Kai Austin

Software Systems

March 6, 201

Repo: KaminariNoKage/Software Systems/hw03/bacon.c

1) Installation

Mongo DB is free for all, licensed under Creative Commons by MongoDB, Inc. Downloads are available for all major OS platforms and is fairly easy to setup.

2) Immersion

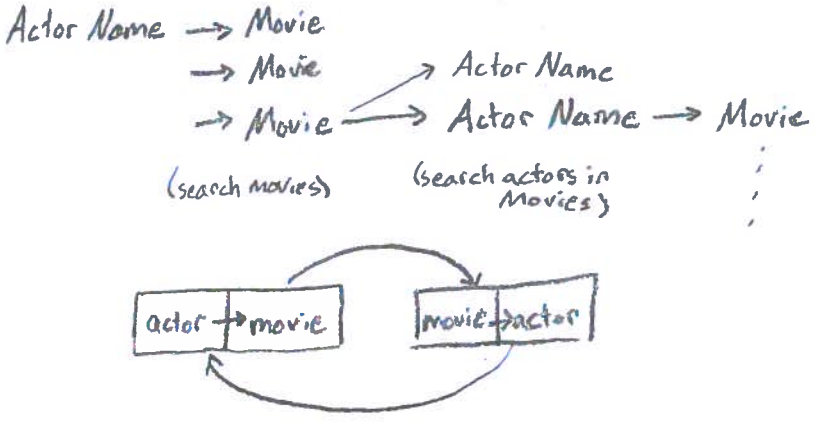
The database has sufficient documentation for setting it up in C and Python (the latter a plausible method for parsing IBDb data and putting it in the database). The key is to get the correct path for the mongo-driver, which is needed for C, or pyMongo library to use the database in Python. There are both examples of each use. They are not always 100% functional and may require a bit of debugging—however this is fairly standard of most random code snippets found in examples. There may still be occasional warnings, however these are not always database related.

3) Semantics

The way data is queried in MongoDB is as follows (and example).

```
{ "actor name" : [ "Movie 1",
                   "Movie 2",
                   "Movie 3",
                   ... etc. ] }
```

Which is fairly easy to handle as the program merely needs to look up a string, which is the "key", and obtain the data the query/key points to. Suppose we store all actor names under the "actor" key, and under each actor we have a list of all movies they have been in. Likewise, we could have a key called "movies" directing to all movies, which each have a list of all actors who play in them. This directly mirrors the structure the algorithms we implement in C will be searching for data, and trying to match actors.

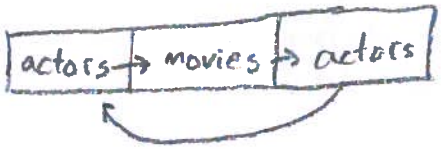


4) Performance

The likely fault of this database mostly comes from the sheer magnitude of data which IMDb has to offer. There are thousands of names and movies. Each step in the algorithm as of now is to look up each actor name, then look up the movie, then look up each actor in the movie, which accumulates time far too quickly. Faster implementations would be to rearrange the database so that each actor points to a movie list and in that movie list, each movie has its own list of actors. There would be no search need for movies seperately.

"Actor Name": ["Movie": ["Actor", "Actor"],

"Movie": ["Actor", "Actor", etc. ..]}



Other possible optimizations include not adding (or searching for) an actor which the given actor in the key has been in a movie with before, since the algorithm would be repeating searches/look ups it has done before.

5) ACIDity

The database is guaranteed to be consistent, and fairly easy to maintain. Should the structure need to be changed, it is much easier than SQL counterparts. Adding and removing data is fairly simplistic as well. It largely depends on chosen structure. The main issue faced is likely speed with magnitude.

6) C interface

The API is a little wonky, but mostly has a learning curve. All that merely needs to be done is to retrieve data. There is no complex parsing or structures to deal with, which is an advantage in efficiency.