PHYS321 - Assignment 1

Nuri Tas

April 2023

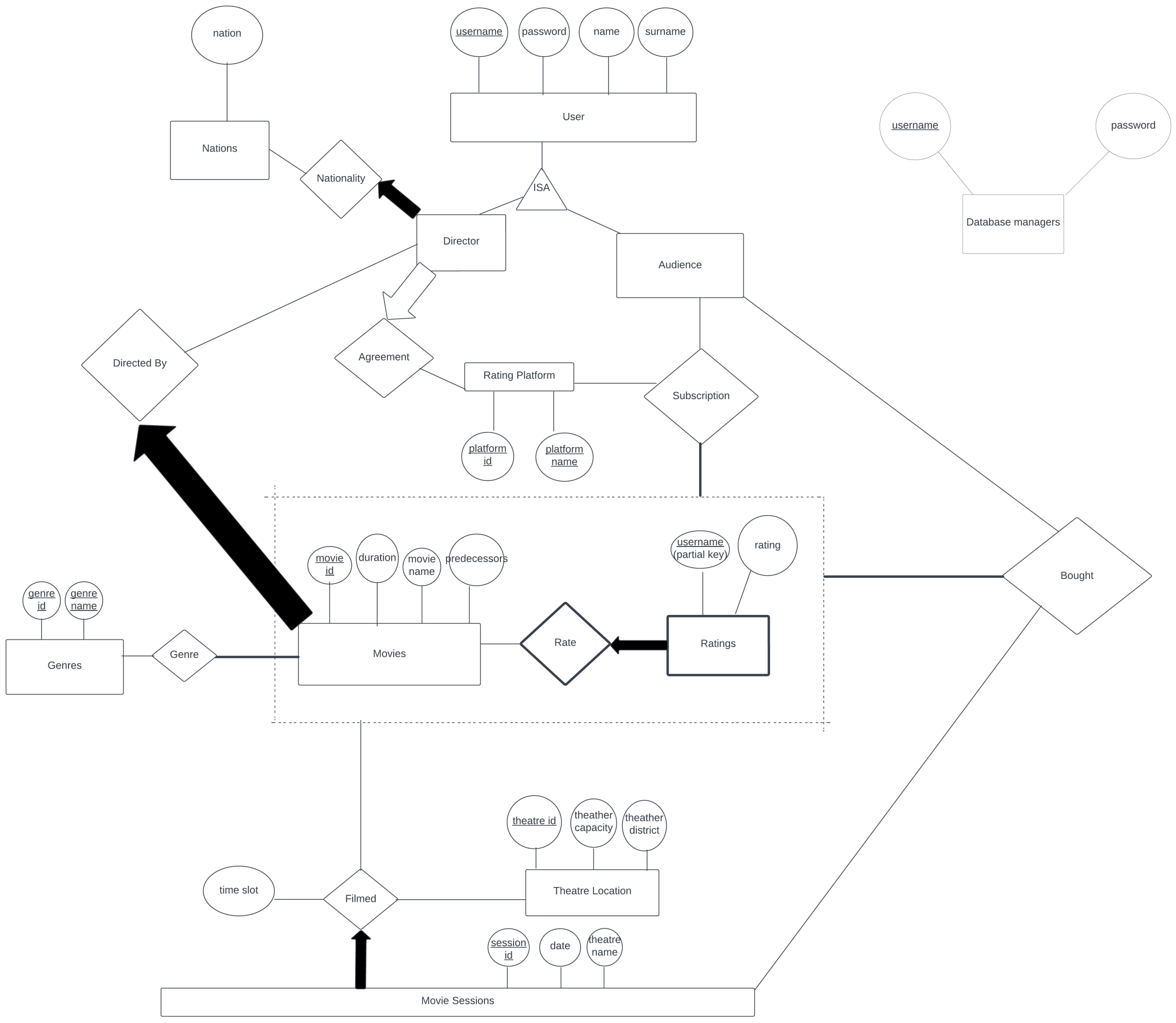
1 ER Diagram

My ER Diagram is given at page 2. The details about the ER diagram is given below.

- *Rate* relation has an aggregation relation with *Subscription*, *Bought*, and *Filmed* relations, whereas the *Movies* entity has relations with *DirectedBy* and *Genre* - who don't form an aggregation relation with *Movies*.
- Director and Audience COVERS User
- Subscription and Bought has an aggregation relation with total participation. That is, a *Rate* entity must be subscribed to the platform and bought the session id of the respective movie.
- Ratings and Movies form a weak entity relation where Movies is the owner and Ratings is the weak entity. Since I wasn't able to underline the partial key (username) with dotted lines, I had to state it in a parenthesis. Participation relations are denoted with thick lines. Key constraint
- The shape sizes are completely irrelevant and are not an indicator of any relation.

2 Schemas for Entities and Relations

The schemas for both entities and relations are given below.
User(username: string, password: string, name: string, surname: string)
Rating Platform (platform id: string, platform name: string)
Directors (username: string);
Nations (nation: string);
Audience (username: string);
Ratings (username: string, rating: real);
Movies (movie id: string, duration: real, movie name: string, predecessors: string);
Genres (genre id: string, genre name: string);


nation
username
password
name
surname
User
Nations
Nationality
ISA
username
password
Database managers
Director
Audience
Directed By
Agreement
Rating Platform
Subscription
platform id
platform name
movie id
duration
movie name
predecessors
username (partial key)
rating
genre id
genre name
Bought
Genres
Genre
Movies
Rate
Ratings
theatre id
theather capacity
theather district
time slot
Filmed
Theatre Location
session id
date
theatre name
Movie Sessions

Theatre Location (theatre id: string, theatre district: string, theatre capacity: integer);

Movie Sessions (session id: string, date: date, theatre name: string);

Database Managers (username: string, password: string)

Nationality (username: string, nation: string)

Rate (username: string, movie id: string),

Genre (movie id: string, genre id: string)

Subscription (username: string, platform id: string, movie id: string)

Bought (username: string, session id: string, movie id: string)

Filmed (session id: string, movie id: string, theatre id: string, time slot: int)

Directed By (movie id: string, username: string)

Agreement (username: string, platform id: string, platform name: string)

3 The requirements that my ER or SQL design can't satisfy

Here are the requirements that I was unable to catch neither in the ER diagram or my SQL DDL:

- No two movie sessions can overlap in terms of theatre and the time it's screened.
- There are four time slots for each day.
- If a movie has any predecessor movies, all predecessor movies need to be watched in order to watch that movie.
- There can be at most 4 database managers registered to the system.
- I was unable to create overall rating for movies.

4 createTables.SQL

Here is the SQL DDL that creates all the entities and relations.

```
-- I will first create tables for entities and then proceed to create the relations

create table User
(username varchar(20), password varchar(20), name varchar(20), surname varchar(20),
primary key (username));


-- Both name and id must be unique for rating platforms
create table Rating_Platform
```

```
(platform_id varchar(20), platform_name varchar(20),
primary key (platform_id),
unique (platform_id, platform_name));

-- Although Directors entity has only one attribute, I show its corresponding
-- relations later on including the nationality, platform_id, etc.
create table Directors
(username varchar(20),
primary key (username),
foreign key (username) references User(username));


create table Nations
(nation varchar(20) not null,
unique (nation));


-- Similarly, I show the corresponding relations for Audience later on
-- including the sessions bought, platforms subscribed, etc.
-- Audience and Directors COVER Users
create table Audience
(username varchar(20),
primary key (username),
foreign key (username) references User(username) on delete cascade);




-- Note, a username can rate different movies. We will show the Rate relation later
-- on to make sure a user can rate a movie only once.
-- Additionally, we ensure that a rating score is a float number between 0 and 5
-- Also, a user may choose to not rate a movie at all.
create table Ratings
(username varchar(20), rating decimal(2,1), check (rating <= 5 and rating >= 0),
foreign key (username) references Audience(username) on delete cascade);


create table Movies
(movie_id varchar(20), duration decimal(4,2), movie_name varchar(20),
 predecessors varchar(20),
primary key (movie_id));


create table Genres
(genre_id varchar(20), genre_name varchar(20),
primary key (genre_id, genre_name));
```

```
-- a theatre id determines the corresponding theatre district and capacity
create table Theatre_Location
(theatre_id varchar(20), theatre_district varchar(20), theatre_capacity int,
primary key (theatre_id));


create table Movie_Sessions
(session_id varchar(20), date timestamp, theatre_name varchar(20),
primary key (session_id));


-- I couldn't create a table such that at most 4 database managers can be
-- registered to the system
create table Database_Managers
(username varchar(20), password varchar(20) not null,
primary key (username));


-- RELATIONS
-- We will now show the relations between entities


-- Each director must have only one nation
-- we satisfy the participation constraint by setting nation not null and
-- having the director's username as the primary key. Note, we don't even
-- need to unique constraint since primary key will ensure
-- there are no duplicate directors and not null constraint for nation
-- will assign only one nation per director.
create table Nationality
(username varchar(20), nation varchar(20) not null,
primary key (username),
foreign key (username) references Directors (username) on delete cascade,
foreign key (nation) references Nations (nation) on delete cascade);


-- Rate is a weak entity set where an user can rate a movie only once
create table Rate
(username varchar(20), movie_id varchar(20),
primary key (username, movie_id),
foreign key (username) references Ratings (username) on delete cascade);


-- Genre relation ensures the participation constraint - that is each movie has
-- at least one genre
```

```
create table Genre
(movie_id varchar(20), genre_id varchar(20) not null,
primary key (movie_id),
foreign key (movie_id) references Movies (movie_id) on delete cascade,
foreign key (genre_id) references Genres (genre_id) on delete cascade);


-- shows the Subscription relation between Audiences and Rating platforms.
-- An audience can subscribe to different platforms
-- Moreover, Subscription relation has an aggregation relation with
-- the Rates relation where Rates has a total participation to the Subscription
create table Subscription
(username varchar(20), platform_id varchar(20), movie_id varchar(20),
primary key (username, platform_id, movie_id),
foreign key (movie_id) references Movies (movie_id) on delete cascade,
foreign key (username) references Audience (username) on delete cascade,
foreign key (platform_id) references
Rating_Platform (platform_id) on delete cascade);

-- shows the Bought sessions relation between Audiences and Movie Sessions.
-- An audience can buy tickets to different sessions
-- Similary, Bought has an aggregation relation to the Rates, and Rates must have
-- total participation to the Bought sessions as well. Overall, Subscription and Bought
-- relations will ensure a user can rate a movie if s/he is subscribed to the movie's
-- platform and bought a ticket to the movie
create table Bought
(username varchar(20), session_id varchar(20), movie_id varchar(20),
primary key (username, session_id, movie_id),
foreign key (username) references Audience (username) on delete cascade,
foreign key (session_id) references Movie_Sessions (session_id) on delete cascade,
foreign key (movie_id) references Movies (movie_id) on delete cascade);



-- Filmed relation will make sure there will be no overlap of two films at the
-- same date and theatre Filmed has a total participation constraint with
-- both Theatre location and movies.
-- Note, I wasn't able to create 4 different time slots for each day
create table Filmed
(session_id varchar(20), movie_id varchar(20) not null,
theatre_id varchar(20) not null,
 time_slot int,
primary key (session_id, movie_id, theatre_id),
foreign key (movie_id) references
 Movies (movie_id) on delete cascade,
foreign key (theatre_id) references
Theatre_Location (theatre_id) on delete cascade);


-- Directed By relation will ensure a movie will have only one director
--  and has the same platform with the director.
-- Directed By has a total participation constraint where each movie needs
--  to have exactly one director
create table Directed_By
(movie_id varchar(20), username varchar(20) not null,
primary key (movie_id),
foreign key (username) references Directors (username) on delete cascade);


-- Agreement relation will ensure the key constraint between directors and rating platforms
-- That is each director can have an agreement with at most one platform
create table Agreement
(username varchar(20), platform_id varchar(20), platform_name varchar(20),
primary key (username),
unique (username, platform_id, platform_name),
foreign key (platform_id, platform_name) references
 Rating_Platform (platform_id, platform_name) on delete cascade);


-- END
```

5 dropTables.SQL

Here are the SQL queries that drop all tables created in createTables.SQL

```
drop table Rate;

drop table Ratings;

drop table Genre;

drop table Genres;

drop table Agreement;

drop table Subscription;

drop table Rating_Platform;

drop table Bought;
drop table Filmed;

drop table Theatre_Location;

drop table Movies;

drop table Movie_Sessions;

drop table Nationality;

drop table Nations;

drop table Directed_By;

drop table Directors;

drop table Audience;

drop tables User;

drop table Database_Managers;
```