CS421/621 Lab05

MongoDB & Poker API

Objectives:

- A quick overview of MongoDB
- Connecting to MongoDB command line client
- Poker API exercise

MongoDB (or Mongo, for short) is a database that allows us to store data on disk, but not in a relational format. Instead, Mongo is a document-oriented database that conceptually allows us to store objects organized as collections in the JSON or JavaScript Object Notation format. (Technically, MongoDB stores its data in the BSON format, but from our perspective we can think of it as JSON.) In addition, it allows us to interact with it completely in JavaScript smoothly. Mongo can be used for more complex data storage needs, such as storing user accounts or comments in blog posts. It can even be used to store binary data like images. Today we are going to show how to interact with the MongoDB command line client.

Like Redis, Mongo offers a command-line client that allows us to directly interact with the data store. You can fire up the Mongo client by typing "mongo" at the command line of your guest machine:

Firstly, navigate to Chapter7 folder and start Vagrant
$ vagrant up

$vagrant ssh

Now you can type mongo and hit enter to get into mongodb.

```
login as: vagrant
Authenticating with public key "imported-openssh-key"
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.4.0-151-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.

New release '18.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.


Last login: Mon Jun 24 22:54:33 2019 from 10.0.2.2
vagrant@ubuntu-xenial:~$ mongo
MongoDB shell version: 2.6.10
connecting to: test
Server has startup warnings:
2019-06-25T20:33:32.853+0000 [initandlisten]
2019-06-25T20:33:32.853+0000 [initandlisten] ** NOTE: This is a 32 bit MongoDB binary.
2019-06-25T20:33:32.853+0000 [initandlisten] **       32 bit builds are limited to less than 2GB of data (or less with --journal).
2019-06-25T20:33:32.853+0000 [initandlisten] **       See http://dochub.mongodb.org/core/32bit
2019-06-25T20:33:32.853+0000 [initandlisten]
>
```

You may see some start-up warnings when Mongo first launches, but that’s totally normal.

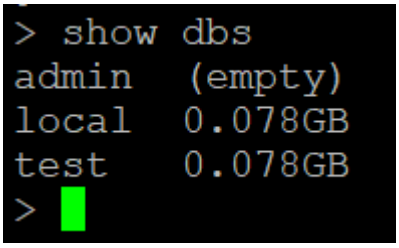
One immediate difference between Redis and Mongo is that we can interact with it using JavaScript. For example, here we create a variable called card and store an object in it:

```
> var card = {"rank":"ace","suit":"clubs"};
> card
{ "rank" : "ace", "suit" : "clubs" }
>
```

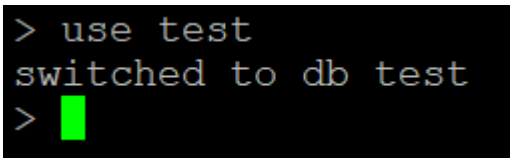
Likewise, we can create and manipulate arrays. Notice that in this example, we don't complete our JavaScript statement before the end of each line. When we press Enter, Mongo responds with three dots letting us know that the previous statement was incomplete. Mongo will automatically execute the first full JavaScript statement:

```
> var clubs = [];
> ["one","two","three","four"].forEach(
... function(rank){
... clubs.push({"rank":rank,"suit":"clubs"})
... });
> clubs
[
        {
                "rank" : "one",
                "suit" : "clubs"
        },
        {
                "rank" : "two",
                "suit" : "clubs"
        },
        {
                "rank" : "three",
                "suit" : "clubs"
        },
        {
                "rank" : "four",
                "suit" : "clubs"
        }
]
>
```

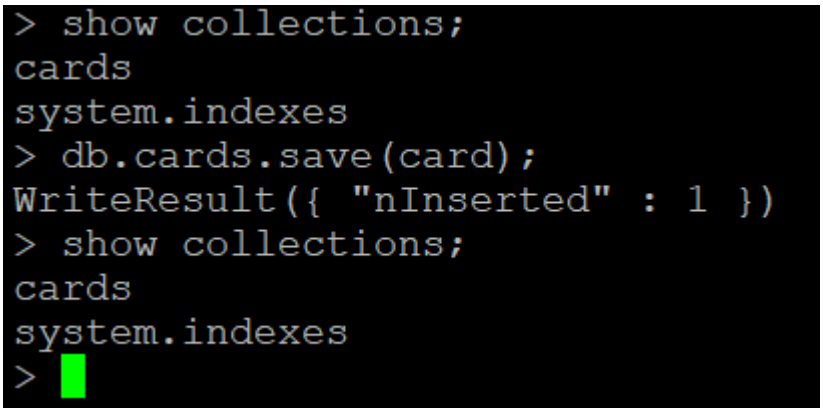
In other words, the Mongo command-line client works a little bit like the JavaScript console in Chrome. But those similarities end when we want to start storing data. Mongo organizes data as documents, which we can think of as JSON objects. It stores collections of documents in databases. We can see the databases in our MongoDB by using the show dbs command:



The local db is always there. We can switch to a different database with the use command:



Once we've selected a database to use, we can access it through the db object. We can save objects to a collection by calling the save function on the collection. If the collection doesn't already exist, Mongo will create it for us. Here, we save the card that we created earlier in our collection:



You'll see that the cards collection doesn't exist before we save an object to it. We can use the collection's find function with no arguments to see what documents are stored:

In addition to rank and suit properties, a card also has an _id associated with it. For the most part, every document in a MongoDB collection has one of these associated with it. In addition to saving a single item, we can also add multiple documents to the collection in one call to save. Here we do that with the club's array that we created previously:

```
> db.cards.save(clubs);
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 4,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
> db.cards.find();
{ "_id" : ObjectId("5d12899cfccc197267ecf944"), "rank" : "ace", "suit" : "clubs" }
{ "_id" : ObjectId("5d128a42fccc197267ecf946"), "rank" : "one", "suit" : "clubs" }
{ "_id" : ObjectId("5d128a42fccc197267ecf947"), "rank" : "two", "suit" : "clubs" }
{ "_id" : ObjectId("5d128a42fccc197267ecf948"), "rank" : "three", "suit" : "clubs" }
{ "_id" : ObjectId("5d128a42fccc197267ecf949"), "rank" : "four", "suit" : "clubs" }
>
```

We can also add more objects to the db:

```
> hearts = [];
[ ]
> ["four","five","six","seven"].
... forEach(function(rank)
... { hearts.push({"rank":rank,"suit":"hearts"})
... });
> db.cards.save(hearts);
BulkWriteResult({
        "writeErrors" : [ ],
        "writeConcernErrors" : [ ],
        "nInserted" : 4,
        "nUpserted" : 0,
        "nMatched" : 0,
        "nModified" : 0,
        "nRemoved" : 0,
        "upserted" : [ ]
})
```

Now, lets find the entire collection of objects by doing db.cards.find();

```
> db.cards.find();
{ "_id" : ObjectId("5d12899cfccc197267ecf944"), "rank" : "ace", "suit" : "clubs"
 }
{ "_id" : ObjectId("5d128a42fccc197267ecf946"), "rank" : "one", "suit" : "clubs"
 }
{ "_id" : ObjectId("5d128a42fccc197267ecf947"), "rank" : "two", "suit" : "clubs"
 }
{ "_id" : ObjectId("5d128a42fccc197267ecf948"), "rank" : "three", "suit" : "club
s" }
{ "_id" : ObjectId("5d128a42fccc197267ecf949"), "rank" : "four", "suit" : "clubs
" }
{ "_id" : ObjectId("5d128acefccc197267ecf94b"), "rank" : "four", "suit" : "heart
s" }
{ "_id" : ObjectId("5d128acefccc197267ecf94c"), "rank" : "five", "suit" : "heart
s" }
{ "_id" : ObjectId("5d128acefccc197267ecf94d"), "rank" : "six", "suit" : "hearts
" }
{ "_id" : ObjectId("5d128acefccc197267ecf94e"), "rank" : "seven", "suit" : "hear
ts" }
>
```

Once we have enough varying documents in our collection, we can retrieve them by creating queries from JSON objects that represent the properties of the documents we want to retrieve. For example, we can retrieve all card documents with a rank of two and store them in a variable called twos:

```
> var twos = db.cards.find({"rank":"two"});
> twos
{ "_id" : ObjectId("5d128a42fccc197267ecf947"), "rank" : "two", "suit" : "clubs"
 }
>
```

Let us try once using another object from that collection.

```
> var apps = db.cards.find({"rank":"six"});
> apps
{ "_id" : ObjectId("5d128acefccc197267ecf94d"), "rank" : "six", "suit" : "hearts
" }
>
```

We can also remove the elements from the collection by calling the remove method and sending in a query:

```
> db.cards.remove({"rank":"two"});
WriteResult({ "nRemoved" : 1 })
> db.cards.find();
{ "_id" : ObjectId("5d12899cfccc197267ecf944"), "rank" : "ace", "suit" : "clubs"
 }
{ "_id" : ObjectId("5d128a42fccc197267ecf946"), "rank" : "one", "suit" : "clubs"
 }
{ "_id" : ObjectId("5d128a42fccc197267ecf948"), "rank" : "three", "suit" : "club
s" }
{ "_id" : ObjectId("5d128a42fccc197267ecf949"), "rank" : "four", "suit" : "clubs
" }
{ "_id" : ObjectId("5d128acefccc197267ecf94b"), "rank" : "four", "suit" : "heart
s" }
{ "_id" : ObjectId("5d128acefccc197267ecf94c"), "rank" : "five", "suit" : "heart
s" }
{ "_id" : ObjectId("5d128acefccc197267ecf94d"), "rank" : "six", "suit" : "hearts
" }
{ "_id" : ObjectId("5d128acefccc197267ecf94e"), "rank" : "seven", "suit" : "hear
ts" }
>
```

Let us try for another object.

```
> db.cards.remove({"rank":"three"});
WriteResult({ "nRemoved" : 1 })
> db.cards.find();
{ "_id" : ObjectId("5d12899cfccc197267ecf944"), "rank" : "ace", "suit" : "clubs" }
{ "_id" : ObjectId("5d128a42fccc197267ecf946"), "rank" : "one", "suit" : "clubs" }
{ "_id" : ObjectId("5d128a42fccc197267ecf949"), "rank" : "four", "suit" : "clubs" }
{ "_id" : ObjectId("5d128acefccc197267ecf94b"), "rank" : "four", "suit" : "hearts" }
{ "_id" : ObjectId("5d128acefccc197267ecf94c"), "rank" : "five", "suit" : "hearts" }
{ "_id" : ObjectId("5d128acefccc197267ecf94d"), "rank" : "six", "suit" : "hearts" }
{ "_id" : ObjectId("5d128acefccc197267ecf94e"), "rank" : "seven", "suit" : "hearts" }
>
```

You must have noticed that it worked, similarly we need to remove for all the objects. Either it can be done separately or by removing all the documents from the collection.

Or we can remove all the documents from the collection by calling remove with an empty query. It goes like db.cards.remove(); Then, please do db.cards.find(); again to check, you should be seeing all the objects removed. You can quit from the MongoDB shell by typing exit. Once you do that, you should see something like this:

```
> exit
bye
vagrant@ubuntu-xenial:~$
```

If you need help with any commands, please type help in the MongoDB shell. You should see something like this:

```
> help
        db.help()                    help on db methods
        db.mycoll.help()             help on collection methods
        sh.help()                    sharding helpers
        rs.help()                    replica set helpers
        help admin                   administrative help
        help connect                 connecting to a db help
        help keys                    key shortcuts
        help misc                    misc things to know
        help mr                      mapreduce

        show dbs                     show database names
        show collections             show collections in current database
        show users                   show users in current database
        show profile                 show most recent system.profile entries with time >= 1ms
        show logs                    show the accessible logger names
        show log [name]              prints out the last segment of log in memory, 'global' is default
        use <db_name>                set current database
        db.foo.find()                list objects in collection foo
        db.foo.find( { a : 1 } )     list objects in foo where a == 1
        it                           result of the last line evaluated; use to further iterate
        DBQuery.shellBatchSize = x   set default number of items to display on shell
        exit                         quit the mongo shell
```

Similar to Redis, MongoDB offers an interactive MongoDB tutorial https://university.mongodb.com/) that you can try out in your web browser. I encourage you to work your way through this tutorial to learn a little more about Mongo's functionality and types of queries that are available.