CMPE321

Introduction to Database Systems

Project #1

Designing a Movie Database

Student Name : Beyza Akçınar

Student ID : 2019400156

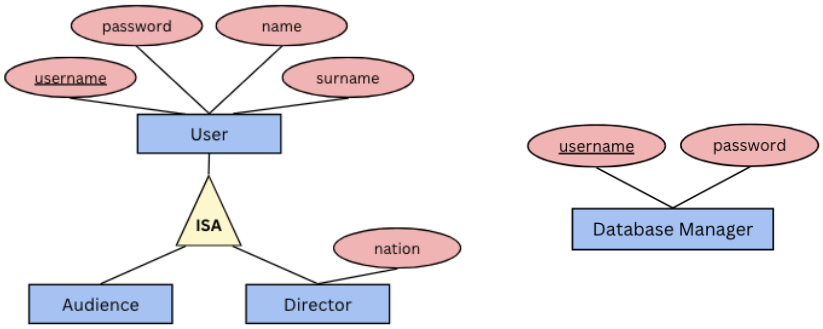
Semester : Spring 2023

1 Conceptual Design

1.1 ER Diagram

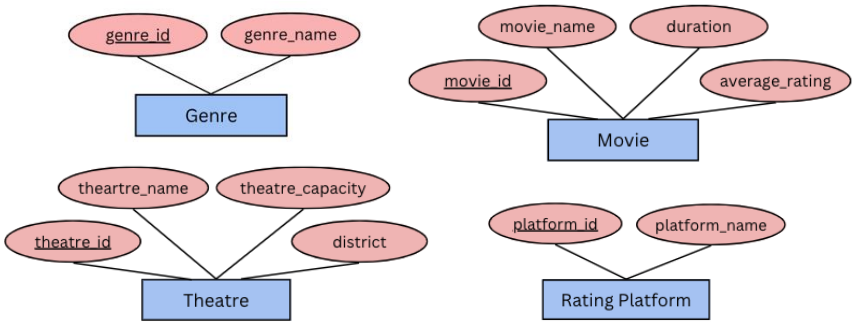
1.1.1 Entities

I started the diagram by modeling the main entities for the database.



Audience and Director COVER User

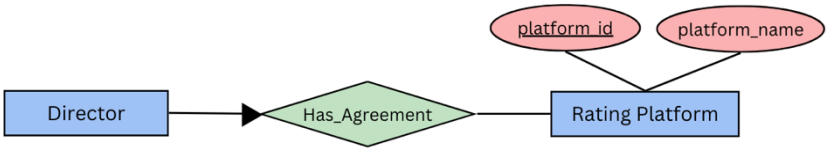
A user is either an audience or a director, meaning that audience and director entity sets cover all users and there is no overlap between them. It is important to note that the constraint that every director should have one nation cannot be satisfied in this ER design.



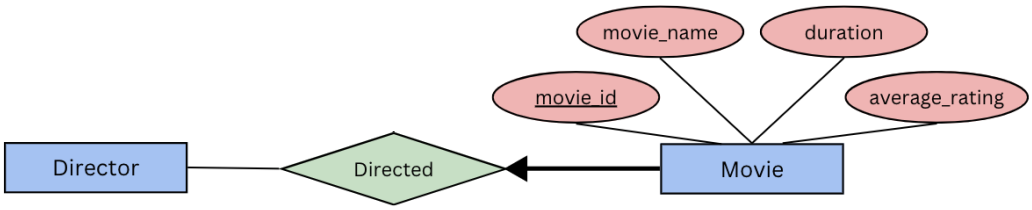
Since I do not want the database to store any redundant data, I decided to group all related information together as much as I can.

1.1.2 Relationships

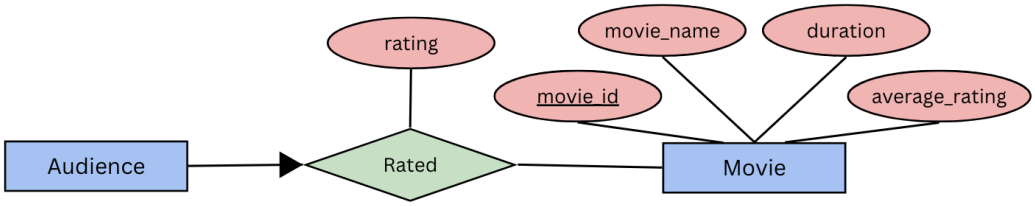
After deciding on entities, I tried to figure out the possible relations between them.



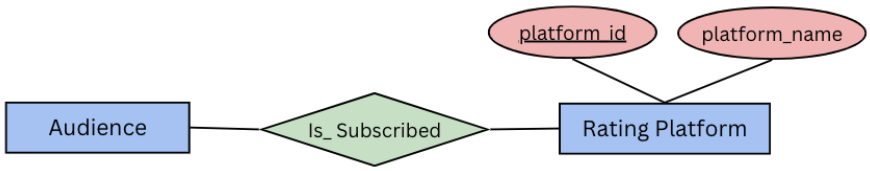
A director can have an agreement with only one rating platform, and this is ensured by the key constraint.



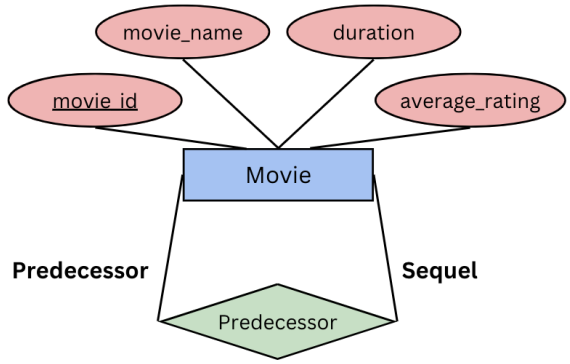
One director can have many movies, but a movie is directed by only one director. Since the platform of the movie is the same as its director's, the platform information of a movie is already accessible through its director.



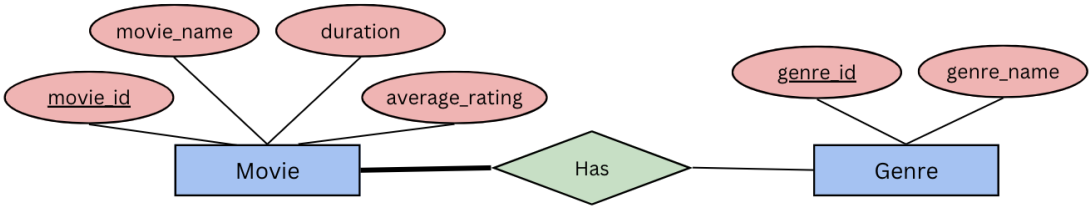
An audience can rate one movie only once.



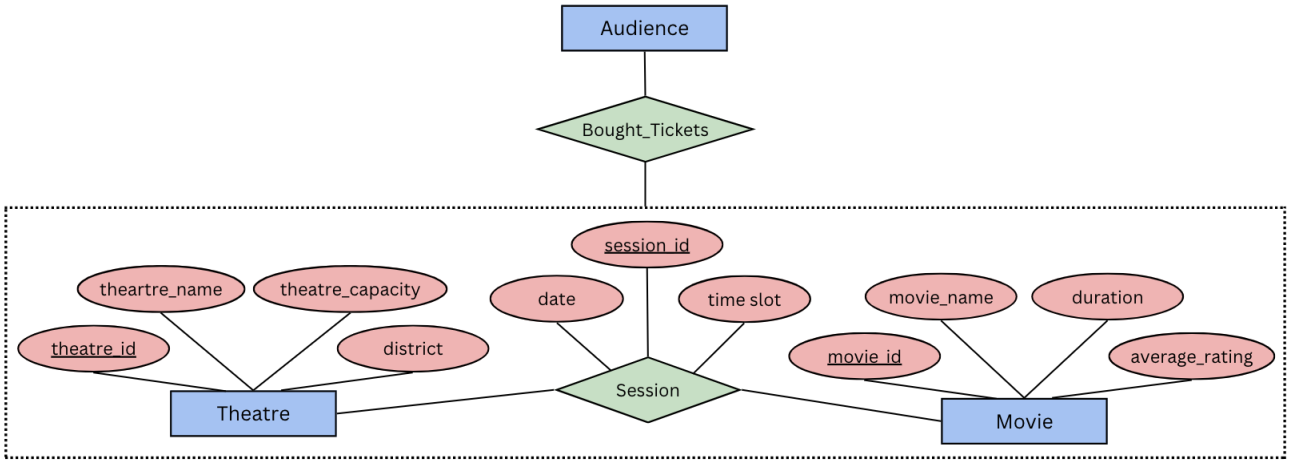
An audience can subscribe to zero or more Rating Platforms, and a rating platform can have zero or more subscribers.



A movie can have zero or more predecessors or sequels.



A movie has at least one genre, this is ensured by the total participation constraint.



An audience can buy tickets to movie sessions. Since I modeled a movie session to be a relationship, to create the Bought_Tickets relationship, I used aggregation property.

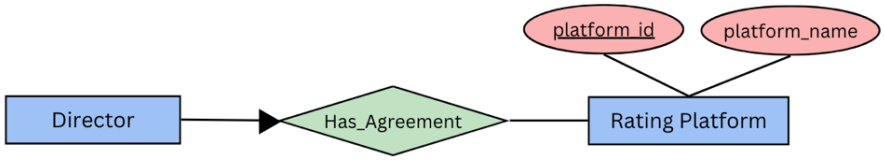
2 Logical Design

2.1 Relations

2.1.1 Schemas

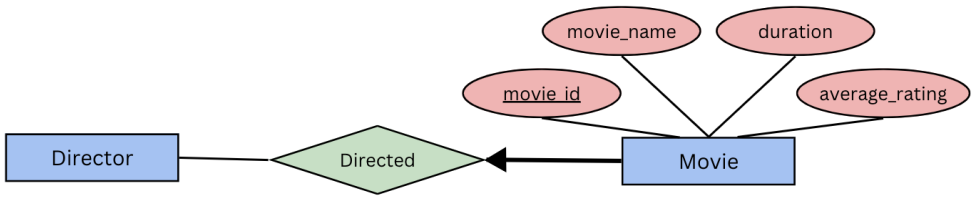
Some entities are directly translated into relations with their corresponding attributes.

- *Audience(username: string, password: string, name: string, surname: string)*
- *Database_Manager(username: string, password: string)*
- *Rating_Platform(platform_id: integer, platform_name: string)*
- *Theatre(theatre_id: integer, theatre_name: string, theatre_capacity: integer, district: string)*
- *Movie(movie_id: integer, movie_name: string, duration: integer, average_rating: float)*
- *Genre(genre_id: integer, genre_name: string)*

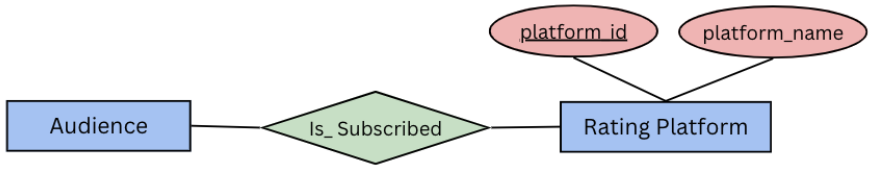


Since every director has an agreement with only one rating platform, I decided to keep this information under director relation.

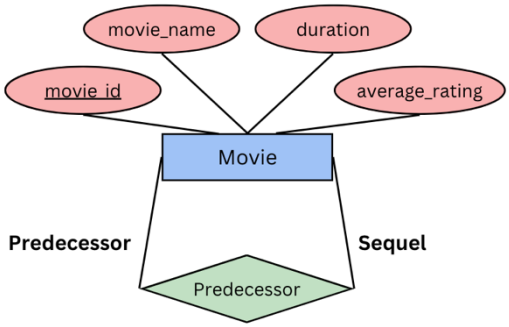
- *Director(username: string, password: string, name: string, surname: string, nation: string, platform_id: integer)*



- *Directed_By(username: string, movie_id: integer)*

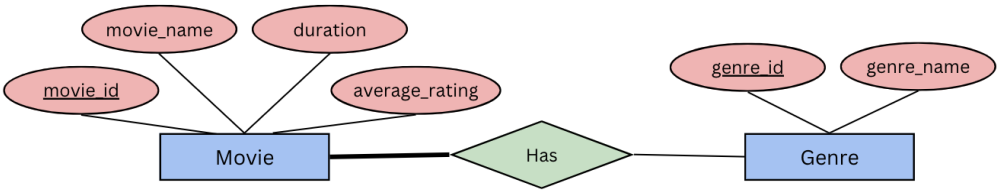


- *Subscription(username: string, platform_id: integer)*



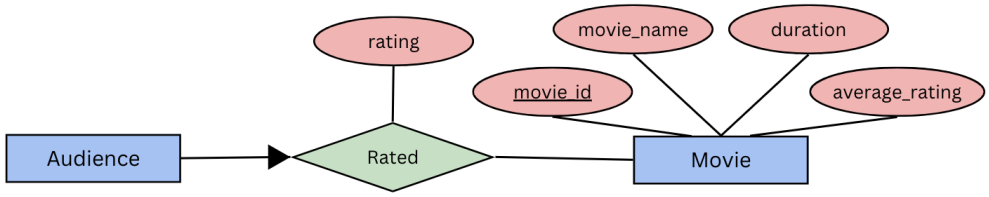
This relationship is converted to a relation with both movie_ids of the predecessor and the sequel movies.

- *Predecessor(sequel_id: integer, predecessor_id: integer)*



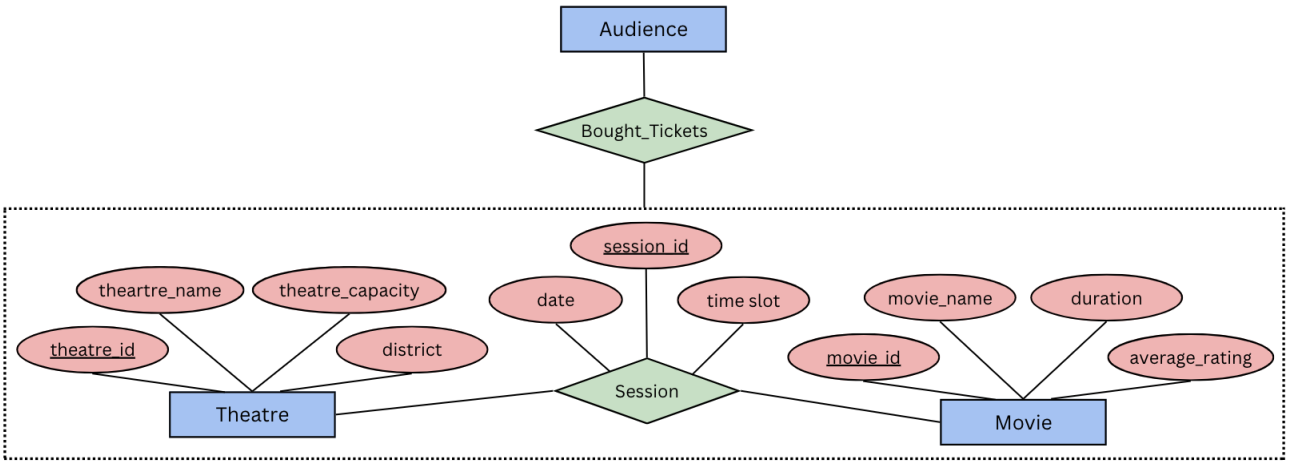
Every movie has at least one genre.

- *Genre_List(movie_id: integer, genre_id: integer)*



A user can rate a movie only once.

- *Rating(username: string, movie_id: integer, rating: float)*



This aggregation can be converted to two relations. Session_id is also unique, but by having time_slot, date, and theatre_id attributes as composite key for session, it is possible to forbid a session entry that has the exact same time and location. However, since ticket relation has username and session_id as composite key, is it only possible for an audience to buy a ticket on his/her own name.

- *Session(session_id: integer, movie_id: integer, theatre_id: integer, date: date, time_slot: integer)*
- *Ticket(username: string, session_id: integer)*

2.1.2 Example Data

(The dataset provided is not complete, only few instances are shown)

username	password	name	surname
alan.turing	tmachine	Alan	Turing
bakcinar	12345	Beyza	Akçınar
steve.jobs	apple123	Steven	Jobs

theatre_id	theatre_name	theatre_capacity	district
123	Şişli_1	340	Şişli
134	Beşiktaş_1	450	Beşiktaş

username	password
manager1	managerpass1
manager2	managerpass2

platform_id	platform_name
80022	HBO
80234	Netflix
81226	IMDB

genre_id	genre_name
12082	thriller
12973	action
23784	horror

director

username	password	name	surname	nation	platform_id
chris.nolan	inception	Christopher	Nolan	British	13
oriol.paulo	plottwist	Oriol	Paulo	Spanish	12

rating

username	movie_id	rating
bakcinar	3498	5
steve.jobs	3864	5

subscription

username	platform_id
bakcinar	80234
bakcinar	81226
steve.jobs	80234

directed_by

username	movie_id
chris.nolan	3498
oriol.paulo	3495
oriol.paulo	3864

genre_list

movie_id	genre_id
3498	12082
3498	12973

movie_id	movie_name	duration	average_rating
3200	Frozen	108	4.2
3403	Frozen 2	103	4
3495	El Cuerpo	110	4.6
3498	Tenet	150	4.8
3708	X-Men First Class	132	4.6
3812	X-Men Days of Future Past	132	4.7
3864	Contratiempo	106	4.7

session

session_id	movie_id	theatre_id	date	time_slot
60225	3498	123	2023-02-01	2
60713	3498	134	2023-01-04	3

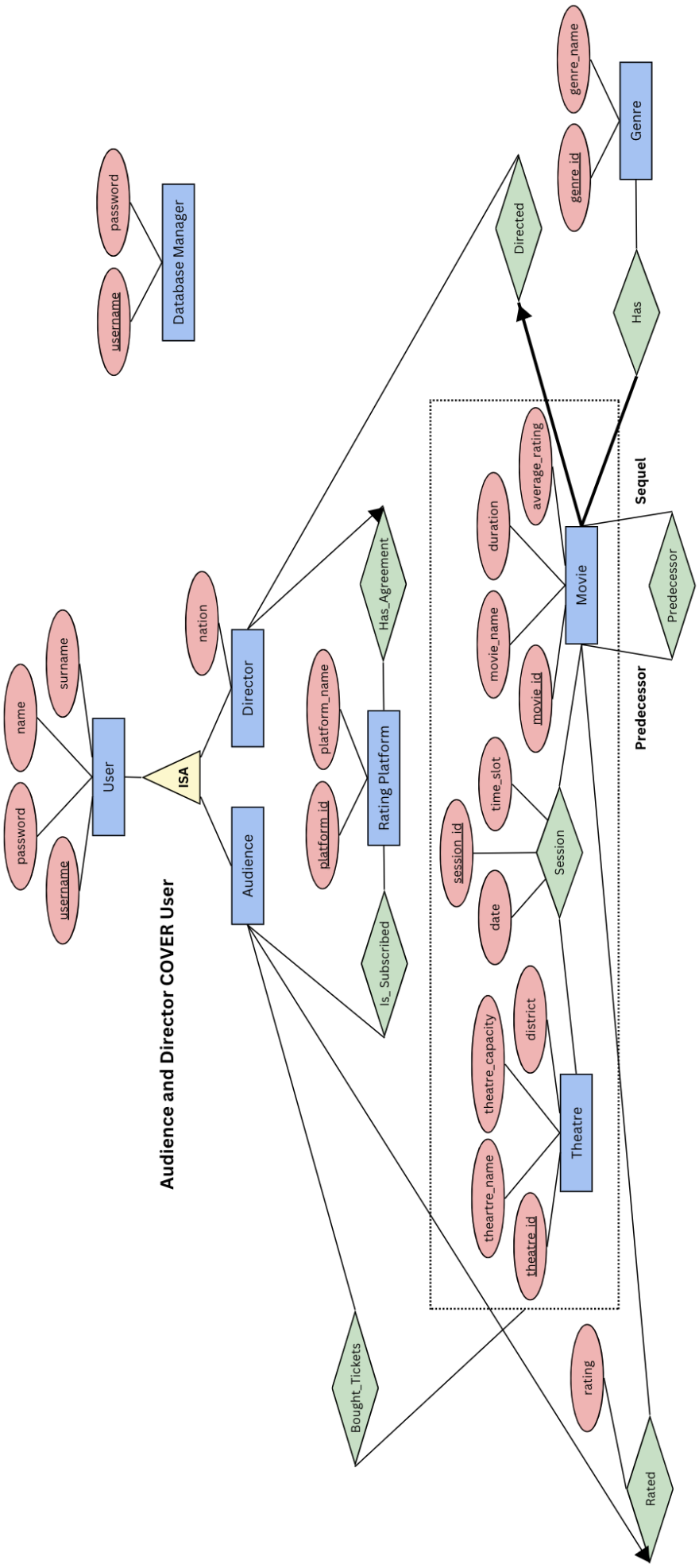
predecessor

movie_id1	movie_id2
3403	3200
3812	3708

ticket

username	session_id
alan.turing	60713
bakcinar	60713

Full ER Diagram



3 Additional Notes

Since this project only requires the design of the database tables, there are constraints given that cannot be captured by the limited rules available.

- Booking mechanism is only valid for 1 week.
- There are four time slots for each day.
- The movie starts at the given time slot and the ending time is determined by its duration. (This design makes sure that starting time of the movies do not overlap, but without taking the duration of the movie into account.)
- All predecessors should be watched before watching a movie.
- A user can rate a movie if they are already subscribed to the platform that the movie can be rated and bought a ticket for the movie.
- There can be at most 4 database managers registered to the system.