Project 2 - Databasic Instinct

CECS 323 – Sec 05

Lorenzo Murillo IV, ID#: 028112355

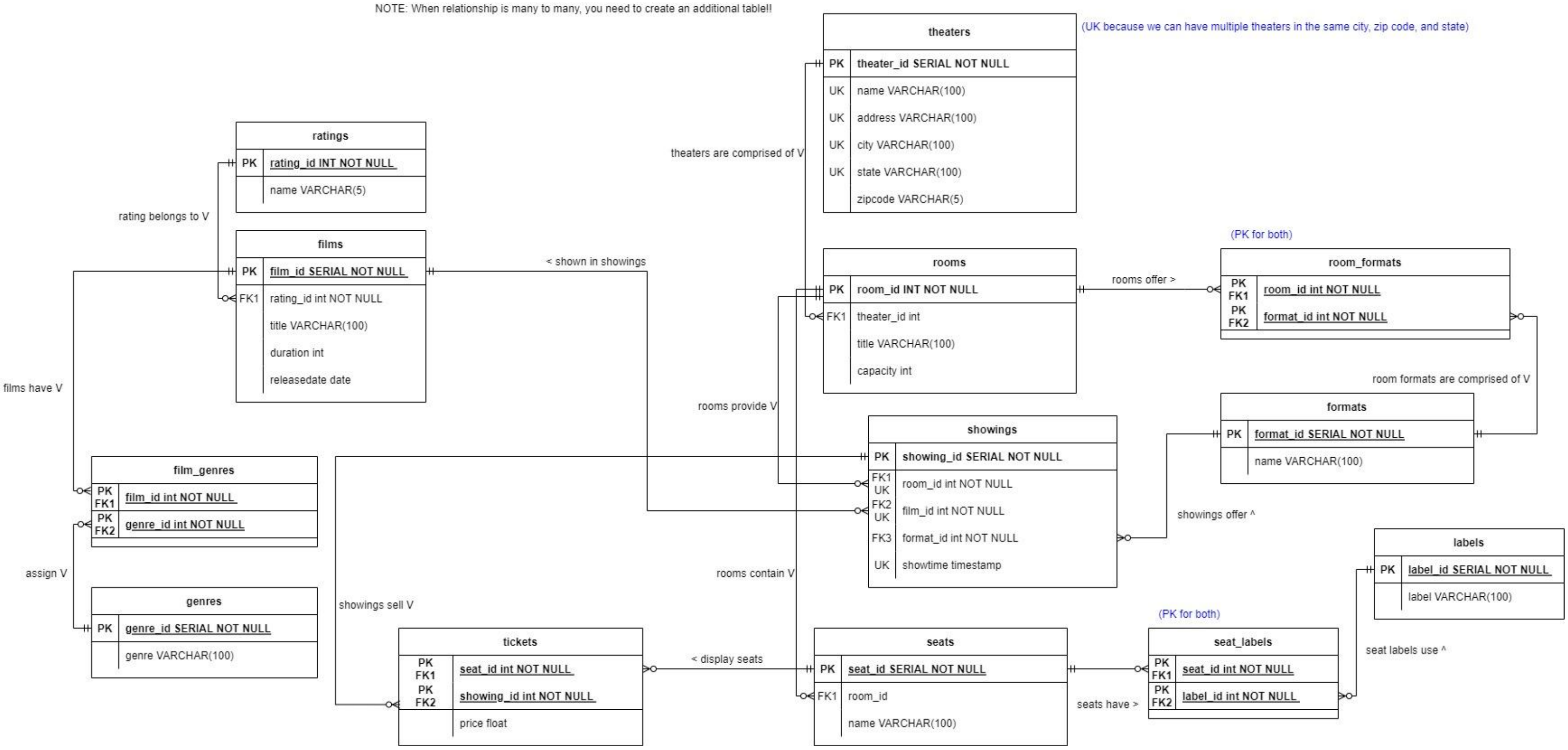
Joakim Eckerman, ID#: 028731311

California State University, Long Beach

College of Engineering

Project 2 - Databasic Instinct

Entity Relationship Diagram:


NOTE: When relationship is many to many, you need to create an additional table!!
theaters
(UK because we can have multiple theaters in the same city, zip code, and state)
PK theater_id SERIAL NOT NULL
UK name VARCHAR(100)
UK address VARCHAR(100)
UK city VARCHAR(100)
UK state VARCHAR(100)
zipcode VARCHAR(5)
theaters are comprised of V
ratings
PK rating_id INT NOT NULL
name VARCHAR(5)
rating belongs to V
films
PK film_id SERIAL NOT NULL
FK1 rating_id int NOT NULL
title VARCHAR(100)
duration int
releasedate date
< shown in showings
films have V
rooms
PK room_id INT NOT NULL
FK1 theater_id int
title VARCHAR(100)
capacity int
rooms offer >
(PK for both)
room_formats
PK FK1 room_id int NOT NULL
PK FK2 format_id int NOT NULL
room formats are comprised of V
formats
PK format_id SERIAL NOT NULL
name VARCHAR(100)
rooms provide V
showings
PK showing_id SERIAL NOT NULL
FK1 UK room_id int NOT NULL
FK2 UK film_id int NOT NULL
FK3 format_id int NOT NULL
UK showtime timestamp
showings offer ^
film_genres
PK FK1 film_id int NOT NULL
PK FK2 genre_id int NOT NULL
assign V
genres
PK genre_id SERIAL NOT NULL
genre VARCHAR(100)
showings sell V
rooms contain V
labels
PK label_id SERIAL NOT NULL
label VARCHAR(100)
tickets
PK FK1 seat_id int NOT NULL
PK FK2 showing_id int NOT NULL
price float
< display seats
seats
PK seat_id SERIAL NOT NULL
FK1 room_id
name VARCHAR(100)
seats have >
(PK for both)
seat_labels
PK FK1 seat_id int NOT NULL
PK FK2 label_id int NOT NULL
seat labels use ^

This image is rather small and I am including a link that will allow you to view the image to scale: https://viewer.diagrams.net/?tags=%7B%7D&highlight=0000ff&edit=_blank&layers=1&nav=1&title=Project%202%20ER%20Diagram#Uhttps%3A%2F%2Fdrive.google.com%2Fuc%3Fid%3D1pzzbuU42m8VwjMjPh3lZynhSmDyfrhei%26export%3Ddownload

CREATE Statements:

```
create table theaters
(
   theater_id serial
      constraint theaters_pk
         primary key,
   name      varchar(100),
   address   varchar(100),
   city      varchar(100),
   state     varchar(100),
   zipcode   varchar(5),
   unique (name, address, city, state)
);

alter table theaters
   owner to postgres;

create table rooms
(
   room_id   integer not null
      constraint rooms_pk
         primary key,
   theater_id integer
      constraint theater_id
         references theaters,
   title     varchar(100),
   capacity  integer
);

alter table rooms
   owner to postgres;

create table seats
(
   seat_id serial
      constraint seats_pk
         primary key,
   name   varchar(100)
      constraint seats_uk
         unique,
   room_id integer
      constraint room_id_fk
         references rooms
);
```

```
alter table seats
    owner to postgres;

create table formats
(
    format_id serial
        constraint formats_pk
            primary key,
    name      varchar(100)
        constraint formats_uk
            unique
);

alter table formats
    owner to postgres;

create table room_formats
(
    room_id   integer not null
        constraint room_id
            references rooms,
    format_id integer not null
        constraint room_formats_fk
            references formats,
    constraint room_formats_pk
        primary key (room_id, format_id)
);

alter table room_formats
    owner to postgres;

create table labels
(
    label_id serial
        constraint labels_pk
            primary key,
    label    varchar(100)
        constraint labels_uk
            unique
);

alter table labels
    owner to postgres;
```

```
create table seat_labels
(
    seat_id  integer not null
        constraint seat_id_fk
            references seats,
    label_id integer not null
        constraint label_id_fk
            references labels,
    constraint seat_labels_pk
        primary key (seat_id, label_id)
);

alter table seat_labels
    owner to postgres;

create table ratings
(
    rating_id integer not null
        constraint ratings_pk
            primary key,
    name      varchar(100)
        constraint ratings_uk
            unique
);

alter table ratings
    owner to postgres;

create table films
(
    films_id  serial
        constraint films_pk
            primary key,
    rating_id integer not null
        constraint rating_id_fk
            references ratings,
    title     varchar(100),
    duration  integer,
    release   date
);

alter table films
    owner to postgres;
```

```
create table showings
(
   showing_id serial
      constraint showings_pk
         primary key,
   room_id    integer not null
      constraint room_id_fk
         references rooms,
   films_id   integer not null
      constraint films_id_fk
         references films,
   showtime   timestamp,
   format_id  integer
      constraint format_id_fk
         references formats,
   constraint showtime_room_uk
      unique (showtime, room_id)
);

alter table showings
   owner to postgres;

create table tickets
(
   seat_id    integer not null
      constraint seat_id_fk
         references seats,
   showing_id integer not null
      constraint showing_id_fk
         references showings,
   price      double precision,
   constraint tickekts_pk
      primary key (seat_id, showing_id)
);

alter table tickets
   owner to postgres;
```

```
create table genres
(
    genre_id serial
        constraint genres_pk
            primary key,
    genre    varchar(100)
        constraint genres_uk
            unique
);

alter table genres
    owner to postgres;

create table film_genres
(
    genre_id integer not null
        constraint genres_fk
            references genres,
    films_id integer not null
        constraint films_id_fk
            references films,
    constraint film_genres_pk
        primary key (genre_id, films_id)
);

alter table film_genres
    owner to postgres;
```

INSERT Statements:

```
insert into Project_2_Databasic_Instinct.theaters (theater_id,
name, address, city, state, zipcode)
values  (1, 'Regal Edwards Long Beach', '7501 E. Carson St.',
'Long Beach', 'CA', '90808'),
       (2, 'AMC Marina Pacifica', '6346 E. Pacific Coast Hwy',
'Long Beach', 'CA', '90803'),
       (3, 'Art Theatre of Long Beach', '2023 E. 4th Street',
'Long Beach', 'CA', '90814'),
       (4, 'Cinemark at The Pike Outlets', '99 S. Pine Ave',
'Long Beach', 'CA', '90802');
```

```
insert into Project_2_Databasic_Instinct.rooms (room_id,
theater_id, title, capacity)
values  (1, 1, 'Screen 1', 5),
       (2, 2, 'Screen 2', 10),
       (3, 1, 'Screen 2', 30),
       (4, 3, 'Main Theatre', 120),
       (5, 4, 'Spielberg Room', 15);
```

```
insert into Project_2_Databasic_Instinct.room_formats (room_id,
format_id)
values  (1, 1),
       (1, 2),
       (2, 1);
```

```
insert into Project_2_Databasic_Instinct.formats (format_id,
name)
values  (1, 'Standard'),
       (2, 'IMAX');
```

```
insert into Project_2_Databasic_Instinct.showings (showing_id,
room_id, films_id, showtime, format_id)
values  (1, 1, 1, '2022-11-10 15:45:00.000000', 1),
       (2, 1, 1, '2022-11-10 19:00:00.000000', 2),
       (3, 3, 4, '2022-11-05 13:15:00.000000', null),
       (4, 3, 4, '2022-11-05 04:30:00.000000', null);
```

```
insert into Project_2_Databasic_Instinct.tickets (seat_id,
showing_id, price)
values  (1, 1, 18),
       (1, 2, 22),
       (5, 2, 15),
       (6, 3, 35),
       (7, 3, 35),
       (8, 3, 35),
       (6, 4, 45),
       (7, 4, 45),
       (8, 4, 45);
```

```
insert into Project_2_Databasic_Instinct.seats (seat_id, name,
room_id)
values  (2, 'Seat 2', 1),
       (1, 'Seat 1', 1),
       (4, 'Seat 4', 1),
       (3, 'Seat 3', 1),
       (5, 'Seat 5', 1),
       (8, '1C', 5),
       (6, '1A', 5),
       (7, '1B', 5);
```

```
insert into Project_2_Databasic_Instinct.seat_labels (seat_id,
label_id)
values  (1, 1),
       (2, 1),
       (3, 1),
       (4, 1),
       (5, 2),
       (5, 3),
       (6, 4),
       (7, 4),
       (8, 4);
```

```
insert into Project_2_Databasic_Instinct.labels (label_id,
label)
values  (1, 'reclining'),
       (2, 'non-reclining'),
       (3, 'accessible seating'),
       (4, 'premiere seating');
```

```
insert into Project_2_Databasic_Instinct.films (films_id,
rating_id, title, duration, release)
values  (1, 3, 'Wakanda Forever', 161, '2022-11-10'),
       (2, 4, 'Everything Everywhere All At Once', 139,
'2022-03-25'),
       (3, 3, 'Mean Girls', 97, '2004-04-30'),
       (4, 4, 'Tar', 158, '2022-10-07');
```

```
insert into Project_2_Databasic_Instinct.ratings (rating_id,
name)
```

```
values  (1, 'G'),
       (2, 'PG'),
       (3, 'PG-13'),
       (4, 'R'),
       (5, 'NC-17');
```

```
insert into Project_2_Databasic_Instinct.film_genres (genre_id,
films_id)
values  (1, 1),
       (2, 1),
       (3, 1),
       (1, 2),
       (8, 2),
       (9, 3),
       (5, 3),
       (10, 4);
```

```
insert into Project_2_Databasic_Instinct.genres (genre_id,
genre)
values  (2, 'Adventure'),
       (1, 'Action'),
       (3, 'Superhero'),
       (4, 'Romance'),
       (5, 'Comedy'),
       (6, 'Horror'),
       (7, 'Thriller'),
       (8, 'Sci-fi'),
       (9, 'Teen'),
       (10, 'Drama');
```

Queries:

```
/*
1) Select the title of all films that have at least one showing
in the IMAX format. [Be wary of duplicates!]
*/
select distinct title as titles_offered_in_IMAX
from films
   inner join showings s on films.films_id = s.films_id
   inner join formats f on f.format_id = s.format_id
where f.name like 'IMAX';

/*
2) Select the name of all theaters that have no showings in the
IMAX format.
*/
select name as theaters_without_imax
from theaters
where name not in (
   select t.name
   from theaters as t
       inner join rooms r on t.theater_id = r.theater_id
       inner join showings s on r.room_id = s.room_id
       inner join formats f on f.format_id = s.format_id
   where f.name like 'IMAX');

/*
3) Select the name of all rooms that do not have any seats with
the "accessible seating" label.
*/
select title as rooms_without_accessible_seating
from rooms
where title not in (
   select title
   from rooms
       inner join seats s on rooms.room_id = s.room_id
       inner join seat_labels sl on s.seat_id = sl.seat_id
       inner join labels l on l.label_id = sl.label_id
   where label like '%accessible%');
```

```
/*
4) Select the primary key of the showing that has brought in the
most income:
```

```
the sum of the price of every ticket sold for that showing.
*/
select s.showing_id, sum(price) as showing_sales
from showings as s
   inner join tickets t on s.showing_id = t.showing_id
group by s.showing_id
order by showing_sales desc
fetch first 1 rows with ties;

/*
5) Count the number of "short", "average", and "long" films.
A short film is under 90 minutes; an average film is between 90
and 120 minutes; a long film is over 120 minutes.
*/
select count(films_id) as number_of_films,
      case
          when duration < 90 then 'short'
          when duration >= 90 and duration < 120 then 'average'
          when duration >= 120 then 'long'
end as film_length
from films as f
group by film_length;
```