQL databases are designed to be easier to scale horizontally
- Document databases make data modeling easier and data access more efficient for certain applications
- Using one language (i.e. JavaScript) for everything is appealing
- But …

… NoSQL vs Relational

- Giving up ACID, SQL, and tried-and-true reliability of RDMBS is no small sacrifice
- Scalability and performance of RDBMS have continued to improve
- Understanding application requirements and data modeling is important for both
- Use the right database for the right job

Readings

- MongoDB Manual
- MongoDB Node.js Driver
- Why SQL is beating NoSQL