CS143: MongoDB (NoSQL)

Book Chapters

(7th) Chapter 10.2

MongoDB

- Database for JSON objects
  - “NoSQL database”
- Schema-less: no predefined schema
  - MongoDB will store anything with no complaint!
  - No normalization or joins
  - Use Mongoose for ensuring structure in the data
- Adopts JavaScript philosophy
  - “Laissez faire” policy
    * Don’t be too strict! Handle user request in a “reasonable” way
  - Both blessing and curse

Document in MongoDB

- Data is stored as a collection of documents
  - Document: (almost) JSON object
  - Collection: group of “similar” documents

- Example

```json
{
  "_id": ObjectId("8df38ad8902c"),
  "title": "MongoDB",
  "description": "MongoDB is NoSQL database",
  "tags": ["mongodb", "database", "NoSQL"],
  "likes": 100,
  "comments": [
    {
      "user": "lover",
      "comment": "Great book!"
    },
    {
      "user": "hater",
      "comment": "Worst ever!"
    }
  ]
}
```
• **_id** field: primary key
  – Its value must be unique in the collection
  – May be of any type other than array
  – If not provided, _id is automatically added with a unique ObjectId value

• Stored as BSON (Binary representation of JSON)
  – Supports more data types than JSON
  – Does not require double quotes for field names

• Analogy
  – Document in MongoDB ≈ row in RDB
  – Collection in MongoDB ≈ table in RDB

**MongoDB vs RDB**

**MongoDB document**
• Preserves structure
  – Nested objects
• Potential redundancy
• Hierarchical view of a particular app
• Retrieving data with different “view” is difficult

**RDB relation**
• “Flattens” data
  – Set of flat rows
• Removes redundancy
• Flat schema based on the intrinsic nature of data
• Easy to obtain different “view” using efficient “joins”

**Basic MongoDB Commands**

• Basic administration
  – *mongo*: start MongoDB shell
  – *use <dbName>*: use the database
  – *show dbs*: show list of databases
  – *show collections*: show list of collections
  – *db.colName.drop()*: delete colName collection
  – *db.dropDatabase()*: delete current database

• CRUD operations
  – Create: *insertOne()*, *insertMany()*
Retrieve: `findOne`, `find`
Update: `updateOne`, `updateMany`
Delete: `deleteOne`, `deleteMany`

**MongoDB commands for CRUD**

- Create: `insertX(doc(s))`
  
  ```
  db.books.insertOne({title: "MongoDB", likes: 100})
  db.books.insertMany([{{title: "a"}, {title: "b"}}])
  ```

- Retrieve: `findX(condition)`
  
  ```
  db.books.findOne({likes: 100})
  db.books.find({$and: [{likes: {$gte: 10}}, {likes: {$lt: 20}}]})
  ```

  - `findOne()` returns the first (?) matching document for multiple matches
  - Other boolean/comparison operators: `$or`, `$not`, `$gt`, `$ne`, ...

- Update: `updateX(condition, update_op)`
  
  ```
  db.books.updateMany({title: "MongoDB"}, {$inc: {likes: 1}})
  ```

  - Other update operators: `$mul` (multiply), `$unset` (remove the field), ...

- Delete: `deleteX(condition)`
  
  ```
  db.books.deleteOne({title: "MongoDB"})
  db.books.deleteMany({likes: {$lt: 100}})
  ```

**Index**

- Indexes can be built for efficient retrieval
  - `db.books.createIndex({title:1, likes:-1})`

  - Create one index on combined attributes “title” and “likes”
  - 1 means ascending order, -1 means descending order

**More on MongoDB**

- We learned just the basic

  - Enough for our project

- But MongoDB has many more features:

  - Aggregate queries
  - Transactions
  - Replication
- (Auto)sharding
- ...

- Read MongoDB documentation and online tutorials to learn more