Database Project 2 Report

2013-11431 정현진

0. 개발 환경

다음과 같은 환경 속에서 프로젝트를 진행하였다.

- OS: Windows 10 (Version 2004, Build 19041.264)
- IDE: Visual Studio Code (Version 1.46.0) with the Python extension
- Python: Python 3.7.6 on Conda
- PyMySql: 0.9.3

1. 핵심 모듈

프로젝트 2 는 다음과 같은 세 가지 파일로 이루어져 있다.

1. main.py
2. database.py
3. actions.py

1-1. main.py

프로그램의 진입점이며, 프롬프트 출력과 명령어를 입력 받은 뒤 해당 명령어에 맞는 함수를 실행하는 역할을 한다.

1-2. database.py

데이터베이스 서버 연결, 초기화, 쿼리문 수행 등 데이터베이스와 관련된 클래스를 정의하였다.

1-3. actions.py

프로젝트 명세서에 정의된 다양한 명령어를 구현하는 함수들을 정의해둔 파일이다. main.py 에 같이 두기엔 코드가 너무 길어져서 따로 모듈화하였다.

2. 구현 내용 및 알고리즘

본 프로그램의 실행 순서를 간략히 살펴보면 다음과 같다:

1. 명령어를 입력 받는다.
2. 해당 명령어를 수행한다.
3. 종료 명령어가 아니면 1 번으로 돌아가서 반복한다.

상당히 간단한데, 이를 조금 더 자세히 풀어 쓰면 다음과 같다.

우선 main.py 에서 초기 프롬프트를 출력한 후, 명령어를 입력 받는다. 그 후 입력 받은 명령어 번호에 따라 actions.py 에 정의된 함수들을 실행한다. actions.py 파일의 함수들은 database.py 에 정의되어 있는 Database 클래스의 인스턴스를 인자로 받아 데이터베이스와 연동하여 필요한 작업들을 수행하게 된다. 이 때 Database 클래스의 인스턴스는 프로그램에서 단 하나만 존재하면 되므로, Database 클래스는 Singleton 패턴으로 구현하였다.

database.py 파일의 구현에 대해서는 각 명령어들을 실행할 때 모든 명령어를 Database 클래스 안의 메소드로 정의해서 실행하는 방법도 있지만, 해당 방법은 하나의 클래스가 너무 많은 책임을 진다고 생각해 Database 클래스 내부에는 INSERT, DELETE 문에 대응하는 'execute()' 함수와 SELECT 문에 대응하는 'fetch()' 함수만을 정의한 뒤, actions.py 파일의 각각 함수에서 필요한 SQL 을 생성해서 두 함수들을 사용하도록 구현하였다. 다만 Database 클래스가 생성되거나 reset 될 때 '_initialize_database()' 함수를 통해 테이블 스키마를 생성하는데, 해당 부분은 CREATE TABLE 문을 내부에서 직접 사용하고 있다.

주어진 프로젝트의 데이터 구성이 그렇게 복잡하기 않아서 actions.py 파일의 구현은 크게 어려운 부분이 없었고 모두 직관적이었다. 해당 파일 구현 중 특이점으로 반올림 함수를 직접 구현하였는데, 이는 Python 의 내장 round() 함수가 일반적으로 생각하는 방식과 다른 점이 있기 때문이다 (예를 들어 round(3.5)와 round(4.5) 가 모두 4 를 반환한다). 그리고 결과 출력을 포맷해주는 함수인 'format_results()'의 출력 양식은 프로젝트 명세서의 예시와 동일하게 구현하였다. 또한 15 번 명령어인 프로그램 종료는 적절한 처리를 위해 actions.py 파일에서 구현하지 않고 main.py 파일에서 직접 구현하였다.

프로그램의 전체 테이블 스키마를 살펴보면, 아래와 같이 5 개의 테이블이 존재한다.

1. building (공연장)
2. performance (공연)
3. audience (관객)
4. assign (공연장에 공연 할당)
5. book (공연에 관객 예매)

1-3 번 테이블은 프로젝트 명세서의 2 번 항목 그대로 정의했으며, 4 번과 5 번 테이블은 각각 공연장-공연 관계와 공연-관객 관계를 나타낸다.

assign 테이블은 performance_id 와 building_id 를 칼럼으로 가지고 있으며 둘 모두 foreign key 이다. 그리고 performance_id 를 primary key 로 지정해 하나의 공연장에 여러 공연이 할당될 수 있게 했으며, 각 foreign key 에 ON DELETE CASCADE 를 적용해 공연장이나 공연의 row 가 삭제될 경우 연관된 assign 의 row 도 모두 삭제되게 구현하였다.

book 테이블은 performance_id, audience_id, seat_number 를 칼럼으로 가지고 있으며 (공연장, 좌석 번호)의 조합은 유일해야 하기 때문에 performance_id 와 seat_number 를 primary key 로 정의하였고, performance_id 와 audience_id 를 foreign key 로 설정하였다. 이 때 performance_id 는 performance 테이블이 아닌 assign 테이블을 참조하도록 하여 할당되지 않은 공연은 예매할 수 없다는 개념을 적용시켰다. book 테이블에서도 foreign key 에 각각 ON DELETE CASCADE 를 적용해주었다.

따라서 공연장이 삭제될 경우 building -〉 assign -〉 book 의 순서로 삭제가 이어지게 되고, 공연은 performance -〉 assign -〉 book, 관객이 삭제되면 audience -〉 book 으로 삭제가 이어진다.

3. 구현하지 못한 내용

프로젝트 명세서에 기재되어 있는 내용 중 구현하지 못한 부분은 없는 것 같다.

4. 가정한 것들

프로젝트 명세서의 명령어 중 update 와 관련된 부분은 없었기 때문에, 테이블들을 정의할 때 foreign key 의 update 는 없다고 가정하고 설계하였다.

5. 컴파일과 실행 방법

Python 으로 프로젝트를 진행했기 때문에 별도의 컴파일은 진행하지 않았고, PyMySql 은 'pip install pymysql' 명령어를 통해 설치해주었다.

다만 코드에서 SQL 을 정의할 때 f-string 을 많이 사용했는데, 이는 Python 3.6 버전부터 도입된 기능이기 때문에 3.6 버전 이상의 Python 이 설치되어 있어야 코드가 오류 없이 잘 작동한다.

코드의 실행은 압축 파일을 푼 후 main.py 파일의 경로에서 'python main.py'를 통해 실행한다.

6. 느낀 점

이번 프로젝트 2 는 프로젝트 1 에 비하면 간단한 과제였던 것 같고, 큰 스트레스를 받지 않고 즐겁게 구현했던 것 같다. 또한 테이블 스키마를 설계하고 구현하면서 수업 시간에 배웠던 내용들을 전체적으로 가볍게 복습할 수 있었다.