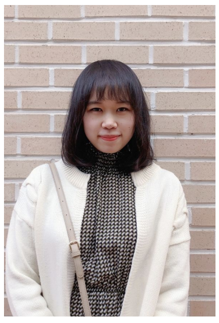
DataBase Midterm Report

: Movie Website with Objective Info & Customization Recommendation

21500095 김시온	21500643 정선일	21600752 하재경	21700034 곽영혜
			

1. **Problem specification**

o Problem

The application “Naver films”, currently available in Korea, provides services regarding films. However, there are some limits that users find unsatisfactory. It’s limits are as follows:

1. Place too much weight on local films
2. Does not have enough review written by film critics
3. Does not have film recommendation system that meets the needs of users.

To provide better experiences to users, our team aims to build a web application that addresses these limits by combining data from “Naver films” with the data from “Rotten Tomatoes”; A web page popular for having lots of reviews written by professional film critics and top 100 lists sorted by movie genre.

o Database and Application Introduction

Our team plan to use database that is largely divided into three parts. First part is for data from Naver films, second for Rotten Tomatoes, and third for users. First two parts of databases will be used to store and compare data from the two websites and the last part will be used to manage users.
From our database, our application will be able to extract and compare data from two websites and also use specific user’s information to provide several recommendation functions. The recommendation systems will be able to use the strengths of both “Naver films” (preferred movies by age, movies by gender, movies by country) and “Rotten Tomatoes” (objective movie rankings through many expert reviews, vast amounts of data from overseas data). Detailed database structures can be in **Designed Database** section below.

o Summary/Requirements

Our database and application is designed to:

1. Provide information on broader range of movies
2. Provide more objective evaluation scores
3. Provide a variety of recommendation services based on user’s following information:
    a. favorite genres (based on survey and previously watched movies)
    b. favorite actor (based on previously watched movies)
    c. favorite director (based on previously watched movies)
    d. favorite country (based on previously watched movies)
    e. Age group (Certain age group’s favorite movies)
4. Provide information about weekly, monthly and overall popular movies on an hourly basis.

2. Application Proposal

o Goal & Main features

The goal of our application is to provide more reliable, diverse, and customized information to users by recommending movies from foreign countries as well as domestic using information about the user. With this, user is expected to watch diverse movies that suit their tastes better.

Application provide:

1. More diverse movie information with movies from foreign countries
    a. Use movie data pool of both Naver films and Rotten Tomatoes
2. Objective information
    a. Give combined rating of two websites
    b. Divide webpage into two part and show information from Naver films on one and from Rotten Tomatoes on the other
3. Customized recommendations
    a. Use user’s information such as age, gender, genre for recommendation
    b. The criterias for recommendation are movie’s cumulative audience, rating, number of likes, and number of comments.

o Brief Design of Application

Main Page

a. User Login section for user login
b. Search section for movie search
c. Various “Recommendation Pages” sections with diverse themes listed

Recommendation Page (when a list is selected)

a. Recommended movies of the selected theme

Movie Page (when a movie is selected)

a. Page divided in half
    i. Half shows the data from Naver films
    ii. Other half shows data from Rotten Tomatoes
b. Review section for user evaluation of movie and to see other users' evaluations.

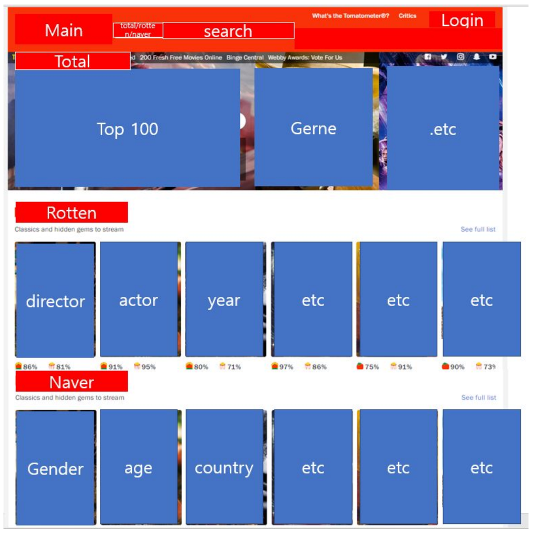


figure 1: Main Page Example

o Used Platforms

Server : AWS server **Database :** mysql
Web Application : php **Data Accumulation :** python (crawling)

3. **Designed database**

o Table Explanation

userList : Include user information through sign up as a member. In gender, 0 represents male and 1 represents female. PK - user_id

watching_history : Record the User’s movie watching history to make sure you have already watched it.

naver_evaluation, rotten_evaluation : It contains user_id, movie_id, evaluation score, review, and evaluation time for movie evaluation.

interest : It contains data of the genre that users are interested in. Make sure to select a genre of interest early after joining the membership, and then implement it so that the information can be contained.

movieList_naver, movieList_rotten : The movie is crawled by year in Rotten Tomato and Naver. And, the movie contains data of each movie_id, name, ranking by year, expert review count, evaluation score, and director id. Used to show movie information of Rotten Tomato, annual ranking movies, score movies, and director movies. PK - movie_id, FK - director_id(in directorList(Contains director information.))

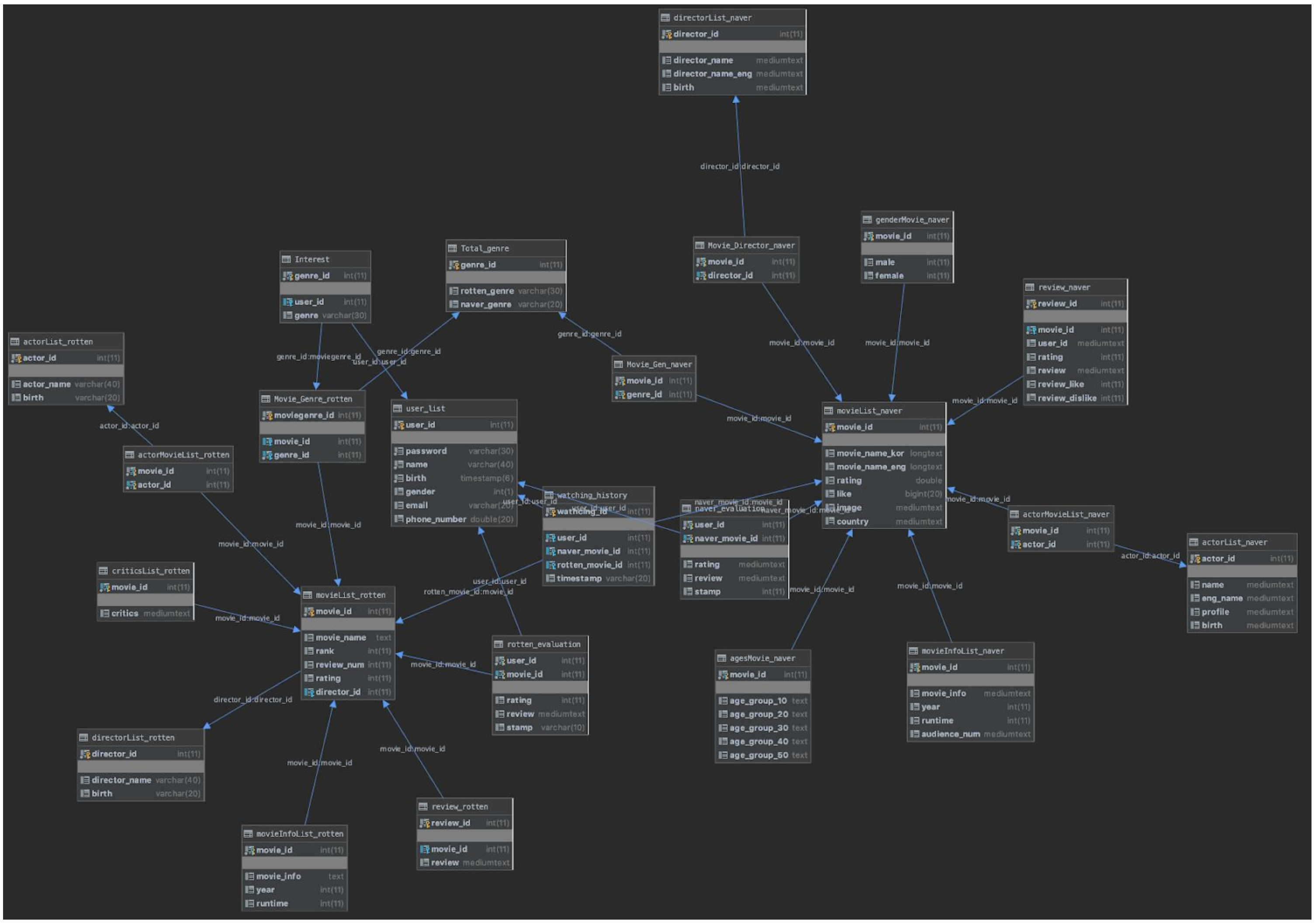
movieInfoList_naver, movieInfoList_rotten, criticsList : Since putting all the movie information in one table makes the table heavy, it creates a "movieInfoList," make the movie_id of "movieList_rotten" to be kept as a foreign key, and insert data from movie_info(description), year, and runtime. Critics information is also included in the "criticsList".

actorList_naver, actorList_rotten, actorMovieList_naver, actorMovieList_rotten, directorList_naver, directorList_rotten : The table of actorsList and directorList contains information of actors and directors. "actormovieList_rotten" takes the movie_id from "movieList_rotten" and includes the actor_id who appeared in that movie. PK - movie_id(FK in movieList), actor_id(FK in actorList)

review_naver, review_rotten : review_rotten has has reviews information according to each movies that obtain from ‘movieList_rotten’.

Total_Genre, Movie_Genre_Naver, Movie_Genre_Rotten : ‘Movie_Genre_Naver’ and ‘Movie_Genre_Rotten’ has ‘movie_id’ and ‘genre_id’ attributes. And Total_Genre has genre_id, rotten_genre, and naver_genre. So, we can access the genre of movie that we want to know by querying.

o E-R Diagram



o Example Queries Used for Functions

1. Provide information on broader range of movies

```
CREATE VIEW total_movie AS
SELECT distinct A.movie_id as movie_id_rotten, B.movie_id as movie_id_naver
FROM movieList_rotten AS A, movieInfoList_rotten AS C, movieList_naver AS B,
movieInfoList_naver AS D
WHERE A.movie_name = B.movie_name_eng AND C.year = D.year
```

2. Provide more objective evaluation scores

```
SELECT A.rating/10+B.rating as total_rating
FROM movieList_rotten AS A, movieInfoList_rotten AS C, movieList_naver AS B,
movieInfoList_naver AS D
WHERE A.movie_name = B.movie_name_eng AND C.year = D.year
```

3. Provide a variety of recommendation services based on user's following information:
    a. favorite genres (based on survey and previously watched movies)

```
SELECT C.* FROM Movie_Gen_naver as MD, movieList_naver as C WHERE
MD.genre_id IN (SELECT MAX(distinct C.country) FROM movieList_naver AS C,
movieInfoList_naver AS D, watching_history AS H, Movie_Gen_naver AS MD
WHERE H.user_id = 2 AND H.naver_movie_id=C.movie_id AND C.movie_id =
MD.movie_id GROUP BY MD.genre_id) AND MD.movie_id AND C.movie_id
```

b. favorite actor (based on previously watched movies)
   ```
   SELECT C.* FROM actorMovieList_naver as MD, movieList_naver as C WHERE
   MD.actor_id IN (SELECT MAX(distinct C.country) FROM movieList_naver AS C,
   movieInfoList_naver AS D, watching_history AS H, actorMovieList_naver AS MD
   WHERE H.user_id = 2 AND H.naver_movie_id=C.movie_id AND C.movie_id =
   MD.movie_id GROUP BY MD.actor_id) AND MD.movie_id AND C.movie_id
   ```
c. favorite director (based on previously watched movies)
   ```
   SELECT C.* FROM Movie_Director_naver as MD, movieList_naver as C WHERE
   MD.director_id IN (SELECT MAX(distinct C.country) FROM movieList_naver AS C,
   movieInfoList_naver AS D, watching_history AS H, Movie_Director_naver AS MD
   WHERE H.user_id = 2 AND H.naver_movie_id=C.movie_id AND C.movie_id =
   MD.movie_id GROUP BY MD.director_id) AND MD.movie_id AND C.movie_id
   ```
d. favorite country (based on previously watched movies)
   : Show the top 10 list of “Country” movies which user is most frequently saw and sorting them by the movie’s rating
   ```
   SELECT C.* FROM movieList_naver as C WHERE C.country IN (SELECT
   MAX(distinct C.country) FROM movieList_naver AS C, movieInfoList_naver AS D,
   watching_history AS H WHERE H.user_id = 2 AND H.naver_movie_id=C.movie_id
   GROUP BY C.country) ORDER BY C.rating desc limit 10
   ```
e. age (Certain age group’s favorite movies)
   Top 10 Most Viewed by 20s :
   ```
   select M.* FROM agesMovie_naver AS A, movieList_naver AS M WHERE
   M.movie_id = A.movie_id order by CAST(SUBSTRING_INDEX(A.age_group_20, '%',
   1) as unsigned) desc limit 10
   ```

o **Data Insertion**

Data using “Crawling”

1. AWS builds its own servers so that they can be used for mysql and Web site implementation.
2. Using Python BeautifulSoup and urlopen library, data is crawled from each Naver movie, Rotten Tomato site and saved as a csv file.
3. Connect to the server we created by AWS with datagrip.
4. Import a crawled csv file from datagrip, then create a table and specify a type to insert data from the csv file.

Data using Web Application

1. Information about movie is gathered from web (e.g. title, director, year, span, etc.)
2. Using php mysql query, new record is inserted to database