Basic MongoDB Tutorial

NoSQL (Not Only SQL) was designed to compensate for the major weakness of relational databases, namely; their inability to cope with large scale data storage and their inability to deal with dynamic storage of data. NoSQL databases are unstructured and normally have dynamic schemas allowing the storage of data to be flexible and the databases are designed to be distributed meaning they can meet the scalability and storage needs of storing large amounts of data.

MongoDB is a popular NoSQL database, namely a document oriented database.

Installing

Install your shit, its not that hard. Dont forget to add the *data/db* folder somewhere

Download MongoDB for Windows

Tutorial

First download the following simple datasets:

Restaurants (Point data)

Neighborhoods (Polygon data)

1. **Loading The Spatial Data**

```
mongoimport <path to restaurants.json> -c restaurants
mongoimport <path to neighborhoods.json> -c neighborhood
```

2. **Indexing**

   - Note: that Mongo automatically indexes *_id*
   - Indexing Spatial Data

```
db.restaurants.createIndex({ location: "2dsphere" }) //Create spatial
index
```

   - Single Field Indexing

```
// There's no difference in performance whether you make it ascending
or descending
db.restaurants.createIndex({ name: -1 }) //create single field
descending index
```

- Compound Indexing

```
// Create descending index on name then creates index on location field
db.restaurants.createIndex({ name: -1,  location: 1})
```

3. **Basic Selections**

- Find One

```
db.restaurants.findOne()
```

- Find All

```
db.restaurants.find()
```

- Find Filter

```
// Find restaurants that have the words mori in the name
db.restaurants.find({"name": /.*mori.*/})
// Find exact match
db.restaurants.find({name: "Morris Park Bake Shop"})
// OR filter
db.restaurants.find( { $or: [ { name: "Amorino" }, { name: "Morris Park
Bake Shop" } ] } )
// AND filter
db.restaurants.find( { "name": "Amorino", "location.type": "Point" })
// Note the use of dot notation to search for nested JSON objects
```

4. **Finding Features In A Given Bounding Box**

- To find Points within a bounding box make use of the below query. The $box takes two params, the bottom left and upper right coordinates

```
db.restaurants.find( {
    location: { $geoWithin: { $box:  [ [ 0, 0 ], [ 200, 100 ] ] } }
} )
```

5. **Intersections**

- Query below will find the first polygon that intersects with the givin point location

```

```

```
db.neighborhoods.findOne({ geometry: {
    $geoIntersects: {
        $geometry: { type: "Point", coordinates: [ -73.93414657,
40.82302903 ]
        }
    }
} })
```

6. **Selecting Features Based On Their Distance From Another Feature**

   - Note MongoDB makes use of Miles instead of Kilometers when calculating distance
   - There are two methods of getting features based on distance (Ordered and Unordered)
   - Ordered: Will return results from nearest to farthest from central point

```
// Finds all restaurants within 2 miles of center point
var METERS_PER_MILE = 1609.34
db.restaurants.find({
    location: {
        $nearSphere: {
            $geometry: { type: "Point", coordinates: [ -73.93414657,
40.82302903 ] },
            $maxDistance: 2 * METERS_PER_MILE
        }
    }
})
```

   - Unordered

```
// Finds all restaurants within 2 miles of center point
// $centerSphere takes 2 params: the center point and radius in radians
(divide radius by radius of earth in miles)
db.restaurants.find({
    location:{
        $geoWithin:{
            $centerSphere: [ [ -73.93414657, 40.82302903 ], 2 / 3963.2
] }
    }
})
```

7. **Extra**

Reference

Alejandro Corbellini, Cristian Mateos, Alejandro Zunino, Daniela Godoy, Silvia Schiaffino,Persisting big-data: The NoSQL landscape,Information Systems,Volume 63,2017,Pages 1-23,ISSN 0306-4379, https://doi.org/10.1016/j.is.2016.07.009. (http://www.sciencedirect.com/science/article/pii/S0306437916303210)