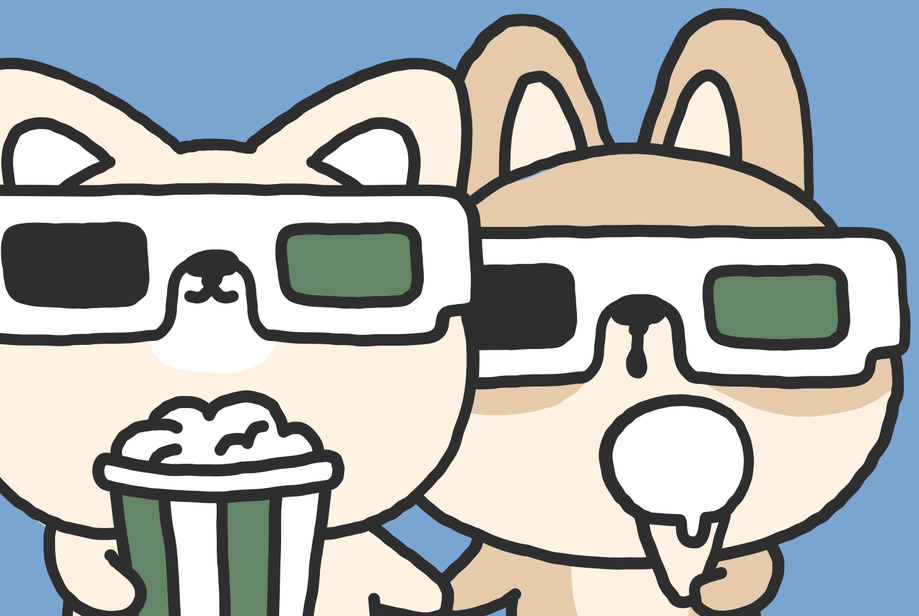
Final Project Database Theory

Movie Database System



Student: Akmal and Nurbek

Group: IT3-2205

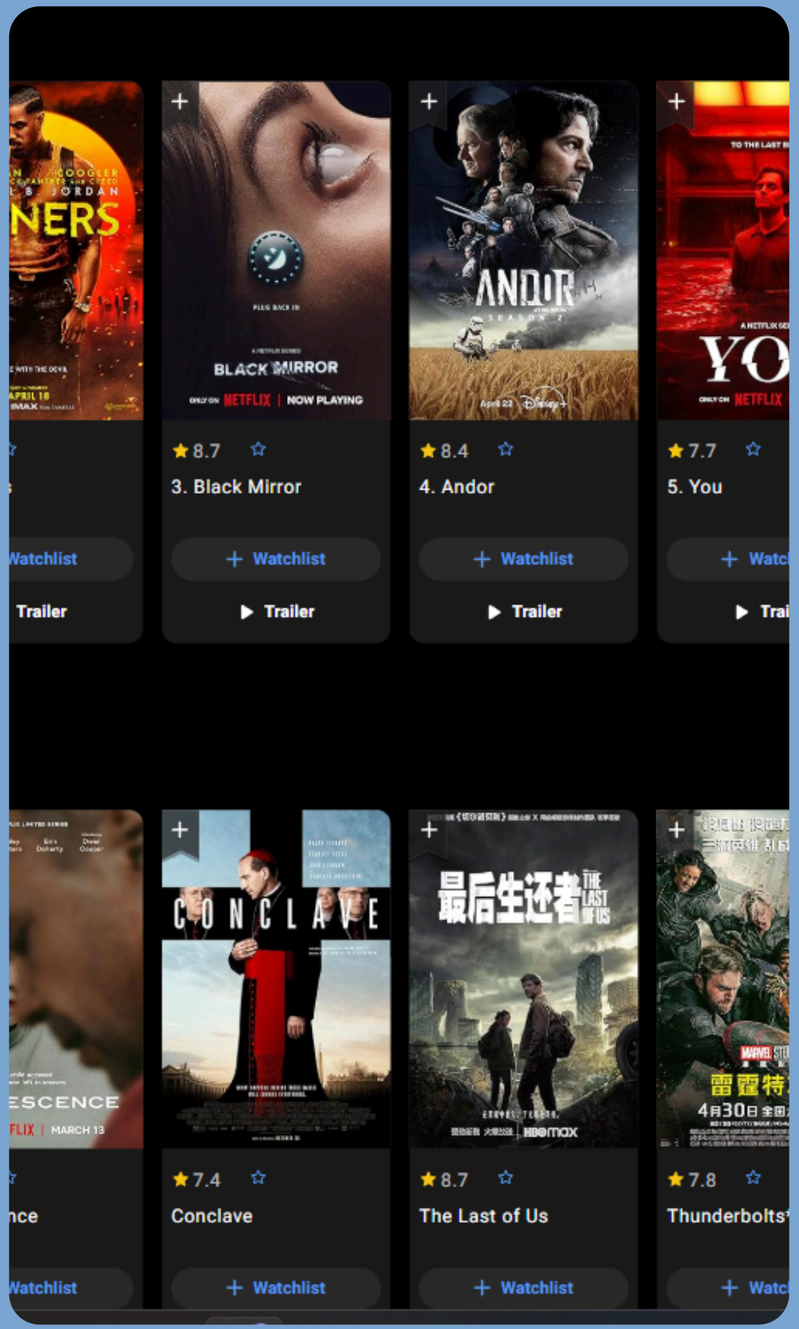
Table of Content

Introduction

Design a relational database to manage movie information.

Store data about movies, genres, actors, directors, and ratings

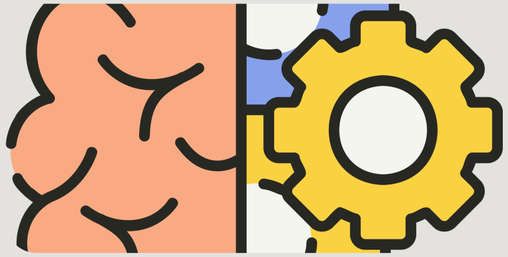
Tool : PostgreSql



DATABASE DESIGN

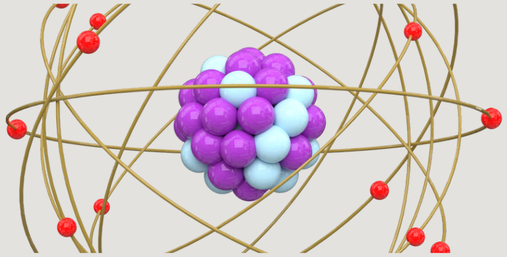
SUBJECT AREA ANALYSIS AND CONCEPTUAL DESIGN

The conceptual design tell the main structure of the movie database, and including movies name, genres, people involved, user ratings and so on.



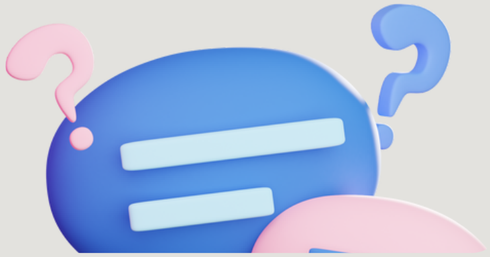
LOGICAL DESIGN

We turned the concept map into a table structure , and set primary keys and foreign keys to keep the data correct, also the design follows normalization



PHYSICAL DESIGN AND CONTENT

Use CREATE TABLE to create each table, handle the type (such as INT), set primary key, foreign key, constraints, and add real data to it



QUERIES

find useful information from the movie database using query statements such as searching, filtering, linking tables, aggregation, etc.

Subject Area Analysis and Conceptual Design

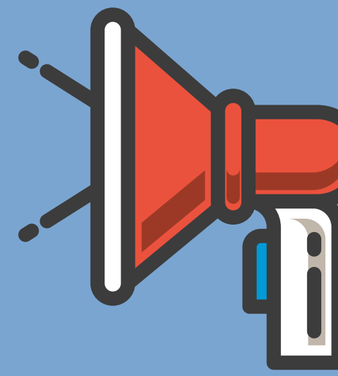
Description of the Subject Area

The database is designed to manage information related to movies, including details about movies, actors, directors, genres, awards, users, reviews, votes, and trailers. The database structure is based on IMDb-like data storage, allowing efficient organization and retrieval of movie-related data.

Subject Area Analysis and Conceptual Design

The system will store information about:

- Movies (title, release year, rating, duration, description, etc.).
- Actors and their roles in movies.
- Directors and their contributions to movies.
- Genres associated with movies.
- Awards and their relation to movies.
- Users who can submit reviews and vote on movies.
- Reviews and ratings given by users.
- Votes for movies by users.
- Trailers linked to movies.
-

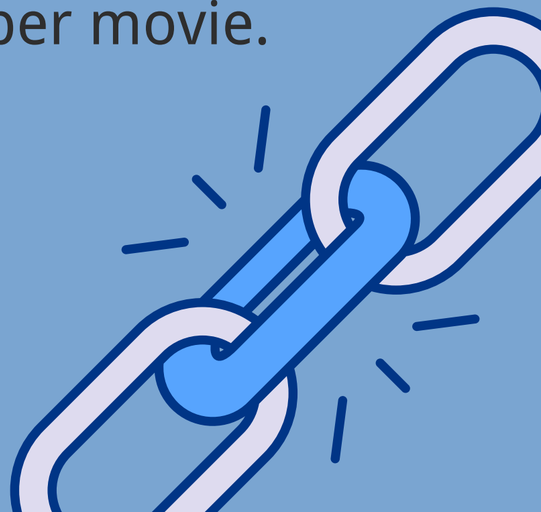


The database will help users find information about movies, explore details about cast and crew, see movie ratings, and read user reviews. It ensures a structured way to manage relationships between different entities.

Subject Area Analysis and Conceptual Design

Constraints

- Each movie must have a unique Movie_ID as the primary key.
- Each actor and director must have a unique Actor_ID and Director_ID.
- A movie can belong to multiple genres, but each genre must be unique.
- Users must have a unique User_ID and an email.
- A user can only submit one review per movie.



Subject Area Analysis and Conceptual Design

Potential Queries from Users

- Retrieve all movies released in a specific year.
- Get the highest-rated movies.
- Find movies by a specific director.
- List all actors who played in a given movie.
- Find movies of a specific genre.
- Get all movies that won a specific award.
- Retrieve the average rating of a particular movie.
- Find movies with the most user reviews.
- List top-rated movies based on user votes.
- Retrieve all trailers for a specific movie.
- Find the most reviewed movies by users.
- Get all reviews for a particular movie.
- Find movies with a duration longer than a given time.
- Get movies sorted by rating in descending order.
- List all awards won by a specific movie.
-

Subject Area Analysis and Conceptual Design

ER-Diagram

- ·Movies (PK: Movie_ID)
- ·Actors (PK: Actor_ID)
- ·Movie_Actors (FK: Movie_ID, Actor_ID)
- ·Directors (PK: Director_ID)
- ·Movie_Directors (FK: Movie_ID, Director_ID)
- ·Genres (PK: Genre_ID)
- ·Movie_Genres (FK: Movie_ID, Genre_ID)
- ·Users (PK: User_ID)
- ·Reviews (PK: Review_ID, FK: Movie_ID, User_ID)
- ·Votes (PK: Vote_ID, FK: Movie_ID, User_ID)
- ·Awards (PK: Award_ID)
- ·Movie_Awards (FK: Movie_ID, Award_ID)
- ·Trailers (PK: Trailer_ID, FK: Movie_ID)

ER-DIAGRAM FOLLOW THIS RULE----------

ONE - ONE
ONE - MANY
MANY - MANY
MANY - ONE

Logical Design

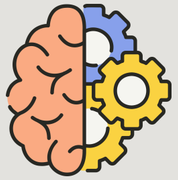


TABLE: MOVIES

Field's name	Field's content	Type, length	Note (constraints)
Movie_ID	Movie ID	N(7)	PK
Title	Title	C(255)	NOT NULL
Release_year	Release year	N(4)	
Duration	Duration	N(5)	
Description	Description	C(1000)	

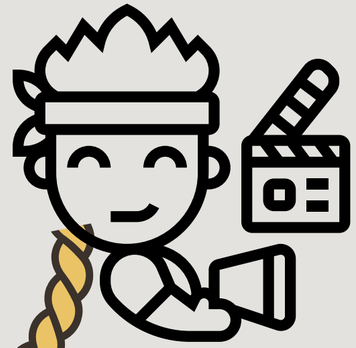


TABLE: DIRECTORS

Field's name	Field's content	Type, length	Note (constraints)
Director_ID	Director ID	N(7)	PK
Director_name	Director name	C(255)	NOT NULL
Birth_year	Birth year	N(4)	

Logical Design

TABLE: MOVIE_GENRES

Field's name	Field's content	Type, length	Note (constraints)
Movie_ID	Movie ID	N(7)	FK to Movies
Genre_ID	Genre ID	N(7)	FK to Genres

TABLE: ACTORSS

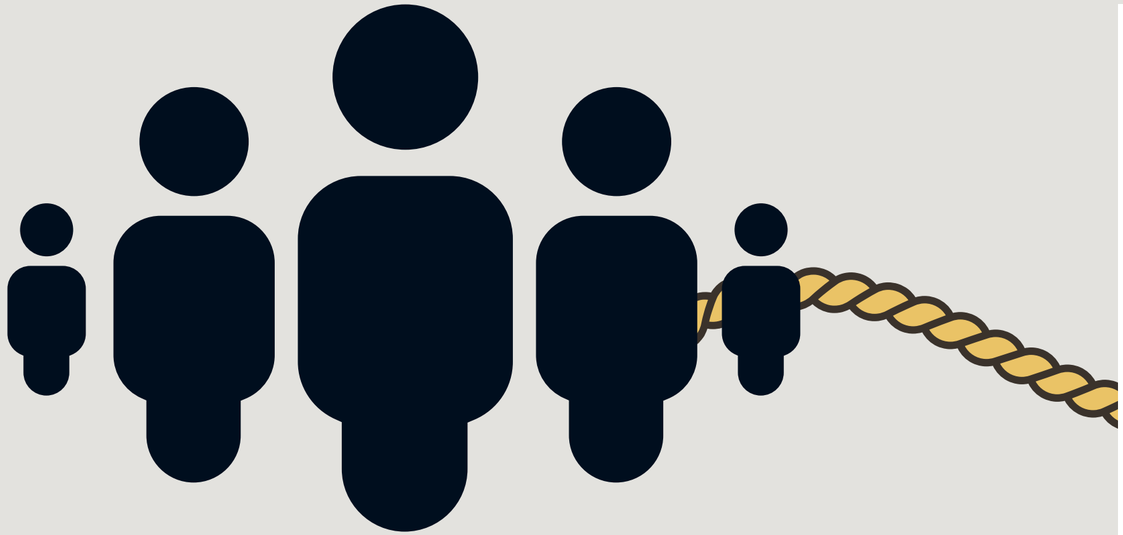
Field's name	Field's content	Type, length	Note (constraints)
Actor_ID	Actor ID	N(7)	PK
Actor_Name	Actor Name	C(255)	NOT NULL
Birth_date	Birth date	D	

Logical Design

TABLE: MOVIE_ACTORS

Field's name	Field's content	Type, length	Note (constraints)
Movie_ID	Movie ID	N(7)	FK to Movies
Actor_ID	Actor ID	N(7)	FK to Actors

TABLE: USERS



Field's name	Field's content	Type, length	Note (constraints)
User_ID	User ID	N(7)	PK
Username	Username	C(255)	NOT NULL
Email	Email	C(255)	UNIQUE, NOT NULL
Registration_date	Registration Date	D	

Logical Design

TABLE: REVIEWS

Field's name	Field's content	Type, length	Note (constraints)
Review_ID	Review ID	N(7)	PK
Movie_ID	Movie ID	N(7)	FK to Movies
User_ID	User ID	N(7)	FK to Users
Rating	Rating	N(2)	
Review_date	Review Date	D	

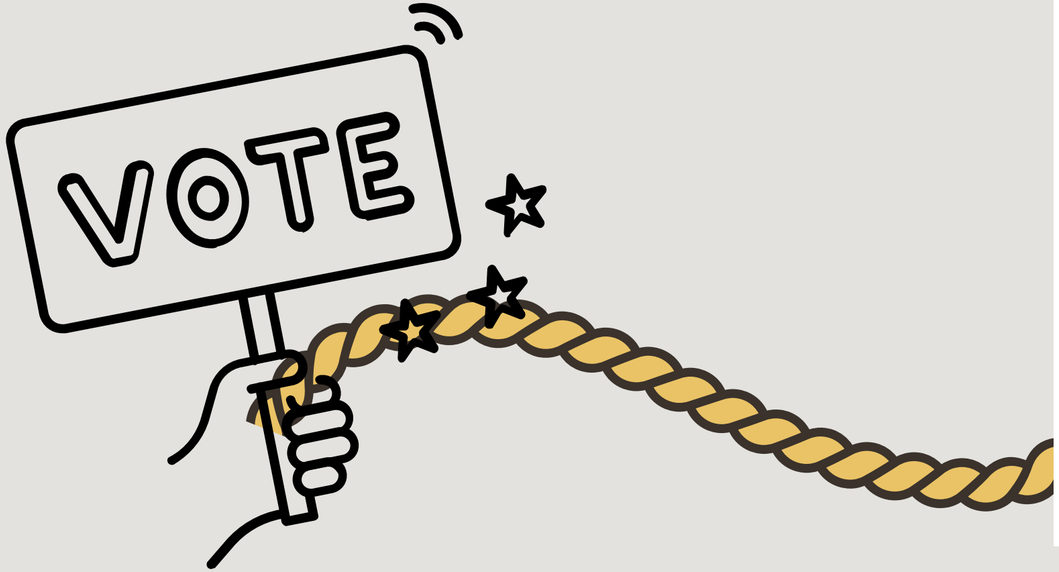


TABLE: VOTES

Field's name	Field's content	Type, length	Note (constraints)
Vote_ID	Vote ID	N(7)	PK
Movie_ID	Movie ID	N(7)	FK to Movies
User_ID	User ID	N(7)	FK to Users
Rating_Value	Rating Value	N(2)	

Logical Design

TABLE: TRAILERS

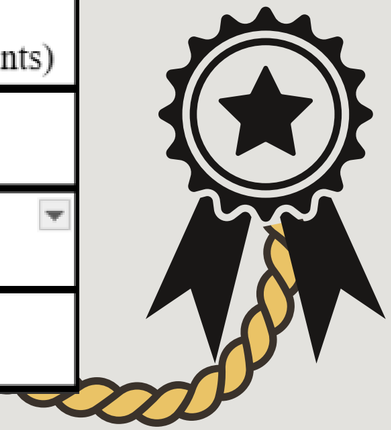
Field's name	Field's content	Type, length	Note (constraints)
Trailer_ID	Trailer ID	N(7)	PK
Movie_ID	Movie ID	N(7)	FK to Movies
URL	Trailer URL	C(500)	
Release_Date	Release Date	D	

TABLE: MOVIE_AWARDS

Field's name	Field's content	Type, length	Note (constraints)
Movie_ID	Movie ID	N(7)	FK to Movies
Award_ID	Award ID	N(7)	FK to Awards
Category	Award Category	C(255)	

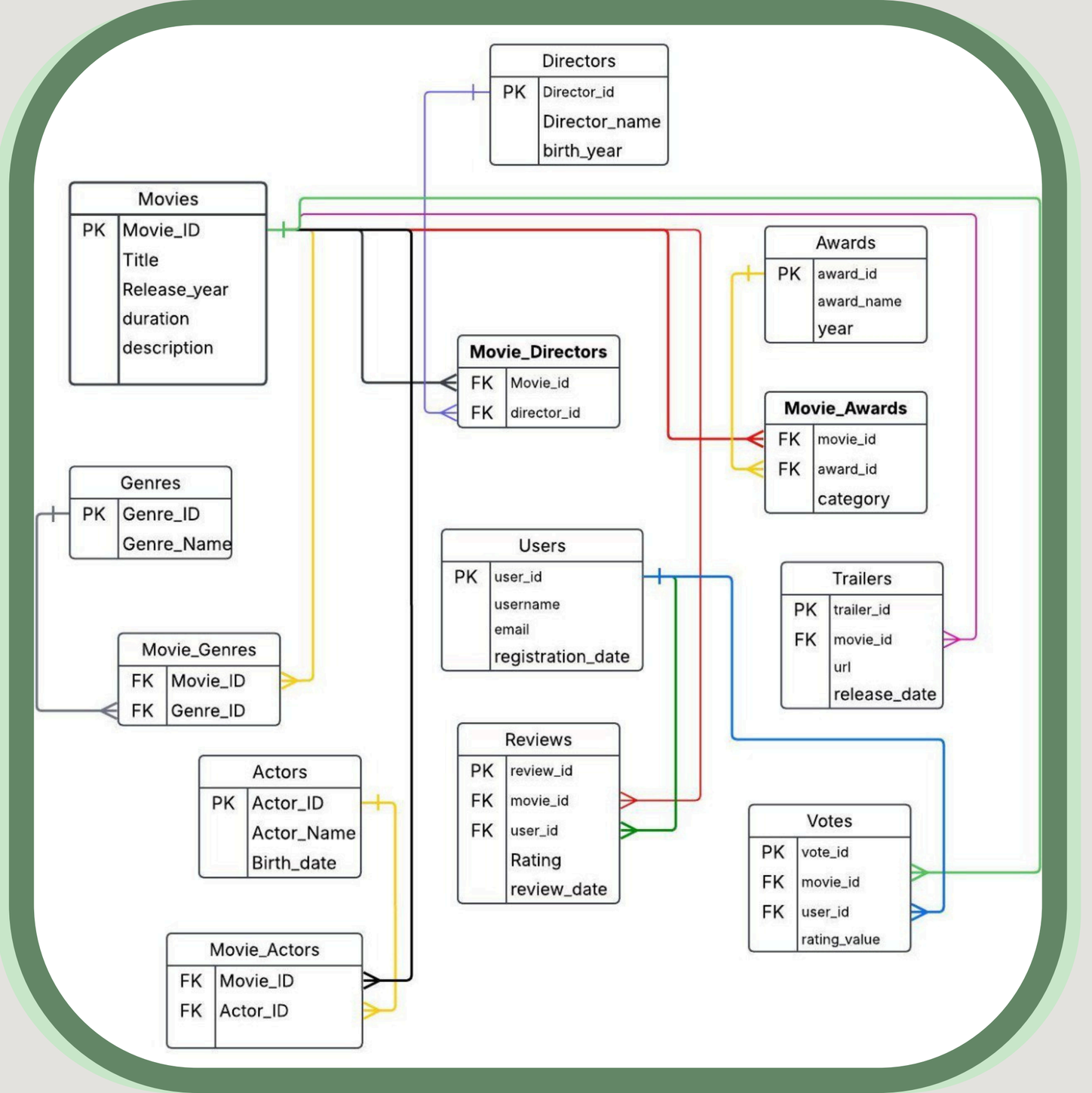
TABLE: AWARDS

Field's name	Field's content	Type, length	Note (constraints)
Award_ID	Award ID	N(7)	PK
Award_Name	Award Name	C(255)	NOT NULL
Year	Year	N(4)	



Logical Design

ER-DIAGRAM



Physical Design and Content

The Table by Created called Movies

```sql
CREATE TABLE Movies (
    Movie_ID SERIAL PRIMARY KEY,
    Title VARCHAR(255) NOT NULL,
    Release_year INT,
    Duration INT,
    Description TEXT
);
```

The data by Inserted

```sql
INSERT INTO Movies (Title, Release_year, Duration, Description) VALUES
('The Shawshank Redemption', 1994, 142, 'Two imprisoned men bond over a number of years...'),
('The Godfather', 1972, 175, 'The aging patriarch of an organized crime dynasty...'),
('The Dark Knight', 2008, 152, 'When the menace known as the Joker wreaks havoc...'),
('Pulp Fiction', 1994, 154, 'The lives of two mob hitmen, a boxer, and others...'),
('Fight Club', 1999, 139, 'An insomniac office worker and a soap maker...'),
('Forrest Gump', 1994, 142, 'The presidencies of Kennedy and Johnson...'),
('Inception', 2010, 148, 'A thief who steals corporate secrets through dream-sharing...'),
('The Matrix', 1999, 136, 'A hacker discovers the reality is a simulation...'),
('The Lord of the Rings: The Fellowship of the Ring', 2001, 178, 'A meek Hobbit sets out to destroy a powerful ring...'),
('Interstellar', 2014, 169, 'A team travels through a wormhole in space...'),
('Gladiator', 2000, 155, 'A former Roman general seeks revenge...'),
('Titanic', 1997, 195, 'A seventeen-year-old aristocrat falls in love...'),
('Avengers: Endgame', 2019, 181, 'After the devastating events of Infinity War...'),
('The Silence of the Lambs', 1991, 118, 'A young FBI cadet must confide in a manipulative killer...'),
('Schindler''s List', 1993, 195, 'In German-occupied Poland during WWII...');
```

	movie_id [PK] integer	title character varying (255)	release_year integer	duration integer	description text
1	1	The Shawshank Redemption	1994	142	Two imprisoned men bond over a number of years...
2	2	The Godfather	1972	175	The aging patriarch of an organized crime dynasty...
3	3	The Dark Knight	2008	152	When the menace known as the Joker wreaks havoc...
4	4	Pulp Fiction	1994	154	The lives of two mob hitmen, a boxer, and others...
5	5	Fight Club	1999	139	An insomniac office worker and a soap maker...
6	6	Forrest Gump	1994	142	The presidencies of Kennedy and Johnson...
7	7	Inception	2010	148	A thief who steals corporate secrets through dream-sharing...
8	8	The Matrix	1999	136	A hacker discovers the reality is a simulation...
9	9	The Lord of the Rings: The Fellowship of the Ring	2001	178	A meek Hobbit sets out to destroy a powerful ring...
10	10	Interstellar	2014	169	A team travels through a wormhole in space...
11	11	Gladiator	2000	155	A former Roman general seeks revenge...
12	12	Titanic	1997	195	A seventeen-year-old aristocrat falls in love...
13	13	Avengers: Endgame	2019	181	After the devastating events of Infinity War...
14	14	The Silence of the Lambs	1991	118	A young FBI cadet must confide in a manipulative killer...
15	15	Schindler's List	1993	195	In German-occupied Poland during WWII...

The table look like

Physical Design and Content

The Table by Created called Directors

```
CREATE TABLE Directors (
    Director_ID SERIAL PRIMARY KEY,
    Director_name VARCHAR(255) NOT NULL,
    Birth_year INT
);
```

The data by Inserted to the Directors

```
INSERT INTO Directors (Director_name, Birth_year) VALUES
('Frank Darabont', 1959),
('Francis Ford Coppola', 1939),
('Christopher Nolan', 1970),
('Quentin Tarantino', 1963),
('David Fincher', 1962),
('Robert Zemeckis', 1952),
('Lana Wachowski', 1965),
('Peter Jackson', 1961),
('Ridley Scott', 1937),
('James Cameron', 1954),
('Anthony Russo', 1970),
('Joe Russo', 1971),
('Jonathan Demme', 1944),
('Steven Spielberg', 1946),
('Denis Villeneuve', 1967);
```

The table look like

	director_id [PK] integer	director_name character varying (255)	birth_year integer
1	1	Frank Darabont	1959
2	2	Francis Ford Coppola	1939
3	3	Christopher Nolan	1970
4	4	Quentin Tarantino	1963
5	5	David Fincher	1962
6	6	Robert Zemeckis	1952
7	7	Lana Wachowski	1965
8	8	Peter Jackson	1961
9	9	Ridley Scott	1937
10	10	James Cameron	1954
11	11	Anthony Russo	1970
12	12	Joe Russo	1971
13	13	Jonathan Demme	1944
14	14	Steven Spielberg	1946
15	15	Denis Villeneuve	1967

Physical Design and Content

As you know we get 13 tables to created, and it is hard to show all of them , but the point is that all of them are get the following things :

Each table contains real from IMDb.

The data is connected through foreign keys.

The types of data by inserted, including users, reviews,actors trailers, awards ..... and so onnnnnnnn.....

Average Rating of Each Movie

	title character varying (255)	average_rating numeric
1	The Matrix	8.00
2	Fight Club	10.00
3	The Shawshank Redemption	10.00
4	Titanic	8.00
5	Inception	9.00
6	Pulp Fiction	9.00
7	Interstellar	9.00
8	The Dark Knight	8.00
9	Schindler's List	10.00
10	The Lord of the Rings: The Fellowship of the Ring	10.00
11	Forrest Gump	8.00
12	Avengers: Endgame	9.00
13	The Silence of the Lambs	8.00
14	The Godfather	9.00
15	Gladiator	7.00

```sql
SELECT Title, ROUND(AVG(Rating), 2) AS Average_Rating
FROM Movies
JOIN Reviews ON Movies.Movie_ID = Reviews.Movie_ID
GROUP BY Title;
```

This query calculates the average user rating for each movie. It helps identify which films are the most appreciated by users

Using about Queries

Actors Who Starred in Both "The Godfather" and "Pulp Fiction"

	actor_name character varying (255)
1	Marlon Brando
2	Al Pacino
3	John Travolta
4	Samuel L. Jackson

```
SELECT Actor_Name FROM Actors
JOIN Movie_Actors ON Actors.Actor_ID = Movie_Actors.Actor_ID
JOIN Movies ON Movie_Actors.Movie_ID = Movies.Movie_ID
WHERE Movies.Title IN ('The Godfather', 'Pulp Fiction');
```

This query finds actors who acted in both classic films, demonstrating the use of many-to-many relationships and filtering with IN

Using about Queries

Movies Released After 2000 That Won "Best Picture"

	title character varying (255)
1	The Lord of the Rings: The Fellowship of the Ring

```sql
SELECT Title FROM Movies
JOIN Movie_Awards ON Movies.Movie_ID = Movie_Awards.Movie_ID
JOIN Awards ON Movie_Awards.Award_ID = Awards.Award_ID
WHERE Awards.Award_Name = 'Best Picture' AND Movies.Release_year > 2000;
```

Retrieves award-winning movies that were released after 2000 by joining multiple related tables

Using about Queries

Each User’s Highest-Rated Movie

	username character varying (255)	title character varying (255)	highest_rating integer
1	user1	The Shawshank Redemption	10
2	user2	The Godfather	9
3	user3	The Dark Knight	8
4	user4	Pulp Fiction	9
5	user5	Fight Club	10
6	user6	Forrest Gump	8
7	user7	Inception	9
8	user8	The Matrix	8
9	user9	The Lord of the Rings: The Fellowship of the Ring	10
10	user10	Interstellar	9
11	user11	Gladiator	7
12	user12	Titanic	8
13	user13	Avengers: Endgame	9
14	user14	The Silence of the Lambs	8
15	user15	Schindler's List	10

```sql
SELECT u.Username, m.Title, r.Rating AS Highest_Rating
FROM Users u
JOIN Reviews r ON u.User_ID = r.User_ID
JOIN Movies m ON r.Movie_ID = m.Movie_ID
WHERE (u.User_ID, r.Rating) IN (
    SELECT r2.User_ID, MAX(r2.Rating)
    FROM Reviews r2
    GROUP BY r2.User_ID
);
```

This query identifies the favorite (highest-rated) movie for each user. It uses grouping and a correlated subquery.

Using about Queries

Movies with No Reviews

	title character varying (255)

```
SELECT Title FROM Movies
LEFT JOIN Reviews ON Movies.Movie_ID = Reviews.Movie_ID
WHERE Reviews.Review_ID IS NULL;
```

This query lists movies that have not received any reviews yet. It uses a LEFT JOIN and NULL filter to find unmatched records.

Using about Queries

Conclusion

We built a movie database that stores information about movies, actors, directors, genres, users, and reviews. The project includes database design, sample data, and useful SQL queries. It works well and show every detail information that you want.

Thank you !

REFERENCES

- lecture from 1 to 12,
- IMDb site
- https://www.w3schools.com/sql/
- https://sql-academy.org/
- https://www.geeksforgeeks.org/sql-tutorial/