The Chinese University of Hong Kong, Shenzhen

CSC 3170

Database System

Group 10 Report: Rotten Potatoes

Authors:	*Student Number:*
郑时飞	119010465
朱伯源	119010485
李易	119010156
施天昊	120090472
汪明杰	119010300

Wednesday 11th May, 2022

Contents

1 Introduction

2 Design

In this section, we focus on the design of Entity-Relationship Model, the reduction from ER diagram into relational schemas, constraint and index.

2.1 Entity-Relationship Model

As shown in the ER digram below, there are 6 entities and 5 relationships.

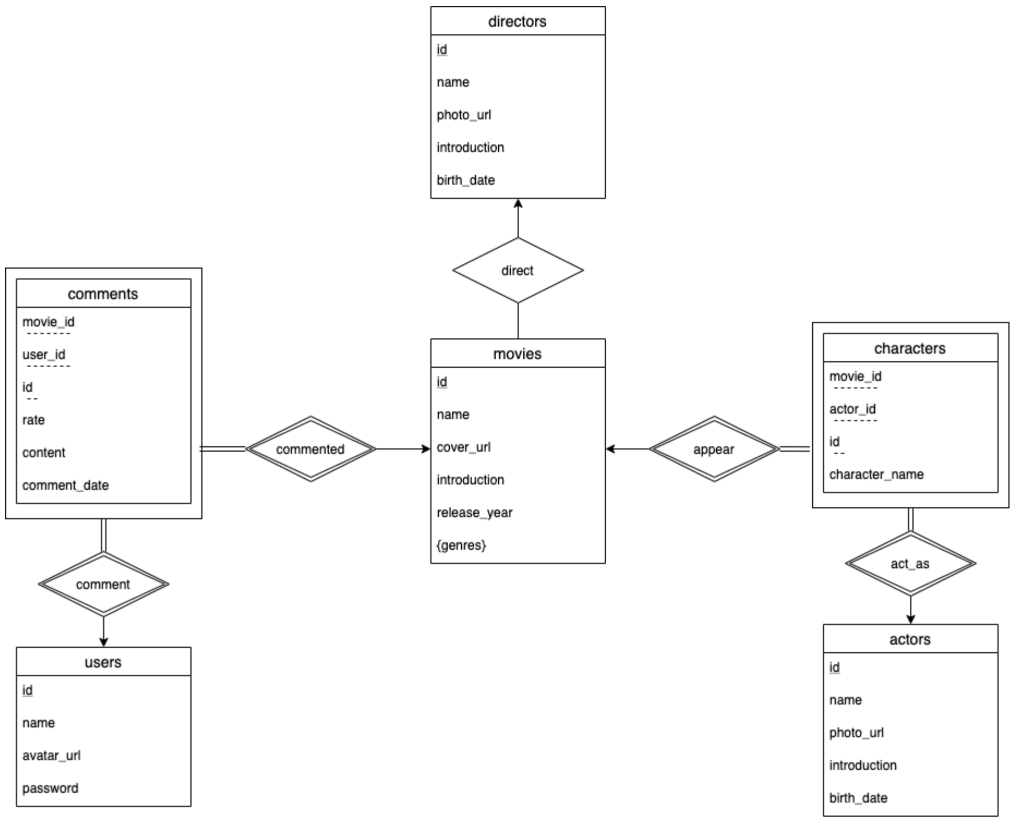

Figure 1: ER Diagram

Entity "movies" To store id, name, cover url, introduction, release year and genres of a movie, where genres is a multivariate attribute. Identified by id.

Entity "directors" To store id, name, photo url, introduction and birth date of a director. This entity has a one-to-many relationship "direct" with the entity "movies", which means a director can directs multiple movies and a movie can be directed by only one director in our assumption. Identified by id.

Entity "actors" To store id, name, photo url, introduction and birth date of an actor. This entity has a many-to-many relationship with the entity "movies", which means an actor can act in multiple movies and a movie can be acted by multiple actors in our assumption. Identified by id.

Entity "users" To store id, name, avatar url, password of a user. This entity has a many-to-many relationship with the entity "movies", which means a user can comment on multiple movies and a movie can be commented by multiple users in our assumption. Identified by id.

Entity "characters" To store movie id (of the movie where this character appears), actor id (of the actor who acts as this character), id, character name of a character. This entity is a weak entity identified by entity "movies" through relationship "appear" and entity "actors" through relationship "act as", and also by its own id, which means a character can be acted by exactly one actor and appear in exactly one movie in our assumption (we treat characters of the same name appearing in multiple movies or acted by multiple actors as multiple different characters for simplicity). Note that since this entity is also identified by its own id, it is allowed that an actor acts as multiple characters in the same movie.

Entity "comments" To store movie id (of the movie commented by this comment), user id (of the user who makes this comment), id, rate (from 0 to 10), content and comment date of a comment. This entity is a weak entity identified by entity "movies" and entity "users", and also by its own id, which means a comment is on exactly one movie and is made by exactly one user in our assumption. Note that since this entity is also identified by its own id, it is allowed that a user makes multiple comments on the same movie.

2.2 Relational Schema

As shown in the relational schema digram below, there are 7 schemas.

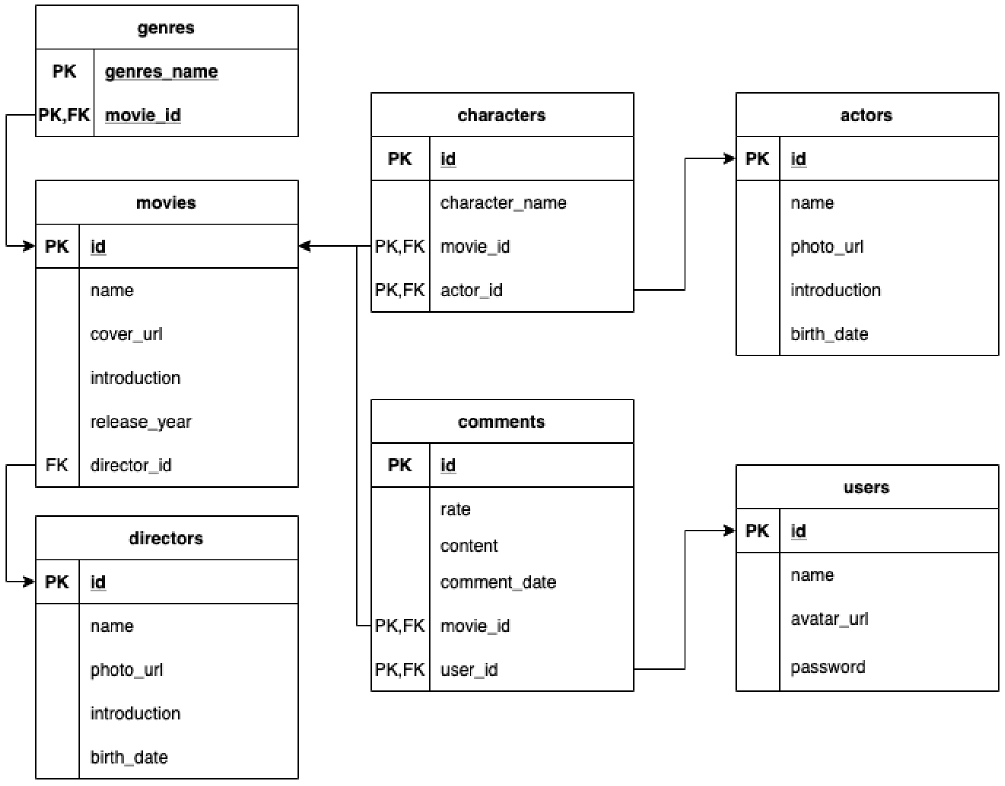

Figure 2: Relational Schema Diagram

The following reductions are made:

- The attribute “genres” in entity “movies” is reduced to schema “genres” with attributes “genres_name” and “movie_id” as a foreign key, both of which forms a primary key to make sure no redundant genres of a movie.
- The relationship “direct” between entity “movies” and “directors” is reduced to attribute “director_id”as a foreign key in schema “movies” so that a movie is directed by exactly one director.
- All attributes identifying the entity are reduced to primary keys.
- “movie_id”, “actor_id” of entity “characters”, “movie_id”, “user_id” of entity “comments” are reduced to foreign keys in their schemas referencing their corresponding identifying strong entities.

2.3 Constraint

3 types of constraints are further added:

Not Null Constraints

- All “id” attributes as they are primary key and thus automatically becoming not null.
- “name” attribute of schema “movies”.
- “name” and “director_id” attributes of schema “directors”, as a movie is directed by exactly one director.
- “name” attribute of schema “actors”.
- “name” and “password” attributes of schema “users”.
- “actor_id”, “movie_id” and “character_name” attributes of schema “characters”, as it is a weak entity of schemas “actors” and “movies”.
- “user_id”, “movie_id”, “rate”, “content” and “comment_date” attributes of schema “comments”, as it is a weak entity of schemas “users” and “movies”.
- “genres_name”, “movie_id” attributes of schema “genres”, as it is a multivariate attribute of schema “movies”.

Unique Constraint A unique constraint is added to “name” attribute of schema “users” as by our assumption there should be no repeating user names.

Check Constraint A check constraint is added to “rate” attribute of schema “comments” to make sure the rate is from 0 to 10.

2.4 Index

The following attributes are indexed to make search faster:

- "name" and "release_year" attributes of schema "movies".
- "name" and "birth_date" attributes of schema "directors".
- "name" and "birth_date" attributes of schema "actors".
- "name" attribute of schema "users".

3 Implementation

3.1 Frontend

3.2 Backend

3.3 Web Crawler

3.4 Data Analysis

4 Result

5 Conclusion

6 Self-Evaluation

7 Contribution