NoSQL DBs and MongoDB

DATA SCIENCE BOOTCAMP

Terminology

DBMS: Database management system

- Software which controls the storage, retrieval, deletion, security, and integrity of data within the database
- Examples: MySQL, mongoDB

RDBMS: Relational database management system

- Relational database stores data in tables
- Organized in columns
- Each column stores one type of data

Terminology

CRUD: basic DB functionality

Create, read, update, delete

Schema:

A method of data modeling; a framework that describes the relationships in your data, how they are stored in tables, and how tables relate to each other

Principles of Relational Databases

Schemas are planned in advance and are relatively static.

- Changes require tacking on new tables and joins, or complete schema overhauls

Data for a single entity can be split among many tables

- Reassembled using link tables and joins

Issues with relational databases

Slow or expensive to reassemble fragmented data quickly

- One machine is best – sometimes must be one extremely large system
- Multiple machines require difficult technical overhead, expertise, and maintenance, vulnerable to downtime in any one piece of the system

Enter: Non-relational databases

NoSQL = “Not Only SQL”

Some examples of NoSQL databases:

- Document databases: mongoDB, couchDB
- Key-value stores: Riak, Voldemort, Redis
- Graph databases: Neo4j, HyperGraph
- Wide-column stores: Cassandra, HBase

mongoDB

Mongo is the most popular NRDBMS / NoSQL database

Source: http://db-engines.com/en/ranking

Mongo concepts

Stores information in *documents* rather than in rows

- Documents are data structures like objects, dictionaries, hashes, maps, associative arrays

MongoDB documents are BSON documents

- JSON = javascript serial object notation
- BSON = binary (javascript) serial object notation

mongoDB document

```
{
    one_field: one_value,
    another_field: [an,
                    array,
                    of,
                    values]
}
```

mongoDB document

```
{
    name: “Sue”,
    age: 20,
    status: “A”,
    groups: [“news”, “sports”]
}
```

Mongo concepts

Dynamic schemas:

- New fields can be entered on-the-fly
- No enforcement of pre-defined columns

"Horizontal scalability"

- "Sharding": data may be spread across multiple machines
- Replication and fault tolerance

Mongo concepts

Unstructured data

- Well-suited for holding sloppy information like text, web pages, etc.
- CRUD operations also allow for storage now, structure later

Semi-structured data

Fields in document databases can be:

- added on the fly
- present or absent
- lists, subdocuments (hierarchical), links, etc.

SQL-to-mongo phrasebook

SQL	**Mongo**
database	database
table	collection
row	document
column	field
index	index
table joins	embedded documents / linking

More at: http://docs.mongodb.org/manual/reference/sql-comparison/

Consider using a NoSQL database like MongoDB instead of a Relational Database like MySQL when:

- You don’t have a predetermined schema for your data, and instead need something more flexible
- You don’t really need to do joins between databases from different servers
- Your data is rather large (5-10 GB per table or more if you put it in a SQL database)

Getting started in mongo

```
brew update
brew install mongodb
```

--or--

```
brew unlink mongodb
brew install –upgrade mongodb
```

Run the mongo server

First time?

```
sudo mkdir -p /data/db
```

(check permissions – user needs readwrite access)

```
sudo chmod -R ugo+rw /data/db
```

start service:

```
mongod
```

service runs in the background while we do our business elsewhere.

More troubleshooting:

http://docs.mongodb.org/manual/tutorial/install-mongodb-on-os-x/

Try out the mongo console

```
> mongo
> show dbs
> use dsbc
> show dbs
```

Try out the mongo console

```
> j = { name: “Eddie” }
> k = { name: “Felicity” }
> l = { nationality: “British” }
> db.testData.insert(j)
> db.testData.insert(k)
> db.testData.insert(l)
> show dbs
> show collections
```

Try out the mongo console

```
> db.testData.find()
> td = db.testData
> td.find()
```

Try out the mongo console

Cursors

```
> var c = td.find()
> while ( c.hasNext() )
    printjson( c.next() )
> td.find( {x: { $lt: 20 } } )
```

Try out the mongo console

Insert documents with a for loop

```
> for (var i=1; i<=25; i++) {
...  td.insert( {x : i } )
...  }
> td.find()
> it
```

← For more records (stands for "iterate")

Try out the mongo console

Queries

```
> td.find( {x: {$lt : 20 } } )
> td.find(
              {$or:
                [
                  { name: “Eddie”},
                  {x: { $gt : 22 } }
                 ]
               }
             )
```

Query tutorial: http://docs.mongodb.org/manual/tutorial/query-documents/

pymongo

```
> pip install [--upgrade] pymongo
```

(warning: still need a mongod instance running)

Try out pymongo

```
> ipython
[1] from pymongo import MongoClient
[2] client = MongoClient()
[3] db = client.dsbc
[4] db.collection_names()
[5] td = db.testData
[6] cursor = td.find()
[7] cursor.next()
[8] list(cursor)
```

Try out pymongo with: Heavy metal movies

```
> ipython
[1] from pymongo import MongoClient
[2] client = MongoClient()
[3] import pickle
[4] with open('heavy_metal_parsed.pkl', 'r') as infile:
          reviews = pickle.load(infile)
[5] reviews[0].keys()
[6] reviews[0]
[7] len(reviews)
```

data source: http://www.bazillionpoints.com/shop/heavy-metal-movies-by-mike-mcbeardo-mcpadden/

Heavy metal movies >> mongo

```
> ipython
 [1] hmm = client.dsbc.hmm
 [2] hmm.insert(reviews[0])
 [3] hmm.find().next()
 [4] for review in reviews[1:]:
           hmm.save(review)
 [5] cursor = hmm.find()
 [6] len(list(cursor)) // beware
 [7] hmm.count()
```