CMPE321
PROJECT 1 REPORT

HASAN BAKİ KÜÇÜKÇAKIROĞLU - 2018400141
ALP TUNA – 2019400288

***Read.me file is located in the MovieDB directory.**

Logical Database Design

DatabaseManagers(username: string, password_: string)

RatingPlatforms(platform_id: string, platform_name: string)

Genres(genre_id: string, genre_*name: string);*

Audiences(username: string, password: string, name*: string, surname: string)*

Directors(username: string, password: string, name_: string, surname: string, nationality: string, platform_id: string)

Movies(movie_id: string, movie_name: string, duration: integer, director_username: string, average_rating: real)

MovieHasGenres(movie_id: string, genre_id: string)

Subscribes(username: string, platform_id: string)

Rates(username: string, movie_id: string, rating)

Theaters(theater_id: string, theater_name: string, theater_capacity: integer, theater_district: string)

MovieSessions(session_id: string, movie_id: string, theater*id: string, date*: date, time_slot: integer)

BoughtTickets(username: string, session_id: string)

MoviePrerequisites(movie_id_predecessor: string, movie_id_successor: string)

Schema Refinement Steps

To check if the relation is in Boyce-Codd Normal Form (BCNF), we need to ensure that for every non-trivial FD (X → Y), X is a key or a superkey. A key or a superkey is a set of attributes that uniquely identifies a tuple in the relation.

1. *DatabaseManagers(username, password_)*

Non-trivial Functional Dependencies:

- *username → username, password_*

Based on: “There exists only one database manager with a certain username.”

Since the username is the primary key of the relation, it is a key. Therefore, ***DatabaseManagers*** relation satisfies BCNF.

2. *RatingPlatforms(platform_id, platform_name)*

Non-trivial Functional Dependencies:

- *platform_id → platform_id, platform_name*
- *platform_name → platform_id, platform_name*

Based on: “Both name and id must be unique.”. We picked platform_id as the primary_key.

Since the platform_id is the primary key and platform_name is candidate key for the relation, it is a key. Therefore, ***RatingPlatforms*** relation satisfies BCNF.

3. *Genres(genre_id, genre_name);*

Non-trivial Functional Dependencies:

- *genre_id→ genre_id, genre_name*
- *genre_name→ genre_id, genre_name*

Based on: “Both name and id must be unique.”. We picked genre_id as the primary_key.

genre_id is key and genre_name is candidate key for the relation, therefore ***Genres*** relation is in BCNF.

4. ***Audiences(<u>username</u>, password*, name*, surname)***

Non-trivial Functional Dependencies:

- *username → username, password, name, surname*

Based on : "Each user has a unique username and each user is either an audience or a director."

username is the primary key of the relation and we cannot generate any functional dependency from those in which X is not a key or a superkey, therefore the ***Audiences*** relation is in BCNF.

5. ***Directors(<u>username</u>, password, name_, surname, nationality, platform_id)***

Non-trivial Functional Dependencies:

- *username → username, password, name_, surname, nationality, platform_id*

Based on: "Each user has a unique username and each user is either an audience or a director."

username is the primary key of the relation and we cannot generate any functional dependency from those in which X is not a key or a superkey, therefore ***Directors*** relation is in BCNF.

6. ***Movies(<u>movie_id</u>, movie_name, duration, director_username, average_rating)***

Non-trivial Functional Dependencies:

- *movie_id→ movie_name, duration, director_username, average_rating*

Based on: "Movie id determines the movie name, duration, genre list, overall rating, director username, and platform id.",

movie_id is the primary key of the relation and we cannot generate any functional dependency from those in which X is not a key or a superkey, therefore ***Movies*** relation is in BCNF.

7. ***MovieHasGenres(<u>movie_id</u>, <u>genre_id</u>)***

Functional Dependencies:

movie_id, genre_id → movie_id, genre_id

Based on: “Every movie needs to have at least one genre.”

This is the only dependency and it is trivial, therefore ***MovieHasGenres*** relation is in BCNF.

8. ***Subscribes(<u>username</u>, <u>platform_id</u>)***

Functional Dependencies:

- username, platform_id → username, platform_id (This is the only dependency and it is trivial, therefore ***Subscribes*** relation is in BCNF.)

Based on: “Audience can subscribe to different rating platforms such as IMDB and Letterboxd.

9. ***Rates(<u>username</u>, <u>movie_id</u>, rating)***

Non-trivial Functional Dependencies:

- *username, movie_id→ username, movie_id, rating*

Based on: “A user can rate the same movie only once.”

username and movie_id is the primary key of the relation and we cannot generate any functional dependency from those in which X is not a key or a superkey, therefore ***Rates*** relation is in BCNF.

10. ***Theaters(<u>theater_id</u>, theater_name, theater_capacity, theater_district)***

Non-trivial Functional Dependencies:

- *theater_id → theater_id, theater_name, theater_capacity, theater_district*

Based on: “Each theater id corresponds to a physical location. Hence, theater capacity and theater district depend solely on the theater id.”

theater_id is the primary key of the relation and we cannot generate any functional dependency from those in which X is not a key or a superkey, therefore ***Theaters*** relation is in BCNF.

11. ***MovieSessions(<u>session_id</u>, movie_id, theater_id, date, time_slot)***

Non-trivial Functional Dependencies:

Based on: “The session id must be unique.”

- *session_id → session_id, movie_id, theater_id, date, time_slot*
- *theater_id, date, time_slot → session_id, movie_id, theater_id, date, time_slot*

session_id is the primary key of the relation and *theater_id, date, time_slot are another candidate key.* We cannot generate any functional dependency from those in which X is not a key or a superkey, therefore ***MovieSessions*** relation is in BCNF.

12. *BoughtTickets(username, session_id)*

Functional Dependencies:

username, session_id → username, session_id

This is the only dependency and it is trivial, therefore ***BoughtTickets*** relation is in BCNF.

13. *MoviePrerequisites(movie_id_predecessor, movie_id_successor)*

Functional Dependencies:

movie_id_predecessor, movie_id_successor → movie_id_predecessor, movie_id_successor

This is the only dependency and it is trivial, therefore ***MoviePrerequisites*** relation is in BCNF.

Constraints Captured Using Triggers

1. **insertRating:** This trigger runs before inserting the rating in order to ensure that a user can rate a movie only once.

2. **updateAverageRating:** This trigger runs after inserting the rating in order to calculate the average rating of the movie and update its value accordingly.
3. **InsertDatabaseManager:** This trigger runs before inserting into databaseManagers table in order to prevent more than 4 database managers in the system.

4. **checkSubscriptionBeforeRating:** This trigger runs before inserting the rating and checks whether the audience has already subscribed to the platform of the movie that s/he is trying to rate.

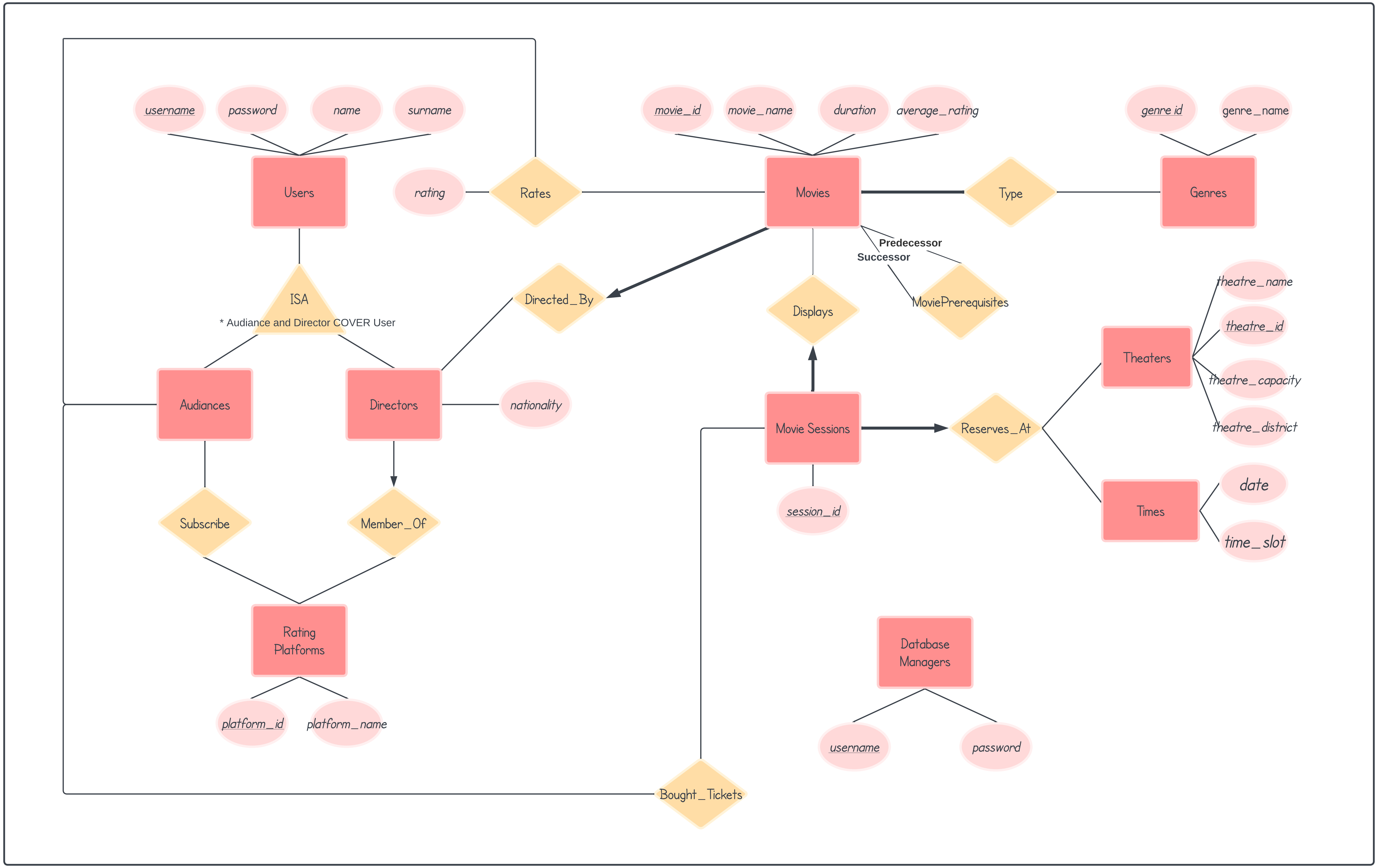
5. **checkPredecessorMovies:** This trigger runs before inserting into boughtTickets table in order to ensure that user has already watched the prerequisite movies.

6. **checkTicketBeforeRating:** This trigger runs before inserting the rating in order to ensure that user has bought a ticket for the movie s/he is trying to rate.

7. **checkCapacityBeforeBuyingTicket:** This trigger runs before user buys a ticket in order to ensure that theater capacity is not exceeded.

8. **checkMovieDuration:** This trigger runs before inserting a movie and checks its duration. Its duration must be between 1 and 4.

9. **checkRateLimit:** This trigger runs before inserting a rating and checks its value. Its value must be between 0 and 5.


username
password
name
surname
Users
rating
Rates
movie_id
movie_name
duration
average_rating
Movies
Type
genre id
genre_name
Genres
ISA
* Audiance and Director COVER User
Directed_By
Predecessor
Successor
Displays
MoviePrerequisites
theatre_name
theatre_id
theatre_capacity
theatre_district
Theaters
Audiances
Directors
nationality
Movie Sessions
Reserves_At
session_id
Times
date
time_slot
Subscribe
Member_Of
Rating Platforms
platform_id
platform_name
Database Managers
username
password
Bought_Tickets