Group 5: ~~samima Hassan,~~

~~Defene sahal , Zeynep dilay~~

online movie system

Users table: users are identified uniquely by **users_id** and every user has their unique id which automatically updated by the big-serial of PostgreSQL

Users table has the attributes

- Name which is a composite attribute. It's the combination of **first name** and **last name**
- Country
- Password
- Email

Subscriptions table: subscription table is uniquely identified by the **sub_id** and every subscriber has a unique id, subscription table has **1:N relationship with payment** table and **payment id** in the subscription is a foreign key reference **to payment table** id to check if the user has paid money for the subscription, the other attributes of the subscription table

- Price
- Subscription date
- Subscription type
- Payment id --→a foreign key reference to the payment table

user_subscription

Users have **N: M** relationship with subscription table which is held in a new table called **user_subscription** the table that holds the id of users and sub_id of from subscription table to identify those users who have a subscription. User_subsrciption uniquely identified by the **user_id** and **sub_id**

Movie table: movie table is uniquely identified by the **movie_id** which is automatically given by the Postgresql big-serial domain the other attribute of the movie are

- Title → name of the movie
- Country → which country made the movie
- Duration → time the movie takes to watch
- Year release → the year movie came out to the cinema
- Price →the price of the movie

Movie table has **N: M** relationship with users which is kept in <**rating**> where **user_id** and **movie_id** the unique identifier and this relationship has 2 more attributes

- Year_rate → what was the date when the user rated the movie
- Rate_star → the star user decided to give the movie

Rating relationship table keeps the record of the rating users gives to the movies

Movie table has **N: M** relationship with users which is kept in <**user_movie >** where **user_id** and **movie_id** is the unique identifier this relationship has one more attribute

- Purchase_date → the date when the user bought the movie

User_movie relationship table keeps the track of movies which are purchased by the users

Actor: actor uniquely identified by the **actor_id**, each actor has its unique id which is updated automatically by the big-serial of the Postgresql, the other attributes of the actor are

- Name → which combination of first name and last name
- Country → where the actor or actress is from
- Gender → which identifies whether the player is male or female

By adding the gender part in an actor we didn't need to create another **entity for actress**

actor_movie:

the actor has N: M relationship with the movie table which is **actor_movie**, this relationship uniquely identified by the actor_id and movie_id. which are foreign keys from movie and actor entities

This relationship keeps the record of the movies which actor worked in.

Genre: uniquely identified by the **genre_id**, the only other attribute of the genre entity is genre type

Genre_movie:

Genre has N: M relationship with movie entity which movie_genre, this relationship uniquely identified by the genre_id and movie_id.

Genre_movie relationship keeps the track what are the genre of the movie and which movie In a genre

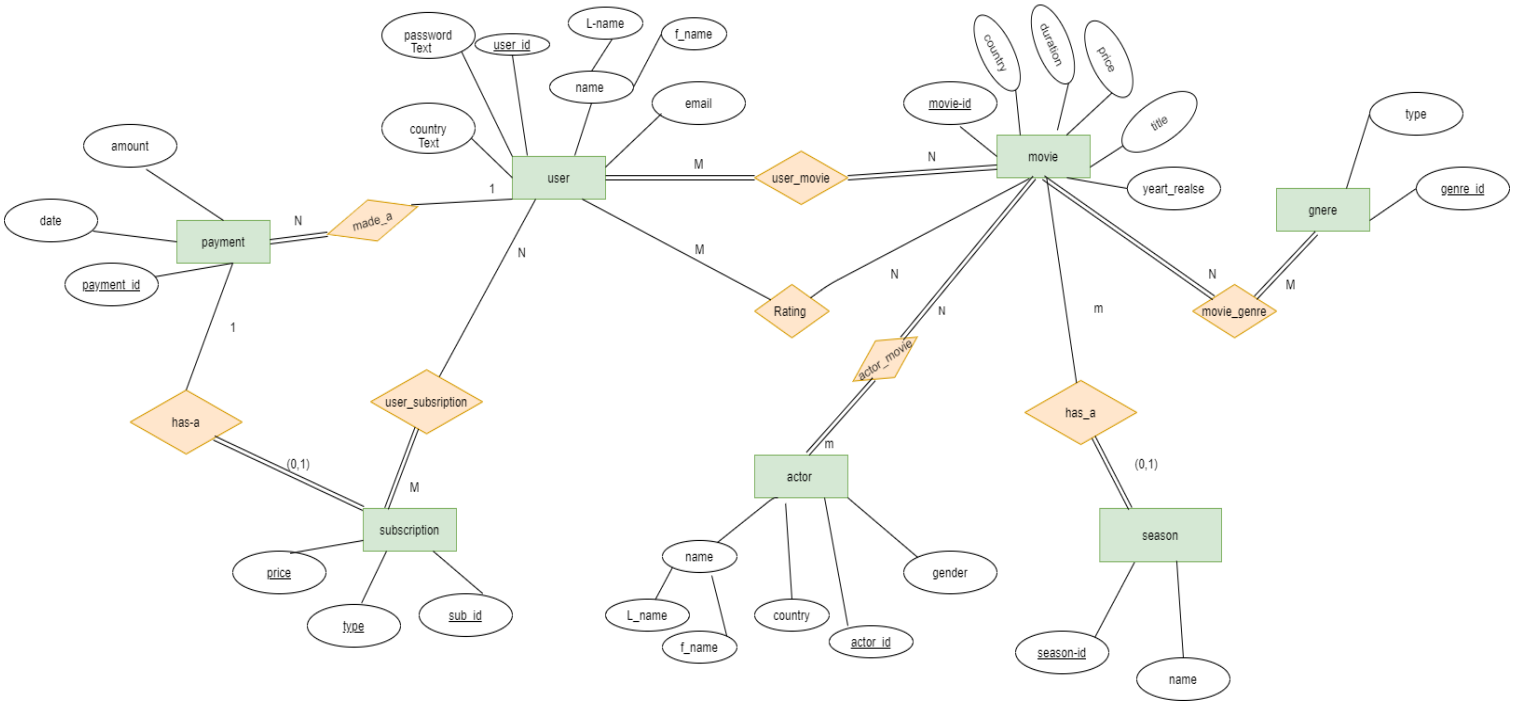
Season uniquely identified by the season_id, another attribute is name

The season has **1:N** relationship with movie

Payment uniquely identified by the payment id other attributes of the payment are

- Amount → amount paid by the user
- Date → the date payment was made

Payment has a made_**a 1:N r**elationship with the user



ER DIAGARAM :

REDUCTION :

Phases of reduction

Phase 1: Strong entities

user

user_id	F_name	L-name	email	password	country

Movie

movie-id	title	country	language	duration	year_realse	price

Gnere

Genre_id	type

actor

actor_id	f_name	L_name	country

actress

actress_id	f_name	L_name	country

subscription

sub_id	type	price

payment

payment_id	amount	date

season

season_id	name

Phase 2: weak entities

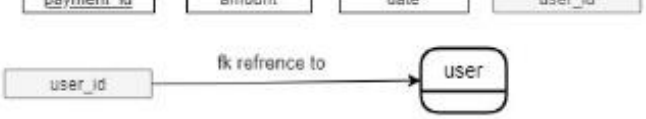
no weak entities

Phase 3 : 1:1 Relationship

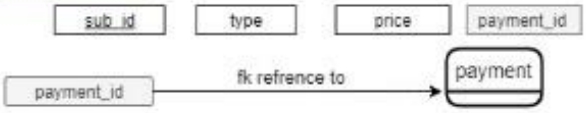
no 1:1 relationship

Phase 4: 1:N relationship

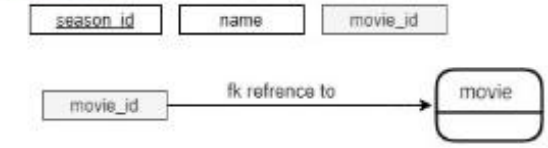
payment



subscription

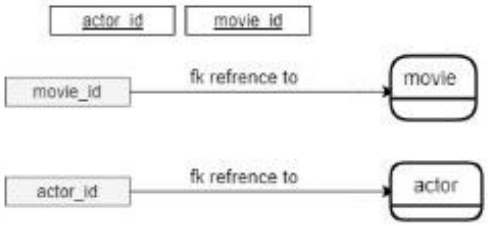


season

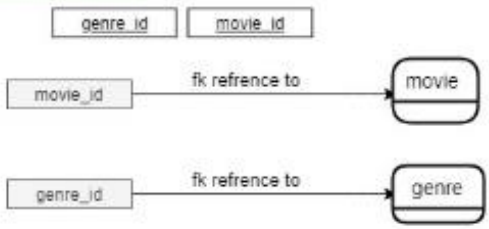


Phase 5: M:N relationship

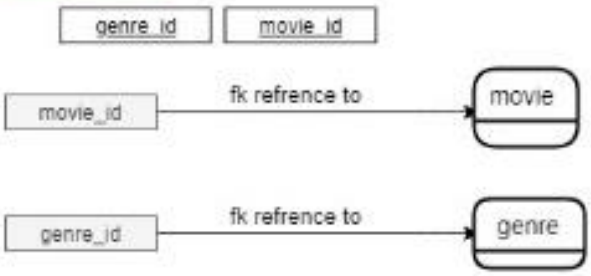
actor_movie



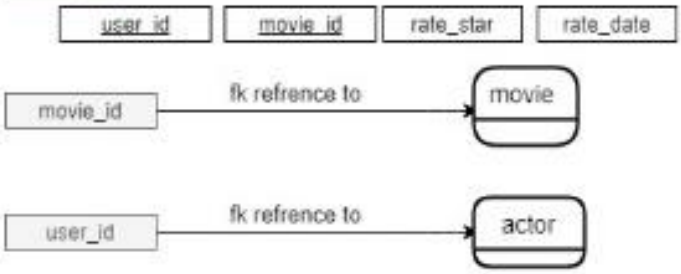
movie_genre



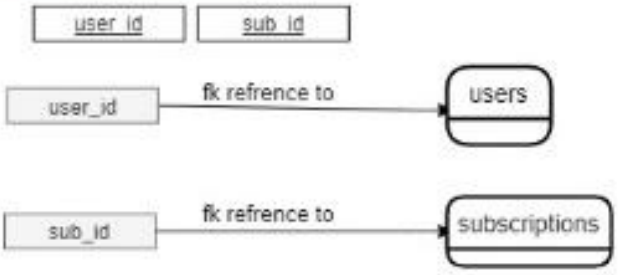
movie_genre



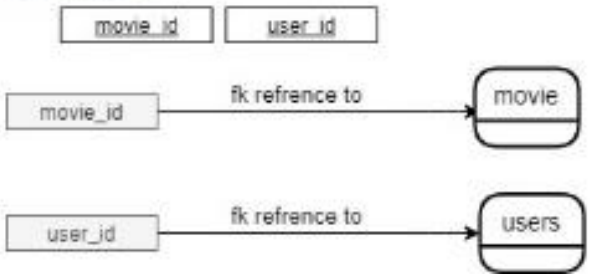
rating



usert_subscription



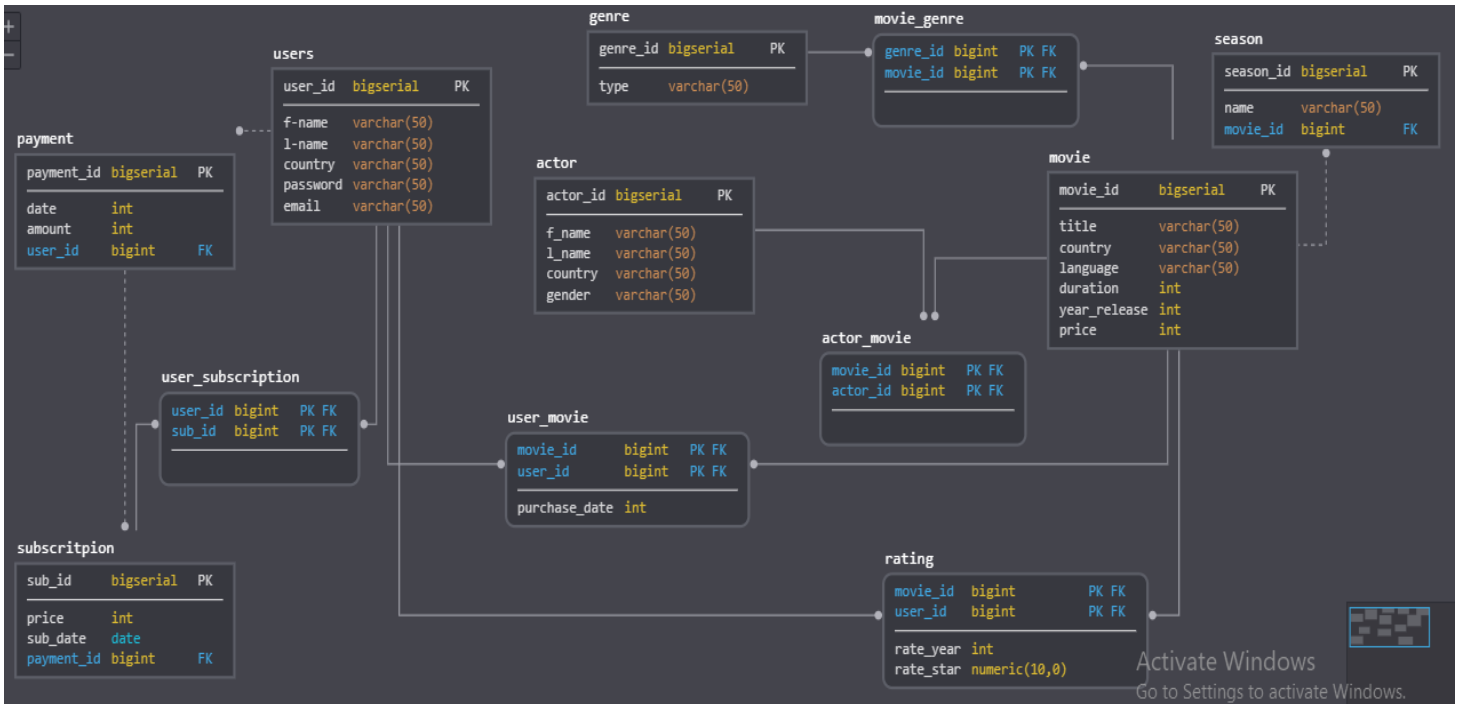
user_movie



Blue colour → Foreign_key

Final schema :

Users	***User_id***	F_name	L_name	country	password	Email
Movie	***Movie_id***	title	country	duration	Year_realse	Price
Subscription	***Sub_id***	amount	Sub_date	Sub_type		
Payment	***Payment_id***	amount	Payment_date	User_id		
Genre	***Genre_id***	Genre_type				
Actor	***Actor_id***	F_name	L_name	Country		
Season	***Season_id***	name	Movie_id			
Actor_movie	***Movie_id***	***Actor_id***				
User_movie	***User_id***	***Movie_id***	Purchase_date			
Genre_movie	***Genre_id***	***Movie_id***				
rating	***User_id***	***Movie_id***	Rate_date	Rate_star		
User_subsription	***User_id***	***Sub_id***				



Changes made from phase one :

1. The trending movie table was dropped due to redundancy

2. attribute gender is added to the actor table where we can recognize the gender of the actor

3. table rating is added where the user can rate the movie

4. table payment is added where user make payment for subscription and movie

4. table season is added where we can see which season the movie belong to

5. script has been changed and updated

JAVA PROJECT REPORT

Java project is made up of 3 parts for 3 tables, movie, actor, and user. To make changes or updates in any of these tables the user must enter the first letter of the table, for example, m for the movie a for actor and u for the user.

If the users wish to make changes in the movie then there will be another menu with 8 options and the user must enter the number for the corresponding task for example

1. to see all the movies the user must enter 1 by keyboard, the list of all movies with their id, price and name will appear of the console.

2. For adding the movie to movie table user must enter 2 and the application will ask the user to enter the title, duration, price, year realise and price of the movie.

3. For deleting a movie the user needs to enter the id of the movie.

4. Option 4 is to see movies from a specific country so the user has to enter the name of the country and will all the movie from that country.

5.option 5 is for rating the use will be asked to enter user_id, movie_id, year, and the rate from 1 to 10

Which how much rate they want to give,

6. options 6 is to see all the movie where the price is less than 100

7. option 7 is if the user wants to see the genre of the movie

8. option 8 is for the user if want to update the price of the movie

If the user chooses the actor part then the menu of the actor part will appear on the console which is consist of the 4 options.

1. option 1 is for viewing all actors with the movies they have worked

2. option 2 is for seeing a movie from a specific country

3. option 3 is for adding actress to the list and the application will ask for first name, last name, country and gender

4. option 4 is for to delete and actor and the user must enter the id of the actor to be deleted

The last part is for the user table part and it consists of the following tasks

1. option first of adding a user to the users' account, the application will ask for first name, last name, country,

Email, password

2. option to add into subscriptions users, the user must choose one package between these: weekly, monthly, annual. the price for these packages is already declared.

3. option3 is for list of all users who have a subscription