M1_01_relational_database

July 20, 2022

[53]:
```
#Data Viz
import matplotlib.pyplot as plt
import matplotlib.image as img

#Data Manipulation
import pandas as pd

#Data Base
import mysql.connector
```

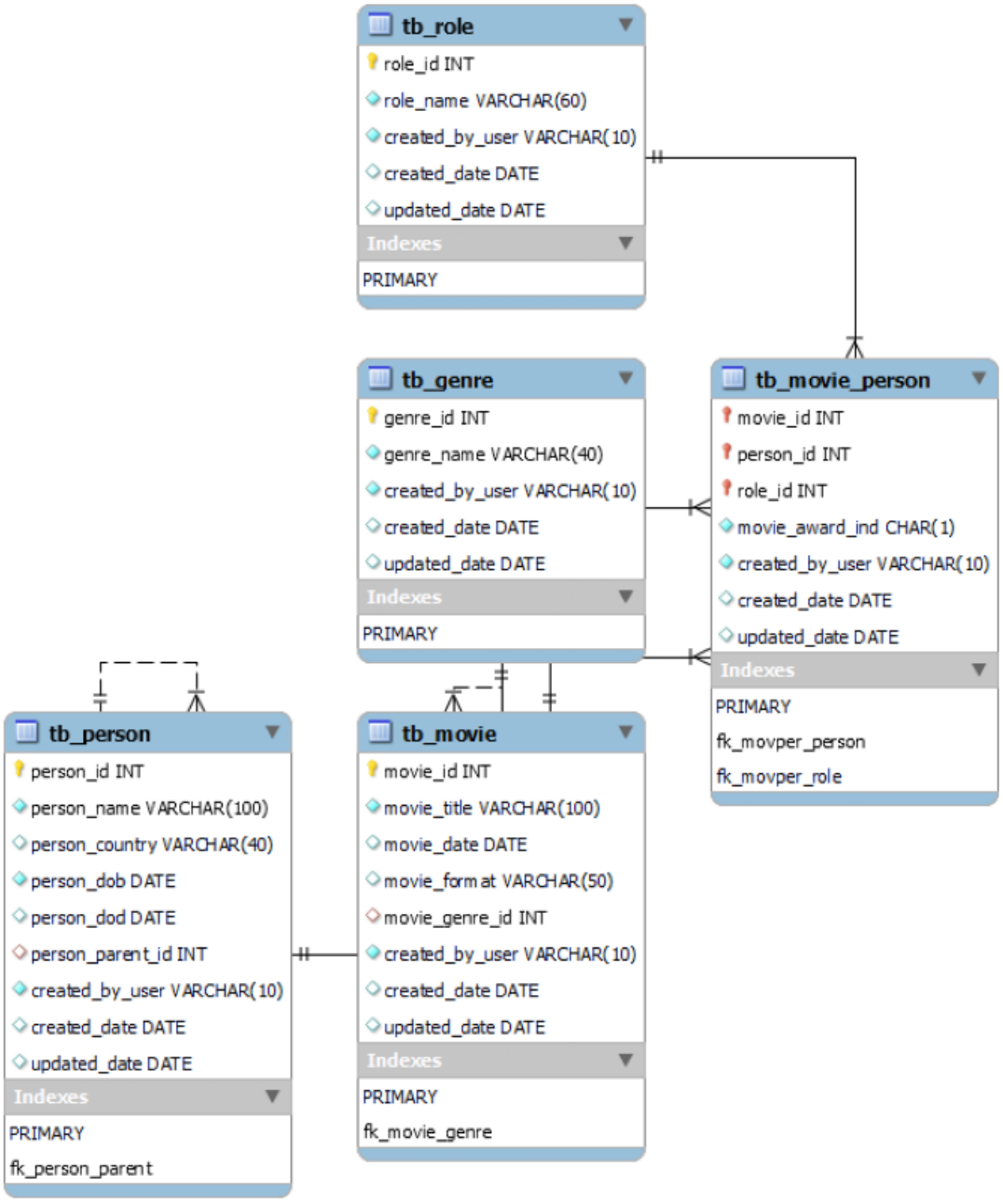
M1 01 : Relational DataBase

[54]:
```
user_name = 'root'
password = 'Freegalado-SQL-22'
```

Exercises 1

From the attached documents (structure and data), create a database with MySQL. It shows the main characteristics of the created scheme and explains the different tables and variables that exist.

[55]:
```
fig, ax = plt.subplots(figsize=(15, 15))
fig = img.imread('EER Movies.png')
plt.imshow(fig)
plt.axis('off')
plt.show()
```



The EER diagram was made with the SQL workbench and shows the relation between the tables and principal keys

Exercises 2

Make the following query on the newly created database:

You must obtain the name, country and date of birth of those persons for whom a date of death is not stated and sort the data from the oldest person to the youngest

person.

[56]:
```
#Conect to local DB
mydb = mysql.connector.connect(
  host="localhost",
  user=user_name,
  passwd=password,
  database="movies"
)


print(mydb)
```

```
<mysql.connector.connection.MySQLConnection object at 0x0000011691D7F080>
```

[57]:
```
#Creating a cursor object using the cursor() method
myCursor = mydb.cursor()
```

[58]:
```
#query
sql = "SELECT person_name,person_country, person_dob FROM tb_person WHERE␣
 ↪person_dod IS NULL ORDER by person_dob"
```

[59]:
```
#show the databases available in mysql

myCursor.execute("SHOW TABLES")

for dbs in myCursor:
  print(dbs)
```

```
('tb_genre',)
('tb_movie',)
('tb_movie_person',)
('tb_person',)
('tb_role',)
```

[60]:
```
pd.read_sql(sql,mydb)
```

[60]:

	person_name	person_country	person_dob
0	John Williams	United States	1928-08-08
1	Vera Miles	United States	1929-08-23
2	Sean Connery	Scotland	1930-07-08
3	Robert Duvall	United States	1931-01-05
4	Morgan Freeman	United States	1935-10-01
5	Francis Ford Coppola	United States	1939-04-07
6	Gary Kurtz	United States	1940-07-27
7	Martin Sheen	United States	1940-08-03
8	Harrison Ford	United States	1942-07-13
9	George Lucas	United States	1944-05-14

```
10         Carmine Coppola  United States  1945-07-08
11        Steven Spielberg  United States  1946-12-18
12             Tim Robbins  United States  1949-06-07
13              Mel Gibson      Australia  1950-08-09
14               Tom Hanks  United States  1956-07-09
15  Emilio Martinez Lazaro          Spain  1956-09-09
16          Karra Elejalde          Spain  1960-03-06
17          Emilio Estevez  United States  1962-05-12
18           Ramón Estevez  United States  1963-08-07
19            Carmen Machi          Spain  1964-08-09
20           Charlie Sheen  United States  1965-09-03
21           Reneé Estevez  United States  1967-04-02
22    Catherine Zeta-Jones          Wales  1969-09-25
23            Robert Alamo          Spain  1970-05-06
24  Daniel Sanchez Arevalo          Spain  1970-06-08
25          Quim Gutierrez          Spain  1981-03-27
26        Veronica Echegui          Spain  1983-03-14
27     Taylor Levi Estevez  United States  1984-06-22
28              Dani Rovira          Spain  1984-07-01
29      Paula Speert Sheen  United States  1986-01-06
30      Paloma Rae Estevez  United States  1986-02-15
31               Clara Lago          Spain  1986-04-17
32          Patrick Criado          Spain  1995-09-23
33       Paula Jones-Sheen  United States  2003-07-06
34                Sam Sheen  United States  2004-03-09
35              Lola Sheen  United States  2005-06-01
36                Bob Sheen  United States  2009-05-01
37                Max Sheen  United States  2009-05-01
```

The query result is sorted by date of birth.

Exercises 3

Make the following query on the newly created database:

You have to get the name of the genre and the total number of movies of that genre and sort it in descending order of total number of movies.

[61]:
```
mydb = mysql.connector.connect(
  host="localhost",
  user=user_name,
  passwd=password,
  database="movies"
)
```

[62]:
```
#query
sql3 = "SELECT tb_genre.genre_name, COUNT(tb_movie.movie_title) AS numTitles␣
 ↪FROM tb_movie \
       LEFT JOIN tb_genre ON tb_genre.genre_id = tb_movie.movie_genre_id\
```

```
        GROUP BY tb_genre.genre_name\
        ORDER by numTitles DESC"
```

[63]:
```
pd.read_sql(sql3,mydb)
```

[63]:

	genre_name	numTitles
0	Comedia	3
1	Ciencia Ficción	2
2	Acción	2
3	Suspense	2
4	Drama	2
5	Bélico	1
6	Terror	1
7	Fantasía	1
8	Romance	1

The query result show us that the genre Comedy is the most produced genre

Exercises 4

Make the following query on the newly created database:

You have to get, for each person, their name and the maximum number of different roles they have played in the same movie. Subsequently, it shows only those people who have taken on more than one role in the same film.

[64]:
```
mydb = mysql.connector.connect(
  host="localhost",
  user=user_name,
  passwd=password,
  database="movies"
)
```

[65]:
```
#query
sql4_1 = "SELECT  tb_person.person_name, max(numRol) as numRol FROM (\
        SELECT tb_person.person_id, tb_person.person_name, tb_movie.movie_id,␣
 ↪tb_movie.movie_title, count(*) AS numRol\
        FROM tb_person \
        JOIN tb_movie_person ON tb_person.person_id = tb_movie_person.person_id\
        JOIN tb_movie  ON tb_movie_person.movie_id = tb_movie.movie_id \
        JOIN tb_role ON tb_movie_person.role_id = tb_role.role_id\
        GROUP BY tb_person.person_id, tb_movie.movie_id\
        ORDER BY numRol desc) tb_person\
        GROUP BY tb_person.person_id"
```

[66]:
```
pd.read_sql(sql4_1,mydb)
```

[66]:

	person_name	numRol
0	Alfred Joseph Hitchcock	3

```
1      Francis Ford Coppola       3
2              George Lucas       2
3          Steven Spielberg       2
4              Martin Sheen       1
5            Harrison Ford       1
6            Charlie Sheen       1
7                Tom Hanks       1
8      Catherine Zeta-Jones       1
9              Dani Rovira       1
10             Clara Lago       1
11            Carmen Machi       1
12          Karra Elejalde       1
13           Marlon Brando       1
14           Robert Duvall       1
15         Anthony Perkins       1
16             Vera Miles       1
17   Emilio Martinez Lazaro       1
18             Gary Kurtz       1
19         Carmine Coppola       1
20           John Williams       1
```

The query result show us the maximum number of roles that a person have in a movie.

[67]:
```
sql4_2 = "SELECT  tb_person.person_name, max(numRol) as numRol FROM (\
        SELECT tb_person.person_id, tb_person.person_name, tb_movie.movie_id,␣
 ↪tb_movie.movie_title, count(*) as numRol \
        FROM tb_person \
        JOIN tb_movie_person ON tb_person.person_id = tb_movie_person.person_id\
        JOIN tb_movie  ON tb_movie_person.movie_id = tb_movie.movie_id \
        JOIN tb_role ON tb_movie_person.role_id = tb_role.role_id\
        GROUP BY tb_person.person_id, tb_movie.movie_id\
        HAVING numRol > 1\
        ORDER BY numRol desc) tb_person\
        GROUP BY tb_person.person_id"
```

[68]:
```
pd.read_sql(sql4_2,mydb)
```

[68]:
```
              person_name  numRol
0  Alfred Joseph Hitchcock       3
1     Francis Ford Coppola       3
2             George Lucas       2
3         Steven Spielberg       2
```

in this query the HAVING clause is used as conditional to keep only the people with more than 1 role in a movie.

Exercises 5

Make the following query on the newly created database:

You have to create a new genre called "Documentary" which has the number 69 as its identifier.

[69]:
```
mydb = mysql.connector.connect(
  host="localhost",
  user=user_name,
  passwd=password,
  database="movies"
)
```

```
INSERT INTO movies.tb_genre (genre_id, genre_name, created_by_user, created_date, updated_date
```

I preferred to use MySQL Workbench for this exercise, the above query was used.

[70]:
```
sql5_view = "SELECT * FROM tb_genre"
```

[71]:
```
pd.read_sql(sql5_view,mydb)
```

[71]:

	genre_id	genre_name	created_by_user	created_date	updated_date
0	1	Acción	OS_SGAD	None	None
1	2	Ciencia Ficción	OS_SGAD	None	None
2	3	Comedia	OS_SGAD	None	None
3	4	Drama	OS_SGAD	None	None
4	5	Fantasía	apermag	None	None
5	6	Melodrama	apermag	2018-09-01	2018-09-27
6	7	Musical	OS_SGAD	None	None
7	8	Romance	OS_SGAD	None	None
8	9	Suspense	OS_SGAD	None	None
9	10	Terror	OS_SGAD	None	None
10	11	Bélico	OS_SGAD	None	None
11	12	Documental	FcoReg	2022-07-17	2022-07-19

The Documental genre was added to the genre table.

Exercises 6

Make the following query on the newly created database:

Removes the movie "La Gran Familia Española" from the database.

[72]:
```
mydb = mysql.connector.connect(
  host="localhost",
  user=user_name,
  passwd=password,
  database="movies"
)
```

```
DELETE FROM movies.tb_movie WHERE (movie_id = 11)
```

I preferred to use MySQL Workbench for this exercise, the above query was used.

[73]:
```
#query
sql6_view = "SELECT * FROM tb_movie"
```

[74]:
```
pd.read_sql(sql6_view,mydb)
```

[74]:

	movie_id	movie_title	movie_date	movie_format	movie_genre_id	created_by_user	created_date	updated_date
0	1	Apocalypse Now	1979-05-10	Film	11	OS_SGAD	None	None
1	2	Star Wars:Episode IV - A New Hope	1977-05-25	Film	2	OS_SGAD	None	None
2	3	Indiana Jones and the Temple of Doom	1984-05-08	Film	1	OS_SGAD	None	None
3	4	The Terminal	2004-06-18	Digital	3	OS_SGAD	None	None
4	5	Jaws	1975-01-01	Film	10	OS_SGAD	None	None
5	6	ET The Extraterrestrial	1982-07-25	Film	5	OS_SGAD	None	None
6	7	Psycho	1960-05-06	Film	9	OS_SGAD	None	None
7	8	Ocho Apellidos Vascos	2014-03-14	Digital	3	OS_SGAD	None	None
8	9	Ocho Apellidos Catalanes	2016-06-09	Digital	3	OS_SGAD	None	None
9	10	El otro lado de la cama	2002-09-04	Digital	8	OS_SGAD	None	None
10	12	El dia de la bestia	1994-12-25	Film	1	OS_SGAD	None	None
11	13	Braveheart	1995-08-08	Film	4	OS_SGAD	None	None
12	14	The Shawshank Redemption	1992-01-07	Film	4	OS_SGAD	None	None
13	15	Las brujas de Zugarramurdi	2009-10-07	Digital	9	OS_SGAD	None	None
14	16	Blade Runner	1982-12-25	Digital	2	OS_SGAD	None	None

Exercises 7

Make the following query on the newly created database:

It changes the genre of the film "Ocho apellidos catalanes" so that it is listed as a comedy and not as a romantic one.

[75]:
```
mydb = mysql.connector.connect(
  host="localhost",
```

```
  user=user_name,
  passwd=password,
  database="movies"
)
```

```
UPDATE movies.tb_movie SET movie_genre_id = 3 WHERE (movie_id = 9)
```

I preferred to use MySQL Workbench for this exercise, the above query was used.

[76]:
```
#query
sql7_view = "SELECT tb_movie.movie_title,  tb_genre.genre_id, tb_genre.
 ↪genre_name\
            FROM tb_movie\
            LEFT JOIN tb_genre ON tb_genre.genre_id = tb_movie.movie_genre_id"
```

[77]:
```
pd.read_sql(sql7_view,mydb)
```

[77]:

	movie_title	genre_id	genre_name
0	Apocalypse Now	11	Bélico
1	Star Wars:Episode IV - A New Hope	2	Ciencia Ficción
2	Indiana Jones and the Temple of Doom	1	Acción
3	The Terminal	3	Comedia
4	Jaws	10	Terror
5	ET The Extraterrestrial	5	Fantasía
6	Psycho	9	Suspense
7	Ocho Apellidos Vascos	3	Comedia
8	Ocho Apellidos Catalanes	3	Comedia
9	El otro lado de la cama	8	Romance
10	El dia de la bestia	1	Acción
11	Braveheart	4	Drama
12	The Shawshank Redemption	4	Drama
13	Las brujas de Zugarramurdi	9	Suspense
14	Blade Runner	2	Ciencia Ficción

The query result show us that the movie Ocho Apellidos Catalanes have a genre update from Romance to Comedy.