

2023-1 데이터베이스 기말과제

영화 정보 DB 구축 및 검색

소프트웨어학과

201711357 최병찬

목차

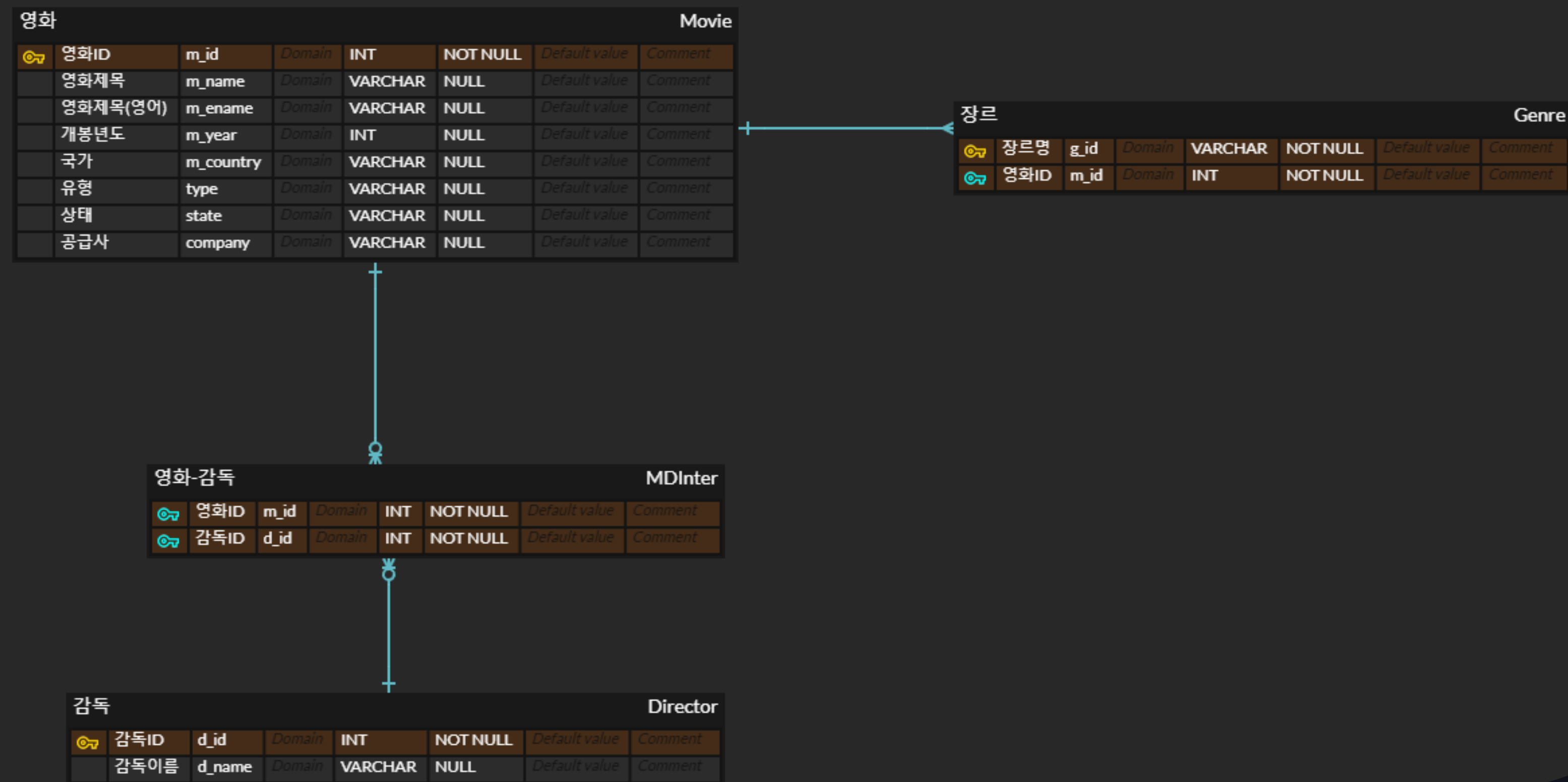
ER DIAGRAM

PYTHON CODE

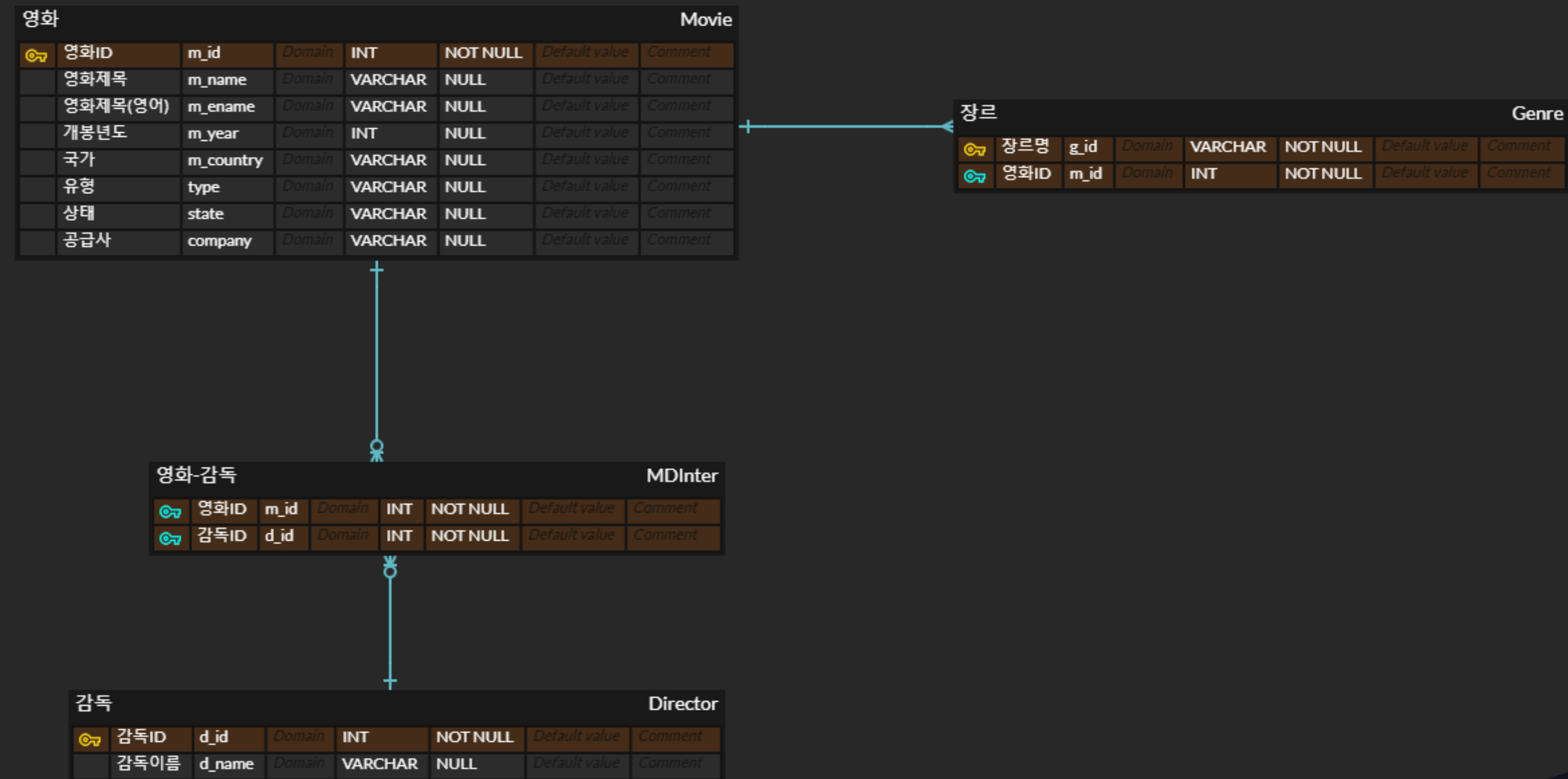
쿼리문 Workbench 결과

프로그램 시연

ERD



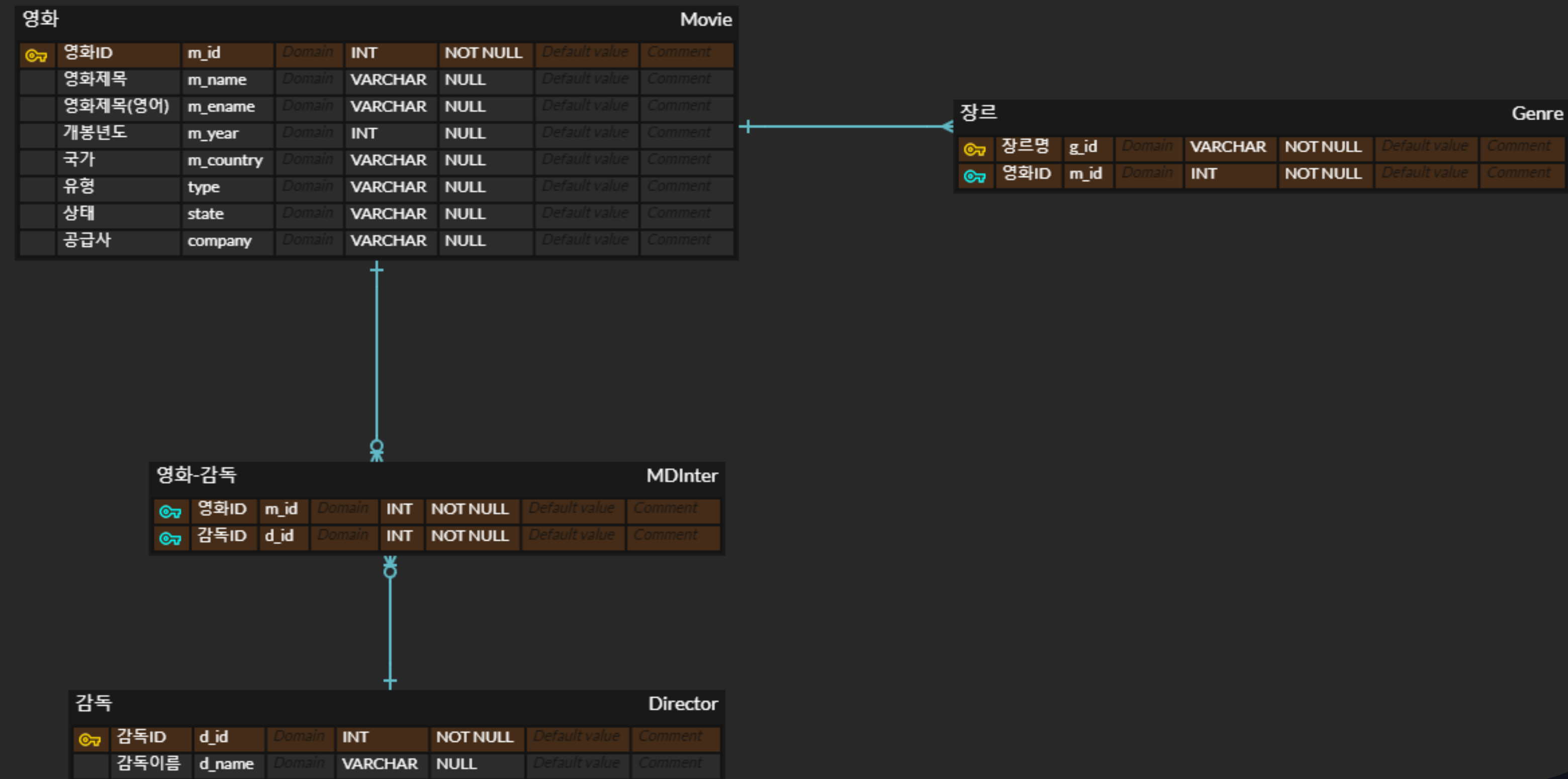
ERD



Movie - Direction ; M:N Strong Relationship

→ Intersection Table 생성

ERD



Movie - Genre ; multi-valued attribute (1:N)

→ Genre Table 생성 (ID-Dependent)

PYTHON CODE : 테이블 생성

```
def create_table()
```

총 4개의 테이블 생성

Movie , Director , Genre , mdInter

Movie , Director : PK로 ID 생성

Genre : 식별자로 MovieID와 GenreName

mdInter : 식별자로 MovieID와 DirectorID

PYTHON CODE : 데이터 삽입

```
def xls_to_table()
```

pandas module : excel 데이터 추출

예외처리 : 중간테이블 중복 값

동일한 이름의 감독은 동일한 ID로 설정

PYTHON CODE : 데이터 삽입

예외처리 : 중간테이블 중복 값

```
sql = f"Select * from mdinter where m_id = {m_id} and d_id = {d_id}"
cursor.execute(sql)

if cursor.fetchone() is None :
    sql = "INSERT INTO mdinter (m_id,d_id) values (%s,%s)"
    val =(m_id,d_id)
    cursor.execute(sql,val)
else :
    print(m_name , d_name , m_id , d_id)
```


PYTHON CODE : 데이터 삽입

동일한 이름의 감독은 동일한 ID로 설정

```
sql = "SELECT d_id from director where d_name = %s"  
val = (d_name,)   
cursor.execute(sql,val)   
res = cursor.fetchone()
```

```
#감독 아이디가 없는 경우
```

```
if res is None :
```

```
# 감독 아이디가 이미 존재하는 경우
```

```
else :
```

```
    sql = f"Select * from mdinter where m_id = {m_id} and d_id = {res['d_id']}"  
    cursor.execute(sql)
```

PYTHON CODE : 인덱스 설정

```
def create_index()
```

검색에 자주 쓰이는 어트리뷰트에 대한 인덱스 생성

영화 , 감독 , 장르명

PYTHON CODE : 인덱스 설정

검색에 자주 쓰이는 어트리뷰트에 대한 인덱스 생성

```
def create_idx() :  
    conn ,cursor = sql_connect()  
  
    cursor.execute("CREATE INDEX idx_m_name on movie (m_name)")  
    cursor.execute("CREATE INDEX idx_d_name on director (d_name)")  
    cursor.execute("CREATE INDEX idx_g_name on genre (g_name)")  
  
    commit_n_close(conn)
```

Workbench 결과

1. 2020년에 제작된 다큐멘터리 한국 영화의 영화명과 감독을 영화명 순으로 검색하라

```
3 • select m.m_name , d.d_name
4 from movie m ,director d ,mdinter md , genre g
5 where m.m_id = md.m_id and g.m_id = m.m_id
6 and d.d_id = md.d_id and g.g_name = "다큐멘터리"
7 and m.m_country = "한국" and m.m_year = 2020
8 order by m_name
```

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

Result Grid

m_name	d_name
100년 도시재생 vs 5년 도시재생	정용택
1980사북	박봉남
1984 최동원	조은성
1포 10kg 100개의 생애	조기현
2020년의 봄	차경미
40	양주연
CO-OP	김정호

Output

#	Time	Action	Message	Duration / Fetch
7	22:27:46	EXPLAIN select m.m_name , d.d_name from movie m ,director d ,m...	OK	0.000 sec
8	22:27:46	EXPLAIN FORMAT=JSON select m.m_name , d.d_name from movi...	OK	0.000 sec
9	22:27:56	select m.m_name , d.d_name from movie m ,director d ,mdinter md ,...	2000 row(s) returned	0.078 sec / 0.000 sec
10	22:28:39	EXPLAIN select m.m_name , d.d_name from movie m ,director d ,m...	OK	0.000 sec
11	22:28:39	EXPLAIN FORMAT=JSON select m.m_name , d.d_name from movi...	OK	0.000 sec
12	22:28:48	select m.m_name , d.d_name from movie m ,director d ,mdinter md ,...	127 row(s) returned	0.016 sec / 0.000 sec
13	22:30:39	select m.m_name , d.d_name from movie m ,director d ,mdinter md ,...	127 row(s) returned	0.016 sec / 0.000 sec

Workbench 결과

2. '봉준호' 감독의 영화를 제작년도 순으로 검색하라

The screenshot displays the MySQL Workbench interface. The SQL Editor at the top contains the following query:

```
2
3 • select m.m_name , d.d_name
4 from movie m ,director d ,mdinter md , genre g
5 where m.m_id = md.m_id and g.m_id = m.m_id and d.d_id = md.d_id
6 and d.d_name = "봉준호"
7 order by m.m_year
```

Below the editor, the 'Result Grid' shows the results of the query. The columns are 'm_name' and 'd_name'. The results are as follows:

m_name	d_name
프레임 속의 기억들	봉준호
지리멸렬	봉준호
백색인	봉준호
플란다스의 개	봉준호
플란다스의 개	봉준호
싱크 & 라이즈	봉준호
사이이 초여	봉준호

At the bottom, the 'Output' tab shows the 'Action Output' log:

#	Time	Action	Message	Duration / Fetch
✓ 10	22:28:39	EXPLAIN select m.m_name , d.d_name from movie m ,director d ,m...	OK	0.000 sec
✓ 11	22:28:39	EXPLAIN FORMAT=JSON select m.m_name , d.d_name from movi...	OK	0.000 sec
✓ 12	22:28:48	select m.m_name , d.d_name from movie m ,director d ,mdinter md ,...	127 row(s) returned	0.016 sec / 0.000 sec
✓ 13	22:30:39	select m.m_name , d.d_name from movie m ,director d ,mdinter md ,...	127 row(s) returned	0.016 sec / 0.000 sec
✓ 14	22:35:11	EXPLAIN select m.m_name , d.d_name from movie m ,director d ,m...	OK	0.000 sec
✓ 15	22:35:11	EXPLAIN FORMAT=JSON select m.m_name , d.d_name from movi...	OK	0.000 sec
✓ 16	22:35:18	select m.m_name , d.d_name from movie m ,director d ,mdinter md ,...	20 row(s) returned	0.047 sec / 0.000 sec

Workbench 결과

3. 각 연도별 제작된 영화의 편수를 검색하되, 연도별로 출력하라

The screenshot displays the MySQL Workbench interface. The top pane shows a SQL query in the 'SQL File 5*' editor:

```
1 • use movie_info;
2
3 • select m.m_year , count(*) as m_num
4   from movie m
5  group by m.m_year
6
```

The bottom pane shows the 'Result Grid' with the following data:

m_year	m_num
2022	3655
2023	466
2021	4521
2013	3186
2012	1947
2018	4962
2020	5500

The bottom right pane shows the 'Output' tab with the 'Action Output' table:

#	Time	Action	Message	Duration / Fetch
17	22:37:58	EXPLAIN select m.m_name , count(*) as m_num from movie m gro...	Error Code: 1055 Expression #1 of SELECT list is not in GROUP B...	
18	22:38:38	EXPLAIN select count(*) as m_num from movie m group by m.m_y...	OK	0.000 sec
19	22:38:38	EXPLAIN FORMAT=JSON select count(*) as m_num from movie m ...	OK	0.000 sec
20	22:38:45	select count(*) as m_num from movie m group by m.m_year LIMIT ...	120 row(s) returned	0.031 sec / 0.000 sec
21	22:39:05	EXPLAIN select m.m_year , count(*) as m_num from movie m grou...	OK	0.000 sec
22	22:39:05	EXPLAIN FORMAT=JSON select m.m_year , count(*) as m_num fr...	OK	0.000 sec
23	22:39:09	select m.m_year , count(*) as m_num from movie m group by m.m_...	120 row(s) returned	0.032 sec / 0.000 sec

Workbench 결과

4. 각 나라별 가장 많이 제작된 영화 장르를 검색하라

```
SELECT m_country, g_name, COUNT(*) AS genre_count
FROM movie
JOIN genre ON movie.m_id = genre.m_id
GROUP BY m_country, g_name
HAVING genre_count = (
    SELECT MAX(cnt)
    FROM (
        SELECT m_country, COUNT(*) AS cnt
        FROM movie
        JOIN genre ON movie.m_id = genre.m_id
        GROUP BY m_country, g_name
    ) AS counts
    WHERE counts.m_country = movie.m_country
)
order by m_country desc
```

Workbench 결과

4. 각 나라별 가장 많이 제작된 영화 장르를 검색하라

The screenshot displays the MySQL Workbench interface. The top pane shows a SQL query for finding the most common genre per country. The middle pane shows the 'Result Grid' with 11 rows of data. The bottom pane shows the 'Output' window with a log of database actions.

SQL Query:

```
3 • SELECT m_country, g_name, COUNT(*) AS genre_count
4 FROM movie
5 JOIN genre ON movie.m_id = genre.m_id
6 GROUP BY m_country, g_name
7 HAVING genre_count = (
8     SELECT MAX(cnt)
9     FROM (
```

Result Grid:

m_country	g_name	genre_count
한국, 네덜란드, ...	멜로/로맨스	1
한국, 네덜란드	기타	1
한국, 네덜란드	다큐멘터리	1
한국, 기타	기타	5
한국	드라마	9007
필리핀, 한국	다큐멘터리	1
필리핀, 한국	드라마	1

Output Window:

#	Time	Action	Message	Duration / Fetch
✓ 26	22:48:49	SELECT m_country, g_name, COUNT(*) AS genre_count FROM m...	2000 row(s) returned	3.454 sec / 0.000 sec
✓ 27	22:50:44	SELECT m_country, g_name, COUNT(*) AS genre_count FROM m...	2000 row(s) returned	3.469 sec / 0.000 sec
✓ 28	22:51:38	SELECT m_country, g_name, COUNT(*) AS genre_count FROM m...	2000 row(s) returned	3.360 sec / 0.000 sec
✓ 29	22:53:31	EXPLAIN SELECT m_country, g_name, COUNT(*) AS genre_coun...	OK	0.000 sec
✓ 30	22:53:31	EXPLAIN FORMAT=JSON SELECT m_country, g_name, COUNT(...	OK	0.000 sec
✓ 31	22:54:46	SELECT m_country, g_name, COUNT(*) AS genre_count FROM m...	2000 row(s) returned	3.281 sec / 0.000 sec
✓ 32	22:55:13	SELECT m_country, g_name, COUNT(*) AS genre_count FROM m...	2000 row(s) returned	3.360 sec / 0.000 sec

Workbench 결과

5. 한국 일본 중국 세나라에 대하여 각 나라별 영화를 가장 많이 감독한 감독의 이름을 검색하라

```
SELECT m_country, d_name, COUNT(*) AS movie_count
FROM movie
JOIN mdinter ON movie.m_id = mdinter.m_id
JOIN director ON mdinter.d_id = director.d_id
WHERE m_country IN ('한국', '일본', '중국')
GROUP BY m_country, d_name
HAVING movie_count = (
    SELECT MAX(cnt)
    FROM (
        SELECT m_country, COUNT(*) AS cnt
        FROM movie
        JOIN mdinter ON movie.m_id = mdinter.m_id
        JOIN director ON mdinter.d_id = director.d_id
        WHERE m_country IN ('한국', '일본', '중국')
        GROUP BY m_country, d_name
    ) AS counts
    WHERE counts.m_country = movie.m_country
)
```

Workbench 결과

5. 한국 일본 중국 세나라에 대하여 각 나라별 영화를 가장 많이 감독한 감독의 이름을 검색하라

The screenshot displays the Oracle SQL Developer interface. The top pane shows a SQL query that counts movies by director and country. The bottom pane shows the results of the query in a grid format.

SQL Query:

```
3 • SELECT m_country, d_name, COUNT(*) AS movie_count
4 FROM movie
5 JOIN mdinter ON movie.m_id = mdinter.m_id
6 JOIN director ON mdinter.d_id = director.d_id
7 WHERE m_country IN ('한국', '일본', '중국')
8 GROUP BY m_country, d_name
9 HAVING movie_count = (
```

Result Grid:

m_country	d_name	movie_count
일본	코지마 유키오	382
한국	최성은	97
중국	류신의	24

Output Log:

#	Time	Action	Message	Duration / Fetch
29	22:53:31	EXPLAIN SELECT m_country, g_name, COUNT(*) AS genre_cou...	OK	0.000 sec
30	22:53:31	EXPLAIN FORMAT=JSON SELECT m_country, g_name, COUNT(...	OK	0.000 sec
31	22:54:46	SELECT m_country, g_name, COUNT(*) AS genre_count FROM ...	2000 row(s) returned	3.281 sec / 0.000 sec
32	22:55:13	SELECT m_country, g_name, COUNT(*) AS genre_count FROM ...	2000 row(s) returned	3.360 sec / 0.000 sec
33	23:01:28	EXPLAIN SELECT m_country, d_name, COUNT(*) AS movie_cou...	OK	0.000 sec
34	23:01:28	EXPLAIN FORMAT=JSON SELECT m_country, d_name, COUNT(...	OK	0.000 sec
35	23:01:34	SELECT m_country, d_name, COUNT(*) AS movie_count FROM ...	3 row(s) returned	15.062 sec / 0.000 sec

프로그램 시연

사용자로부터 어트리뷰트 입력받고
해당 어트리뷰트에 대해 AND 연산 수행 후
검색결과 출력