12
NoSQL and MongoDB

Adding data persistence to our Node/Express app

- So far we have been using data we hard coded into arrays.
- If we want to make an app that is useful to others we will need the ability to store and retrieve data between user sessions.
- For this app we will look at using MongoDB to store our data.
- We will read data from the database on our HTTP GET routes.
- We will write new data to the database on our HTTP POST routes.
- We will change existing data in the database on our HTTP PUT routes.
- We will remove data from the database on our HTTP DELETE routes.

Relational Databases (RDBMS)

- Relational model conceived in the early 1970s - IBM E.F. Codd et al.
- Makes use of Tables, Columns, Keys, Relationships, Indices, SQL
- Schema provides structure to data
- Data is normalized (duplication is minimized)
- Joins are used to collect related data from different tables
- Referential Integrity can be enforced easily due to the database design
- Transactional in nature - locking used to ensure data is not corrupted
- Can be difficult to scale across many nodes

SQL

- Structured Query Language
- The language used to interact with relational databases and their data
- Two flavors - Data Definition Language (DDL) and Data Manipulation Language (DML). DML is what we use to write CRUD statements.
- DDL Example:
  - CREATE TABLE Students (Id, Name, Major, GPA, Year)
- DML Example:
  - SELECT Id, Name FROM Students WHERE GPA > 3.0
- NoSQL databases still provide support for querying and the syntax can be similar to SQL.

NoSQL Databases

- “No SQL” or “Not-Only SQL”. We will focus on document-oriented databases
- No schema - data is organized as a collection of documents
- Documents do not all have to be the same (no schema)
- Collections provide a way to group related documents
- Some data duplication can occur if relationships exist between documents
- Highly scalable across many nodes using sharding
- Can easily scale out horizontally using cheap hardware without performance penalty - think Facebook, Snapchat, Google Docs, etc
- Tradeoffs include loss of referential integrity - would have to use application level software to verify the relationships between data

ACID

- Atomicity, Consistency, Isolation, Durability
- Atomicity - Transactions must be atomic (all or nothing) - they either succeed or fail, even when the power goes out, database crashes, etc.
- Consistency - Data is always left in a consistent, valid state
- Isolation - Each transaction should not affect other transactions
- Durability - Transactions will be recorded permanently regardless of system failure, loss of power, no data loss

CAP Theorem

- CAP is an acronym for Consistency, Availability, and Partition Tolerance

- The CAP Theorem states that you can pick 2 out of the 3 properties, but never all 3 at once.

- Consistency - Every read receives the most recent write, or an error
- Availability - Every request receives a (non-error) response – without guarantee that it contains the most recent write
- Partition Tolerance - The system continues to operate despite an arbitrary number of messages being dropped (or delayed) by the network between nodes, or nodes failing (crashing)

BASE

- Basically Available, Soft state, Eventual Consistency
- Whereas ACID prioritizes Consistency over Availability, BASE takes the opposite approach and prioritizes Availability over Consistency
- Eventual Consistency - Data has to be reconciled so the state of all the nodes are consistent. Until that happens reads can return inaccurate or unpredictable data.
- Important to understand there are trade-offs made when scaling a data storage system across many nodes

MongoDB

- MongoDB is an open source document-oriented NoSQL database using JSON formatted documents.
- The M in the MEAN stack - completes the JavaScript all the way paradigm.
- Dynamic Schema - documents do not all have to have the same fields.
- One of the most popular open source NoSQL databases in use.
- Can query data by field, range, and regular expressions.
- Can use MongoDB in hosted cloud services to store data for web applications - https://mlab.com/
- Mongoose and other ODM software can provide a modeling layer for web apps built with Node and Express.

Installing MongoDB on Windows

https://docs.mongodb.com/manual/tutorial/install-mongodb-on-windows/

- Download the .msi installer and run it
- Change the install location to C:\mongodb
- Create a new directory C:\mongodb\data\db
- Run the following command: mongod --directoryperdb --dbpath C:\mongodb\data\db --logpath C:\mongodb\log\mongo.log --logappend --rest --install
- On Windows MongoDB runs as a Windows Service. net start MongoDB
- Run the MongoDB shell: mongo

Installing MongoDB on MacOS

https://docs.mongodb.com/manual/tutorial/install-mongodb-on-os-x/

- To install MongoDB using Homebrew - brew install mongodb
- Create the database directory - mkdir -p /data/db
- Set permissions for the database directory - sudo chown <user> /data/db/
- Start the MongoDB process - mongod
- In another terminal run the MongoDB shell - mongo
- Test that everything works - show dbs
- Create a new database - use customers
- Create a new user - db.createUser({ user: “username”, pwd: “pwd”, roles: [“readWrite”, “dbAdmin”]});

Creating a MongoDB user

```
  show dbs
admin  0.000GB
local  0.000GB
> use customers
switched to db customers
> show dbs
admin  0.000GB
local  0.000GB
> db
customers
> db.createUser({
...     user: "student",
...     pwd: "1234",
...     roles: [ "readWrite", "dbAdmin" ]
... });
Successfully added user: { "user" : "student", "roles" : [ "readWrite", "dbAdmin" ] }
>
```

Creating collections and inserting documents

```
> db.createCollection("customers");
{ "ok" : 1 }
> show collections
customers
> db.customers.insert( { first_name: "Derick", last_name: "Beckwith" } );
WriteResult({ "nInserted" : 1 })
> db.customers.find();
{ "_id" : ObjectId("58e9bcf150b1909b6ab14eeb"), "first_name" : "Derick", "last_name" : "Beckwith" }
> db.customers.find().pretty();
{
        "_id" : ObjectId("58e9bcf150b1909b6ab14eeb"),
        "first_name" : "Derick",
        "last_name" : "Beckwith"
}
> _
```

Updating documents

```
> db.customers.update( { first_name: "Derick" }, { first_name: "Jimmy", last_name: "McGill", occupation: "Attorney" } );
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
> db.customers.find().pretty();
{
        "_id" : ObjectId("58e9bcf150b1909b6ab14eeb"),
        "first_name" : "Jimmy",
        "last_name" : "McGill",
        "occupation" : "Attorney"
}
>
```

Removing documents

```
> db.customers.remove( { first_name: "Jimmy" }, { justOne: true } );
WriteResult({ "nRemoved" : 1 })
> db.customers.find().pretty();
> _
```

MongoJS

- MongoJS is an npm package that makes connecting our Node/Express app to a MongoDB database much easier.
- To install MongoJS - npm install mongojs --save
- To require the dependency - var mongojs = require('mongojs')
- To configure the database connection - var db = mongojs('students', ['students'])
- To retrieve documents from a collection - db.students.find(function (err, docs)
- To insert a new document - db.students.insert(student, function (err, result)
- To update an existing document - db.students.update(student, function(err...
- To remove a document - db.students.remove({ student._id }, function(err...)

Steps to delete a rest resource (student in this app)

- Add a Delete link onto each student in the UI.
- Add a function using jQuery to send an ajax http delete request for the specific document id the user clicked on.
- Add a delete route in our Express app to handle the ajax delete request.
- Remove the deleted student document from the database and redirect the browser back to the index view.

Steps to update a rest resource (student in this app)

- Add an update link in the index view beside each student.
- Add an update view containing a form with fields populated from the selected student object and an update button to submit the changes in the form.
- Since we are not using a front-end JS framework, we can use jQuery to send an ajax XHR request to the PUT route.
- Add a PUT route in our Express app to handle the ajax request.
- Update the student in the database with the version of data in the request.

Additional NoSQL Resources

- NoSQL Introduction - https://www.thoughtworks.com/insights/blog/nosql-databases-overview
- NoSQL on Wikipedia - https://en.wikipedia.org/wiki/NoSQL
- List of various NoSQL databases - http://nosql-database.org/

- Document oriented databases - https://en.wikipedia.org/wiki/Document-oriented_database
- Explanation of ACID vs BASE - http://www.dataversity.net/acid-vs-base-the-shifting-ph-of-database-transaction-processing/

Additional MongoDB Resources

- MongoDB home - https://www.mongodb.com/
- MongoDB code - https://github.com/mongodb/mongo
- Setting up MongoDB - https://www.youtube.com/watch?v=pWbMrx5rVBE
- Setting up MongoJS - https://github.com/mafintosh/mongojs
- Using MongoDB with NodeJS - https://www.youtube.com/watch?v=5if-d-JhRKc