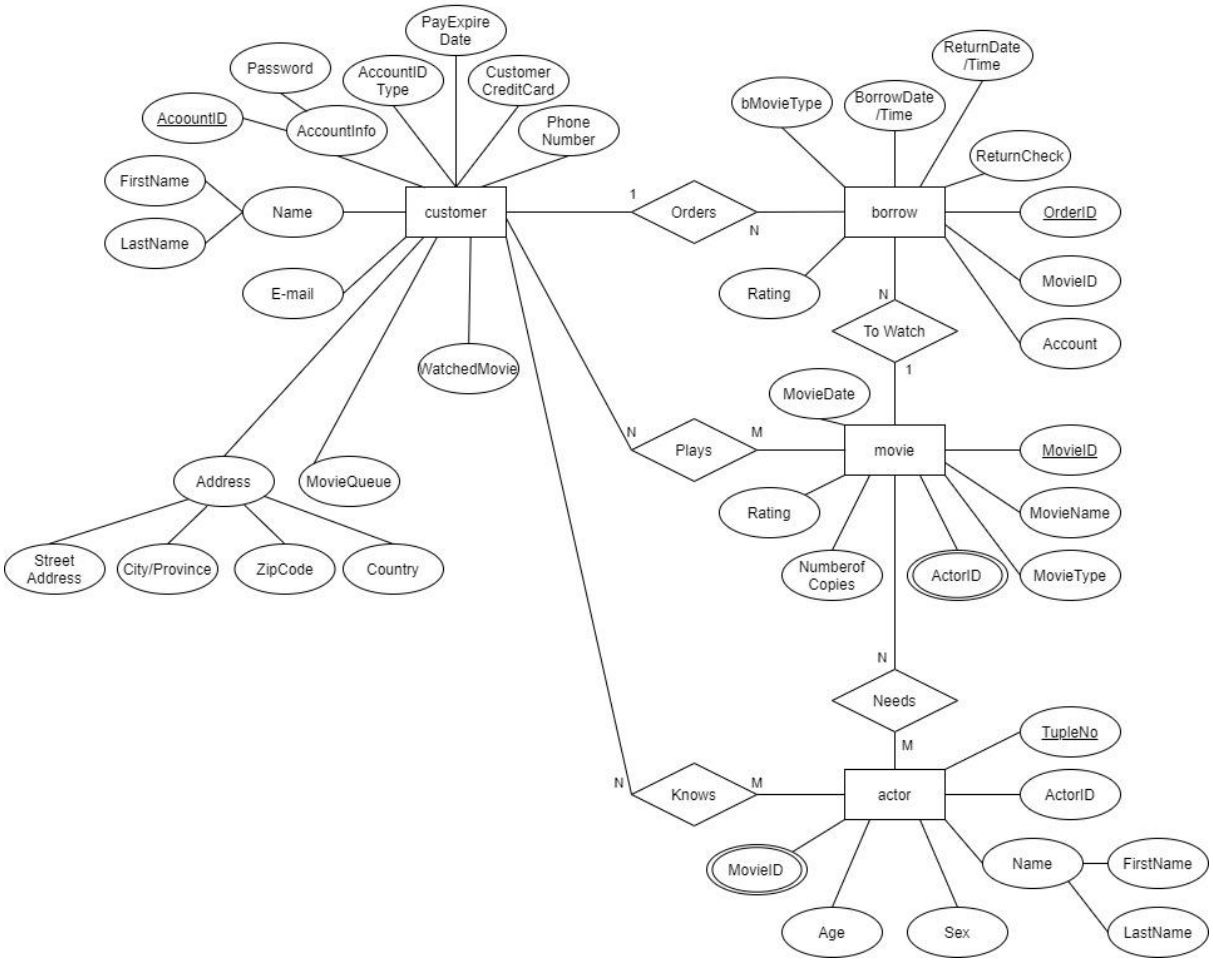
DB Project Part 3 Assignment

201620931 소프트웨어학과 김영우

Watch Network



이번 3차, 그리고 FLASK을 이용한 웹 프로그래밍에서 백엔드로 사용한 ER 다이어그램입니다. 이전 2차 ER 다이어그램과 비교하여 거의 바뀐 것은 없었습니다. 제 ER 다이어그램에는 총 4개의 테이블이 있습니다. 따라서 저의 프로그램에도 총 4개의 데이터베이스 테이블을 가지고 있습니다. 각각 테이블의 이름은 customer, borrow, movie, actor입니다. Customer는 사용자, 유저의 정보를 가지고 있는 테이블입니다. Borrow는 영화를 빌려간 내역, 정보를 저장하고 있습니다. Movie는 영화의 정보를 가지고 있고, actor는 영화에 출연한 배우들의 정보를 가지고 있습니다. Customer의 Primary Key는 AccountID입니다. 이 ID가 유저를 대표할 수 있는 가장 대표적인 속성값이라고 생각했습니다. Borrow의 Primary Key는 OrderID입니다. OrderID를 보면 모든 속성을 구분해내는데 가장 적합합니다. Movie의 Primary Key는 MovieID, Actor의 Primary Key는 TupleNo입니다. Actor ID를 저장하는데 중복되게 저장을 할 수 밖에 없었습니다. Actor는 정확히 어떻게 보면 출연한 영화를 기준으로 튜플을 저장하는데, 배우가 하나의 영화에만 촬영하는 것은 아니기 때문에, 어쩔 수

없이 TupleNo로 값들을 구분해 주었습니다.

다음은 최종 SQL statements와 해당 statement의 결과입니다.

```
INSERT INTO customer(FirstName, LastName, AccountID, Password,
StreetAddress, City_Province, ZipCode, Country, Email, PhoneNumber)
VALUES(%s, %s, %s, %s, %s, %s, %s, %s, %s, %s)", (firstname, lastname,
accountid, password, streetaddress, city_province, zipcode, country, email,
phonenumber)
```

해당 SQL statement는 Customer에 Tuple을 추가하는 구문입니다. INSERT INTO를 이용해 튜플을 추가하고, customer테이블에 값을 추가합니다. 뒤에 따라오는 FirstName, LastName, AccountID...는 customer 테이블의 attribute값입니다. values에는 각각 해당하는 값이 들어갑니다.

해당 구문을 FLASK에서 입력하면, Customer에 Tuple이 생성됩니다. 즉 이 구문을 통해 유저를 추가할 수 있습니다.

```
ysql> SELECT * FROM customer;
+-----------+----------+-------------+----------+-------------------+-----------------+---------+---------+
| FirstName | LastName | AccountID   | Password | StreetAddress     | City_Province   | ZipCode | Country |
+-----------+----------+-------------+----------+-------------------+-----------------+---------+---------+
| Shang     | Yang     | 111-11-1111 | 1111     | 123 Success Street | New York_NY     |   11790 | USA     |
| Victor    | Du       | 222-22-2222 | 2222     | 456 Fortune Road  | West Lafayette_IN |   11790 | USA     |
| John      | Smith    | 333-33-3333 | 3333     | 789 Peace Blvd.   | Los Angeles_CA  |   93536 | USA     |
| Lewis     | Phillip  | 444-44-4444 | 4444     | 135 Knowledge Lane | Stony Brook_NY  |   11794 | USA     |
| Young     | Kim      | 555-55-5555 | 5555     | East Street       | Seoul           |    8090 | KOREA   |
+-----------+----------+-------------+----------+-------------------+-----------------+---------+---------+
5 rows in set (0.00 sec)
```

```
+---------------+---------------+--------------------+---------------------+-------------+------------+--------------+
| AccountIDType | PayExpireDate | CustomerCreditCard | Email               | PhoneNumber | MovieQueue | WatchedMovie |
+---------------+---------------+--------------------+---------------------+-------------+------------+--------------+
| L             | 2020-12-17    | 1234-5678-1234-5678 | syang@ajou.ac.kr    | 010-1111-1111 |          1 |           23 |
| U             | 2020-12-15    | 5678-1234-5678-1234 | vicdu@ajou.ac.kr    | 010-2222-2222 |          2 |           10 |
| U             | 2020-12-04    | 2345-6789-2345-6789 | jsmith@ajou.ac.kr   | 010-3333-3333 |          0 |          230 |
| L             | 2020-12-31    | 6789-2345-6789-2345 | pml@ajou.ac.kr      | 010-4444-4444 |          0 |           35 |
| NULL          | NULL          | NULL                | skysaver00@ajou.ac.kr | 010-555-5555 |          0 |            0 |
+---------------+---------------+--------------------+---------------------+-------------+------------+--------------+
```

이렇게 customer에 새로운 Tuple인 Name Attribute Young Kim이 들어가게 되는 것을 확인할 수 있습니다.

다음 SQL statement는 customer의 account setting을 확인하는 구문입니다.

```
"SELECT * FROM customer WHERE AccountID = %s AND Password = %s", [name,
pas]
```

다음 구문을 이용하면 사용자는 자신이 기억하는 ID와 비밀번호를 입력해야만 합니다. WHERE AccountID = %s AND Password = %s를 통해 이를 제한해 줄 수 있습니다. 이렇게 해서 개인정보를 보호할 수 있습니다. 해당 statement가 실행이 되면,

Profile

Welcome! ('Shang', 'Yang')

FirstName	LastName	AccountID	Password	StreetAddress	City, Province	ZipCode	Country	Account Type	PayExpireDate	Credit Card	Email	Phone Number	Movie Queue	Watched Movie
Shang	Yang	111-11-1111	1111	123 Success Street	New York_NY	11790	USA	L	2020-12-17	1234-5678-1234-5678	syang@ajou.ac.kr	010-1111-1111	1	23

다음 사진은 Flask을 사용하여 만든 웹 서비스에서 적절한 ID와 비밀번호를 입력했을 때 나오는 값입니다. 사진의 테이블을 보면, 유저의 개인정보가 쭉 나와있는 것을 볼 수 있습니다.

다음 SQL statement는 유저가 현재 가지고 있는 영화, 즉 빌린 영화를 가지고 오는 구문입니다.

```
SELECT DISTINCT b.MovieID, MovieName, Account FROM customer, borrow as b,
Movie as m WHERE b.MovieID = m.MovieID AND Account = %s", [accountid]
```

빌린 영화의 ID와 영화 테이블의 ID가 서로 같은 것들을 확인하고, Account = %s로 해당하는 값이 맞는 Account를 입력하면 b.MovieID, MovieName, Account를 가지고 옵니다. 이때, 튜플들이 서로 섞여서 가지고 오기 때문에 DISTINCT를 통해 이를 제한해 줄 수 있습니다.


Held Movie Results

Movie ID	Movie Name	Account ID
1	The Godfather	111-11-1111
3	Borat	111-11-1111

Main Page

위의 Statement가 실행되면 적합한 값들이 출력되는 것을 확인할 수 있습니다.

```
"SELECT DISTINCT MovieQueue, FirstName, LastName, AccountID FROM customer
WHERE MovieQueue >= 1 AND AccountID = %s", [accountid]
```

해당 구문은 영화의 Queue를 가지고 오는 구문입니다. DISTINCT를 지정해주고, MovieQueue, FirstName, LastName, AccountID를 가져와서 테이블을 이해할 수 있는 값들을 가져옵니다. 또한, Queue에 값이 0개이면 가지고 올 이유가 없기 때문에, MovieQueue >= 1으로, 그리고 AccountID가 같은 값들을 가지고 옵니다.


Movie Queue Results

Movie Queue	Fist Name	Last Name	Account ID
1	Shang	Yang	111-11-1111

Main Page

위의 구문을 실행하면 다음과 같은 결과가 나옵니다.

```
"SELECT MovieID, MovieName, MovieType FROM Movie WHERE MovieType = %s",
[movietype]
```

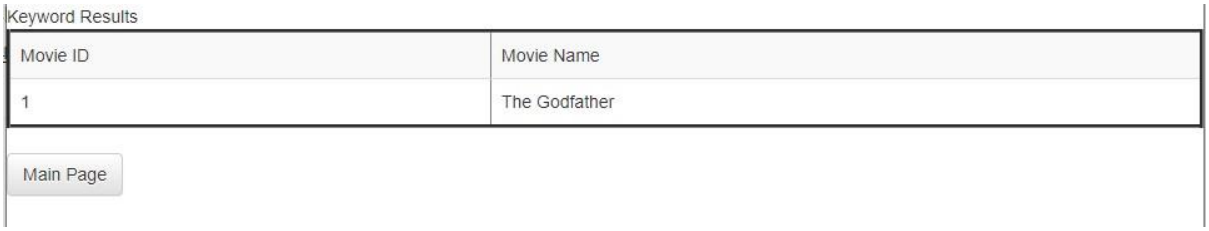
이 구문은 Drama와 같은 Movie Type을 입력했을 때, Drama의 장르에 속해 있는 튜플들을 가지고 옵니다. WHERE에서 MovieType = %s로 %s에는 입력한 Movie Type이 들어갑니다.

결과는 다음과 같습니다. Movie Type을 확인할 수 있는 속성값을 출력합니다.

```
"SELECT MovieID, MovieName FROM movie WHERE MovieName LIKE %s", [keyword]
```

특정 단어를 포함하는 영화를 가지고 오는 구문입니다.

LIKE %s로 keyword 변수에 들어있는 단어가 포함된 영화들을 가지고 옵니다. Keyword는 사용자가 입력할 수 있습니다.



이와 같이 영화 ID, 영화 이름이 포함된 결과값이 테이블로 나오게 됩니다.

다음은 영화의 특정 배우의 이름으로 영화를 찾을 수 있는 SQL 구문입니다.

```
"SELECT DISTINCT m.MovieID, m.MovieName, a.ActorID, a.FirstName, a.LastName
FROM movie AS m, actor AS a WHERE m.ActorID = a.ActorID AND a.FirstName
= %s AND a.LastName = %s", [firstname, lastname])
```

이 구문은 영화의 이름, 영화 ID, 배우 ID, 배우 이름을 가지고 옵니다. 구분하는 조건은 영화에 출연한 배우의 ID를 먼저 비교하고, 배우의 이름을 사용자가 입력한 값으로부터 가지고 온다음 SQL statement를 실행합니다.

Actor Results

Movie ID	Movie Name	Actor ID	First Name	Last Name
1	The Godfather	1	Al	Pacino
3	Borat	1	Al	Pacino

Main Page

결과는 다음과 같습니다. 사용자는 First Name에 al, Last Name에 pacino를 입력했습니다. 이렇게 하면 알파치노가 출연한 영화가 출력됩니다.

다음에는 Recommendation에 해당하는 SQL 구문입니다. 여기서는 extra credit에 해당하는 SQL statement의 결과를 만들어주는 값들을 넣어두었습니다.

베스트셀러를 출력하는 SQL statement는 다음과 같습니다.

```
"SELECT MAX(NumberofCopies), MovieName FROM movie"
```

이를 출력하면 NumberofCopies의 최대값을 선택해서 가져옵니다.

Best Seller! We Recommend Copies: (3, 'The Godfather')

현재 제일 많이 빌려간 영화는 The Godfather인데, 총 3번 빌려갔다는 것을 알 수 있습니다.

다음은 Personalized Movie Suggestion List를 출력하는 SQL 구문입니다. 저는 이 뜻을 영화를 보고 아직 보지 못한 영화들 중에서 장르가 같은 영화를 출력하는 것이 가장 올바르다고 생각했습니다. 이렇게 해서 값을 출력했습니다.

```
"SELECT DISTINCT c.FirstName, c.LastName, m.MovieName, m.MovieType FROM
customer AS c, borrow AS b, movie AS m WHERE b.MovieID <> m.MovieID AND
c.AccountID = b.Account AND b.bMovieType = m.MovieType AND c.AccountID
= %s", [accountid]
```

이 SQL statement는 앞서 말한 값을 출력하는 구문입니다. 만약에 여기에 사용자의 ID를 넣으면

Movie Suggestion

First Name	Last Name	Movie Name	Movie Type
Shang	Yang	Shawshank Redemption	Drama

Main Page

이렇게 값이 출력이 됩니다.

마지막으로 Rating을 넣는 방법입니다.

```
"UPDATE borrow SET Rating = %s WHERE Account = %s AND MovieID = %s",
[rating, accountid, movieid]
```

이것을 넣으면 Rating을 넣을 수 있습니다. 사용자가 입력해야 하는 값은 총 3개로 사용자의 ID, 영화 ID, 그리고 점수입니다. 점수는 라디오버튼으로 구현이 되어있어 해당하는 값에 클릭해주고 제출하면 값이 들어갑니다.

Rating Successfully done!

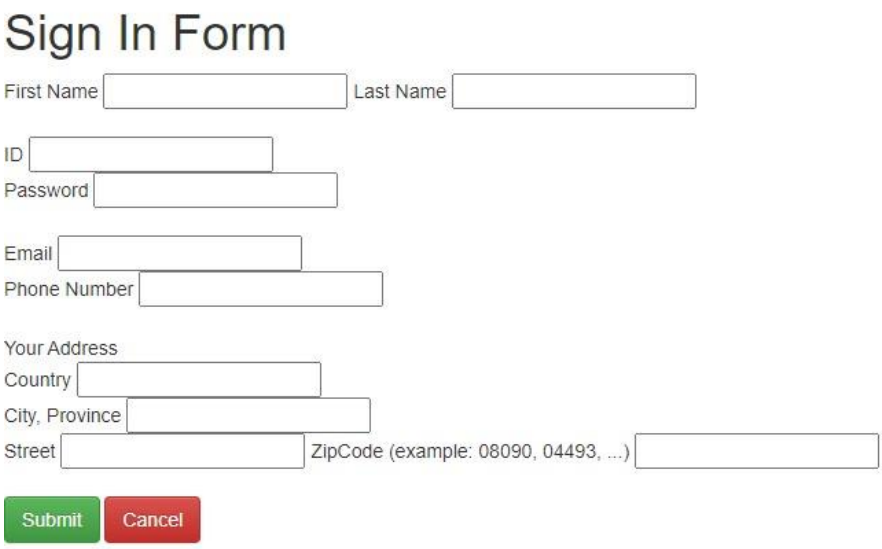
Order ID	Movie ID	Account	Movie Type	Borrowed Date	Returned Date	Return Check	Rating
1	1	111-11-1111	Drama	2011-09-11 10:00:00	2011-09-14 00:00:00	T	0
3	3	111-11-1111	Comedy	2011-09-12 09:30:00	None	N	5

Main Page

다음은 Rating을 주었을 때 나오는 값입니다. 오른쪽 Rating에 5점을 준 것을 확인할 수 있습니다.

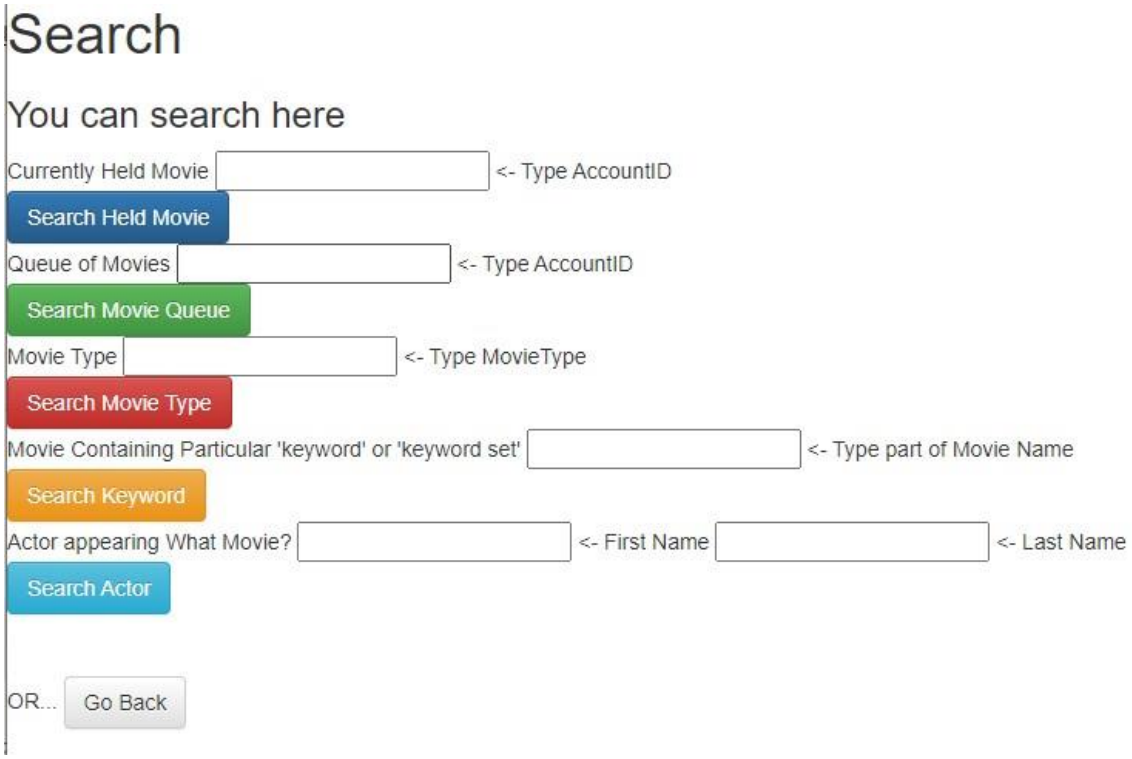
마지막으로 유저 매뉴얼을 설명하도록 하겠습니다. 맨 처음 들어갈 때, http://127.0.0.1:5000/main 으로 가주어야 합니다. '/'에 해당하는 사이트가 메인인데, 이 프로젝트를 만들 때 어떤 사이트가 가장 메인으로 들어갈지 아직 정해지지 않아서 공백으로 만들었습니다. 만일 위의 사이트로 들어가신다면, 다음과 같은 화면이 나옵니다.

이 사이트가 현재 가장 메인이 되는 사이트입니다. ID, 비밀번호를 입력해 프로파일(개인정보)을 확인하거나 오른쪽의 Sign In을 눌러 회원 가입을 할 수 있습니다.

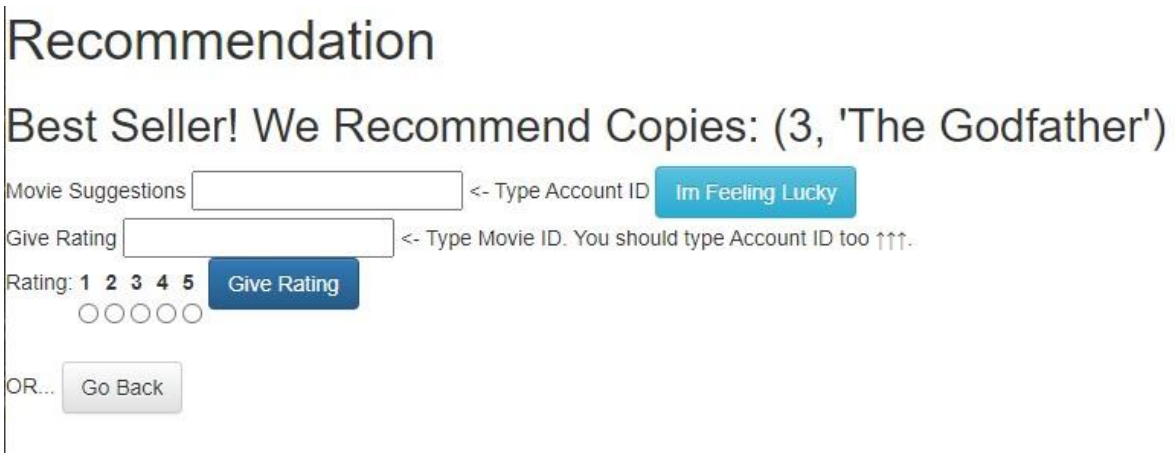


이 사이트는 하늘색 Sign In을 눌렀을 때 나오는 회원가입입니다. 여기에 올바른 값들을 입력하면 Check Profile에서 값을 확인할 수 있습니다.

초록색 버튼 Submit를 누르면 값이 데이터베이스에 저장되고, 만일 Cancel을 누르면 이전으로 돌아갑니다.



다음은 밑의 Search 버튼을 눌렀을 때 나오는 화면입니다. 여기서는 값을 각각 넣어서 지시사항의 결과를 확인할 수 있습니다.



마지막으로 Recommendation에 해당하는 화면입니다. 여기서는 Best Seller, Movie Suggestion, Rating을 확인할 수 있습니다. Recommendation 바로 밑에 Best Seller! 해서 Godfather가 베스트 셀러라는 것을 확인할 수 있습니다.

특이사항으로 Rating을 줄 때는 Account Id, MovieID, 그리고 라디오 버튼을 모두 지정해야 합니다. 이 3개를 모두 값을 부여할 경우, Rating을 줄 수 있습니다.

마지막으로 메인화면에서 Bye Bye를 누르면 www.google.com 으로 링크를 지정해두었습니다.