# Big Data Architecture Lab 2 Report

**Task 0: setup**

- Open a terminal and make a new folder named server_1 then launch a server.
  cd /Users/cherilyn/Downloads/lab_2_datasets
  mkdir server_1
  mongod --dbpath /Users/cherilyn/Downloads/lab_2_datasets --port 27018

- Open a new terminal and import the document into the server.
  cd /Users/cherilyn/Downloads/lab_2_datasets
  mongoimport -c movies --port 27018 ./moviepeople-10.jsonl

- Open the third terminal and launch a client.
  mongo --port 27018
  Then we can write all the querying in this client.

- To retrieve all the documents:
  db.movies.find()

**Task 1: import and querying**

1. Add two more documents into the server in task 0.

mongoimport -c movies --port 27018 ./moviepeople-3000.jsonl
mongoimport -c cities --port 27018 ./cities.jsonl

2. Queries

a) > db.movies.find({"person-name":"Teixeira, Anabela"})

{ "_id" : ObjectId("5da2498fe39a96ff50f1e88d"), "person-name" : "Teixeira, Anabela", "info" : { "trivia" : [ "Her favorite actor is 'Marcello Mastroianni' (qv) and her favorite actress is 'Juliette Binoche' (qv).", "Partner of musician Frederico Pereira.", "Has two brothers, one biological and one adopted.", "Vice-President of the Portuguese Cinema Academy.", "She was first noticed by the public with the Tv series A Viúva do Enforcado (1992).", "Made her theater debut in 1992 in the play Os Processos of Dostolevsky.", "Did some workshops of dance: dance in movement with Peter Diez and dance classes with Madalena Vitorino.", "Besides Portuguese, she speaks English and French." ], "birthnotes" : [ "Lisbon, Portugal" ], "birthdate" : [ "18 May 1973" ], "birthname" : [ "Teixeira, Anabela Cristina Alves" ], "interviews" : [ "TV Guia (Portugal), 1997, Iss. 960, pg. 24-25, by: Pedro Teixeira", "A Capital (Portugal), 30

April 1998, pg. 55, by: Helena Mata" ] } }

After observing the format of jsonl file, we found that for the person-name, it should be “lastname, firstname”.

b) > db.movies.find({"person-name":"Spielberg, Steven"},{"info.birthnotes":1})
{ "_id" : ObjectId("5da2498fe39a96ff50f1e9d9"), "info" : { "birthnotes" : [ "Cincinnati, Ohio, USA" ] } }

Using the projection operator to show the birthnotes.

c) > db.movies.count({"info.birthnotes":/^Lisbon/})
171

Count the number of people whose birthnotes start with “Lisbon”.

d) > db.movies.find({$and:[{"info.height":/cm/},{"info.height":{$gt:"170 cm"}}]},{"person-name":1}).limit(1)
{ "_id" : ObjectId("5da2493cdba579ecd1a611c6"), "person-name" : "Cáceres, Luciano" }

In this question, first I select the people whose height contains “cm”, then select the people whose height is bigger than 170. Because there are so many people that are eligible, so I just show one person here.

e) For this question, I didn’t find a solution to query all the content in “info”, but I noticed that there are many fields who contain “Opera”. I tried to use createIndex but it didn’t work either. So, I wrote it like below.

```
> db.movies.count({"info.trivia":/Opera/},{"person-name":1})
25
> db.movies.count({"info.article":/Opera/},{"person-name":1})
25
> db.movies.count({"info.otherworks":/Opera/},{"person-name":1})
28
> db.movies.count({"info.interviews":/Opera/},{"person-name":1})
44
> db.movies.count({"info.minibiography":/Opera/},{"person-name":1})
35
> db.movies.count({"info.magazinecoverphoto":/Opera/},{"person-name":1})
45
> db.movies.count({"info.pictorial":/Opera/},{"person-name":1})
8
> db.movies.count({"info.books":/Opera/},{"person-name":1})
1
> db.movies.findOne({"info.trivia":/Opera/},{"person-name":1})
{
        "_id" : ObjectId("5da2498fe39a96ff50f1e775"),
        "person-name" : "Rolfe, James"
}
```

f) > db.cities.find(

```
    {
        "name":
            {
                $in: db.movies.distinct("info.birthnotes")}},
    {   "_id":0,
        "name":1,
        "population":1,
        "location.latitude":1,
        "location.longitude":1
    }
)
```

We need to find the cities whose name is also in “info.birthnotes” and every city appears only once. What we want to see, are the name, the population, the latitude and the longitude of city. We don’t care the id.

```
{ "name" : "Cuba", "population" : 3427, "location" : { "latitude" : 38.06282, "l
ongitude" : -91.40348 } }
{ "name" : "Mexico", "population" : 10788, "location" : { "latitude" : 39.16976,
 "longitude" : -91.88295 } }
{ "name" : "Cuba", "population" : 1418, "location" : { "latitude" : 40.49282, "l
ongitude" : -90.19068 } }
{ "name" : "Cuba", "population" : 1361, "location" : { "latitude" : 42.18391, "l
ongitude" : -88.19091 } }
{ "name" : "Angola", "population" : 7932, "location" : { "latitude" : 41.63477,
"longitude" : -84.99941 } }
{ "name" : "Mexico", "population" : 1840, "location" : { "latitude" : 44.5609, "
longitude" : -70.54534 } }
{ "name" : "Norway", "population" : 5731, "location" : { "latitude" : 44.21396,
"longitude" : -70.54478 } }
{ "name" : "Norway", "population" : 2853, "location" : { "latitude" : 45.7869, "
longitude" : -87.90374 } }
{ "name" : "Angola", "population" : 2198, "location" : { "latitude" : 42.63839,
"longitude" : -79.02782 } }
{ "name" : "Cuba", "population" : 1583, "location" : { "latitude" : 42.21757, "l
ongitude" : -78.27529 } }
{ "name" : "Mexico", "population" : 1560, "location" : { "latitude" : 43.45951,
"longitude" : -76.22882 } }
{ "name" : "Belgium", "population" : 2100, "location" : { "latitude" : 43.49972,
 "longitude" : -87.85037 } }
```


**Task 2: replication**

1. Open a new terminal, go to the folder we want, then create 3 new folders
   mkdir ./mongo1 ./mongo2 ./mongo3

2. Open another 3 terminals, run the 3 commands respectively
   mongod --replSet small-movie --dbpath ./mongo1 --port 27011
   mongod --replSet small-movie --dbpath ./mongo2 --port 27012
   mongod --replSet small-movie --dbpath ./mongo3 --port 27013

3. Open another terminal as client
   mongo --port 27011

4. In the client terminal, we initialize the replication

```
rs.initiate(
    {
        _id: "small-movie",
        members: [
            { _id: 1, host : "localhost:27011" },
            { _id: 2, host : "localhost:27012" }
        ]
    }
)
```

rs.addArb("localhost:27013")

After this step from the arbiter we can see:

```
2019-10-13T15:31:30.814+0200 I  REPL     [replexec-0] This node is localhost:270
13 in the config
2019-10-13T15:31:30.814+0200 I  REPL     [replexec-0] transition to ARBITER from
 STARTUP
2019-10-13T15:31:30.815+0200 I  REPL     [replexec-1] Member localhost:27012 is
now in state SECONDARY
2019-10-13T15:31:30.815+0200 I  REPL     [replexec-0] Member localhost:27011 is
now in state PRIMARY
```

5. rs.status()

   With this command we can see the informations of all the members. Here I put a screenshot of Primary.


6. Open another new terminal and import the document
   mongoimport -c movies --port 27011 ./moviepeople-1000.jsonl

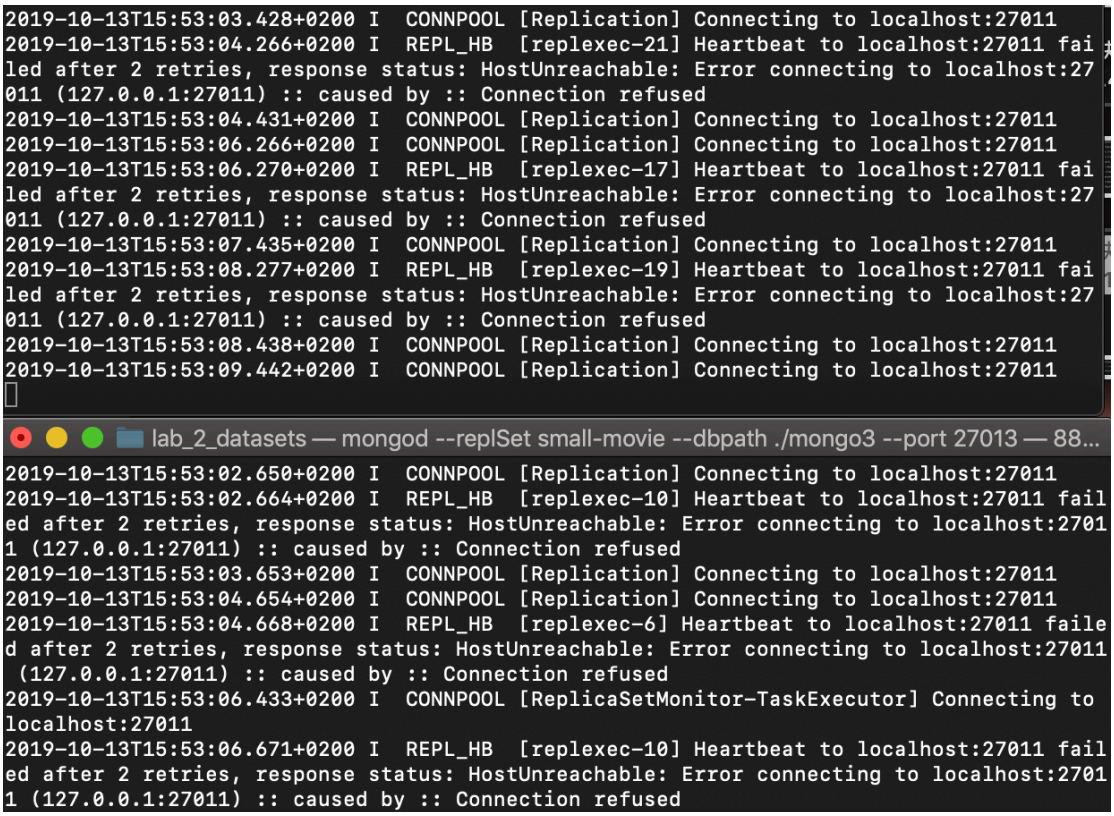
   the output of the secondary server:

```
2019-10-13T15:47:17.989+0200 I  NETWORK  [ReplicaSetMonitor-TaskExecutor] Confirmed rep
lica set for small-movie is small-movie/localhost:27011,localhost:27012
2019-10-13T15:47:17.990+0200 I  NETWORK  [LogicalSessionCacheRefresh] Starting new repl
ica set monitor for small-movie/localhost:27011,localhost:27012
2019-10-13T15:47:17.990+0200 I  NETWORK  [ReplicaSetMonitor-TaskExecutor] Confirmed rep
lica set for small-movie is small-movie/localhost:27011,localhost:27012
2019-10-13T15:47:17.995+0200 I  NETWORK  [conn20] end connection 127.0.0.1:54959 (4 con
nections now open)
2019-10-13T15:47:52.382+0200 I  STORAGE  [repl-writer-worker-7] createCollection: test.
movies with provided UUID: c8c8b2e7-8563-4dff-8a28-ac9df6dcbf99 and options: { uuid: UU
ID("c8c8b2e7-8563-4dff-8a28-ac9df6dcbf99") }
2019-10-13T15:47:52.413+0200 I  INDEX    [repl-writer-worker-7] index build: done build
ing index _id_ on ns test.movies
2019-10-13T15:47:52.420+0200 I  SHARDING [repl-writer-worker-6] Marking collection test
.movies as collection version: <unsharded>
2019-10-13T15:48:39.895+0200 I  NETWORK  [listener] connection accepted from 127.0.0.1:
55083 #30 (5 connections now open)
2019-10-13T15:48:39.895+0200 I  NETWORK  [conn30] received client metadata from 127.0.0
.1:55083 conn30: { driver: { name: "NetworkInterfaceTL", version: "4.2.0" }, os: { type
: "Darwin", name: "Mac OS X", architecture: "x86_64", version: "18.7.0" } }
2019-10-13T15:48:39.908+0200 I  NETWORK  [conn26] end connection 127.0.0.1:55063 (4 con
nections now open)
```

the output of the arbiter server:

```
2019-10-13T15:48:39.893+0200 I  NETWORK  [LogicalSessionCacheRefresh] Starting n
ew replica set monitor for small-movie/localhost:27011,localhost:27012
2019-10-13T15:48:39.893+0200 I  CONNPOOL [ReplicaSetMonitor-TaskExecutor] Droppi
ng all pooled connections to localhost:27011 due to ShutdownInProgress: Pool for
 localhost:27011 has expired.
2019-10-13T15:48:39.893+0200 I  CONTROL  [LogicalSessionCacheReap] Sessions coll
ection is not set up; waiting until next sessions reap interval: config.system.s
essions does not exist
2019-10-13T15:48:39.893+0200 I  CONNPOOL [ReplicaSetMonitor-TaskExecutor] Connec
ting to localhost:27012
2019-10-13T15:48:39.893+0200 I  CONNPOOL [ReplicaSetMonitor-TaskExecutor] Connec
ting to localhost:27011
2019-10-13T15:48:39.896+0200 I  NETWORK  [ReplicaSetMonitor-TaskExecutor] Confir
med replica set for small-movie is small-movie/localhost:27011,localhost:27012
2019-10-13T15:48:39.898+0200 I  NETWORK  [LogicalSessionCacheRefresh] Successful
ly connected to localhost:27011 (1 connections now open to localhost:27011 with
a 0 second timeout)
```

7. The output of the two servers:




**Task 3: sharding**

- First, we create two folders like above. Then open 2 servers.
  mkdir ./mongo4 ./mongo5
  mongod --shardsvr --dbpath ./mongo4 --port 27014
  mongod --shardsvr --dbpath ./mongo5 --port 27015

- Open another terminal as config server. Here we must make it a replica set.
  mkdir ./mongoconfig
  mongod --replSet cities --dbpath ./mongoconfig --port 27016 --configsvr

- Open another terminal to connect with the replica set and initialize it.
  mongo --port 27016
  rs.initiate()

- Now we can start mongos connected to the config
  mongos --configdb cities/localhost:27016 --port 27020

- Import cities.jsonl like always
  mongoimport -c cities --port 27020 ./cities.jsonl

- Open another terminal to operate sharding. Unless the collection is empty, the index must exist prior to the shardCollection command.

  mongo localhost:27020

  mongos> sh.addShard( "localhost:27014" )

  mongos> sh.addShard( "localhost:27015" )

  mongos> sh.enableSharding("test")

  mongos> db.cities.createIndex({"country":1})

  mongos> sh.shardCollection("test.cities", { country: 1 } )

  Now the sharding is done. We can check it.

```
mongos> db.cities.stats().sharded
true
mongos> db.cities.getShardDistribution()

Shard shard0001 at localhost:27015
 data : 14.78MiB docs : 99838 chunks : 1
 estimated data per chunk : 14.78MiB
 estimated docs per chunk : 99838

Totals
 data : 14.78MiB docs : 99838 chunks : 1
 Shard shard0001 contains 100% data, 100% docs in cluster, avg obj size on shard
 : 155B
```