Paige Rosynek

CS 3860 041

Dr. Magana

11.8.2022

**Final Lab – MongoDB**

**Introduction**

For the final project, I chose to redo Lab 03 in MongoDB as suggested by the lab document. In order to interact with the MongoDB database, I used the Python library PyMongo which is a Python driver for interacting with MongoDB databases and Atlas. All the code mentioned in the report below can be found in accompanied zip file.

PyMongo: <https://www.mongodb.com/docs/drivers/pymongo/>

**Data Model**

Graphical user interface, text, application, email

Description automatically generated

Text

Description automatically generatedFigure MongoDB database representation of Lab03 data.

Figure 2. (right) Another example of a document in the films collection.

For my data model, I used one collection called films in order to represent all the data from the three datasets from Lab 03: video\_recordings, video\_actors, and video\_categories. To do this, I first exported the mentioned tables from MySQL as JSON files then, after some data cleaning for punctuation, I loaded the files into MongoDB. I then used PyMongo to denormalize the data into a single collection as shown in the screenshot above. The data model I chose for this project utilizes a single collection, films, that represents all the data and relationships across the three datasets. The collection contains a total of 55 documents which is the MySQL equivalent of the total number of rows in a table. Each document in the collection represents a unique recording\_id and its relational data from the three datasets, therefore, there were 55 unique recording\_id’s, or films, in the dataset. This way, each document represents a unique film. The fields of each document are the same as the attributes from the video\_recordings dataset, with the addition of the actors field. From Lab03, we found that there is a one to many relationship between a film (recording\_id) and actors; in other words, we know that one film can have many actors. In order to represent this relationship in MongoDB, I included an array of actors for each film (see ‘actor’ field in screenshot above). I chose this structure for my data because for all the fields of each document (except for actors) only a single value is needed. In addition, since PyMongo uses dictionary data structures to interact with collections, the actors in each film (document) can easily be accessed with the collection key, ‘actors’, where the value is the list of actors for the film, which can be looped through in Python code, so it should help with answering the questions in the report. I kept the recording\_id as a field in MongoDB because I wasn’t sure if the recording\_id would be needed for the purpose other than identifying films, but it can be observed that MongoDB creates a unique ObjectId in the \_id field which uniquely identifies each document within the collection. The Python script I used to combine the datasets and create the films collection can be found in the accompanying zip folder, in the file denormalize.py.

**Query Results**

**Advantages & Disadvantages of NoSQL**

One advantage of using a NoSQL database system like MongoDB for this lab, is that you can write functions in code to perform similar queries to the database which can reduce the amount of code needed to be written. In addition, the data model for the lab in MongoDB is more effective than