CS3860 – Database Systems

Lab 8 (Final Project)

Nigel Nelson & Mitchell Johnstone

Aparna Joshi

**\*Note: Demoed database implementation and queries in lab\***

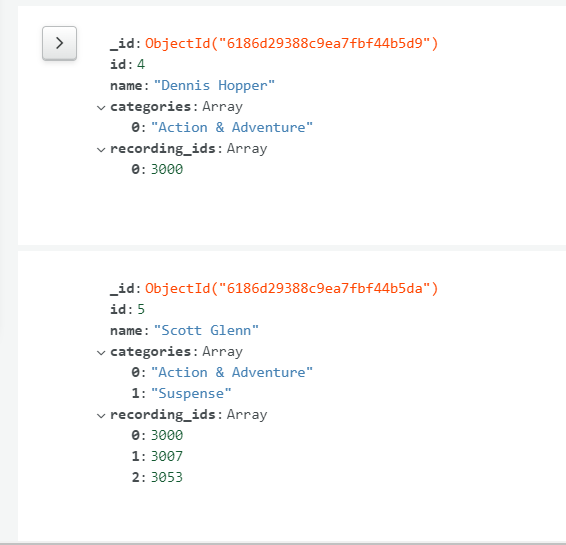
Process to create final implementation:

1. Exported video tables from MySQL to .json files
2. Used MongoDB-Compass to load .json files into MongoDB database
3. Used python scripts to de-normalize the collections in order to make NoSQL queries easier to execute

* Document your data model. At a minimum this should include a discussion of your Collection(s) – what you choose to include in each document (entry) and why. This should relate to the questions (lab3 queries) that we are trying to answer with MongoDB.

Main design: 2 collections, one for the actors and one for the records.

Actors Collection:

* Our Actors collection maintained almost all the data contained in its initial form from the relational database. This includes the actor’s id as well as their name. However, we dropped the recording\_id field in favor of including a recording\_ids array field. This allowed us to delete / aggregate multiple documents referring to the same actor and store all the recording\_id’s that an actor references into a single document. In addition, we added a categories array field to make our Lab 3 queries much easier. This is because many queries wanted to know the categories that actors appeared in, and by storing all these categories as an array in an actor document, it made these queries much easier, since we didn’t have to do any joins on other collections/documents, we only had to worry about the actors and their categories. This helped us simplify our queries 5-8 immensely.
* 

Records Collection:

* Similar to the Actors collection, we took all of the data from the MySQL video\_recording table. We found that while we didn’t really need all the data fields, we still would like to have that data for the future. However, we did modify the fields to better apply to our queries. We converted the category field from a number into the actual category title so we could better parse through the different types of categories without needing to join on a category table. This helped particularly in queries 3 and 4, for displaying the number of recordings for each category. As well, we added another array for all the actors that feature in the given recording so we can better filter actors based on their recordings.
* 
* Results of each query (Lab 3 queries 3-8, or equivalent) AND a quick description of how Method/Code/Way you obtain those results. If the results are too large, then provide the first 10 entries of results.

**Query Results and methods of obtaining results have already been demonstrated to the instructor (Dr. Joshi) during the designated lab period on 11/9/2021. She requested us to include that in our report.**

* Each team member answer this question separately: **Analysis of benefits and drawbacks of using a NoSQL database instead of a relational database for this data/project. (Mark your answer)**

Mitchell’s Response:

* Benefits:
  + The NoSQL was a much looser framework. We could design it to have different values, if necessary, to allow for more varied structures of documents in the collection, although we did not utilize this feature too much.
  + We could also have nested documents and arrays, which meant we could maintain multiple values that related to a single document. This allowed us to have only 1 document for each actor, for example, instead of having multiple for each recording the given actor is in.
  + As a result, we could design our collections to optimize the quality of our queries. We didn’t have to make our data applicable to every situation, only for the queries we’re given.
* Drawbacks:
  + In order to process the queries more easily, we had to duplicate our data multiple times, such as listing the actors for each recording or all of the categories for each actor.
  + Our queries were direct enough that we could contain all the information easily in single objects, but if we hadn’t it would’ve been difficult to join collections together in the NoSQL framework.

Nigel’s Response:

* Benefits:
  + One of the primary benefits of NoSQL databases is their ease of use. Due to less normalization, it can be much simpler as an end user to get the information you need from a database when compared to using a relational database. The reason for this that instead of having to process several inner joins to get information that relates to given document, NoSQL allows for nested objects and nested arrays of objects to be stored, so that all relevant information is stored inside a given document. So, for example, instead of having to inner join video\_recordings and video\_actors to get which categories an actor had acted in, using NoSQL we’re able to store all of the categories an actor has acted in inside each actor document.
  + Another benefit of NoSQL is its use of partitioning leads to greater scalability. In the relational videos database, even at this smaller scale it was difficult to maintain the relations between all of the normalized tables. One can imagine that if this data set grew, all of these relationships and tables would be difficult to maintain in addition to the fact that it would become more and more inefficient as entries were stacked vertically on each other in these tables. However, thanks to the partitioning that is grantable by NoSQL this database could scale horizontally, due to the fact that each document is essentially a key value pair. This ultimately leads to greater scalability and efficiency at scale.
* Drawbacks:
  + One of the greatest drawbacks of NoSQL is the redundancy of data. Due to the lack of normalization, data is often repeated in NoSQL databases, leading to the same data taking up multiple locations in memory. An example of this is that in our final NoSQL database the actor’s names are contained both in the video\_recordings collection, as well as in the video\_actors collection.
  + Another drawback of the NoSQL implementation for this project is the fact that personally, I found it much less intuitive to write NoSQL queries as opposed to relational database queries using SQL. SQL query syntax is much closer to natural language in how an end user would think of a desired query, whereas the NoSQL version of these queries resembles a programming language much more than it resembles natural language.