Rest of the Semester

- 4/4 Map-Reduce (Al) & TF-IDF
- 4/6 Page Rank 31.3-4 (Ellie F) , Inverted Indices 31.5-6, Google structure (based on "https://snap.stanford.edu/class/cs224w-readings/Brin98Anatomy.pdf" by Page and Brin)
- 4/11Transactions 17.1-4, 17.5-7
- 4/13 Concurrency 18.1-2, 18.3-4, 32.5
- 4/18 Databases in Spreadsheets (2 people)
- 4/20 Parallel/Distributed Hadoop, sail 22, Mongo Sharding mongo 14-17
  - Or a topic of your own suggestion on a date of my choice

- 4/25 Project presentations
- 4/27 Project presentations

Mongo Java

Basics

- need a jar file to compile/run
  - unlike Postgres where only need jar to run
  - mongo-java-driver-3.12.10.jar
    - should be on class web site

- javac -cp mongo-java-driver-3.12.10.jar:. XXX.java
- java -cp mongo-java-driver-3.12.10.jar:. XXX
  - cp == classpath
  - You can tell VSC about JAR files so standard compile/run works

Connect

- List PostgreSQL, need to be on machine
  - or have forwarding set up

- public static String CONNECTION_STRING = "mongodb://127.0.0.1:27017";
  - A URI -- mongo default port in 27017

```
import com.mongodb.client.MongoClients;

try (MongoClient mongoClient = MongoClients.create(CONNECTION_STRING);) {

} catch (Exception ee) {
    System.err.println("Problem talking to Mongo " + ee);
    ee.printStackTrace();
}
```

Explore

- Unlike SQL and ODBC where you just passed the exact query string that you would use in the interactive shell, here you need to know function names.
  - They are usually really close to the ones in mongosh

```
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import org.bson.Document;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoDatabase;

try (MongoClient mongoClient = MongoClients.create(CONNECTION_STRING);) {

    ArrayList<Document> databases = mongoClient.listDatabases().into(new ArrayList<>());
    for (Document db : databases) {
        System.out.println(db.toJson());
    }

    System.out.println("Connect to Univ then list collections");
    MongoDatabase database = mongoClient.getDatabase("univ");
    MongoIterable<String> list = database.listCollectionNames();
    for (String name : list) {
        System.out.println(name);
    }
} catch (Exception ee) {
    System.err.println("Problem talking to Mongo " + ee);
    ee.printStackTrace();
}
```

Convert from awkward iterator to useful ArrayList

Convert from awkward BSON Document type to at least readable JSON

Connect to a particular database

Show all collections in UNIV database

Query and Unmarshall -- 1

- Working with the mgoactor table from Monday
- Create classes to hold values
  - Note here I have a FilmA class specifically for the film extended reference contained in Actor.
    - Could have done it differently
- Set up classes with unmarshalling constructors
  - take a BSON document and fill in values
- Write a toString also

```
private class FilmA {
    int categ;
    String catname;
    int filmid;
    String filmname;

    public FilmA(Document d) {
        categ = d.getInteger("categ");
        catname = d.getString("catname");
        filmid = d.getInteger("filmid");
        filmname = d.getString("filmname");
    }

    public String toString() {
        return filmname;
    }
}
private class Actor {
    int id;
    String first_name;
    String last_name;
    ArrayList<FilmA> films;

    public Actor(Document d) {
        films = new ArrayList<>();
        id = d.getInteger("actor");
        first_name = d.getString("first_name");
        last_name = d.getString("last_name");
        List<Document> extractedfilms = d.getList("films", Document.class);
        for (Document f:extractedfilms)
            films.add(new FilmA(f));
    }

    public String toString() {
        return String.format("<Actor:%d %s %s    films:%s> ", id, first_name, last_name, films);
    }
}
```

Handling null?

Query and Unmarshall -- 1

```
        try (MongoClient mongoClient = MongoClients.create(CONNECTION_STRING);) {


            MongoDatabase database = mongoClient.getDatabase("sakila");
            MongoCollection<Document> actorColl = database.getCollection("mgoactor");
            ArrayList<Document> actorList = actorColl.find(new Document("first_name", new
Document("$regex", "^BE")))   // find actors whose first name starts with BE
                    .into(new ArrayList<>());
            System.out.format("Got %d actors\n", actorList.size());

            ArrayList<Actor> actorsUnmarshalled = new ArrayList<>();
            // unmarshall into java classes
            for (Document d : actorList) {
                actorsUnmarshalled.add(new Actor(d));
            }
            // print java classes via toString
            System.out.println(actorsUnmarshalled);

            HashMap<String, Object> q = new HashMap<>();
            q.put("first_name", "BETTE");
            q.put("last_name", "NICHOLSON");
            actorList = actorColl.find(new Document(q)).into(new ArrayList<>());
            actorsUnmarshalled.clear();
            for (Document d : actorList) {
                actorsUnmarshalled.add(new Actor(d));
            }
            System.out.println(actorsUnmarshalled);


        }
```

Write a Query as a BSON Document

Get query results into an ArrayList

Convert into ArrayList of POJOs using unmarshalling constructors

Print using toString

New query, this time build in a map, then convert Map into document

Marshall and Insert -- 1

- Write marshallers for each object
  - Note that the Marshaller for Actor calls the marshaller for FilmA
  - If only one class for Film / Actor, then may need multiple marshallers

```
private class FilmA {
    int categ;
    String catname;
    int filmid;
    String filmname;

    public Document marshall() {
        Document rtn = new Document("categ", categ);
        rtn.append("catname", catname);
        rtn.append("filmid", filmid);
        rtn.append("filmname", filmname);
        return rtn;
    }

    public String toString() {
        return filmname;
    }
}
private class Actor {
    int id;
    String first_name;
    String last_name;
    ArrayList<FilmA> films;


    public void addFilm(FilmA f) {
        films.add(f);
    }
    public String toString() {
        return String.format("<Actor:%d %s %s    films:%s> ", id, first_name, last_name, films);
    }

    public Document marshall() {
        Document rtn = new Document("actor", id);
        rtn.append("first_name", first_name);
        rtn.append("last_name", last_name);
        ArrayList<Document> ff = new ArrayList<>();
        for (FilmA f : films)
            ff.add(f.marshall());
        rtn.append("films", ff);
        return rtn;
    }
}
```

Handling null?

Marshall and Insert -- 2

- Create object
  - or a list thereof
- Marshall the object
- Pass that to insert

```
try (MongoClient mongoClient = MongoClients.create(CONNECTION_STRING);) {


    MongoDatabase database = mongoClient.getDatabase("sakila");
    MongoCollection<Document> actorColl = database.getCollection("mgoactor");
    Actor geoff = new Actor(2000, "geoff", "towell");
    geoff.addFilm(new FilmA(4, "hello", 10, "DJANGO GESTE"));
    geoff.addFilm(new FilmA(2, "DOCUMENTARY", 20, "LOST IN CANADA"));
    actorColl.insertOne(geoff.marshall());

} catch (Exception ee) {
    System.err.println("problem" + ee);
    ee.printStackTrace();
}
```