Advanced SQL Programming

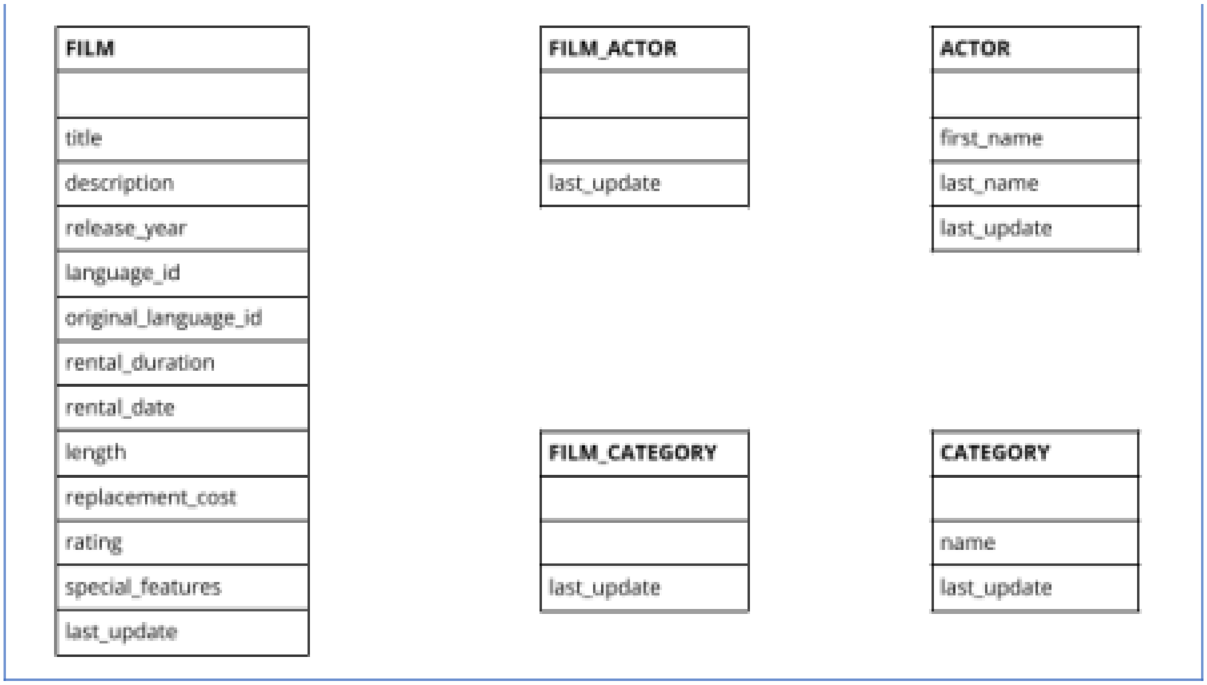
Final Project

Introduction

Sequel Premiere Films specializes in reboots, remakes, and…unsurprisingly… sequels. They need your help to manage their database. You will be working with the existing Sakila sample database in MySQL Workbench.

Task 1: Database Schemas

Below is a small portion of the schema for the Sakila database. Each box represents a table and each row represents a column in the table. Record the missing column names and identify the relationships between the tables by adding lines to connect the two related columns.



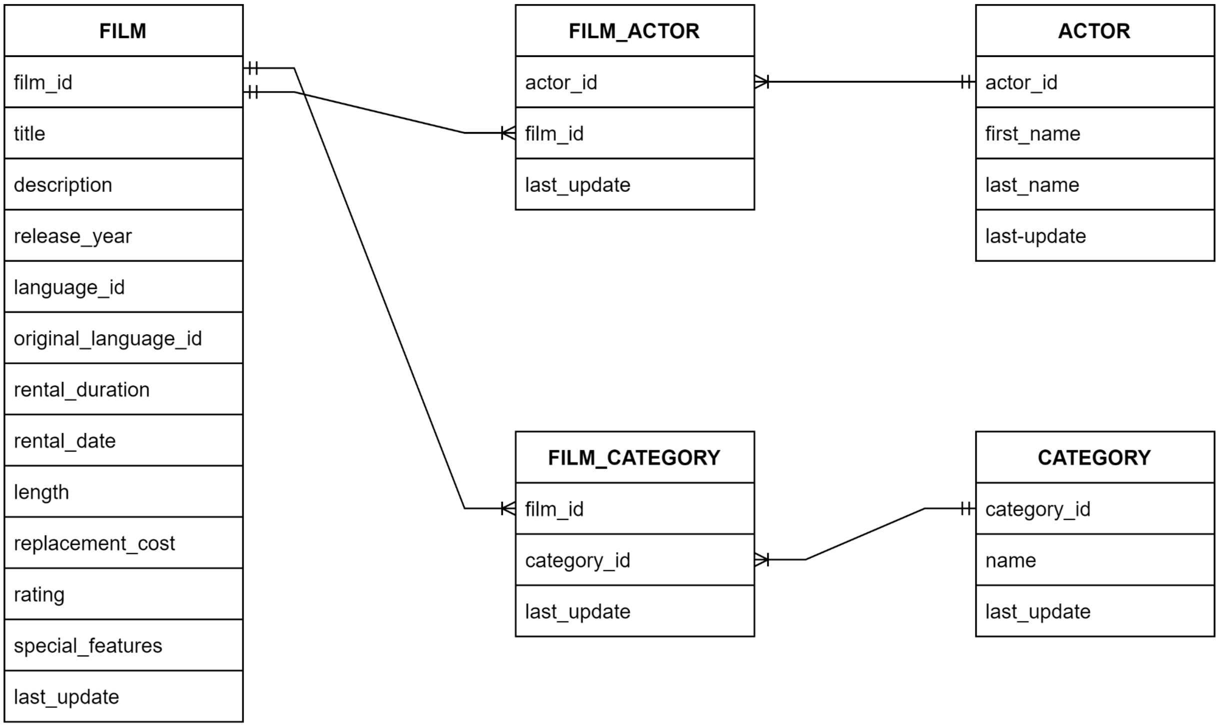
FILM
title
description
release_year
language_id
original_language_id
rental_duration
rental_date
length
replacement_cost
rating
special_features
last_update

FILM_ACTOR
last_update

ACTOR
first_name
last_name
last_update

FILM_CATEGORY
last_update

CATEGORY
name
last_update



FILM
film_id
title
description
release_year
language_id
original_language_id
rental_duration
rental_date
length
replacement_cost
rating
special_features
last_update

FILM_ACTOR
actor_id
film_id
last_update

ACTOR
actor_id
first_name
last_name
last-update

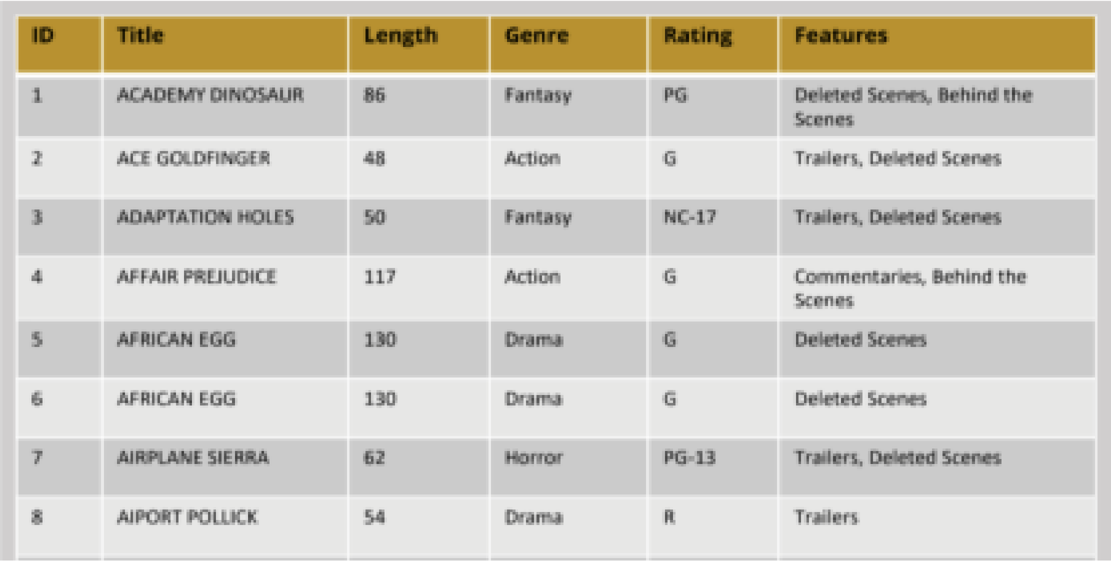
FILM_CATEGORY
film_id
category_id
last_update

CATEGORY
category_id
name
last_update

- All of the tables above have **one-to-many** relationships between them
- The **FILM_ACTOR** and **FILM_CATEGORY** tables are known as **Junction** tables (or as **Bridge** tables), which are used to convert from a many-to-many relationship to two one-to-many relationships.

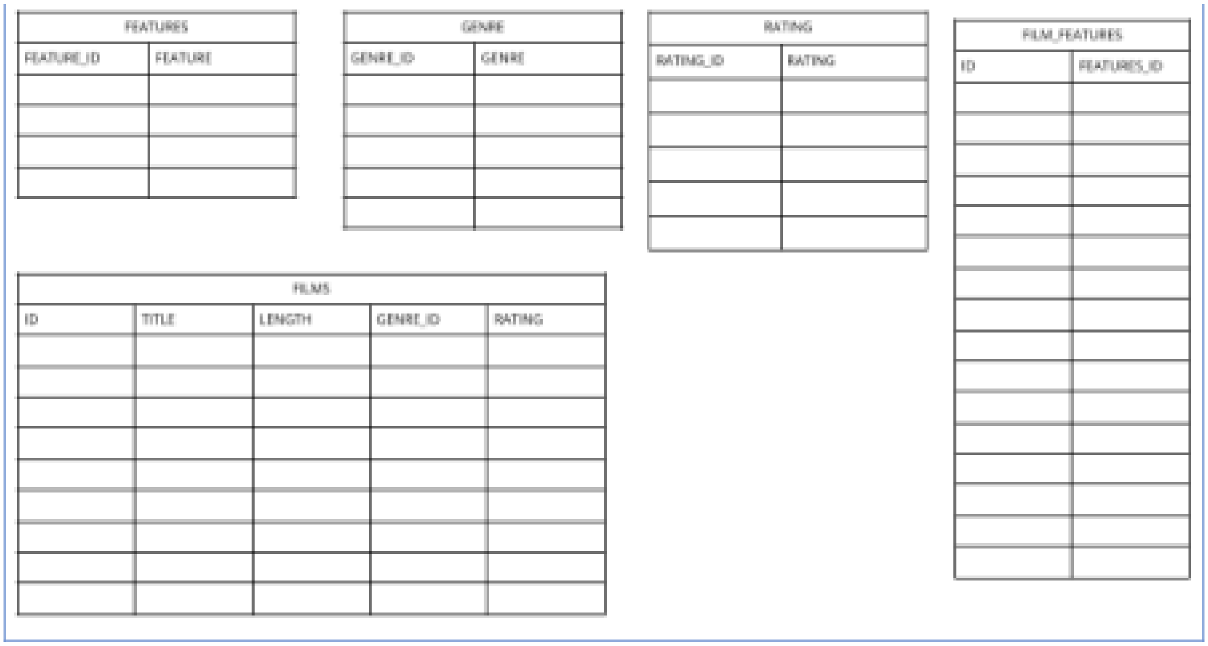
Task 2: Normalization

Below is a section of a table stored in the Sakila database showing information about a specific film. The table has not been normalized. Using the pre-made tables, normalize the table into 3NF form.



ID	Title	Length	Genre	Rating	Features
1	ACADEMY DINOSAUR	86	Fantasy	PG	Deleted Scenes, Behind the Scenes
2	ACE GOLDFINGER	48	Action	G	Trailers, Deleted Scenes
3	ADAPTATION HOLES	50	Fantasy	NC-17	Trailers, Deleted Scenes
4	AFFAIR PREJUDICE	117	Action	G	Commentaries, Behind the Scenes
5	AFRICAN EGG	130	Drama	G	Deleted Scenes
6	AFRICAN EGG	130	Drama	G	Deleted Scenes
7	AIRPLANE SIERRA	62	Horror	PG-13	Trailers, Deleted Scenes
8	AIPORT POLLICK	54	Drama	R	Trailers

From: ThriveDX



FEATURES

FEATURE_ID	FEATURE

GENRE

GENRE_ID	GENRE

RATING

RATING_ID	RATING

FILM_FEATURES

ID	FEATURES_ID

FILMS

ID	TITLE	LENGTH	GENRE_ID	RATING

FEATURES

FEATURE_ID	FEATURE
1	Deleted Scenes
2	Behind The Scenes
3	Trailers
4	Commentaries

GENRE

GENRE_ID	GENRE
1	Fantasy
2	Action
3	Drama
4	Horror
5	Foreign

RATING

RATING_ID	RATING
1	PG
2	G
3	NC-17
4	PG-13
5	R

FILMS

FILM_ID	TITLE	LENGTH	GENRE_ID	RATING_ID
1	ACADEMY DINOSAUR	86	1	1
2	ACE GOLDFINGER	48	2	2
3	ADAPTATION HOLES	50	1	3
4	AFFAIR PREJUDICE	117	2	2
5	AFRICAN EGG	130	3	2
6	AGENT TRUMAN	169	5	1
7	AIRPLANE SIERRA	62	4	4
8	AIRPORT POLLOCK	54	3	5

FILM_FEATURES

FILM_ID	FEATURES_ID
1	1
1	2
2	3
2	1
3	3
3	1
4	4
4	2
5	1
6	1
7	3
7	1
8	3

NOTE: Row 6 (FILM_ID = 6) of the non-normalized films table should have the film title “Agent Truman”, where length = 169, genre = “Foreign”, and rating = “PG”.

Task 3: Cardinality and Degree of Relationship

What is the relationship type (one-to-one, one-to-many, or many-to-many) between FILMS and FILM_FEATURES? Explain the reasoning behind your answer.

One-to-many, because there are many film features for each film

Task 4: Advanced Stored Procedures

Create a stored procedure where users can input the first name and return a list of all of the actors who share that first name.

```
DELIMITER $$
CREATE PROCEDURE sp_actor_first_name(IN actor_firstname VARCHAR(45))
BEGIN
        SELECT first_name, CONCAT(first_name, ' ' , last_name) AS actor
        FROM actor
       WHERE first_name = actor_firstname;
END $$
```

Task 5: Encryption

Describe why it is helpful to use a cryptographic hash when storing sensitive information like a password.

Using a cryptographic hash when storing sensitive information (like passwords) is helpful because it prevents hackers from obtaining sensitive information.