ISQA 8080 Homework 2 - MongoDB

May 22, 2016

1 ISQA 8080 Homework 2.2

Brian Detweiler

```
In [95]: import pymongo
         from pymongo import MongoClient
         import datetime
         import re
         from pymongo import InsertOne, DeleteOne, ReplaceOne
         import datetime

         client = MongoClient()
         client = MongoClient('mongodb://localhost:27017/')
         db = client.homework2
         users = db.users
         movies = db.movies
```

2 Tasks

- The fourth task is to upload a dataset and to write and run different queries on this dataset.
- The fifth task starts with Redis. First, installation and running Redis either on your own computer or running it on
- The sixth task demonstrates three examples of using Redis.
- The seventh tasks asks you to describe a solution to use Redis for storing user ratings.
- The eighth tasks asks you to write a reflection on the discussion boards.

3 Deliverables (Summary, Detailed description in the tasks):

- See task four: In the answer sheet "Homework 2 Answer Sheet.docx", write the 10 queries including a screenshot of the command and the first lines of the result.
- See task seven: In the answer sheet "Homework 2 Answer Sheet. Docx", describe how Redis can be used to store user ratings. You need to include screenshots to show how data would be stored as well as retrieved in Redis.
- Submit the Answer Sheet on Blackboard using the link "Homework 2 Part 2: MongoDB and Redis" (where you downloaded the instructions and the text files).
- See task eight: Write a reflection by replying to the thread "Homework Reflection 2" in the discussion board

```
In [19]: movieList = movies.find({"_id": 1})

         for movie in movieList:
             print movie
```

```
{u'category': [u'Animation', u"Children's", u'Comedy'], u'_id': 1, u'year': 1995, u'title': u'Toy Story
```

4 1. Display all occupations

```
In [28]: occupations = sorted(users.distinct(u'occupation'))

         for occupation in occupations:
             print occupation
```

```
K-12 student
academic/educator
artist
clerical/admin
college/grad student
customer service
doctor/health care
executive/managerial
farmer
homemaker
lawyer
other
programmer
retired
sales/marketing
scientist
self-employed
technician/engineer
tradesman/craftsman
unemployed
writer
```

5 2. Chose an occupation and select all users with this occupation. Only show user information and hide the users' movie ratings.

```
In [107]: # We will choose 'technician/engineer' and hide movie reviews - limit to first 10 results
          engineers = users.find({u'occupation': 'technician/engineer'},
                                 {u'movies':0}).limit(10)

          for engineer in engineers:
              print engineer
```

```
{u'gender': u'M', u'age': u'25-34', u'_id': 9, u'zipcode': u'61614', u'occupation': u'technician/enginee
{u'gender': u'M', u'age': u'45-49', u'_id': 44, u'zipcode': u'98052', u'occupation': u'technician/engine
{u'gender': u'M', u'age': u'25-34', u'_id': 61, u'zipcode': u'95122', u'occupation': u'technician/engine
{u'gender': u'M', u'age': u'25-34', u'_id': 82, u'zipcode': u'48380', u'occupation': u'technician/engine
{u'gender': u'M', u'age': u'25-34', u'_id': 93, u'zipcode': u'95825', u'occupation': u'technician/engine
{u'gender': u'M', u'age': u'25-34', u'_id': 94, u'zipcode': u'28601', u'occupation': u'technician/engine
{u'gender': u'M', u'age': u'35-44', u'_id': 100, u'zipcode': u'95401', u'occupation': u'technician/engin
{u'gender': u'M', u'age': u'25-34', u'_id': 116, u'zipcode': u'55744', u'occupation': u'technician/engin
{u'gender': u'M', u'age': u'25-34', u'_id': 117, u'zipcode': u'33314', u'occupation': u'technician/engin
{u'gender': u'M', u'age': u'35-44', u'_id': 118, u'zipcode': u'22315', u'occupation': u'technician/engin
```

6 3. Count the number of men in the database.

We will choose 'technician/engineer' and hide movie reviews

```
In [108]: men = users.find({u'gender': 'M'}).count()

          print "There are " + str(men) + " men in the database."
```

```
There are 2716 men in the database.
```

7 4. Select all users whose zipcode starts with a 5. Only show the user ID and zipcode.

```
In [82]: regex = re.compile("^5.*")
         by_zipcodes = users.find({u'zipcode': regex},
                                  {u'_id':1, u'zipcode': 1}).limit(10)

         for zipcode in by_zipcodes:
             print zipcode
```

```
{u'_id': 3, u'zipcode': u'55117'}
{u'_id': 5, u'zipcode': u'55455'}
{u'_id': 6, u'zipcode': u'55117'}
{u'_id': 20, u'zipcode': u'55113'}
{u'_id': 22, u'zipcode': u'53706'}
{u'_id': 33, u'zipcode': u'55421'}
{u'_id': 54, u'zipcode': u'56723'}
{u'_id': 55, u'zipcode': u'55303'}
{u'_id': 59, u'zipcode': u'55413'}
{u'_id': 63, u'zipcode': u'54902'}
```

8 5. Select all movies from the year 1998 and category comedy.

```
In [83]: comedies_from_1998 = movies.find({u'year': 1998, u'category': u'Comedy'}).limit(10)

         for comedy in comedies_from_1998:
             print comedy
```

```
{u'category': [u'Comedy'], u'_id': 1679, u'year': 1998, u'title': u'Chairman of the Board '}
{u'category': [u'Comedy', u'Crime', u'Mystery', u'Thriller'], u'_id': 1732, u'year': 1998, u'title': u'B
{u'category': [u'Comedy'], u'_id': 1738, u'year': 1998, u'title': u'Vermin '}
{u'category': [u'Comedy'], u'_id': 1746, u'year': 1998, u'title': u'Senseless '}
{u'category': [u'Comedy'], u'_id': 1753, u'year': 1998, u'title': u'Half Baked '}
{u'category': [u'Comedy', u'Thriller'], u'_id': 1764, u'year': 1998, u'title': u'Tainted '}
{u'category': [u'Action', u'Comedy', u'Musical'], u'_id': 1772, u'year': 1998, u'title': u'Blues Brother
{u'category': [u'Comedy', u'Drama'], u'_id': 1774, u'year': 1998, u'title': u'Mass Transit '}
{u'category': [u'Comedy', u'Romance'], u'_id': 1777, u'year': 1998, u'title': u'Wedding Singer, The '}
{u'category': [u'Comedy', u'Romance'], u'_id': 1782, u'year': 1998, u'title': u'Little City '}
```

9 6. Count the number of movies from the year 1990 and 1995.

```
In [104]: movies_between_1990_and_1995 = movies.find({u'year': {'$gte': 1990, '$lte': 1995}}).count()

          print 'There are ' + str(movies_between_1990_and_1995) + ' movies between 1990 and 1995'
```

```
There are 1003 movies between 1990 and 1995
```

10 7. Display all movies published before the year 1992.

```
In [93]: movies_before_1992 = movies.find({u'year': {'$lt': 1992}})\
                                    .limit(10)\
                                    .sort('year', pymongo.ASCENDING)

         for movie in movies_before_1992:
             print movie
```

```
{u'category': [u'Miami Beach (1988)'], u'_id': 2382, u'year': 0, u'title': u'Police Academy 5 Assi'}
{u'category': [u'Adventure', u'Drama'], u'_id': 2821, u'year': 1919, u'title': u'Male and Female '}
{u'category': [u'Action', u'Drama'], u'_id': 2823, u'year': 1919, u'title': u'Spiders, The (Die Spinnen,
{u'category': [u'Comedy'], u'_id': 3132, u'year': 1919, u'title': u'Daddy Long Legs '}
{u'category': [u'Comedy'], u'_id': 3231, u'year': 1920, u'title': u'Saphead, The '}
{u'category': [u'Comedy'], u'_id': 3309, u'year': 1920, u'title': u"Dog's Life, A "}
{u'category': [u'Action'], u'_id': 3310, u'year': 1921, u'title': u'Kid, The '}
{u'category': [u'Horror'], u'_id': 1348, u'year': 1922, u'title': u'Nosferatu (Nosferatu, eine Symphonie
{u'category': [u'Drama'], u'_id': 3195, u'year': 1922, u'title': u'Tess of the Storm Country '}
{u'category': [u'Comedy'], u'_id': 2230, u'year': 1923, u'title': u'Always Tell Your Wife '}
```

11 8. Imagine that you registered for MovieLens. Create a new user with your user data. Do not include any ratings.

11.1 Note: The _id here is auto-generated, as per MongoDB conventions

I have seen ways to grab the max ID and increment it by one, but this is not safe in general, so I am not implementing it here.

```
In [106]: # users.delete_one({u'zipcode': u'68102'})
```

```
Out[106]: <pymongo.results.DeleteResult at 0x7fb3b1207140>
```

```
In [110]: me = users.find_one({u'zipcode': u'68102'})

          if me is not None:
              print me
          else:
              max_id = users.find_one({u'zipcode': u'68102'})
              myself = {u'gender': u'M',
                        u'age': u'25-34',
                        u'zipcode': u'68102',
                        u'occupation': u'technician/engineer'}
              result = db.users.insert_one(myself)
              my_id = result.inserted_id
              print 'created my profile with id ' + str(my_id)
```

```
{u'gender': u'M', u'age': u'25-34', u'_id': ObjectId('574210b15be4bc706e5d3dc1'), u'zipcode': u'68102',
```

12 9. Update the user record you created in the previous query and insert a new rating for a movie of your choice. You will need to use $addToSet (https://docs.mongodb.com/v3.0/reference/operator/update/addToSet/ to add a value that does not exist in an array to the end of the array. You can use Math.round(new Date().getTime()/1000) to calculate the time in Unix format (seconds since start from epoch).

```
In [111]: movie_review = {u'moveID': 3272, # Bad Lieutenant
                          u'rating': 5,
                          u'timestamp': datetime.datetime.utcnow()}
          users.update_one(
            { u'zipcode': u'68102'},
            { "$addToSet":{"movies": movie_review} },
            upsert=True)

          me = users.find_one({u'zipcode': u'68102'})
          print me
```

```
{u'gender': u'M', u'age': u'25-34', u'zipcode': u'68102', u'movies': [{u'rating': 5, u'moveID': 3272, u
```

13 10. A query of your choice.

Here, we'll find users near Omaha

```
In [112]: regex = re.compile("^681.*")
          by_zipcodes = users.find({u'zipcode': regex},
                                   {u'_id':1, u'zipcode': 1}).limit(10)

          for zipcode in by_zipcodes:
              print zipcode
```

```
{u'_id': 385, u'zipcode': u'68131'}
{u'_id': 389, u'zipcode': u'68128'}
{u'_id': 1081, u'zipcode': u'68144'}
{u'_id': 3666, u'zipcode': u'68144'}
{u'_id': 3679, u'zipcode': u'68108'}
{u'_id': 3792, u'zipcode': u'68108'}
{u'_id': ObjectId('574210b15be4bc706e5d3dc1'), u'zipcode': u'68102'}
```