CMPE321

Introduction to Database Systems

Project #3

MovieDB System

Student Name : Beyza Akçınar

Student ID : 2019400156

Semester : Spring 2023

1 Schema Refinement

Most of the tables I created in the first project were already in BCNF form.

- *Audience(username: string, password: string, name: string, surname: string)*

U->UPNS
Since all fields are dependent on username field, and it is the primary key of the relation, it is in BCNF form.

- *Database_Manager(username: string, password: string)*

U->UP
Since all fields are dependent on username field, and it is the primary key of the relation, it is in BCNF form.

- *Rating_Platform(platform_id: integer, platform_name: string)*

I->IN
Since all fields are dependent on platform_id field, and it is the primary key of the relation, it is in BCNF form.

- *Theatre(theatre_id: integer, theatre_name: string, theatre_capacity: integer, district: string)*

I->INCD
Since all fields are dependent on theatre_id field, and it is the primary key of the relation, it is in BCNF form.

- *Genre(genre_id: integer, genre_name: string)*

I->IN
Since all fields are dependent on genre_id field, and it is the primary key of the relation, it is in BCNF form.

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

U->UPNSTP
Since all fields are dependent on username field, and it is the primary key of the relation, it is in BCNF form.

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

UP->UP
Since username and platform_id fields are the primary key of the relation, it is in BCNF form.

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

MP->MP
Since movie_id and predecessor_id fields are the primary key of the relation, it is in BCNF form.

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

MG->MG
Since movie_id and genre_id fields are the primary key of the relation, it is in BCNF form.

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

UM->UMR
Since username and movie_id fields are the primary key of the relation, it is in BCNF form.

- *Ticket(username: string, session_id: integer)*

US->US
Since username and session_id fields are the primary key of the relation, it is in BCNF form.

There were some changes in some tables.

Since there is only one director for each movie, and it is better to have less tables, director_username field is added to movie table in the new design.

- *Movie(movie_id: integer, movie_name: string, duration: integer, average_rating: float)*
- *Directed_By(username: string, movie_id: integer)*

- *Movie(movie_id: integer, movie_name: string, duration: integer, average_rating: float, director_username: string)*

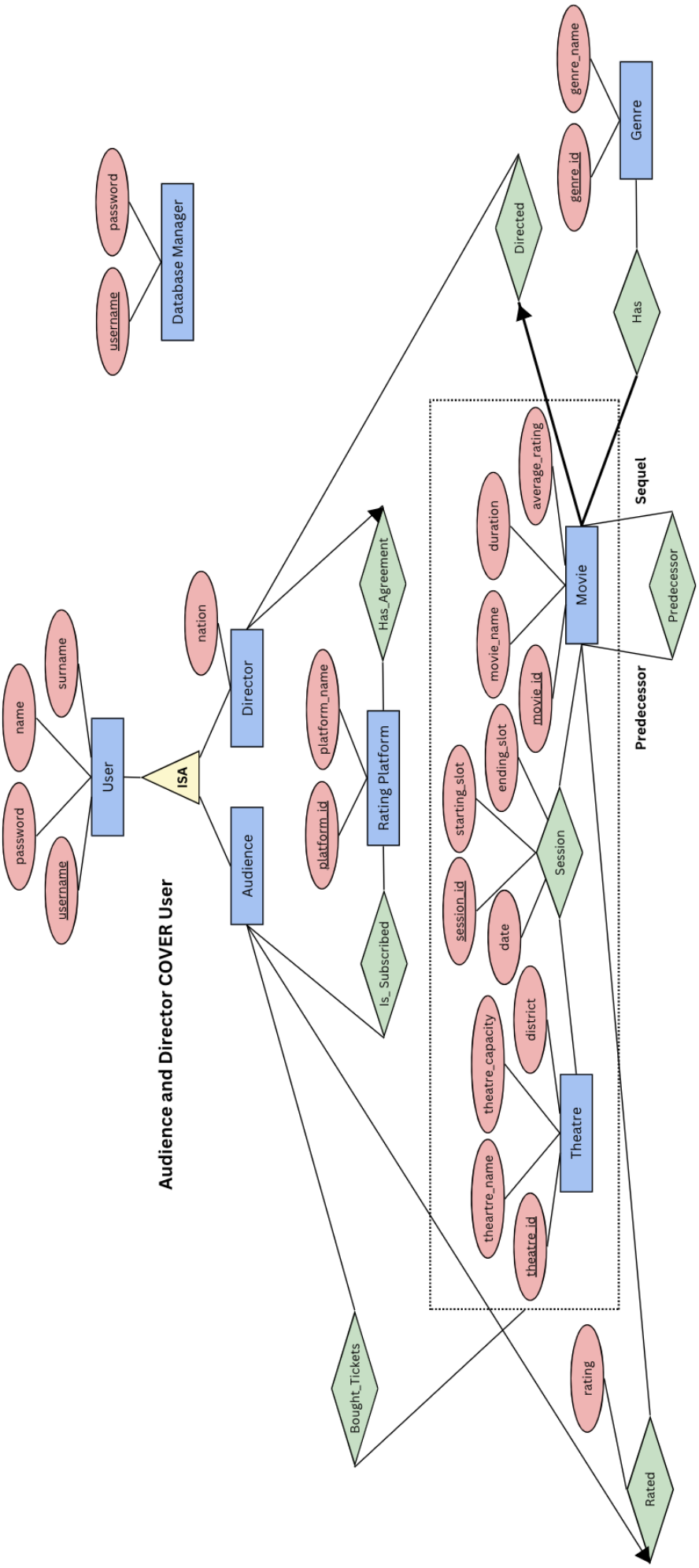
I->INDRS
Since movie_id is the primary key and all the other fields are dependent on it, this relation is BCNF form.

Instead of time_slot, starting_slot and ending_slot values are added to the sessions table.

- *Sessions(session_id: integer, movie_id: integer, theatre_id: integer, date: date, starting_slot: integer, ending_slot: integer)*

I->IMTDSE
Since session_id is the primary key and all the other fields are dependent on it, this relation is BCNF form.

Updated ER Diagram



2 Constraint Handling

Since the first project only required the design of the database tables, there were constraints given that cannot be captured by the limited rules available. In this project, it was possible to handle most of them.

- The movie starts at the given time slot and the ending time is determined by its duration.

The old design made sure that starting time of the movies do not overlap, but without taking the duration of the movie into account. In this design, starting_time and ending_time fields are added to the sessions table. Now that we can make use of a programming language, it is possible to handle this constraint.

- There can be at most 4 database managers registered to the system.

The trigger created for this constraint.

```
CREATE TRIGGER IF NOT EXISTS enforce_max_rows
        BEFORE INSERT ON database_manager
        FOR EACH ROW
        BEGIN
           DECLARE row_count INT;
           SELECT COUNT(*) INTO row_count FROM database_manager;
           IF row_count >= 4 THEN
           SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Maximum number of rows reached';
           END IF;
        END;
```

- Rating is a value between 0 and 5.

This constraint is handled by adding "CHECK(rating>=0 AND rating<=5)" to the end of the CREATE TABLE query for rating table.

- There are four time slots for each day.

This constraint is handled by adding "CHECK (starting_slot IN (1, 2, 3, 4)), CHECK (ending_slot IN (1, 2, 3, 4))" to the end of the CREATE TABLE query for sessions table.

Another trigger for updating the average_rating value after a new rating entry.

```
CREATE TRIGGER IF NOT EXISTS update_average_rating AFTER INSERT ON rating
        FOR EACH ROW
        BEGIN
          UPDATE movie
          SET average_rating = (
            SELECT AVG(rating)
            FROM rating
            WHERE movie_id = NEW.movie_id
          )
          WHERE movie_id = NEW.movie_id;
        END;
```

There are still some constraints that I believe I could not be able to cover fully.

- Booking mechanism is only valid for 1 week.
- 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.