프로젝트 2

2014-15703 송화평

0. 개발 환경

본 프로젝트는 MacOS Catalina 10.15.4 운영체제가 탑재된 컴퓨터에서 Python 3.7.7, MySQL 1.3.12, PyMySQL 0.9.3 버전의 환경에서 실행되었다.

1. 구현한 내용

한 학기 동안 배운 DBMS와 SQL에 대한 지식을 기반으로 간단한 공연 티켓 예매 시스템을 Python을 이용하여 구현하였다. 이 시스템은 사용자와 데이터베이스를 중개해주므로, 사용자는 SQL에 대한 지식 없이도 미리 구현된 여러 API를 통해 DB를 조작할 수 있다.

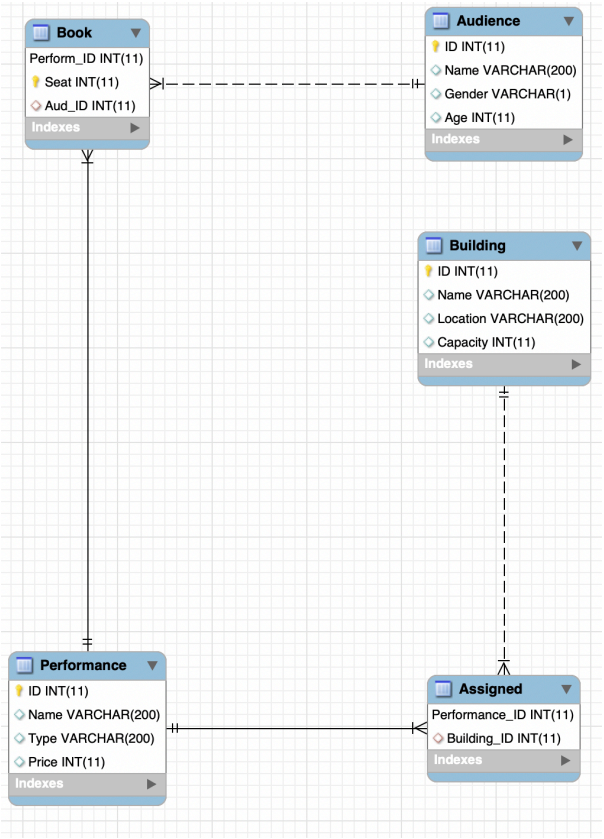
2. 핵심 모듈과 알고리즘

시스템을 구동시켰을 때 무한루프를 돌며 명령을 받아오게 된다. 이 부분은 `if __name__ == '__main__'`으로 감싸, 각 명령에 맞는 함수를 실행하도록 하였다. 이 때 구현을 보조하기 위해 `performanceExist()`, `buildingExist()`, `audienceExist()` 함수들을 추가로 구현하였다. 이들은 단순히 현재 데이터베이스에 인자로 들어온 performance ID, building ID, audience ID에 해당하는 레코드가 존재하는 지 판단하여 True, False를 반환하도록 하였다.

각 스키마는 명세에 맞추어 구현하였다. Building, Performance, Audience 세 스키마 모두 Primary Key로서 `ID`를 가지는데, Auto-increment 옵션을 주어 따로 명세하지 않아도 자동으로 1씩 증가하도록 하였다. 데이터 타입이 문자열인 attribute들은 `VARCHAR(200)`으로 설정하였으며, `'M'` 또는 `'F'`만을 가지는 Audience의 Gender attribute의 경우 `VARCHAR(1)`로 설정하였다.

사전에 명세된 이 세 스키마 뿐만 아니라 구현을 보조하기 위하여 Assigned와 Book의 두 스키마를 추가로 구현하였다. Assigned 스키마는 어떤 performance가 어떤 building에 assign되었다는 정보를 나타내기 위한 것으로, `Performance_ID`, `Building_ID`를 attribute로서 가지며, Primary Key는 `Performance_ID`이다. 이는 하나의 performance는 하나의 building에만 assign될 수 있지만, 하나의 building은 여러 performance에게 assigned to 될 수 있다는 one-to-many constraint를 반영한 것이다. 또한 이 두 attribute는 모두 Building과 Performance를 Foreign Key로서 reference하고 있는데, 두 경우 모두 `ON DELETE CASCADE` 옵션을 설정하여 만약 reference하고 있는 레코드가 삭제될 경우 이를 reference하고 있는 Assigned 테이블의 레코드도 삭제되도록 하였다. 또한 Book 스키마는 어떤 audience가 어떤 performance를 book하였다는 정보를 나타내기 위한 것으로, `Perform_ID`, `Seat`, `Aud_ID`를 attribute로 가지고 있다. `Perform_ID`와 `Seat`가 Primary Key를 구성하고 있으며, `Perform_ID`와 `Aud_ID`가 Foreign Key로서 각각 Performance와 Audience를 reference하고 있다. 이 경우에도 `ON DELETE CASCADE` 옵션을 설정하였다. 여기서 `Perform_ID`와 `Aud_ID`로 attribute를 이름지은 이유는 foreign key namespace에 중복되는 이름이 있어서는 안 된다는 제약 사항이 있었기 때문이다.

이 스키마들을 다음과 같이 EER 다이어그램으로 표현할 수 있었다.



위와 같이 정의한 DB의 테이블과 상호작용할 수 있도록 각 함수를 구현하였다. 대부분의 함수는 입력 – 입력 점검 – 적절한 SQL 생성 – SQL 실행 – 실행 결과 표시와 같은 루틴을 따라가므로 주석만 읽고도 어렵지 않게 그 흐름을 유추할 수 있도록 구현하였다. 추가적인 설명이 필요한 함수는 `assignPerformanceToBuilding()` 함수와 `bookPerformance()` 함수, `printTicketStatus()` 함수, `resetDB()` 함수이다.

`assignPerformanceToBuilding()` 함수는 building ID와 performance ID를 받아 해당 performance를 building에 assign 해주는 함수이다. 이 때 performance가 다른 building에 이미 assign된 상태인지 검사를 하지 않고 바로 SQL을 작성하여 실행한다. 만약 그러한 상태라면, Primary Key Constraint에 의해 예외가 발생하여 except 구문에서 잡아내게 된다. 이 경우의 error number는 1062이므로, 적절한 문구를 출력하고 return하도록 구현하였다. 처음에는 building과 performance가 DB에 있는 지 검사하는 부분 역시 error number를 이용하여 처리하였지만, 3.에서 각 입력이 들어오는 즉시 해당 입력에 대한 유효성 검사를 실시한다는 가정 때문에 위에서 서술한 것과 같이 따로 구현한 함수를 이용하여 검사하도록 하였다.

`bookPerformance()` 함수가 실행되면 performance ID를 입력받아 해당 performance가 실제로 DB에 존재하는 지 확인한 후, 열리는 building의 capacity와 해당 performance의 price 정보를 얻어온다. 만약 이 정보를 얻어오지 못하면 performance는 존재하나 어느 building에도 assign되지 않은 것으로 간주한다. 이후 audience ID를 입력받아 존재성을 검사하고 나이 정보를 얻어온다. 또한 어느 자리를 예매할 것인지에 대한 정보도 입력받아, 해당 자리가 적절한 자리인지, 또는 이미 누군가가 예매한 자리인지 검사한다. 이 절차가 끝나면 모든 검사가 끝난 것이다. 이후 audience의 나이에 맞게 price에 보정을 하여 effective price를 계산하고, 예매한 자리의 수를 곱하여 최종 가격을 산출한 뒤 각 자리를 실제로 예매하도록 SQL을 실행하고, 그 결과를 출력하여 보여준다.

`printTicketStatus()` 함수는 performance ID를 입력받아 해당 performance의 좌석과, 그 좌석이 예매된 경우에는 누가 예매했는지를 보여주는 함수이다. performance ID의 존재성을 검사한 후에 그 performance가 어느 building에서 열리는 지 SQL로 정보를 받아온다. 만약 받아오지 못했으면 이 경우에는 performance는 존재하지만 아직 assign되지 않은 경우이므로 return하도록 하였다. 이어 해당 building의 capacity 정보를 가져오고, Book 테이블로부터 해당 performance의 예매를 불러와 누가 어떤 자리에 예매했는지에 대한 정보 또한 가져온다. 이를 리스트로 가공하여, for문 안에서 각 좌석을 traverse하면서 해당 좌석 번호에 예매자가 있는 경우 예매자의 ID를

출력하도록 구현하였다.

resetDB() 함수를 잘못 실행한 경우 모든 레코드가 날아갈 수 있으므로, 재차 확인하도록 하였다. 확인을 받으면 SQL을 통해 모든 테이블을 drop하도록 하고, 상세에 맞도록 다시 모든 테이블을 create하도록 구현하였다.

3. 가정한 것들

사용자로부터 여러 입력을 받는 경우, 모든 입력을 받고 해당 입력에 대한 유효성 검사를 실시하는 것이 아니라, 매 입력마다 해당 입력에 대한 유효성 검사를 실시하도록 하였다. 또한 어떤 이유로 SQL에서 예외가 발생한 경우 시스템이 종료되지 않도록 하였고, 명세된 문장을 출력하도록 명시된 몇몇 경우를 제외하고는 print(e)처럼 작성하여 SQL의 에러 메시지가 사용자에게 노출될 수 있다고 가정하였다.

4. 컴파일과 실행 방법

PyMySQL을 설치한 후 버전 3.5 이상의 Python에서 '**python main.py**' 또는 '**python3 main.py**'를 입력하여 실행할 수 있다. 본 구현에서는 virtual environment 환경에서 PyMySQL을 설치하고 진행하였다.

5. 느낀 점

다른 과제를 먼저 수행하느라 하루 늦게 진행하였지만, 굉장히 재미있게 수행한 과제였다. 본 수업의 다른 과제들은 에러 핸들링과 엄청난 분량때문에 힘들었지만 이번 과제는 에러가 날 만한 곳이 얼마 없었고, 분량도 750줄 정도로 비교적 짧아 재미있게 수행할 수 있었다. 기말고사가 끝난 후에 수행하는 과제이기 때문에 마치 영화의 에필로그를 보는 느낌이었다.