Data Model & Schema Translation

6.170 Recitation 7
Netflix’s Movie Collection
Exercise: Draw a Data Model

for a movie streaming service where

- a user can browse through the movie collection by genre
- a user can search for a movie by name
- a user can watch movies

Data models crib sheet: Stellar → Materials → General
Movie → Genres

User → watched

genre +
Who's watching?

Guy who pays for the account
Parasite 1
Parasite 2
Parasite 3
Kids

Rick Mac Gillis, Growth Strategist at Dragon Cloud (2016-present)
Answered Mar 14, 2018 · Author has 95 answers and 637.2k answer views

Netflick asks who's watching even if there's only one profile because they haven't figured out how to disable that screen. Their system isn't developed in a modular fashion, and to spare their developers from having to rewrite the software, they pass it off as a feature to the end user.

In short, Netflix doesn't feel like disabling it so you can simply say, “Alexa, play Netflix.” They'd rather defeat Alexa and leave the screen in place.

Exercise: Extend the Data Model
to support the who is watching feature

Who's watching?

Guy who pays for the account  Parasite 1  Parasite 2  Parasite 3  Kids

Data models crib sheet: Stellar → Materials → General
Exercise: Extend the Data Model More!

to make recommendations based on a user’s viewing

Data models crib sheet: Stellar → Materials → General
Recommendation -> Movie

Movie -> Genres

Movie -> User

User -> Account

Movie -> Viewing

Based on recommend

Profile

Account + account
Exercise: Translate Data Model into a SQL Schema

- **Recommendation**
  - Based on: Viewing
  - Movie

- **Viewing**
  - Movie
  - Recommend

- **Movie**
  - Genre

- **Genres**

- **User**
  - Profile

- **Account**
  - Account
1. Add Attributes
2. Pick Primitive Types

Recommendation -> Movie
  - movie
  - recommend
  - based on

Movie -> Genres
  - genre

Viewing -> User
  - profile

User -> Account
  - account

Count
Name
Email
Billing Address
Credit Card #

Email
Billing
Credit
Addr
Payment
Count
Timestamp
Last left off
Name
Email
Addr
Payment
Count
Timestamp
Last left off
Name
Email
Addr
Payment
Count
Timestamp
Last left off
Name
Email
Addr
Payment
3. Reverse 1:Many Relations
4. Break into Tables
5. Write the Tables

CREATE TABLE users (id INT PRIMARY KEY AUTO_INCREMENT, username VARCHAR(10) NOT NULL, FOREIGN KEY(accId) REFERENCES accounts(id))

CREATE TABLE accounts (id INT PRIMARY KEY AUTO_INCREMENT, email TEXT NOT NULL UNIQUE, billingAddr TEXT NOT NULL, creditCardNum INT NOT NULL)

the key?
the whole key?
nothing but the key?
5. Write the Tables

CREATE TABLE movies (
    id INT PRIMARY KEY AUTO_INCREMENT,
    title TEXT NOT NULL
);

CREATE TABLE genres (
    id INT PRIMARY KEY AUTO_INCREMENT,
    name TEXT NOT NULL
);

CREATE TABLE movie-genre (
    id INT PRIMARY KEY AUTO_INCREMENT,
    FOREIGN KEY(movieId) REFERENCES movies(id),
    FOREIGN KEY(genreId) REFERENCES genres(id)
);
5. Write the Tables

```
CREATE TABLE recs (  
id INT PRIMARY KEY AUTO_INCREMENT,  
FOREIGN KEY(movieId) REFERENCES movies(id)  
FOREIGN KEY(userId) REFERENCES users(id)
);
```

```
CREATE TABLE viewing (  
id INT PRIMARY KEY AUTO_INCREMENT,  
count INT NOT NULL,  
lastLeftOff DATETIME NOT NULL,  
FOREIGN KEY(movieId) REFERENCES movies(id)
);
```

```
CREATE TABLE viewing-rec (  
id INT PRIMARY KEY AUTO_INCREMENT,  
FOREIGN KEY(vId) REFERENCES viewings(id),  
FOREIGN KEY(rId) REFERENCES recs(id)
);
```
Evaluate Schema: Sample Queries

- Credit card number of all accounts whose users watched *Slumdog Millionaire*
- The genre that a user watches most frequently
- Number of genres recommended to a user