ERD Example: Movie Ratings

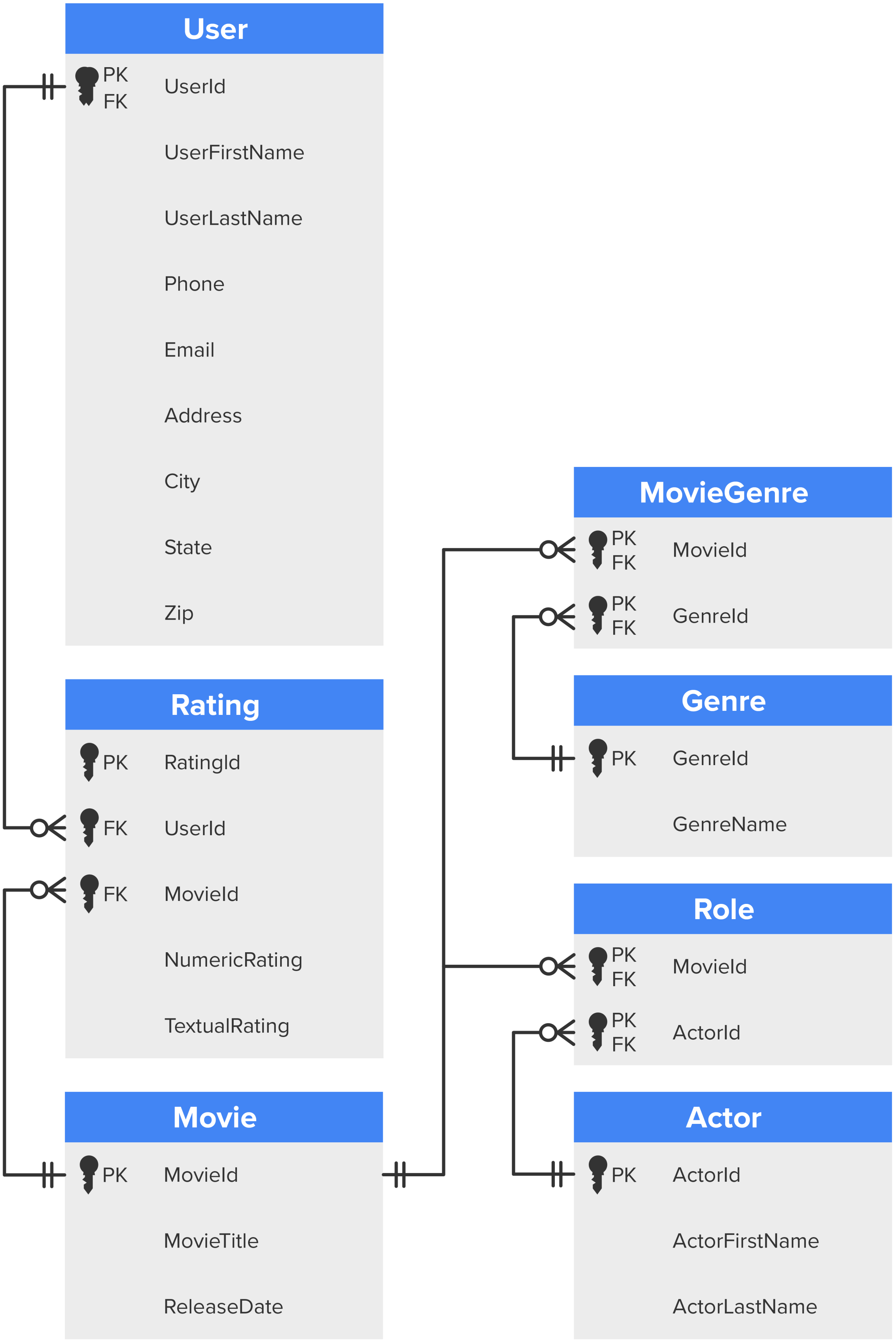
by Sophia

WHAT'S COVERED

In this lesson, you will explore a completed ERD for a movie ratings database structure, in two parts.

1. Introduction

Recall our database design for a movie ratings database from earlier challenges. While the ERD looked complex, in the end, the design fit the needs of the criteria that had been defined, and the resulting design looked like the following:



User

PK FK	UserId
	UserFirstName
	UserLastName
	Phone
	Email
	Address
	City
	State
	Zip

Rating

PK	RatingId
FK	UserId
FK	MovieId
	NumericRating
	TextualRating

Movie

PK	MovieId
	MovieTitle
	ReleaseDate

MovieGenre

PK FK	MovieId
PK FK	GenreId

Genre

PK	GenreId
	GenreName

Role

PK FK	MovieId
PK FK	ActorId

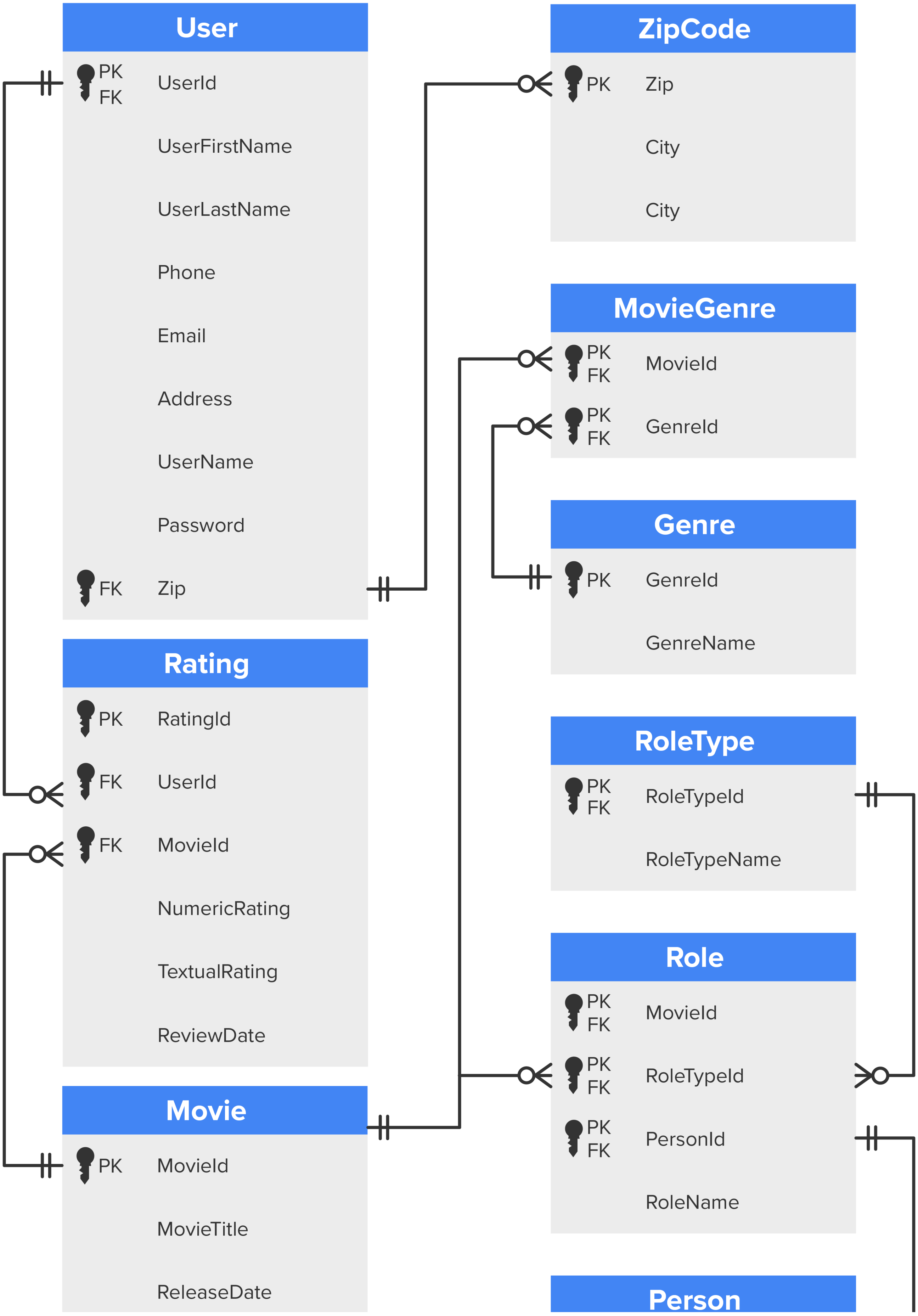
Actor

PK	ActorId
	ActorFirstName
	ActorLastName

2. Improving the Design

This database design works well, but some potential issues could arise. Understanding the business logic and processes better might lead us to expand the schema so it could track non-actor individuals who worked on each movie, such as producers, directors, and costume designers. With each type of work, it might make more sense to generalize the actor table to hold all the personnel on the movie and then add a role_type attribute and a lookup table for its possible values. This way, you could also have an individual play multiple roles within the same movie.

You might also want to reconsider the question of whether one user can review one movie multiple times and give it different ratings and reviews each time. If that is desirable, you might want to add a date and time of submission for each review. You might further want to add an average rating column to the movie table and have it be a derived value from data across all ratings for that movie. Further, in cases where one person rates a movie multiple times, you might want to include only their latest review of the movie when calculating the average; you could determine which is the latest by checking the submission date and time.



User

PK FK	UserId
	UserFirstName
	UserLastName
	Phone
	Email
	Address
	UserName
	Password
FK	Zip

Rating

PK	RatingId
FK	UserId
FK	MovieId
	NumericRating
	TextualRating
	ReviewDate

Movie

PK	MovieId
	MovieTitle
	ReleaseDate

ZipCode

PK	Zip
	City
	City

MovieGenre

PK FK	MovieId
PK FK	GenreId

Genre

PK	GenreId
	GenreName

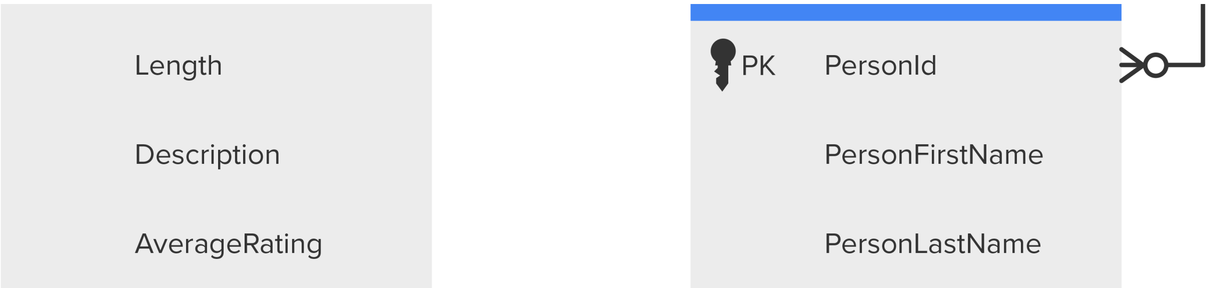
RoleType

PK FK	RoleTypeId
	RoleTypeName

Role

PK FK	MovieId
PK FK	RoleTypeId
PK FK	PersonId
	RoleName

Person



In the revised schema shown above, the role type is a lookup table, which will include all of the types of roles that an individual can play in a movie, allowing a lot more flexibility. In the movie table, there are some added columns that you would typically see in the movie. The user table has the ZIP code split off. As we discussed, this would be in third normal form (3NF), but it would not be crucial to have it implemented. Some of that decision making is dependent on a balance of redundancy over performance. The classic trade-off is between performance and data complexity.

SUMMARY

In this lesson's **introduction**, you recalled the current design of the movie ratings database created over past lessons. Then, you **improved the design** by changing some tables to be more generic, adding a few tables and values so the ERD is a more complete movie ratings database design than defined earlier. The final design represented the classic trade-off between performance and complexity when working with a business process flow.

Source: THIS TUTORIAL WAS AUTHORED BY DR. VINCENT TRAN, PHD (2020) AND Faithe Wempen (2024) FOR SOPHIA LEARNING. PLEASE SEE OUR TERMS OF USE.