DMDII ASSIGMENT I

Maxim Evgrafov

BS18-01

Spring 2020

Introduction

For this assignment I used the following environment:

1. PostgreSQL launched via Docker
2. MongoDB cloud database service Atlas
3. Python 3.7

Transfering data from SQL to NoSQL

1. Establishing connections to MongoDB and PostgreSQL
2. Get data from postgres in JSON format

> SELECT row_to_json(t) FROM <table_name> t;

3. Just insert JSON one by one

> Later I unserstand that I am able to use function **insert_many**, but it was too late 😅

LINK TO GITHUB

Queries

Every MongoDB request needs function **aggregate**. Also, when we call the function, we need to provide **pipeline** argument. Also, sometimes I use python tools to get answer quickier.

LINK TO GITHUB

Query I

LINK TO GITHUB
Main idea is:

1. Get info about customer with all ids of inventory items they ever get

```
Format:
        _id: customer id
        count: # of rentals
        in_ids: list of
                        inventory item id
                        rental year
```

2. Retrieve information about categories for films
3. Join everything, in case to get needed information

Query II

Main idea is:

1. Get information from collection **film_actor**, grouped by *actor_id*
2. Get information from collection **film_actor**, grouped by *film_id*
3. Go trough all actors crossed with all actors and find if they were in the same film
4. Calculate needed data during 3rd step

LINK TO GITHUB

Query III

Main idea is:

1. Get information from collections **inventory** and **rental**, grouped by *film_id*
2. Get information from collections **category** and **film_category**, to get info about category of a particular film
3. Cross every thing, then calculate

LINK TO GITHUB

Query IV

Main idea is:

1. Get info about films, that customers take from stores
2. Get info about *fit_rate* between two people

$$\text{fit\_rate}_{A,B} = 1 - \frac{\text{len}(\{\text{'films, watched by A'}\} - \{\text{'films, watched by B'}\})}{\text{len}(\{\text{'films, watched by A'}\})}$$

> Shortly say, fit_rate means how much B in fit with A in their taste of films

3. Add to *possibly suggested films* $\{\text{'films, watched by B'}\} - \{\text{'films, watched by A'}\}$
4. Rerate everything there by formula

$$\text{film rate} = \frac{\text{film\_rate} + \text{fit\_rate}_{A,B}}{2}$$

5. Sort
6. Return 10 best films for suggesting

LINK TO GITHUB

Query V

Main idea is:

1. Use *output from second query* as weights for graph
2. Just find length of the path between our **actor1** and every other actor

LINK TO GITHUB

Components diagram

Conclusion

NoSQL databases are the best tools for databasing. MongoDB, I choose you.

LINK TO GITHUB ON WHOLE PROJECT