Final Project Report: Movie Ticketing System
CS-GY 6083 A: Principles of Database Systems

Course Name: CS-GY 6083 A: Principles of Database Systems

Project Name: Movie Ticketing System

Port Number: 8672

Team Members:

1. Aniket Lavjibhai Saliya (NetID: as15858)
2. Archil Rameshkumar Chovatiya (NetID: ac9137)

Professor: Julia Stoyanovich

Project Description

- The proliferation of data on the internet has made it possible for organizations to build effective ticketing systems. We plan to take advantage of this by building an online booking system for movie tickets. This system will enable users to check which movies are being screened, their timings, and other relevant information on a web user interface. Based on this information, users will be able to decide whether they would like to watch a particular movie.
- Overall, the online booking system for movie tickets will provide users with real-time information on the availability of movies and their timings. This will enable them to easily book tickets and plan their movie-watching experience. It will also provide employees with the tools they need to manage the cinema screen and ensure that the movie-watching experience is seamless for users.

Entity Sets

- Employee
- User
- Movie
- Screen
- Dependents (Weak Entity)
- Screening
- Tickets
- Cast

Relationship Sets

- Reports
- Plays
- Has
- Features
- Employee_manages
- At
- For
- Contains
- Reserved_by

Attributes

- Movie (movie_name (primary key), release_year, language, duration, genre)
- Cast (actor_name (primary_key), dob, age)
- Tickets (Ticket_id (primary_key))
- Screening (screening_id (primary_key), start_time (primary_key), price)
- Employee (emp_id (primary_key), name, SSN, executive_rank)
- Dependents (name) – It will be a weak entity, dependent on Employee.
- Screen (screen_no (primary_key), capacity, media_technology)
- User (user_id (primary_key), name, membership, dob, gender)

Business Rules

- Users are uniquely identified by their user_id and has name, membership type, dob, gender.
- Screen is uniquely identified by Screen_no and has capacity, media technology.
- Movie contains list of all the movies in the database. They are uniquely identified by the Movie_name and has release_year, duration, languages, genre. Each movie has some cast of at least one.
- Cast is uniquely identified by actor_name and has age, dob.
- Employees are uniquely identified by their emp_id and has name, SSN, executive_rank.
- Employee reports to at most one supervisor.
- Employee has dependents (weak entity). Dependents only exists with occurrence of identified employee.
- Screening entity It is uniquely identified by screening_id and start_timestamp and has price. Each screening plays exactly one movie, but same movie can be played in multiple screening.
- Each screening of movie is managed by at least one employee.
- Tickets are uniquely identified by the ticket_id and has a seat number. For same screening, tickets for same seat can't be sold to multiple user.
- A user can reserve any number of tickets, but each ticket is reserved by exactly one user. The seat_no on each ticket should be recorded.

Data for application

- We have synthesized our database by manual entry of tuples in each table.
- The manual entries were added keeping in mind about the interesting output that each of our functionality would return subsequently.
- This was done in order to ensure that our system was functioning properly and producing the desired outputs

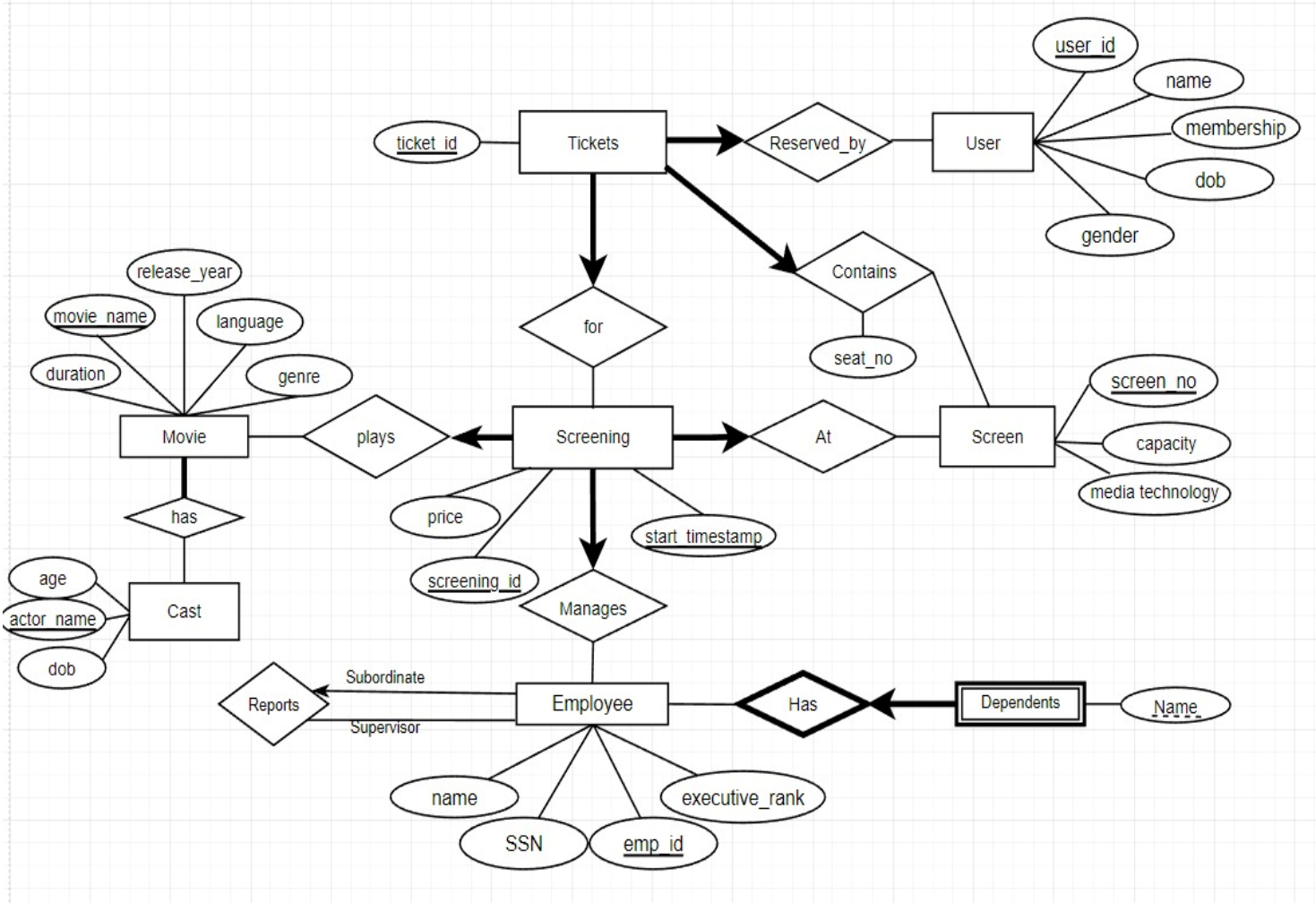
How did we Load our data

cat csvfilename.csv | psql -U Net_id -d Net_id_db -c "COPY table_name from STDIN CSV HEADER"

- We are using the above command in our terminal to load data from all our csv's into our tables.
- Here, **csvfilename** = Name of our csv file, **Net_id** = Our Net_id, **table_name** = Name of the table in our schema where we want to load data of the particular csv file.

```
cat Employee.csv | psql -U as15858 -d as15858_db -c "COPY Employee from STDIN CSV HEADER"
cat Employee_dependents.csv | psql -U as15858 -d as15858_db -c "COPY Employee_dependents from STDIN CSV HEADER"
cat Casting.csv | psql -U as15858 -d as15858_db -c "COPY Casting from STDIN CSV HEADER"
cat Movie.csv | psql -U as15858 -d as15858_db -c "COPY Movie from STDIN CSV HEADER"
cat Movie_has.csv | psql -U as15858 -d as15858_db -c "COPY Movie_has from STDIN CSV HEADER"
cat Screen.csv | psql -U as15858 -d as15858_db -c "COPY Screen from STDIN CSV HEADER"
cat Screening.csv | psql -U as15858 -d as15858_db -c "COPY Screening from STDIN CSV HEADER"
cat Employee_manages.csv | psql -U as15858 -d as15858_db -c "COPY Employee_manages from STDIN CSV HEADER"
cat UserInfo.csv | psql -U as15858 -d as15858_db -c "COPY UserInfo from STDIN CSV HEADER"
cat Ticket.csv | psql -U as15858 -d as15858_db -c "COPY Ticket from STDIN CSV HEADER"
```

ER Diagram


user_id
name
membership
dob
gender
ticket_id
Tickets
Reserved_by
User
Contains
seat_no
release_year
movie_name
language
duration
genre
for
Movie
plays
Screening
At
Screen
screen_no
capacity
media technology
has
price
start_timestamp
screening_id
Manages
age
actor_name
dob
Cast
Subordinate
Reports
Supervisor
Employee
Has
Dependents
Name
name
executive_rank
SSN
emp_id

Schema SQL

```sql
drop table if exists Employee cascade;
drop table if exists Employee_dependents cascade;
drop table if exists Casting cascade;
drop table if exists Movie cascade;
drop table if exists Movie_has cascade;
drop table if exists Screen cascade;
drop table if exists Screening cascade;
drop table if exists Employee_manages cascade;
drop table if exists UserInfo cascade;
drop table if exists Ticket cascade;

create table Employee(
    emp_id char(12) primary key,
    employee_name varchar(32) not null,
    SSN char(9) unique not null,
    executive_rank integer,
    supervisor_id char(12),
    foreign key(supervisor_id) references Employee(emp_id)
);

create table Employee_dependents(
    dependent_name varchar(32) not null,
    emp_id char(12),
    primary key(dependent_name,emp_id),
    foreign key(emp_id) references Employee(emp_id) on delete cascade
);

create table Casting(
    actor_name varchar(32) primary key,
    dob date,
    age integer
);
```

```
34  create table Movie(
35      movie_name varchar(32) primary key,
36      release_year integer,
37      language varchar(32),
38      duration integer,
39      genre varchar(64)
40  );
41
42  create table Movie_has(
43      movie_name varchar(32),
44      actor_name varchar(32),
45      primary key(movie_name,actor_name),
46      foreign key(movie_name) references Movie(movie_name),
47      foreign key(actor_name) references Casting(actor_name)
48  );
49
50  create table Screen(
51      screen_no integer primary key,
52      capacity integer,
53      media_technology varchar(32)
54  );
55
56  create table Screening(
57      screening_id char(10),
58      price decimal not null,
59      start_time timestamp,
60      movie_name varchar(32) not null,
61      screen_no integer not null,
62      primary key(screening_id,start_time),
63      unique(start_time,screen_no),
64      foreign key(movie_name) references Movie(movie_name),
65      foreign key(screen_no) references screen(screen_no)
66  );
```

```
68  create table Employee_manages(
69      screening_id char(10),
70      start_time timestamp,
71      emp_id char(12),
72      primary key(screening_id,start_time,emp_id),
73      foreign key(screening_id,start_time) references Screening(screening_id,start_time),
74      foreign key(emp_id) references Employee(emp_id)
75  );
76
77  create table UserInfo(
78      user_id varchar(32) primary key,
79      name varchar(32) not null,
80      membership varchar(32) not null,
81      dob date,
82      gender char(1),
83      check(gender in ('M','F'))
84  );
85
86  create table Ticket(
87      ticket_id char(10) primary key,
88      screening_id char(10) not null,
89      start_time timestamp not null,
90      user_id  varchar(32) not null,
91      seat_no char(6) not null,
92      screen_no integer not null,
93      unique(screening_id,start_time,seat_no,screen_no),
94      foreign key(screening_id,start_time) references Screening(screening_id,start_time),
95      foreign key(user_id) references UserInfo(user_id),
96      foreign key(screen_no) references Screen(screen_no)
97  )
98
```