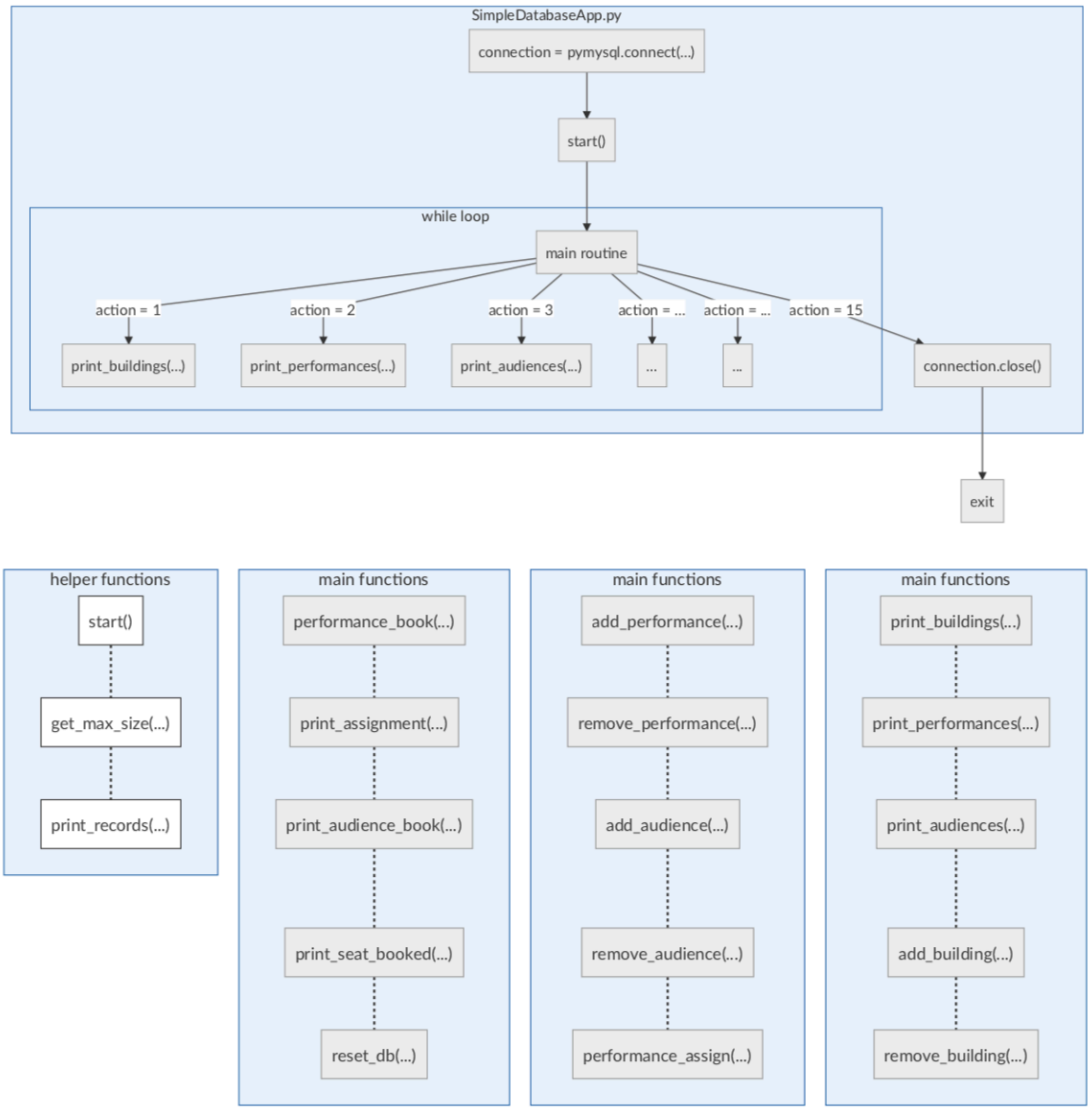
2014-10469 한인성

PROJ 2

IMPLEMENTING A SIMPLE DATABASE APPLICATION

1. 핵심 모듈과 알고리즘에 대한 설명

본 application 은 세가지 helper function 과 각 action 을 처리해주는 함수들 그리고 main routine 으로 구성되어있다.



각 action 에 해당하는 function 은 해당하는 작업을 수행한다.

- 예를 들어서 action 으로 1 을 입력했을 때 수행되는 print_buildings(...)는 DB 에 입력된 모든 공연장을 출력한다.

그 외에 helper function 은 main function 에서 자주 사용되거나 단순 print 인데 내용이 긴 코드를 묶는데 사용했다.

- **start()**
  - application 이 실행될 때 메뉴판을 출력하는 역할을 한다.
- **get_max_size(rows, I)**
  - records(rows)와 각 레코드의 attribute 수(I)를 받은 뒤에 각 attribute 마다 data 가 가장 긴 것의 길이를 찾아내 list 의 형태로 반환한다. 이 반환 list 는 추후에 print_records(...)에서 출력할 내용의 alignment 에 사용된다.

➢ 예를들어서 R = (id, name)이라는 스키마가 있다고 생각해보자

➢ rows = [(1, sujan), (2, john), (3, kim), (4, inseoung)], I = 2 로 주어졌을 경우

➢ id 는 모두 한 자리수로 길이가 같으므로 가장 긴 길이는 1 이고, name 의 경우는 “inseoung”가 길이 8 로 가장 길다. 따라서 [1, 8]을 반환한다.

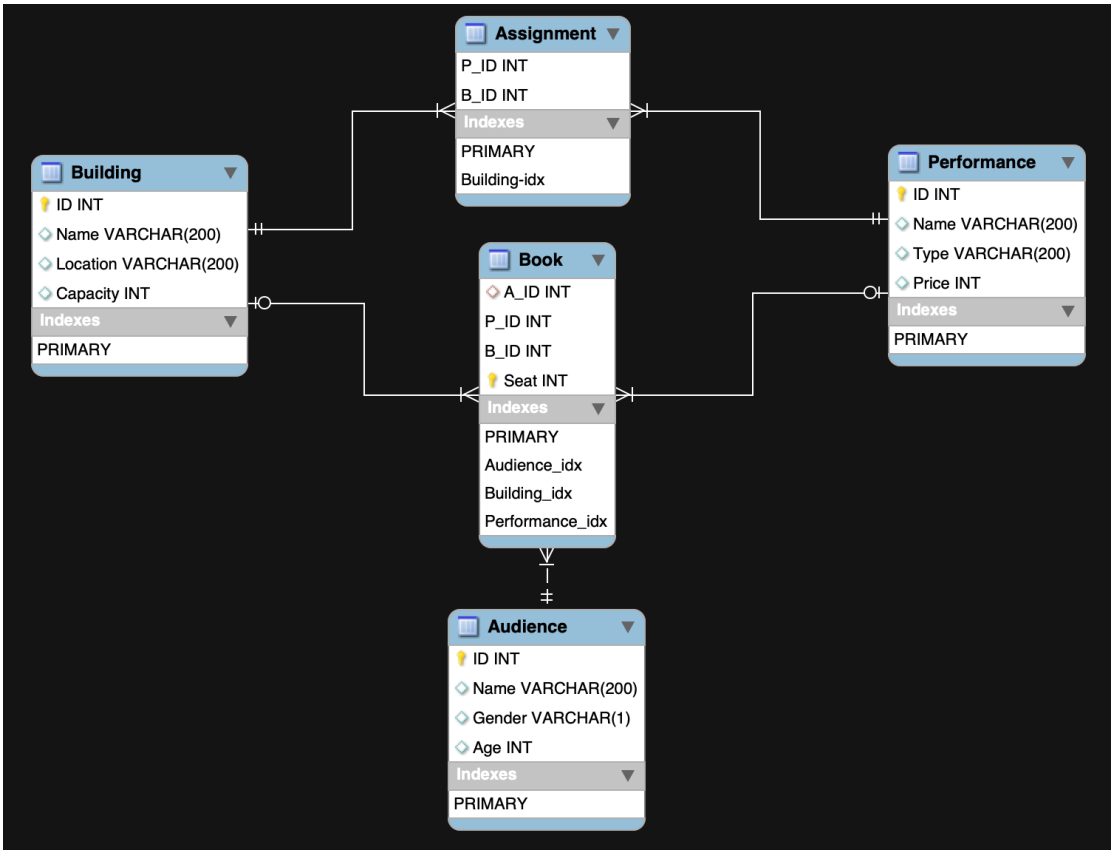
- **print_records(title, rows)**
  - 출력과 관련된 각 action 마다 출력할 record 를 양식에 맞춰 출력해야한다. 양식이 동일하기 때문에 print_record(...) 함수에 title 과 rows 를 주면 자동으로 양식에 맞춰 출력해준다.
  - title
    - record 를 출력하기 전에 맨위에 보여줄 머리글 이름들의 list. list 의 길이는 각 record 의 길이와 동일해야한다.
    - ex) [“id”, “name”]
  - rows
    - 출력할 records

우선 get_max_size(...)에 rows 와 title 의 길이를 넘겨준 뒤 record 에서 각 attribute 의 최대 길이를 구한다. 그리고 거기에 title 의 각 요소와 길이를 한번 더 비교해서 전체 출력할 내용에서 각 column 의 최대 길이를 구한다.

그 길이를 가지고 출력할 format string 을 만든다. 이 format string 을 이용하면 title 이나 record 를 출력할 때 자동으로 alignment 를 해준다.

2. 구현한 내용에 대한 간략한 설명

핵심 모듈과 알고리즘에 대한 설명에서 언급한 모든 helper function 과 main function 을 구현했다. 또한 application 에서 사용될 Database schema 를 MySQLWorkbench 를 이용하여 구현했는데 코드상에서는 RESET_SQL 로 되어있다. RESET_SQL 은 기존에 있던 table 을 모두 drop 하고 새로 table 을 create 한다.



대부분의 action 은 적당한 query 로 record 를 가져와서 작업하는 방식으로 했다.

이때 Database 에 연결할 때 주어진 DB cursor 로 query 를 실행하고 commit 하고 rollback 할 수있는데 cursor.excute(sql) 과 cursor.fetchall() 혹은 cursor.fetchone()으로 query 를 실행하여 record 를 가져오고 cursor.commit()으로 커밋하며 connection.rollback()으로 마지막 커밋상태로 되돌릴 수 있다. rollback 기능은 예매 action 에서만 사용했는데, 예매할 좌석 list 를 순회하면서 우선 Database 에 예매 정보를 넣어놓고 좌석 번호가 이미 예매된 것이거나 올바른 번호가 아닐 경우 rollback 하여 주어진 좌석 list 예매를 atomic 하게 처리했다.

3. 가정한 것들

1. *Gender 대소문자 구분 있음.*
2. *입력 시 각 입력의 데이터 타입은 지켜진다고 가정*
3. *database reset 의 경우(16 번 action) y 가 아니면 모두 n 로 처리했음. 즉, y 를 입력할 경우에만 reset 을 진행함.*
4. *spec 에 명시된 에러 외에 integrity constraint 를 위반하는 입력은 들어오지 않는다. (단, delete 의 경우 cascade delete 가 있으므로 용인)*

4. 컴파일과 실행 방법

- python 3.5+ 설치
  - python 공식홈페이지에서(https://www.python.org) python 3.5 버전 이상을 설치한다.
- pymysql 설치
  - (https://readthedocs.org/projects/pymysql/downloads/)에서 pymysql package 를 설치한다.
- 터미널 실행
  - 이후 터미널에서 다음과 같이 입력해주면 실행할 수 있다.
  - 예를들어 python 버전이 3.7 이면, python3.7 main.py

5. 프로젝트를 하면서 느낀 점

pymysql 을 사용하면서 파이썬으로 실제 DB 를 연동시켜 관련 application 을 구현하는 법을 배웠다. C 와 비교했을 때 파이썬이 범용성이 대단하고 다양한 모듈로 인하여 코드를 간결하게 짤 수 있다는 강점을 다시 한 번 확인하는 계기가 되었다. 또한, 이번 프로젝트를 진행하면서 application 이 DB schema 에 정말 민감하다는 것을 알게 되었다. attribute 가 추가되거나 table 이 추가되는 것은 큰 영향이 없을 것 같지만, 기존 attribute 가 삭제되거나 타입이나 integrity constraint 가 바뀐다거나 table 이 없어지는 경우는 application 의 작동을 못하게 할 수 도 있다. 이런 점에서 보았을 때 DB schema 는 "불변성"이라는 특징이 정말 중요하다고 생각한다. 물론 시대에 필요에 따라 schema 는 일부 변경될 수 있지만 DBA 는 최대한 해당 database 에서 어떤 서비스가 나올지 혹은 추후에 schema 변경이 필요한 경우 최소한의 변경으로 application 에 영향이 크게 가지 않도록 보수적으로 예상해야 할 것이다.