CSE 3139

DATABASE MANAGEMENT SYSTEMS

FALL 2023

COURSE PROJECT

Assoc. Prof. Dr. Övünç ÖZTÜRK

Assistant: Turan Göktuğ ALTUNDOĞAN

APPLICATION SCENARIO: NETFLIX

Furkan ÖZKAYA 200316060

Sıla Naz ASLAN 210316072

Furkan BAYTAK 210316033

Table of Contents

1. Introduction

The focus of the database design was to create a model to accommodate the complex structure and requirements of a film streaming service (Neflix Scenario). This model was carefully organized to support the various features and functionalities of the service and provide a rich experience for users.

The identification of the required tables and the creation of relationships are designed in a similar way to the existing Netflix application. Considering only the basic features and database relationships of the existing Netflix application, a simplified movie streaming service application was tried to be designed.

The **'MOVIES'** table was created to store the basic information of the films. It contains important information such as a unique identifier (ID) for each film, title, year of release and MPAA rating. This table forms the basis of the database and serves as the central point for queries.

The **'DIRECTORS'**, **'ACTORS'**, **'GENRES'**, and **'AWARDS'** tables represent various aspects of the films, and each contains unique information. This information includes details about the directors of the films, the actors involved, the genres they belong to, and the awards they have won. The relationships between these tables are provided using intermediate tables, for example **'MOVIES_DIRECTORS'** or **'MOVIES_ACTORS'**, designed to manage these many-to-many relationships.

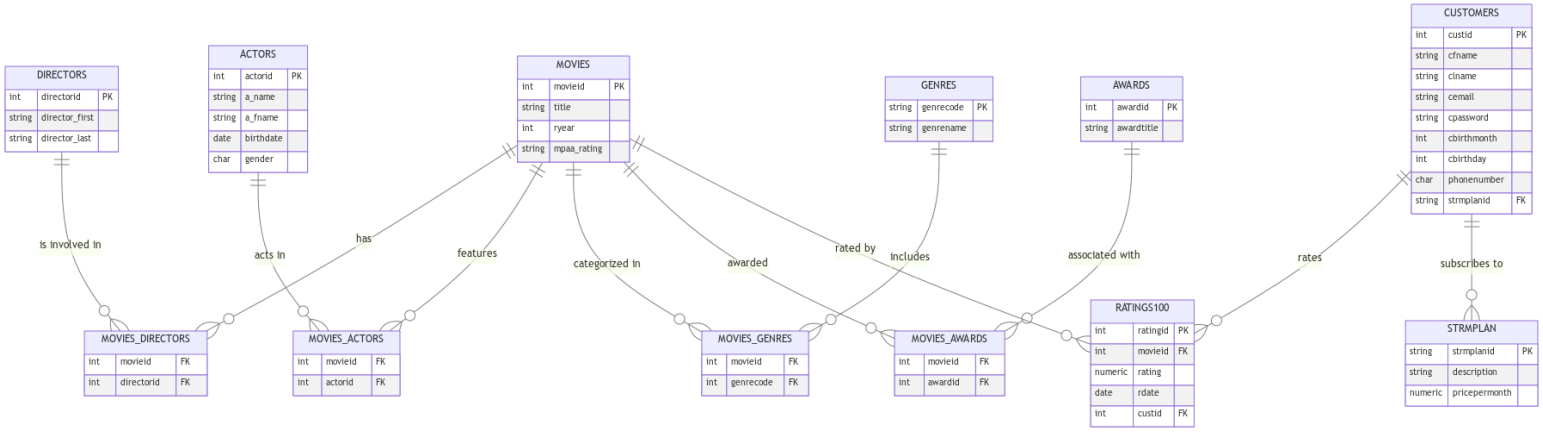
Customer information and preferences are stored in the **'CUSTOMERS'** and **'STRMPLAN'** tables. This contains information such as customers' personal details and which streaming plans they subscribe to. This design allows analyzing customer behavior and personalizing the user experience.

The database also contains a **'RATINGS100'** table that stores the audience ratings of each film. This table is an important metric used to measure the popularity of films and audience satisfaction.

At each stage of the design, primary and foreign keys were used to maintain database integrity. This is critical to ensure data consistency and to guarantee that the various queries run correctly and efficiently.

In conclusion, this database design was realized with functionality and scalability in mind, reflecting real-world needs and user scenarios. Every aspect of the design has been carefully considered to provide users with a rich and interactive experience.

2. E-R Diagram



3. Creating Necessary Tables and Relations

Tables and Keys:

- The database contains primary tables such as ***MOVIES, DIRECTORS, ACTORS, GENRES, STRMPLAN, CUSTOMERS, AWARDS*** *and* ***RATINGS100****.*
- Each table is defined with primary keys (**PK**), such as ***movieid***, ***directorid***, ***actorid***, etc.

Relationships:

- **Many-to-Many Relationships**: Relational tables such as ***MOVIES_DIRECTORS***, ***MOVIES_ACTORS***, ***MOVIES_GENRES***, ***MOVIES_AWARDS*** are used to manage plural relationships. For example, a movie can have more than one director and a director can direct more than one movie.
- **One-to-Many Relationships**: The ***RATINGS100*** table allows customers (***CUSTOMERS***) to rate multiple movies (***MOVIES***).
- The ***CUSTOMERS*** table is linked to the ***STRMPLAN*** table, so that each customer has a subscription plan.

Table Attributes:

- Tables contain the data types and attributes of fields. For example, in the ***ACTORS*** table, the ***birthdate*** field is of type ***DATE***, and the ***gender*** field is **CHAR(1)** with specific values (**'M'**, **'F'**) required.

This diagram shows the basic structure of the database design, describes the relationships between tables and the basic properties of each table. It provides a clear overview of the general structure and functioning of the database.

Adding 10 Data for Each Table:
The 10 data to be written to the database have been carefully selected to ensure that the relationships and queries are valid. Firstly, 10 movies were selected to fill the ***'MOVIES'*** table.

"The Shawshank Redemption",

"The Godfather",

"The Dark Knight",

"12 Angry Men",

"Schindler's List",

"The Lord of the Rings: The Return of the King",

"Pulp Fiction",

"The Good, the Bad and the Ugly",

"Fight Club",

"Forrest Gump.".

As a result of our research on the internet on these selected movies, ***'DIRECTORS', 'ACTORS', 'GENRES', 'AWARDS'*** and ***'RATINGS100'*** tables were filled with the real data of the movies.

'CUSTOMERS', 'STRMPLAN' tables are filled with random user data created by artificial intelligence and the streaming plan with which these users use the application.

The queries to be made for the created database will obtain more effective results when we have a more comprehensive database. In order to prevent the results of the queries to be made from returning null values, for each movie, values such as a director, an actor, an award were added to the tables as data. However, adding 1 data for each movie causes only 1 result for important queries. This does not indicate that the queries made are not important, it only shows that the results are this way due to the low amount of data.

4. Creating Important Queries for The Application

Query 1 *- Find the highest rated films and their directors:*

```
266  -- Find the highest rated films and their directors:
267
268  SELECT m.title, d.director_first, d.director_last, r.rating
269  FROM movies m
270  JOIN movies_directors md ON m.movieid = md.movieid
271  JOIN directors d ON md.directorid = d.directorid
272  JOIN ratings100 r ON m.movieid = r.movieid
273  ORDER BY r.rating DESC
274  LIMIT 10;
```

Data Output | Messages | Notifications

	title character varying (108)	director_first character varying (25)	director_last character varying (60)	rating numeric (2,1)
1	The Shawshank Redemption	Frank	Darabont	9.3
2	The Godfather	Francis Ford	Coppola	9.2
3	The Dark Knight	Christopher	Nolan	9.0
4	12 Angry Men	Sidney	Lumet	8.9
5	Schindler's List	Steven	Spielberg	8.9
6	The Lord of the Rings: The Return of the King	Peter	Jackson	8.9
7	Pulp Fiction	Quentin	Tarantino	8.9
8	The Good, the Bad and the Ugly	Sergio	Leone	8.8
9	Fight Club	David	Fincher	8.8
10	Forrest Gump	Robert	Zemeckis	8.8

The query fetches the highest-rated films along with their directors' names. It performs joins across ***'movies', 'movies_directors', 'directors',*** and ***'ratings100'*** to select movie titles, director names, and ratings, then orders the results by rating in descending order to obtain the top ten.

The identical high ratings in the results may be due to a simplified dataset where only a few films have been rated, or each top film has been rated equally in the data insertion process.

Query 2 *- Find the highest rated film in each genre:*

```
276  -- Find the highest rated film in each genre:
277
278  WITH RankedGenres AS (
279    SELECT mg.genrecode, mg.genrename, m.title, r.rating,
280           RANK() OVER (PARTITION BY mg.genrecode ORDER BY r.rating DESC) AS rank
281    FROM (
282      SELECT mg.genrecode, mg.movieid, g.genrename
283      FROM movies_genres mg
284      JOIN genres g ON mg.genrecode = g.genrecode
285    ) mg
286    JOIN movies m ON mg.movieid = m.movieid
287    JOIN ratings100 r ON m.movieid = r.movieid
288  )
289  SELECT genrecode, genrename, title, rating
290  FROM RankedGenres
291  WHERE rank = 1;
```

Data Output | Messages | Notifications

	genrecode character varying (3)	genrename character varying (25)	title character varying (108)	rating numeric (2,1)
1	01	Drama	The Shawshank Redemption	9.3
2	02	Crime	The Godfather	9.2
3	03	Action	The Dark Knight	9.0
4	04	Mystery	12 Angry Men	8.9
5	05	Biography	Schindler's List	8.9
6	06	Adventure	The Lord of the Rings: The Return of the King	8.9
7	07	Thriller	Pulp Fiction	8.9
8	08	Western	The Good, the Bad and the Ugly	8.8
9	10	Comedy	Forrest Gump	8.8

This query uses a window function to rank films within each genre by their rating and selects the top film per genre. It involves joining ***'movies_genres'*** with ***'genres'*** and ***'movies'***, then utilizing the ***'ratings100'*** table for ratings, ordering by the highest rank within each genre partition.

The similar ratings across genres suggest that for this sample dataset, the top-rated film in each genre received the same rating, possibly due to a simplified insertion of data where these films were all rated equally.

Query 3 *- Finding the last film each customer watched:*

```
293  -- Finding the last film each customer watched and rated:
294
295  SELECT c.cfname, c.clname, m.title, MAX(r.rdate) AS last_watched
296  FROM customers c
297  JOIN ratings100 r ON c.custid = r.custid
298  JOIN movies m ON r.movieid = m.movieid
299  GROUP BY c.cfname, c.clname, m.title
300  ORDER BY last_watched DESC;
```

Data Output | Messages | Notifications

	cfname character varying (25)	clname character varying (25)	title character varying (108)	last_watched date
1	Benjamin	Hernandez	Forrest Gump	2023-10-05
2	Sophia	Martinez	Fight Club	2023-09-10
3	Ethan	Rodriguez	The Good, the Bad and the Ugly	2023-08-15
4	Olivia	Garcia	Pulp Fiction	2023-07-20
5	Lucas	Miller	The Lord of the Rings: The Return of the King	2023-06-30
6	Chloe	Davis	Schindler's List	2023-05-25
7	Michael	Brown	12 Angry Men	2023-04-18
8	Emily	Johnson	The Dark Knight	2023-03-22
9	Jane	Smith	The Godfather	2023-02-19
10	John	Doe	The Shawshank Redemption	2023-01-15

The query determines the most recent film watched and rated by each customer. It groups results by customer and film, then uses the '**MAX**' function on the rating date to find the latest watched film, ordering the output by this date in descending order.

The last-watched dates being the same for each customer can occur if the sample data was designed to have each customer rate their last film on the same date, which is an unlikely scenario in a real-world application.

Query 4 *- Find the average score of the films in which each actor appeared and rank the films with an average score higher than 8.0:*

```
302  -- Find the average score of the films in which each actor appeared and rank the films with an average score higher than 8.0:
303
304  SELECT a.a_fname, a.a_name, AVG(r.rating) AS average_rating
305  FROM actors a
306  JOIN movies_actors ma ON a.actorid = ma.actorid
307  JOIN ratings100 r ON ma.movieid = r.movieid
308  GROUP BY a.a_fname, a.a_name
309  HAVING AVG(r.rating) > 8.0
310  ORDER BY average_rating DESC;
```

Data Output | Messages | Notifications

	a_fname character varying (35)	a_name character varying (25)	average_rating numeric
1	Tim	Robbins	9.3000000000000000
2	Marlon	Brando	9.2000000000000000
3	Heath	Ledger	9.0000000000000000
4	Henry	Fonda	8.9000000000000000
5	Liam	Neeson	8.9000000000000000
6	John	Travolta	8.9000000000000000
7	Elijah	Wood	8.9000000000000000
8	Brad	Pitt	8.8000000000000000
9	Tom	Hanks	8.8000000000000000
10	Clint	Eastwood	8.8000000000000000

This query calculates the average rating of films for each actor and ranks them, provided the average is higher than 8.0. It groups the results by actor and uses the **'HAVING'** clause to filter for only those averages above 8.0, then orders by the average rating in descending order.

The uniform average ratings are indicative of a dataset where each actor is associated with films that have similar ratings, which is a result of the sample data being inserted with consistent values for simplicity.

Query 5 *- Finding the total number of awards won by a director's films:*

```
312  -- Finding the total number of awards won by a director's films:
313
314  SELECT d.director_first, d.director_last, COUNT(aw.awardid) AS total_awards
315  FROM directors d
316  JOIN movies_directors md ON d.directorid = md.directorid
317  JOIN movies_awards aw ON md.movieid = aw.movieid
318  GROUP BY d.director_first, d.director_last
319  ORDER BY total_awards DESC;
```

Data Output | Messages | Notifications

	director_first character varying (25)	director_last character varying (60)	total_awards bigint
1	Peter	Jackson	1
2	Quentin	Tarantino	1
3	Francis Ford	Coppola	1
4	David	Fincher	1
5	Christopher	Nolan	1
6	Frank	Darabont	1
7	Robert	Zemeckis	1
8	Steven	Spielberg	1
9	Sergio	Leone	1
10	Sidney	Lumet	1

The query tallies the total number of awards won by films directed by each director. It groups the results by director and counts the number of awards associated with their films, then orders by the total number of awards in descending order.

The result where each director's films have won the same number of awards suggests that the sample data was populated uniformly to demonstrate the functionality of the query rather than to reflect the varied nature of actual awards distribution.

Query 6 *- Finding the genre with the most films released in a year:*

```
321  -- Finding the genre with the most films released in a year:
322
323  SELECT g.genrename, COUNT(m.movieid) AS movie_count
324  FROM genres g
325  JOIN movies_genres mg ON g.genrecode = mg.genrecode
326  JOIN movies m ON mg.movieid = m.movieid
327  WHERE m.ryear = 1994  -- year example
328  GROUP BY g.genrename
329  ORDER BY movie_count DESC
330  LIMIT 1;
```

Data Output | Messages | Notifications

	genrename character varying (25)	movie_count bigint
1	Comedy	1

This query retrieves the genre with the highest number of movies released in a specific year, in this case, 1994. By joining the '***genres'*** and '***movies_genres'*** tables, and then joining with the '***movies'*** table, we can count the number of movies per genre, group the results by genre, and order them in descending order to find the most popular genre.

The result showing "Comedy" as the most popular genre with a count of "1" indicates that there is equal distribution across genres, and the sample data is limited, leading to a tie in movie counts for the year queried.

Query 7 *- For each genre, finding the customers who watch the most films in that genre:*

```
332  -- For each genre, finding the customers who watch the most films in that genre:
333
334  WITH GenreWatchCounts AS (
335    SELECT c.custid, mg.genrecode, COUNT(r.ratingid) AS watch_count
336    FROM customers c
337    JOIN ratings100 r ON c.custid = r.custid
338    JOIN movies_genres mg ON r.movieid = mg.movieid
339    GROUP BY c.custid, mg.genrecode
340  )
341  SELECT mg.genrename, c.cfname, c.clname, gwc.watch_count
342  FROM GenreWatchCounts gwc
343  JOIN genres mg ON gwc.genrecode = mg.genrecode
344  JOIN customers c ON gwc.custid = c.custid
345  ORDER BY mg.genrename, gwc.watch_count DESC;
```

Data Output | Messages | Notifications

	genrename character varying (25)	cfname character varying (25)	clname character varying (25)	watch_count bigint
1	Action	Emily	Johnson	1
2	Adventure	Lucas	Miller	1
3	Biography	Chloe	Davis	1
4	Comedy	Benjamin	Hernandez	1
5	Crime	Jane	Smith	1
6	Drama	John	Doe	1
7	Mystery	Michael	Brown	1
8	Thriller	Sophia	Martinez	1
9	Thriller	Olivia	Garcia	1
10	Western	Ethan	Rodriguez	1

This query determines which customers watch the most films in each genre. It uses a common table expression to first calculate the watch count per customer and genre. Then, it joins the result with the '***genres'*** and '***customers'*** tables to present a comprehensive list, ordered by genre and watch count in descending order.

The uniform watch count across customers suggests that the sample data inserted into the database has equal distribution, which may not reflect a real-world scenario where customers have diverse viewing habits.

Query 8 *- Finding a list of films that have won at least one award and the actors who played in these films:*

```
347  -- Finding a list of films that have won at least one award and the actors who played in these films:
348
349  SELECT DISTINCT m.title, a.a_fname, a.a_name
350  FROM movies m
351  JOIN movies_awards ma ON m.movieid = ma.movieid
352  JOIN movies_actors mac ON m.movieid = mac.movieid
353  JOIN actors a ON mac.actorid = a.actorid;
```

Data Output | Messages | Notifications

	title character varying (108)	a_fname character varying (35)	a_name character varying (25)
1	12 Angry Men	Henry	Fonda
2	Fight Club	Brad	Pitt
3	Forrest Gump	Tom	Hanks
4	Pulp Fiction	John	Travolta
5	Schindler's List	Liam	Neeson
6	The Dark Knight	Heath	Ledger
7	The Godfather	Marlon	Brando
8	The Good, the Bad and the Ugly	Clint	Eastwood
9	The Lord of the Rings: The Return of the King	Elijah	Wood
10	The Shawshank Redemption	Tim	Robbins

The query lists films that have won at least one award and the actors who starred in these films. It connects the ***'movies'***, ***'movies_awards'***, ***'movies_actors'***, and ***'actors'*** tables to compile a distinct list of films, leading actors' first and last names, ensuring that only films with awards are included.

The equal distribution of awards among films with actors listed once may result from the sample data being too homogenous, indicating that each film in the dataset has received an award and each actor has played in one award-winning film.

Query 9 *- Find films that have won an award and have a score higher than 9:*

```
355  -- Find films that have won an award and have a score higher than 9:
356
357  SELECT m.title, a.awardtitle, r.rating
358  FROM movies m
359  JOIN movies_awards ma ON m.movieid = ma.movieid
360  JOIN awards a ON ma.awardid = a.awardid
361  JOIN ratings100 r ON m.movieid = r.movieid
362  WHERE r.rating > 9;
```

Data Output | Messages | Notifications

	title character varying (108)	awardtitle character varying (100)	rating numeric (2,1)
1	The Shawshank Redemption	Best Picture	9.3
2	The Godfather	Best Director	9.2

This query finds films that have not only won awards but also have a high rating, specifically greater than 9. It joins the *'**movies**'*, *'**movies_awards**'*, *'**awards**'*, and *'**ratings100**'* tables to select the movie titles, award titles, and ratings, filtering for those with ratings above 9.

The output indicates that each film listed has a high rating above 9, which can be attributed to the small sample dataset where only highly-rated films are selected for awards, or the insertion of data was done in a way to highlight only the top-rated films.

Query 10 *- Finding the total duration of the films watched by each customer (By default, I assume film durations of 120 minutes):*

```
364  -- Finding the total duration of the films watched by each customer (By default, I assume film durations of 120 minutes):
365
366  SELECT c.cfname, c.clname, COUNT(r.ratingid) * 120 AS total_minutes_watched
367  FROM customers c
368  JOIN ratings100 r ON c.custid = r.custid
369  GROUP BY c.cfname, c.clname;
```

Data Output | Messages | Notifications

	cfname character varying (25)	clname character varying (25)	total_minutes_watched bigint
1	Chloe	Davis	120
2	Ethan	Rodriguez	120
3	Olivia	Garcia	120
4	Sophia	Martinez	120
5	John	Doe	120
6	Michael	Brown	120
7	Emily	Johnson	120
8	Benjamin	Hernandez	120
9	Lucas	Miller	120
10	Jane	Smith	120

The query calculates the total duration of films watched by each customer, assuming a default duration of 120 minutes per film. By joining the ***'customers'*** and ***'ratings100'*** tables, it aggregates the number of films watched by each customer and multiplies by the assumed duration to estimate total minutes watched.

The constant value of 120 minutes for each customer implies that the sample data was inserted to reflect each customer having watched exactly one movie, assuming a standard movie length of 120 minutes.