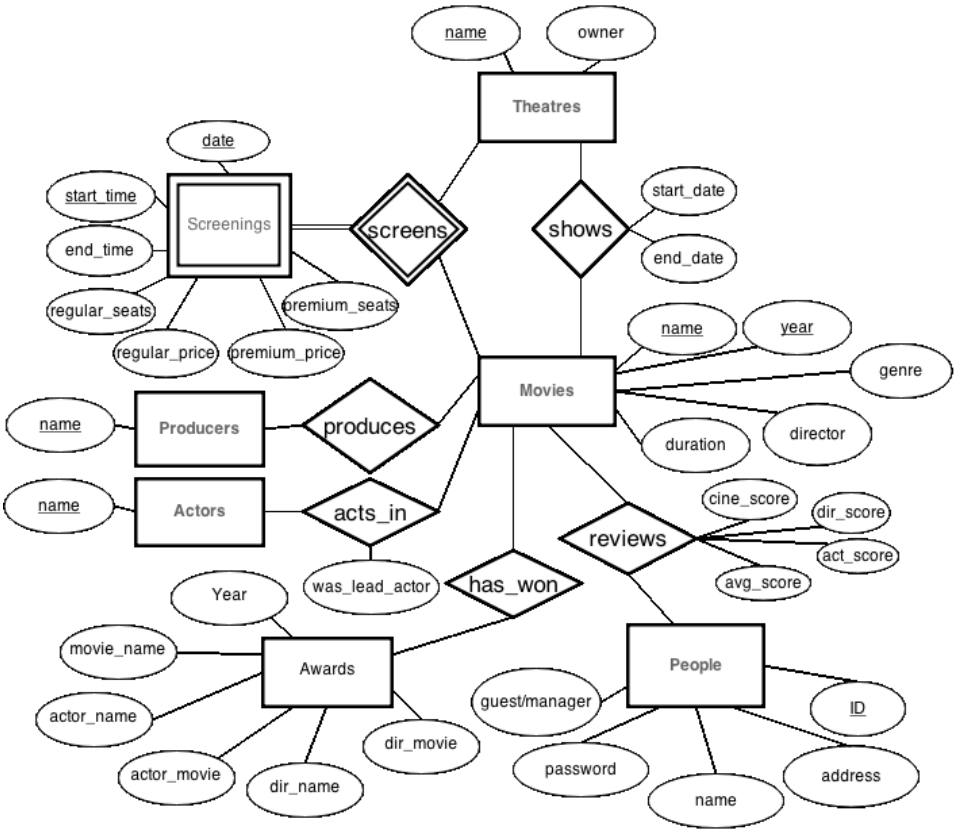
Movie Database Infosheet

by Ankit Mahajan 120100027, Varun Gandhi 120260003

ER Model Diagram



List of functional dependencies:

Movies(duration),Screenings(start_time) → Screenings(end_time)

Reviews(cine_score,dir_score,act_score) → Reviews(avg_score)

Other (obvious) key dependencies

Relational Model

- Relations:

  Theatres - name , owner

  Movies - name, year, genre, duration, director

  Producers relation - name

  produces relation - producer_name , movie_name , year

  Actors relation - name

  acts_in relation - actor_name , movie_name , year, was_lead_actor

Screenings (WE) - movie_name, theatre_name, movie_year, date, start_time, end_time, regular_price, regular_seats, premium_price, premium_seats

screens - movie_name, theatre_name, movie_year, date, start_time (Redundant)

shows - movie_name, theatre_name, movie_year, start_date, end_date

People - username, name, address, phone number, is_manager

reviews - movie_name, movie_year, username, cine_score, dir_score, act_score,

Awards - year, movie_name, actor_name, actor_movie, dir_name, dir_movie

* has_won has not been put as it is redundant. Also screens is present in the current implementation but has not been used in any way.

- Foreign Key constraints

  produces (producer_name) references Producers (name), produces (movie_name, year) references Movies (name, year)

  acts_in (actor_name) references Actors (name), acts_in (movie_name,year) references Movies (name,year)

  Screenings (movie_name, movie_year) references Movies (name, year), Screenings (theatre_name) references Theatres(name)

  screens (movie_name, movie_year, theatre_name, show_date, start_time) references Screenings (movie_name, movie_year, theatre_name, show_date, start_time)

  shows (movie_name, movie_year) references Movies (name, year), theatre_name references Theatres (name)

- Other constraints:

  Use triggers to ensure (ADDED) - see add_award.sql and add_award_trig.sql

  - Awards (movie_name, year-1) references Movies(name, year),
  - Awards (actor, actor_movie, year-1) references acts_in(actor_name, movie_name, year)
  - Awards (dir_name, dir_movie, year-1) references Movies(director, name, year)

  Calculation of end_time and no overlap condition as a trigger (ADDED) - see update_end_time.sql file for function and screening_update_trig.sql for trigger.

  Assertion: For a tuple in shows, year of start_date $\geq$ movie_year and end_date $\geq$ start_date (ADDED)

  Syntax. Alter Table shows Add Constraint no_screenings_before_release Check (Extract (Year From start_date) >= movie_year And end_date>=start_date) ;

  //No quantities are allowed to be null.

—IGNORE BELOW COMMENTARY—

Normalisation (not redone with final schema)

Q. Is current schema in BCNF?

A. No. In reviews, avg_score is determined by attributes which do not form a superkey. Otherwise, everything else seems okay.

Implementation details

All string fields are using `text` data type because there is no significant difference between `char(n)`, `varchar(n)`, `varchar` and `text` in terms of performance (link: http://www.depesz.com/2010/03/02/charx-vs-varcharx-vs-varchar-vs-text/).

Remaining things to implement

"Other constraints" written above with appropriate triggers and/or assertions

Collections should be calculated from bookings

Roles

Table/Roles	Manager	Guest
Theatres		
Movies	add,edit(not year,name)	
Producers		
produces		
Actors	add	
acts_in	add	
Screenings	add,edit	book ticket
screens	add,edit	
shows	add,edit	
People		
reviews		add
Awards	add	

Manager should be able to search collections by a wide set of parameters

- Total collections by (movie_name, movie_year) till date
- Total collections by (theatre_name) from start_date to end_date
- Ratings vs Collections (till date)

CHANGES

Ditch avg_score. Create a view with movie_name, movie_year, avg_rating.

Distinguish between total and available seats for both regular and premium.

Constraint available ≥ 0.

Numerical constraints in HTML.

Whenever actor is being added somewhere (+), first check if it is present in Actors; if not add and (if needed) make changes to acts_in (SQL Trigger).

M1) A manager can enter a new movie to the DB.

Solution: Manager must add name, release year, genre, director, duration (in minutes), at least one producer(+), at least one actor(+) (with exactly one lead actor).

Possible errors: duplicate PK for Movies

M2) A manager can edit movie details but NOT its name or year or release.

Solution: Don't allow changing of duration, producers.

Find tuple in Movies and allow editing of genre/director.

Or change lead actor to some other actor(+). Or add actor or delete actor.

Errors: delete actor who is not present in movie (acc. to acts_in) (Java or Javascript drop-down menu)

M3) Actors can be added but not deleted.

Solution: Allow actor to be added to Actors. Give option for adding filmography of actor to acts_in.

Errors: FK exception movie was not present in Movies.

M4) Award details can be added but not deleted.

Add entry to Awards

Errors: FK exceptions for many things

M5) A manager can add theaters that the movie is showing in currently.

Solution: Add to shows

Errors: Foreign key or Assertion exception

M6) Manager can modify showtimes and seat pricing for the movie in that theatre

Solution: Edit screenings

Errors: FK or Assertion exception

M7) A manager can also change the period for which the movie is available at a theatre.

Solution: BEGIN Delete screenings of movie_name,movie_year,theatre_name, Update shows with new_start_date, new_end_date END

Errors: Foreign key or Assertion exception

M4) A manager can query for the collections of a movie across theaters, of a theatre across movies for a time period. He should be able to rank order the best selling movies, the worst selling movies and search for movies and collections by a wide set of parameters.