Database Project: Movie Application

오유진(인공지능학과, 2277021)

1) Database Schema

- 해당 데이터베이스는 유저의 영화 예매 및 평점 입력과 관련한 시뮬레이션을 진행해주는 어플리케이션을 설계한 것이다. 주요 기능은 문제에서 주어진 바와 같이 크게 총 13가지로 구성되어 있다. 1)데이터베이스 초기화 2)모든 영화 정보 출력 3)모든 고객 정보 출력 4)영화 추가 5)고객 추가 6)영화 삭제 7)고객 삭제 8)영화 예매 9)평점 부여 10)영화 예매 고객 정보 출력 11)고객 예매 영화 정보 출력 12)프로그램 종료 13)데이터베이스 리셋 및 생성.

- 구성한 스키마는 총 5개의 relation(=table)(director, movie_direction, movie, rating, booking, customer)으로 구성됨:

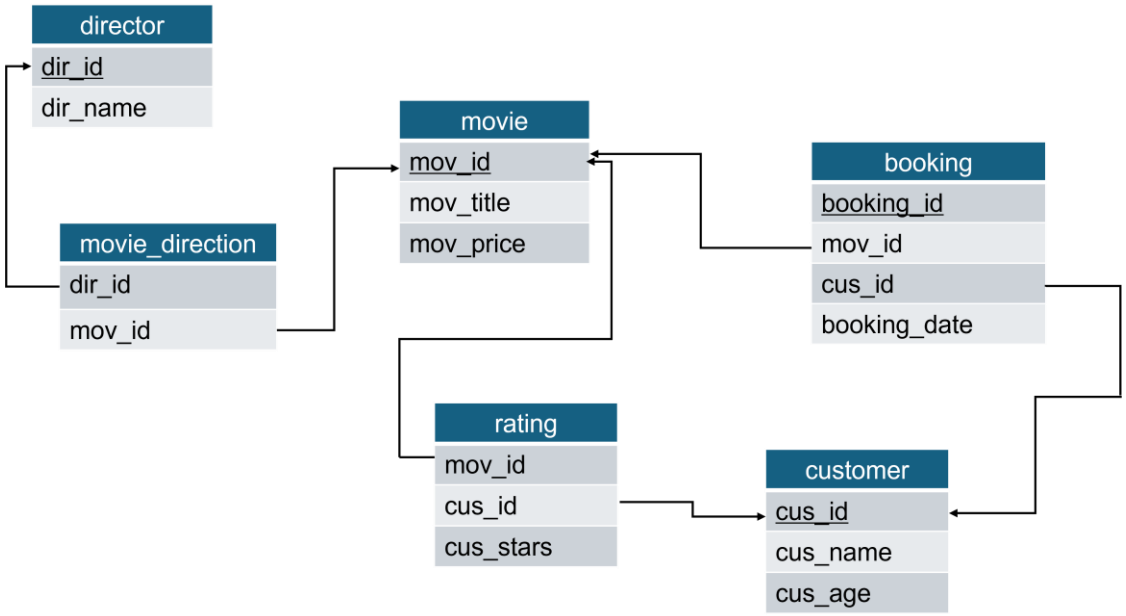


Table 1. director

- Attributes : 감독의 고유 id값, 감독의 이름
- 주어지는 데이터는 감독의 이름이며, 추가 순서에 따라 1부터 차례로 id값을 자동 부여한다.

Table 2. movie

- Attributes : 영화의 고유 id값, 영화의 고유 제목, 영화 가격
- 주어지는 데이터는 영화의 제목과 이에 해당하는 가격이며, 추가되는 순서에 따라 1부터 차례로 id값을 자동 부여한다.

Table 3. movie_direction

- Attributes : 감독의 id, 영화의 id
- 감독의 id는 foreign key로 director table을 참조하며, 영화의 id는 director table을 참조한다. 이때 (감독의 id, 영화의 id)쌍이 primary key로써 사용된다.

Table 4. customer

- Attributes: 고객의 id, 고객의 이름, 고객의 나이
- 주어지는 데이터는 고객의 이름과 나이이며, 추가 순서에 따라 1부터 차례로 id값을 자동 부여한다.

Table 5. booking

- Attributes: 예약 id, 영화의 id, 고객의 id, 예약 시점
- 영화의 id는 foreign key로 movie table을 참조하며, 고객의 id는 customer table을 참조한다. 이때 데이터가 추가되는 순서에 따라 예약 id를 1부터 자동 부여한다. 이때 예약 시점은 booking table에 저장되는 시점이 되도록 하였다.

Table 6. rating

- Attributes: 영화의 id, 고객의 id, 고객의 평점
- id는 foreign key로 movie table을 참조하며, 고객의 id는 customer table을 참조한다.
- Booking이 진행되면 rating table에 해당하는 (mov_id, cus_id)쌍이 insert되며, 이와 함께 고객의 평점은 null으로 초기화되도록 설정하였다.

*연결된 데이터를 한번에 지울 수 있도록 하기 위해, movie_direction, rating, booking table의 foreign key에는 ON DELETE CASCADE라는 foreign key constraints를 부여하였으며 movie_direction의 movie 및 director table과의 연결성을 고려하여 ON UPDATE CASCADE를 추가 부여하였다.

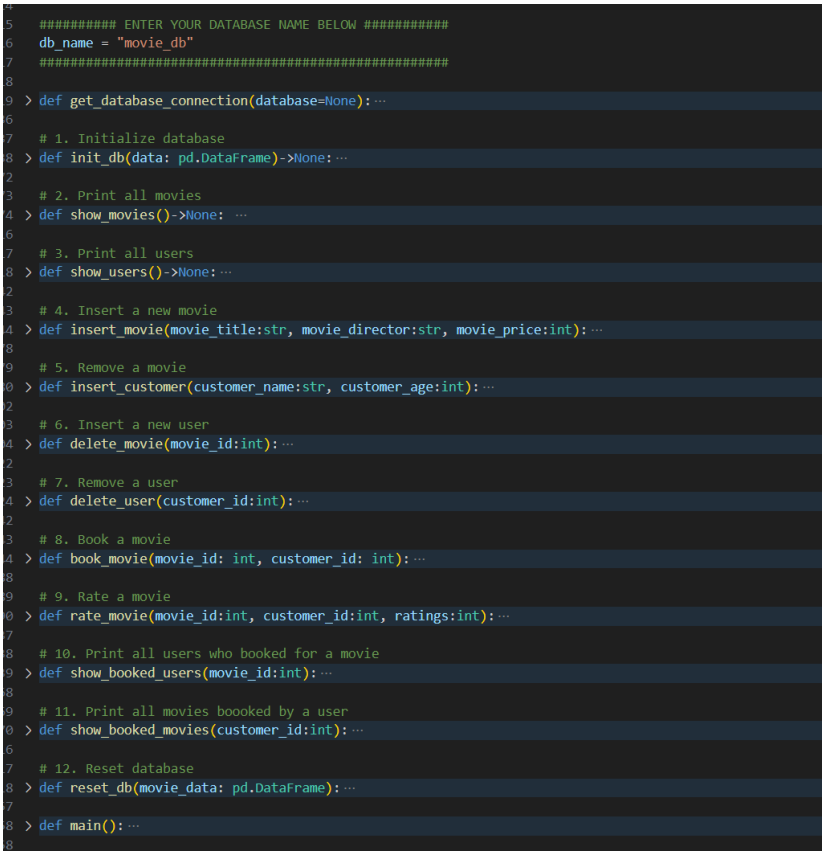
2) 알고리즘 요약 설명

- 핵심 파일은 run.py 및 utils.py에 해당함.

- 사용자가 주어진 매뉴얼에 따라 터미널에 값을 입력한다. 그러면 입력한 번호에 따라 아래에서 해당하는 함수를 호출하게 된다.

a. run.py

- get_database_connection() 함수에서는 본인의 서버에 연결할 수 있도록 하는 connection을 반환한다.



```
########## ENTER YOUR DATABASE NAME BELOW ###########
db_name = "movie_db"
#####################################################

def get_database_connection(database=None):···

# 1. Initialize database
def init_db(data: pd.DataFrame)->None:···

# 2. Print all movies
def show_movies()->None: ···

# 3. Print all users
def show_users()->None:···

# 4. Insert a new movie
def insert_movie(movie_title:str, movie_director:str, movie_price:int):···

# 5. Remove a movie
def insert_customer(customer_name:str, customer_age:int):···

# 6. Insert a new user
def delete_movie(movie_id:int):···

# 7. Remove a user
def delete_user(customer_id:int):···

# 8. Book a movie
def book_movie(movie_id: int, customer_id: int):···

# 9. Rate a movie
def rate_movie(movie_id:int, customer_id:int, ratings:int):···

# 10. Print all users who booked for a movie
def show_booked_users(movie_id:int):···

# 11. Print all movies boooked by a user
def show_booked_movies(customer_id:int):···

# 12. Reset database
def reset_db(movie_data: pd.DataFrame):···

def main():···
```

b. utils.py

- 해당 파일에서는 영화 존재 유무, 고객 존재 유무, 이전 예매 유무, 예매 가능 유무, 평가 가능 유무, 그리고 데이터 베이스 생성 및 데이터 베이스 삭제, 메뉴 출력과 관련한 함수를 정의하였다. Run.py에서 utils가 import되어 해당 함수들이 호출되어 사용된다.

예를 들어, 예매입력 값이 1일 경우 init_db(data)함수가 호출되면서 내부에 table들을 create하고 data.csv를 받아 데이터를 insert한다. 이때 utils.py의 create_table()을 호출함으로써 table이 database에 정상적으로 생성된다.

```
> def get_user_input(prompt, input_type=str): ···
> def show_databases(connection): ···
> def show_ratings(table_name:str)->None: ···
> def create_table(connection, create_table_sql): ···
> def check_movie(movie_id:int)->int: ···
> def check_customer(customer_id:int)->int: ···
> def check_fully_booked(movie_id:int)->int: ···
> def check_reserved(movie_id:int, customer_id:int)->int: ···
> def check_rated(movie_id:int, customer_id:int)->int: ···
> def create_database(connection, db_name): ···
> def drop_database(connection, db_name): ···
  def show_menu()->None:
```

```
connection = get_database_connection(db_name)
create_table(connection, create_customer_table)
create_table(connection, create_director_table)
create_table(connection, create_movie_table)
create_table(connection, create_movie_direction_table)
create_table(connection, create_rating_table)
create_table(connection, create_booking_table)
```

3) 실행 방법

- conda create -n 가상환경이름 python=3.8 을 통해 가상환경 생성(3.8버전)하고 활성화

- pip install -r requirements.txt를 통해 필요한 라이브러리를 설치한다.

- *python run.py 변경*

a. 상단의 주석 박스 내부에서 데이터베이스의 이름을 정할 수 있다.

b. get_database_connection함수 내의 connect(..)에서 본인의 상황에 맞게 host, user, password 등을 변경한다. (해당 실험의 경우 로컬에서 진행함.)

```
########## ENTER YOUR DATABASE NAME BELOW ###########
db_name = "movie_db"
#####################################################

def get_database_connection(database=None):
    """
    Establishes a connection to the MySQL server and returns the connection object.
    If a database name is provided, it connects to that database.
    """
    try:
        connection = connect(
            host='127.0.0.1',
            user='yoojinoh',
            password='cho0312!',
            database=database,
            charset='utf8'
        )
        return connection
    except Error as e:
        print(f"Error: {e}")
        exit()
```

- 위 과정을 모두 완료한 후, command에서 python run.py를 입력해 실행하고, 터미널의 명령에 맞게 입력하면 본격적인 어플리케이션의 실행이 가능하다.

*개발 환경: Visual Studio Code

4) 추가 가정 및 예외처리

- 처음 입력되는 값이 숫자가 아닐 경우 예외문을 출력하도록 하였다.

```
def get_user_input(prompt, input_type=str):
    while True:
        try:
            return input_type(input(prompt))
        except ValueError:
            print(f"Invalid input. Please enter a valid {input_type.__name__}.")
```

- 처음 입력되는 값이 1이 아닐 경우 'You must first initialize the database'가 출력되도록 하였다.

- ratings값이 1이상 5이하가 아닐 경우, Ratings should be from 1 to 5라는 오류를 출력하도록 하였다.

- 문제에서 요했던 'Rating does not exist'는 해당 알고리즘에서 필수적인 메시지가 아니므로 생략하였다.

- 데이터베이스 리셋의 경우(13번), 'y', 'n'이외의 답변 시 'Only y/n answer is available!'이라는 에러 메시지를 출력하도록 하였고, 데이터베이스에 많은 조작을 했을 경우 다시 초기화하는 데에 소요 시간이 길 수 있으므로 스레드를 사용하여 Loading 메시지가 출력될 수 있도록 하였다.

```
    def init_db_with_loading():
        """
        Function to initialize the database in a separate thread
        """
        init_db(movie_data)
        print("Database initialization complete.")

    # Start the database initialization in a separate thread
    init_thread = Thread(target=init_db_with_loading)
    init_thread.start()

    print("Initializing the database, please wait...")
    while init_thread.is_alive():
        for i in range(3):
            print(f"Loading{'.' * (i + 1)}", end='\r')
            time.sleep(1)
    init_thread.join()
```

- 만일 어플리케이션을 시작하고 중간에 재초기화를 시도하고자할 경우,1번이 아니라 13번을 이용하도록 명령하도록 제한했다. (1번은 오직 시작 직후 초기화할때만 사용되도록 하기 위함임.)

```
if action == 1:
    if start:
        init_db(movie_data)
        start = 0
        print()
    else:
        print('If you want to re-initialize the database, please use action 13.')
        print()
```

- 새롭게 예매가 이루어질 경우(booking), 해당하는 movie id와 customer id 정보를 평점은 디폴트값으로(None) 설정한 채로 rating table에도 삽입할 수 있도록 하였다.

- 사용자가 설정한 db_name을 기반으로 데이터 베이스를 성공적으로 생성할 경우 메시지가 출력되도록 하였으며, 기존에 동일 이름의 데이터 베이스가 존재할 경우 이를 drop한 후에 생성하게 된다.

```
def create_database(connection, db_name):
    """
    Creates a new database with the given name.
    """
    try:
        with connection.cursor() as cursor:
            cursor.execute(f"CREATE DATABASE {db_name}")
            print(f"Database {db_name} created successfully.")
    except Error as e:
        print(f"Error creating database {db_name}: {e}")
        exit()
```

```
def drop_database(connection, db_name):
    """
    Drops the database with the given name if it exists
    """
    try:
        with connection.cursor() as cursor:
            # Check if the database exists
            cursor.execute("SHOW DATABASES LIKE %s", (db_name,))
            result = cursor.fetchone()

            if result:
                # Drop the database if it exists
                cursor.execute(f"DROP DATABASE {db_name}")
                print(f"Database {db_name} dropped successfully.")

    except Error as e:
        print(f"Error dropping database {db_name}: {e}")
        exit()
```