Database Project #2 Report

2019-18499 김준혁

핵심 모듈과 알고리즘

1. 중복 쿼리 함수화

코드 상단에 'get_' 또는 'put_'으로 시작하는 함수는 여러 번 사용되는 같은 쿼리를 이용하기 위해 따로 모듈화시킨 것으로, id나 제목, 이름 및 나이 등을 통해 영화/유저/예매 정보를 반환하도록 하였다.

2. table schema

총 3개의 table을 이용하며, 각각의 schema 요약은 다음과 같다. (밑줄 primary key)

movie : id (int, auto_increment), title (varchar), director (varchar), price (int)

user : id (int, auto_increment), name (varchar), age (int), class (varchar)

reservation : movie_id (int), user_id (int), reserve_price (int), rating (int)

예매 및 별점 부여를 reservation 한 테이블에 담아, 별점 부여 시 예매 상태 및 확인하고 별점이 Null인지(이미 부여했는지) 확인하고 부여하도록 하였다.

3. 알고리즘

모든 기능 함수에 대해 모든 에러 체크 이후 데이터 insert/select 등의 직접적 기능을 위한 쿼리가 실행되도록 작성하였다. 기능에 대해서는 대체로 spec 문서를 따라 작성되어 특이사항이 거의 없으며 아래는 비교적 복잡한 쿼리를 사용한 함수에 대한 설명이다.

print_movies() : reservation의 데이터를 통해 movie_id에 따른 avg_price, reserve_num, avg_rating을 구하고, 이를 movie에 LEFT OUTER JOIN (ON id=movie_id)을 하여 출력에 필요한 모든 column을 구하였다.

recommend_popularity() : print_movies()에서와 비슷한 쿼리를 이용하는데, WHERE 절에서 movie의 id가 reservation에서 해당 user_id의 유저가 예매한 movie_id에 포함이 안되는 것만 SELECT 되도록 하여, 유저가 보지 않은 영화만 추천하도록 하였다. 또한 rating에 따라 정렬할 때, avg_rating IS NULL ASC를 먼저 넣어주어, null값이 빠지지 않으면서, 대신 별점이 있는 영화의 뒤로 가도록 하였다.

recommend_item_based() : numpy 라이브러리를 이용하여 반복문으로 구성될 부분을 함수 이용으로 간략화하여 spec에 따라 구현을 하엿다.

구현한 내용

1. table 설계 및 제거 (Problem 1, 15)

모든 CREATE TABLE/DROP TABLE 문을 try~except 문에 넣어 이미 테이블이 존재하는/존재하지 않는 경우에도 에러로 인해 종료되지 않도록 하였다. (table schema는 위에 설명)

2. 데이터 단순 입출력 (Problem 2~7)

spec의 내용에 맞게 자주 사용하는 query를 모듈화한 함수를 이용하여 입출력 코드를 간단하게 구현했다.

3. DB와 Application의 복합적 연동 (Problem 8~13)

problem 8에서는 reservation에 INSERT, 9에서는 reservation의 데이터를 UPDATE하는 내용으로 구현했고, 10, 11에서는 JOIN ~ ON을 이용하여 해당 movie/user id에 해당하는 데이터를 출력해주었다. Problem 12에서는 nested query와 ORDER BY를 활용하여 SELECT 단계에서 필요 데이터를 모두 가져올 수 있도록 하였으며, 마지막 13에서는 numpy를 이용하여 nested for문으로 다수 구성될 계산을 내재된 함수로 축약하여 가독성을 높여 구현했다.

느낀 점 및 기타사항

과제를 진행하면서 나를 가장 당황하게 했던 것은, 프로그램을 exit할 때마다 DB의 데이터가 모두 날라가는 것이었다. mysql-connector 사용에 대해, tutorial에 나와있는 것만 사용하면 문제 없이 될 것이라 생각을 했지만, 분명히 내가 작성한 코드에 대해서 DB의 데이터를 날려버릴만한 코드는 존재하지 않았다. 이에 대해 처음에는 원래 이런 것인가 하며 그냥 놔두었다가, 질문 게시판에서 누군가가 DB의 데이터가 exit할 때마다 날라간다는 질문을 볼 수 있었다. 이에 대해 이 질문자는 구버전의 run.py를 이용하다보니 그랬다는 것이었지만, 나의 경우는 아니었다. 그래서 이것이 문제였다는 것을 깨닫고 질문 게시판에 질문을 올리게 된 것이다. 그러고 나서 잠을 청한 이후 답변을 확인하면서 생각을 해보았다. 추가적인 코드가 데이터 소실에 영향을 주지 않았다면, 오히려 부족한 코드가 영향을 주었는지를 생각해보고, reference를 찾게 되었다. 그리고 깨달았다. connection.commit()을 해주어야 데이터가 commit되어 보존이 된다는 점을. 이 commit()이 필요하다는 것을 알지 못해 코드를 다 짜는 동안에 시행착오를 겪었어야 했다. 그래도 이를 수정한 시점에 다른 모든 코드를 다 짜놓아서 마지막 개인 테스트까지 그리 오랜 시간이 걸리지 않아 다행이었다. 마지막 프로젝트이기도 하고, 구조적으로 꽤나 재밌는 프로젝트였다고 생각한다.