CS 3200 (Database Design) Spring 2018
Final Project: QwikTix

Authors:
Viviano Cantu
Sarina Dass
Kirk Morgan
Zongru Wang

18 April 2018

Abstract

The problem being solved was the lack of a centralized system for getting detailed information about movies and making plans to watch movies. QwikTix can be thought of as "Fandango meets IMDB meets Apple's TV app." When users register with QwikTix, they are able to search and view detailed information about movies, including not only details like release data, genre, and studio, but also details about all of the people credited in the movie, from actors to directors. In addition to learning about movies, users are able to make plans to watch the movie, either by ordering a ticket from a local movie theater, or by ordering the movie from an online streaming service.

In order to provide users with this type of service, we designed a front-end system in HTML with a back-end that uses Python to access the MySQL database which stores all of the information that QwikTix needs. In this system, the homepage contains sections for all of the things that users are allowed to do. For example, they can make an account, love a movie, see the movies they have ordered, see all movies directed by their favorite director, etc. In addition to these tasks, the system allows for a number of reports, many of which are parameterized with user input. These reports provide information such as the most loved movies in every ZIP code of a given country, and the number of orders from all movie theaters and streaming services.

Through partnerships with local movie theaters around the country and around the world, in addition to partnerships with online streaming services, QwikTix will revolutionize the way people think about the movie industry. The combination of movie information, user information, and vendor information into one application allows for collection of massive amounts of data, not only about movies, but also about users. QwikTix plans to use artificial intelligence and machine learning to gather useful information about its users. This data can be used to display targeted advertisements to users to encourage them to either buy movie tickets from local theaters that they've ordered from before, or to order a new movie from a streaming service.

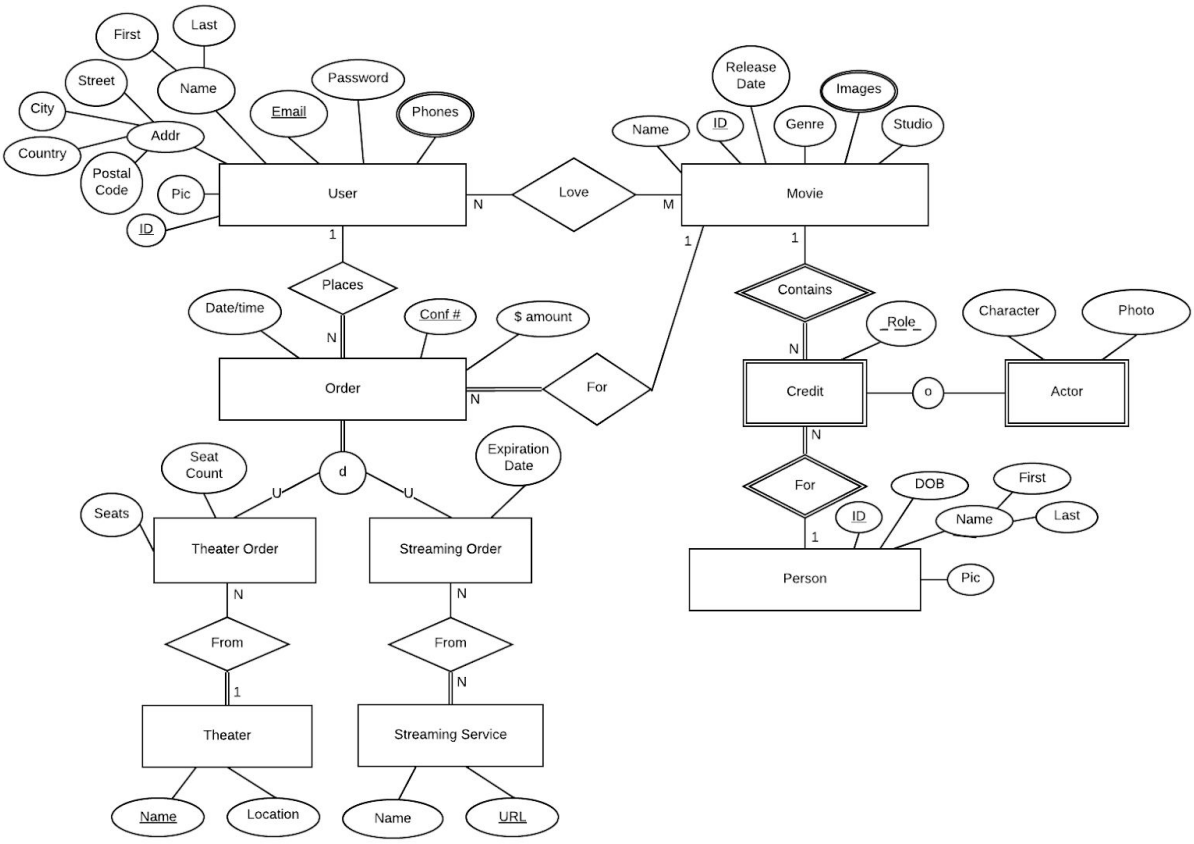
Textual Description

Each user can be identified by their unique email address or their ID. Users have an address that is comprised of a postal code, country, city, and a street. They also have a name that is comprised of a first name and a last name, a password, a profile picture, and any number of phones associated with their account. Users can place any number of orders for movies. Users can also "love" movies.

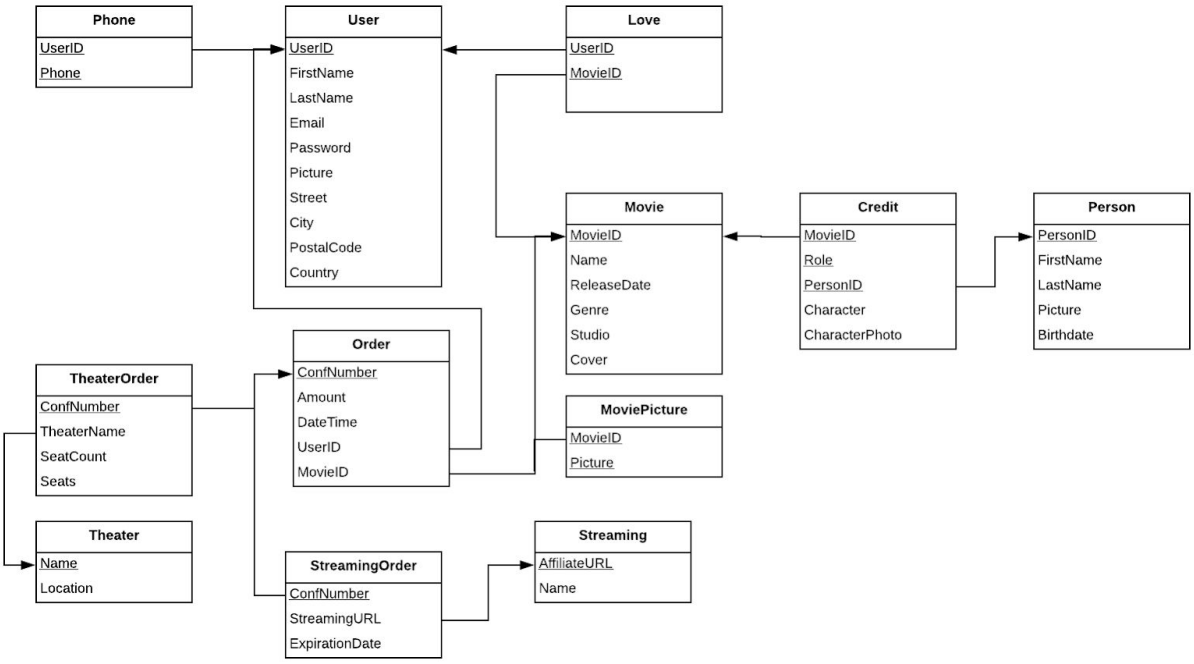
When a user orders a movie, a unique confirmation number is recorded. All orders also have a dollar amount and a date/time. Orders come from a single vendor and are for a single movie. Every order must be either a theater order or a streaming order. In addition to details that all orders have, theater orders also track the number of seats ordered and which seats were ordered. Streaming orders also track the expiration date of the movie ordered. Theater orders have an associated theater, which as a name and a location. Streaming orders have an associated streaming service, which has a name and a URL.

Every movie can be uniquely identified by its ID. In addition, movies have a name, genre, release date, studio, and any number of images. Movies must have a cover image. Movies contain credits, which indicate the role that a person serves. For actors, it is important to know the character that they played and the associated photo. Each credit refers to an individual person, who has an ID, a first and last name, a date of birth, and a photo. A role is played by a single person, but a person can play multiple roles within the same movie. Additionally, a person can be credited in any number of movies. Each credit cannot be identified on its own, but it can be identified by its role in combination with the associated movie and person.

ER Diagram


First
Last
Name
Street
City
Addr
Country
Postal Code
Pic
ID
Email
Password
Phones
User
N
Love
M
Movie
Name
ID
Release Date
Genre
Images
Studio
1
1
1
Places
Date/time
Conf #
$ amount
N
Order
N
For
Contains
N
Role
Credit
o
Character
Photo
Actor
N
For
1
DOB
First
Name
Last
ID
Person
Pic
Seat Count
d
U
U
Expiration Date
Seats
Theater Order
Streaming Order
N
N
From
From
1
N
Theater
Streaming Service
Name
Location
Name
URL

Normalized Relations



Physical Design

The database is currently in third normal form. The use cases that our users have requested thus far do not support the need for any denormalization, or the addition of any indexes beyond those that already exist on the primary keys in each table. Additionally, because the database includes ID's as numeric primary keys on some of the most important tables (e.g. User, Movie, Person), and because the most-run queries involve joining on these ID's, the pre-existing primary key indexes have been sufficiently helpful in ensuring efficiency.

As QwikTix's user base increases, and as the needs of our users evolve over time, we will consider including additional indexes or denormalizing tables as needed to ensure that our users have access to the information they need at reasonable speeds, while maintaining the ability to easily add more information to the database as new movies are released and new users register.

System Overview

a) Register a new user

First name:	John
Last name:	Appleseed
Email:	jappleseed@email.com
Password:	badpassword
Picture:	http://lorempixel.com/outpu
Street:	Elm Street
City:	Boston
Postal Code:	02120
Country:	USA

Table view ☑

Register

Success!

b) Record that a user loves a movie

User ID:	10
Movie ID:	2

Table view ☑

Love

Success!

c) Order a ticket from a local theatre

Amount:	15
User ID:	7
Movie ID:	1
Theater Name:	The Spectrum
Seat Count:	4
Seats:	3G,6J,2A,1G
Table view	☑

Order

ConfirmationNumber
21

d) Credit an existing actress for a movie

Movie ID:	6
Role:	Actor
Person ID:	3
Character Name:	Smith
Character Photo:	http://lorempixel.com/outpu
Table view	☑

Credit

Success!

e) Provide a ranked list of revenue generated from the top-10 studios

Table view ☑

View list

Studio	Revenue
Lucasfilm	185
Warner Bros	45
Marvel	20
Paramount Pictures	5

f) Find all movies directed by a person (supplied via last name)

Last name: Maas

Table view ☑

Find movies

Movie
Forest Gump
The Notebook

g) Load the cover images and names of movies ordered by a particular user

User ID: 1

Table view ☑

Find movies

CoverImage	Name	UserLastName	Userid
https://i.pinimg.com/originals/f8/4b/96/f84b961743b78e47edf22fce1cde3d2c.jpg	Forest Gump	Wang	1
http://nicholassparks.com/wp-content/uploads/2013/07/200406-the-notebook.jpeg	The Notebook	Wang	1
http://nicholassparks.com/wp-content/uploads/2013/07/200406-the-notebook.jpeg	The Notebook	Wang	1

h) Find all movies released this year that a user loves but has not ordered

User ID: 8

Table view ☑

Find movies

Name
A Quiet Place
Love Simon

i) Find all people (name, picture, and role) credited for a particular movie (supplied by name)

Movie name: Anger Management

Table view ☑

See people credited

LastName	FirstName	Picture	Role
NeJame	Robey	http://lorempixel.com/output/people-q-c-400-400-6.jpg	Set Operator

j) Provide a ranked list of revenue generated from the top-3 movie genres

Table view ☑

View list

Genre	Amount
Science Fiction	65
Action	45
Drama	45

Project Retrospective

With respect to this project, we enjoyed designing the database the most. We felt during this part we had the most creative freedom. Creating the relations, fields, and keys gave us the ability to decide how extensively we wanted to map out the system. We also found it entertaining to

design the mock data, as we could use whatever data sources we wanted. We disliked the reporting process the most because we felt that there were some simpler queries that users would have benefitted from that didn't meet the complexity requirements. Additionally, we did have some issues initially with the methods by which we were linking our front-end and our back-end. We solved this, though, by switching from Django to Flask as per the example Python code.

The most difficult tasks of this project were certainly coordinating and creating the linkages between the storage and state of our data (back-end) to the display and representation of our data (front-end). Other things we found difficult were creating an initial design that was scalable. Several times during the project, we had to revise the design of our database to develop our data further. The ERD had to be revised several times. The easier parts of the project were naturally reporting on our process and findings and creating our mock data that would populate the tables.

Doing this larger scale project, we've better learned the relations between the back and front end. We've learned the importance of creating a scalable data representation since revising the data structure in a massive data set would be extremely costly (with respect to computing power) and time consuming. Once we were able to figure this out, the results were extremely rewarding. Completing the project gave us the opportunity to put the principles of database design that we have learned throughout the semester into practice.

Conclusion

The system produced allows users to register, order movies, and retrieve a variety of information regarding movies they may be interested in. Future versions of the system would benefit from more a more stylistic user interface, as well as additional functionality. The current reports are very complex, and do not support simple operations, such as viewing all nearby theaters, or seeing all movies released this year. Future versions of the system may also allow users to leave reviews about movies for other users to read. We also intend to expand our data structures to be as scalable as possible so that as our business successfully grows, we can support a larger and larger user base without having to modify our structure.