PROJECT 2 DB APPLICATION

2015-12740 이예원

핵심 알고리즘 및 모듈 :

간단한 티켓 예약 시스템입니다. 사용자는 관중, 공연, 공연장을 등록하고 삭제할 수 있으며, 공연 예약, 공연장 배정 업무, 예약 확인 업무를 수행할 수 있습니다.

사용자가 콘솔창에 출력된 메뉴를 참고하여 수행할 업무의 번호를 입력하면, dispatcher 모듈이 해당하는 업무 method 를 호출합니다.

각 method는 추가적인 정보가 필요할 경우 사용자의 입력을 받아 해당하는 작업을 수행하고 성공 또는 실패했다는 결과값을 반환합니다.

Dispatcher 모듈이 이 결과값을 반환하면, 성공했을 경우 Database에 결과를 Commit하고 그렇지 않을 경우 다음 업무 번호를 받습니다.

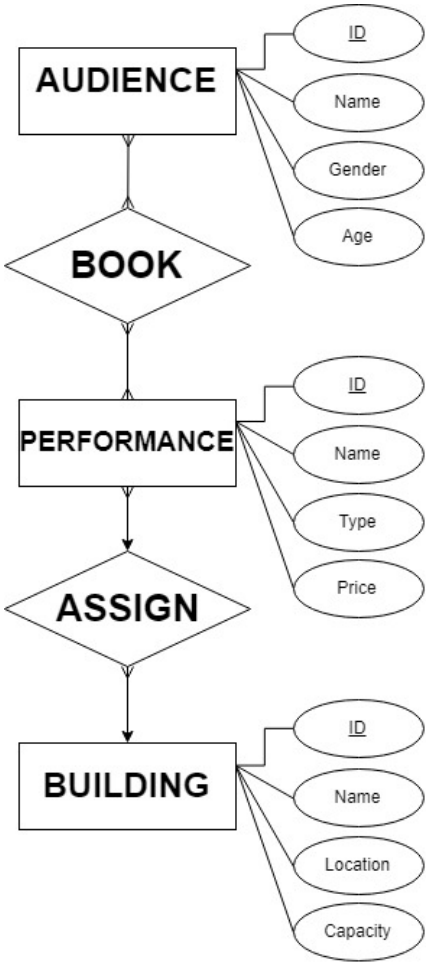
사용자가 15번(프로그램 종료)을 입력할 때까지 계속해서 반복합니다. 사용자가 15번을 입력하면 이 때까지의 결과를 Database에 Commit하고 프로그램을 종료합니다.

개발환경 :

Windows 10, Python 3.7, PyCharm 2019.1.1, pyMySQL, MariaDB 10.3(x64)

구현 내용 :

Audience, Performance, Building, Book 총 네 개의 테이블을 기반으로 동작하는 티켓 예약 시스템입니다. 구현 내용의 이해를 위해 E-R Diagram과 Relational Diagram을 첨부하였습니다.



E-R Diagram

Entity는 총 세 개로 관중, 공연, 무대 테이블이 있습니다.

각각의 테이블은 ID 필드가 Primary Key이며 이름, 장소, 성별 등의 정보를 담고 있습니다.

Relationship은 총 두 개로,

1) 관중 – 공연을 잇는 '예약' relation,

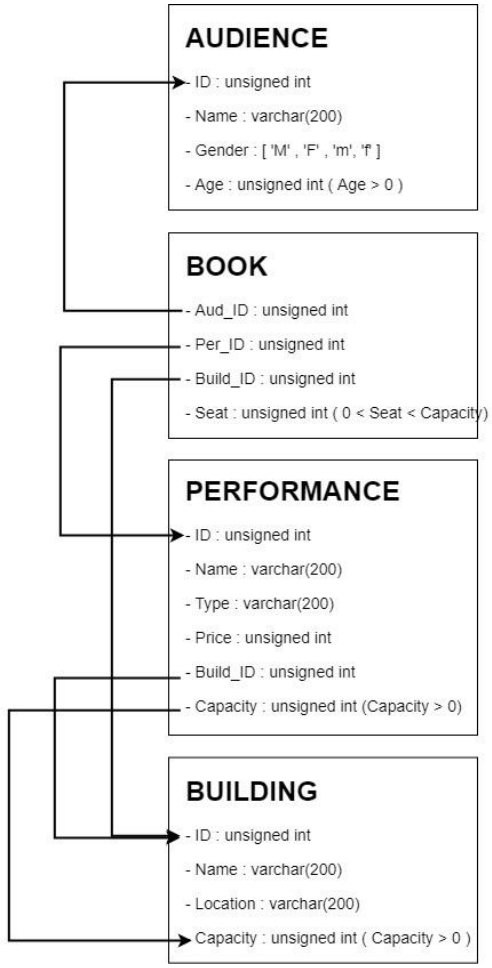
2) 공연 – 무대를 잇는 '배정' relation이 있습니다.

'예약' relation은 seat number 필드를 가지는 Weak entity이며, 관중 id와 공연 id, 좌석 번호를 조합한 PK를 가집니다

이를 실제로 구현할 때에는 Relation의 경우 Book은 테이블로 구현하였으나 ASSIGN의 경우 따로 가지는 필드도 없고 m:1 방식이기 때문에 PERFORMANCE 테이블의 Column 으로 구현하였습니다.

또한 테이블마다 몇몇 필드가 추가되었습니다.

Relational Diagram



관객의 정보를 저장하는 Audience,

예매 정보를 저장하는 Book,

공연 정보를 저장하는 Performance,

공연장 정보를 저장하는 Building

총 네 개의 테이블로 구현하였습니다.

ERD의 field와 달리 추가된 Column에는

예약 시 편의를 위해 Performance table에

Building Id와 Capacity를 추가하였고,

공연장이 하나 삭제될 때 on delete cascade를

구현하기 위하여 Book table에서는 Building id를

참조하도록 하였습니다. 이 때 공연은 삭제되는 것

이 아니라 해당하는 Column 만 null이 되도록

on delete set null 옵션을 사용하였습니다.

프로그램이 실행되면 connect() method가 서버 Database에 연결하며, 이때 연결한 Instance를 global variable로 선언하여 모든 method에서 사용할 수 있도록 합니다. 프로그램이 한 번 실행되면 종료되기 전까지 연결을 끊지 않습니다. 이후 struct() method에서 필요한 스키마가 잘 만들어져 있는지 체크합니다. 만약 스키마가 없다면 새로 만듭니다. Welcome message로 사용자가 선택할 수 있는 메뉴가 출력되며, 사용자가 번호를 입력하면 해당하는 모듈이 데이터베이스에 데이터를 저장하거나, 불러오거나, 수정한 후 사용자에게 결과를 출력합니다. 만약 결과가 성공이면 DB를 Commit하고 그렇지 않으면 바로 다음 업무 번호를 받습니다.

Struct() 모듈 :

DB에 필요한 Schema가 모두 있는지 확인하고 없으면 Create Table Query를 실행하는 모듈입니다. Create table에서 CHECK constraint를 사용하여 나이의 범위, 정원의 범위 등을 제한하고자 하였으나 mySQL 매뉴얼에 따르면 CHECK clause는 파싱만 되고 실제로 동작하지는 않습니다.

("The CHECK clause is parsed but ignored by all storage engines" -https://dev.mysql.com/doc/refman/5.7/en/create-table.html)

따라서 공연장 등록, 공연 등록, 관중 등록을 수행할 때 애플리케이션 수준에서 입력 범위가 제한되도록 구현하였습니다.

Dispatch() 모듈 :

업무 번호를 전달받고 해당하는 method의 결과값을 반환합니다. 이하의 모든 업무 모듈은 True 또는 False 값을 반환합니다.

1번 업무 – printB() 모듈 :

Building Table에서 모든 row를 select합니다. 이후 Performance Table에서 해당 Building ID 값을 가진 row의 개수를 찾아 모든 결과를 출력합니다. 만약 Building Table에 아무것도 존재하지 않으면 헤더 아래에 'There's no building' 메세지를 출력합니다. 만약 어떤 building이 assigned 된 performance가 아무 것도 없으면, 즉 Performance Table에 해당 building ID 값을 가진 row 가 없으면 Assigned 열이 0이 됩니다. 출력이 끝나고 성공/실패 여부를 반환합니다.

이처럼 Building table에서 Assign 필드를 만들어 관리하지 않고 print 명령이 호출될 때마다 Temporal Output을 참고하도록 하여 Table의 Space overhead를 줄이고자 하였습니다.

2번 업무 – printP() 모듈 :

1번과 마찬가지로 동작하지만 1번과는 달리 Performance Table과 Book Table에 접근합니다. Count(distinct aud_id) 조건을 사용하여 중복 없이 해당 공연을 예약한 관중 수를 출력합니다.

3번 업무 – printA() 모듈 :

두 개의 테이블에 접근할 필요 없이 Audience 테이블의 모든 row를 출력합니다.

4번 업무 – insertB() 모듈 :

Building Table에 입력할 row 정보를 입력 받습니다. 입력 받을 때 name 과 location 문자열 길이가 200을 넘으면 Truncate 합니다. 또, Capacity가 0 이하이면 입력을 거절하고 False를 반환합니다. 입력이 올바르면 insert query를 수행하고 True를 반환합니다.

5번 업무 – removeB() 모듈 :

Building Table에서 찾을 ID를 입력 받습니다. Delete query를 수행하여 테이블에 해당 ID가 존재하지 않으면 에러 메시지를 출력하고 False를 반환합니다. 존재할 경우 해당하는 ID를 삭제하고 True를 반환합니다. Struct() 모듈에서 create table로 테이블을 선언 할 때 on delete cascade, on delete set null 옵션이 있기 때문에 공연장이 삭제되면 Performance테이블의 해당하는 row의 buildingID , Capacity 칼럼이 null이 되고 Book 테이블의 해당 row가 사라집니다.

6번 업무 – insertP() 모듈 :

4번 업무와 같은 방식으로 작동하며, 사용자에게 입력 받지 않은 정보인 building ID, Capacity는

null로 설정됩니다.

7번 업무 - removeP() 모듈 :

Performance Table에서 ID를 찾아 삭제합니다. Book 테이블에서 Performacne ID 를 on delete cascade 옵션으로 참조하고 있기 때문에 해당하는 ID가 Book 테이블에 존재하면 해당 row가 삭제됩니다.

8번 업무 – insertA() 모듈 :

4번 업무와 비슷한 방식으로 작동하며, 사용자에게 성별을 입력 받을 때 대소문자 구분 없이 M, F 중 하나만을 입력받는지 체크합니다.

9번 업무 – removeA() 모듈 :

7번 모듈과 같은 방식으로 작동합니다.

10번 업무 – setP2B() 모듈 :

공연장 ID 와 공연 ID를 입력받습니다.

공연장 ID를 이용하여 공연장 Table에서 해당 ID의 capacity 값을 찾아 저장합니다.

공연 테이블에서 공연ID에 해당하는 row를 찾아 buildingID, capacity 칼럼을 업데이트합니다.

이 때 공연 ID와 공연장 ID가 valid한지 각각 확인하고 그렇지 않으면 False를 반환합니다.

또, 만약 두 칼럼이 null이 아니면 이미 배정되었다는 에러를 출력한 후 False를 반환합니다.

성공할 경우 True를 반환합니다.

11번 업무 – book() 모듈 :

공연 ID, 관중 ID, 좌석번호리스트(e.g "2 , 3 , 4")를 입력 받습니다.

공연ID와 관중ID가 valid한지 확인하면서 building id, capacity, price, age 정보를 저장합니다.

만약 공연 ID나 관중 ID가 존재하지 않으면 각각 메시지를 출력하고 False를 반환합니다.

존재한다면 좌석번호 리스트의 각 좌석번호가 capacity보다 크거나 0보다 작지는 않은 지, 그리고 book 테이블에 이미 존재하는지 확인하고 만약 그렇다면 각각 메시지를 출력하고 False를 반환합니다.

모든 검사를 통과하면 각각의 좌석번호가 하나의 row로 book table에 입력됩니다. 입력받은 공연 ID, 관중 ID, 좌석 번호 외에 아까 전에 저장한 공연장 ID를 함께 저장하여 공연장이 삭제될 때 함께 삭제될 수 있도록 합니다.

마지막으로 calprice() method에서 총 가격을 계산하여 출력하고 True를 반환합니다.

11번 업무2 – calprice()모듈 :

11번 업무를 도와주는 support method입니다. 우대 조건에 맞추어 총 가격을 반환합니다.

12번 업무 – printP2B() 모듈 :

공연장 ID를 입력받습니다.

입력받은 ID가 Building Table에 존재하는 Valid한 input인지 확인하고 그렇지 않으면 False를 반환합니다.

존재하는 공연장일 경우, Performance Table에서 공연장 ID를 buildingID 칼럼 값으로 가지는 row를 모두 select합니다.

만약에 아무것도 선택되지 않았으면, 헤더를 출력하고 'There is no performance' 메시지를 출력하고 True를 반환합니다.

Select 된 모든 row마다 performance ID의 값으로 book table에서 count 값을 계산하여 변수에 저장하고 모든 정보를 출력하고 True를 반환합니다.

13번 업무 – printAtoP() 모듈 :

공연 ID를 입력받습니다.

입력받은 ID가 Valid한지 확인하고 그렇지 않으면 False를 반환합니다.

존재하는 공연일 경우, Book table과 Audience Table을 Join하여 공연 ID에 해당하는 row를 모두 select합니다.

아무것도 선택되지 않으면 'There is no Audience' 메시지를 출력하고 False를 반환합니다.

필요한 정보를 출력하고 True를 반환합니다.

14번 업무 – printPstat() 모듈 :

공연 ID를 입력받습니다.

입력 받은 ID가 Valid한지 확인하고 그렇지 않으면 False를 반환합니다.

입력 받은 ID의 row가 Assigned 되었는지 확인하고 그렇지 않으면 False를 반환합니다.

이 과정에서 Capacity 값을 저장합니다.

입력 받은 ID에 해당하는 row를 book table에서 모두 select 합니다.

이후 capacity 값 만큼 loop를 돌며 모든 좌석의 예매 현황을 출력합니다. 매번 select된 결과값의 seat number와 같은 좌석번호가 있는지 확인하고, 있다면 audience id를 출력하고 그렇지 않으면 공란으로 비워둡니다

모두 출력한 후 True를 반환합니다.

15번 업무 - main함수에서 자체적으로 수행합니다.

16번 업무 – clear() :

사용자에게 y/n 입력을 받아 'y' 입력만 통과시키고 나머지는 모두 False를 반환합니다.

Drop Database, Create Database 명령을 수행하여 DB를 삭제하고 다시 만듭니다. 새로 만들어진 DB에 connect()를 이용하여 다시 연결하고 struct()를 수행하여 빈 테이블을 만들고 commit합니다.

구현하지 못한 내용 : Speculation 안내문에 나와있는 모든 입력이 정상적으로 동작하였습니다.

가정한 것들 : 공백만 입력되는 경우는 없다고 가정하였습니다. 또, int 값이 입력되어야 할 때 다른 data type이 입력되지 않는다고 가정하였습니다.

컴파일과 실행방법 :

Python으로 컴파일 하였습니다. 커맨드 창에서 Main.py가 저장된 경로로 간 후 python main.py를 입력하면 프로그램이 실행됩니다. Pymysql이 저장되어 있지 않으면 pip install pymysql로 라이브러리를 설치합니다.

추가 Message :

```
Cannot connect to DB. Closing System . . .
```
서버 데이터베이스에 연결 실패

```
There is no building
```
공연장 테이블이 비어 있음

```
There is no performance
```
공연 테이블이 비어 있음

```
There is no audience
```
관중 테이블이 비어 있음

```
Do you want to reset Database? [y/n]
```
데이터베이스 리셋 확인 메시지

```
Canceling . . .
```
[y/n] 중 n을 입력하였음

```
Invalid input! Back to prompt . . .
```
입력이 y나 n이 아님

```
Printing Performance Failed.
```

```
Printing Building Failed.
Printing Audience Failed.
Remove Building Failed.
Insert Performance Failed.
Remove Performance Failed.
Insert Audience Failed.
Remove Audience Failed.
Performance Setting Failed.
Book Failed.
Printing Performance in the Building Failed
Printing Performance in the Building Failed
Printing Status Failed
Clearing DB Failed.
```

이외 지정되지 않은 오류로 업무 수행 실패

프로젝트를 하면서 느낀점 :

첫번째 프로젝트보다 훨씬 직관적이어서 재미있게 작업할 수 있었습니다. 이 프로젝트 이전에 MariaDB를 Django와 연동하여 사용해 본 적이 있었는데 이번에는 pyMySQL을 통해 사용해보면서 둘의 차이점을 확인할 수 있었습니다. Django의 경우 SQL 문법보다 더 쉬운 새로운 문법으로 DB를 조작하기 때문에 편리하지만 문법을 새로 외우고 제한적으로 사용할 수 있다는 단점이 있었습니다. pyMySQL의 경우는 execute 명령을 통하여 대부분의 SQL Query를 전달하는 방식이기 때문에 수업에서 배운 문법으로 다양한 SQL Query를 직접 짜볼 수 있어서 좋았습니다. 다만 outer join이나 unique같은 여러 요소를 사용해보고 싶었는데 제가 생각한 알고리즘은 다소 간단한 Query만을 사용하였기 때문에 그 부분이 아쉽습니다.

구현하면서 MySQL이 생각한 대로 작동하지 않는 부분이 몇 있었는데 한 예로는 cursor로 execute에 argument를 전달할 때 %d는 전달하지 못한다던가 (예를 들어 insert into A values (%s, %d) 이런 식으로 사용하면 오류가 납니다.), Create Table Query에서 Check clause는 작동하지 않는다던가 하는 부분이 있었습니다. 이 부분은 왜 이럴까 하는 궁금증이 들었고 그 이유를 찾는 과정에서 Manual의 중요성을 다시 한 번 느낄 수 있었습니다.

이번 학기에 데이터베이스 수업을 들으면서 제 개인적으로 여러 프로그래밍 프로젝트를 병행할 기회가 있었습니다. 그런데 웹 서비스 개발을 하든, 앱 개발을 하든 대부분의 소프트웨어는 데이터베이스를 필수적으로 포함하고 있으며 그렇기 때문에 어떤 분야든 SQL이나 데이터베이스 구조에 대한 지식이 꼭 필요하다는 것을 느낄 수 있었습니다. 데이터베이스 수업에서 교수님도 여러 번 강조하셨지만 데이터베이스 자체가 실제 Data에 따라 implementation이 굉장히 다양해지기 때문에 현실과 굉장히 밀접한데, 이번 프로젝트를 통해 실전적 사고를 집중적으로 훈련할 수 있었습니다. 한 학기 동안 가르쳐 주셔서 감사합니다!