Third Normal Form

by Sophia

WHAT'S COVERED

This tutorial explores converting a database design from the second normal form (2NF) to the third normal form (3NF) in three parts:

1. Requirements for Third Normal Form
2. Movie Ratings Example
3. Applying Third Normal Form

1. Requirements for Third Normal Form

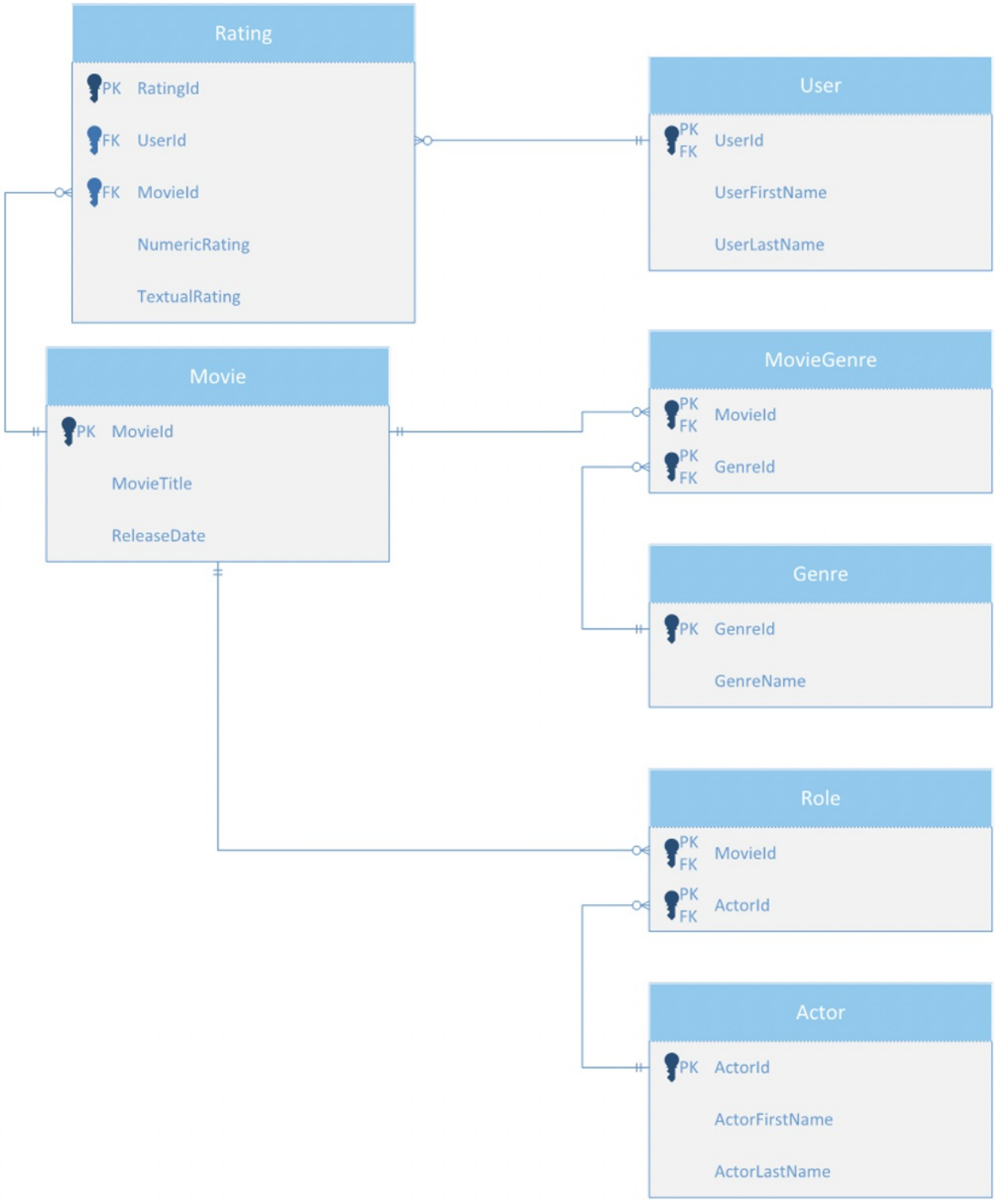
The third normal form (3NF) is generally the final stage of most normalization processes. It requires that the database design first fulfills the requirements of the second normal form (2NF), but also ensures that there is no transitive functional dependency.

BIG IDEA

This means that every attribute that is not the primary key in a table must depend on the primary key, *and only the primary key*. In other words, imagine that column A determines column B, and column B determines column C. You would have a transitive functional dependency because column A then determines column C. To resolve this issue, column C should be removed and placed in a separate table. You need to check this for each column in all of our tables.

2. Movie Ratings Example

Here is our database design that was in second normal form (2NF):



In our case, none of the non-primary key fields depend on something else other than the primary key. So, in essence, after 2NF, our current database design is actually in 3NF. This outcome is quite common.

3. Applying Third Normal Form

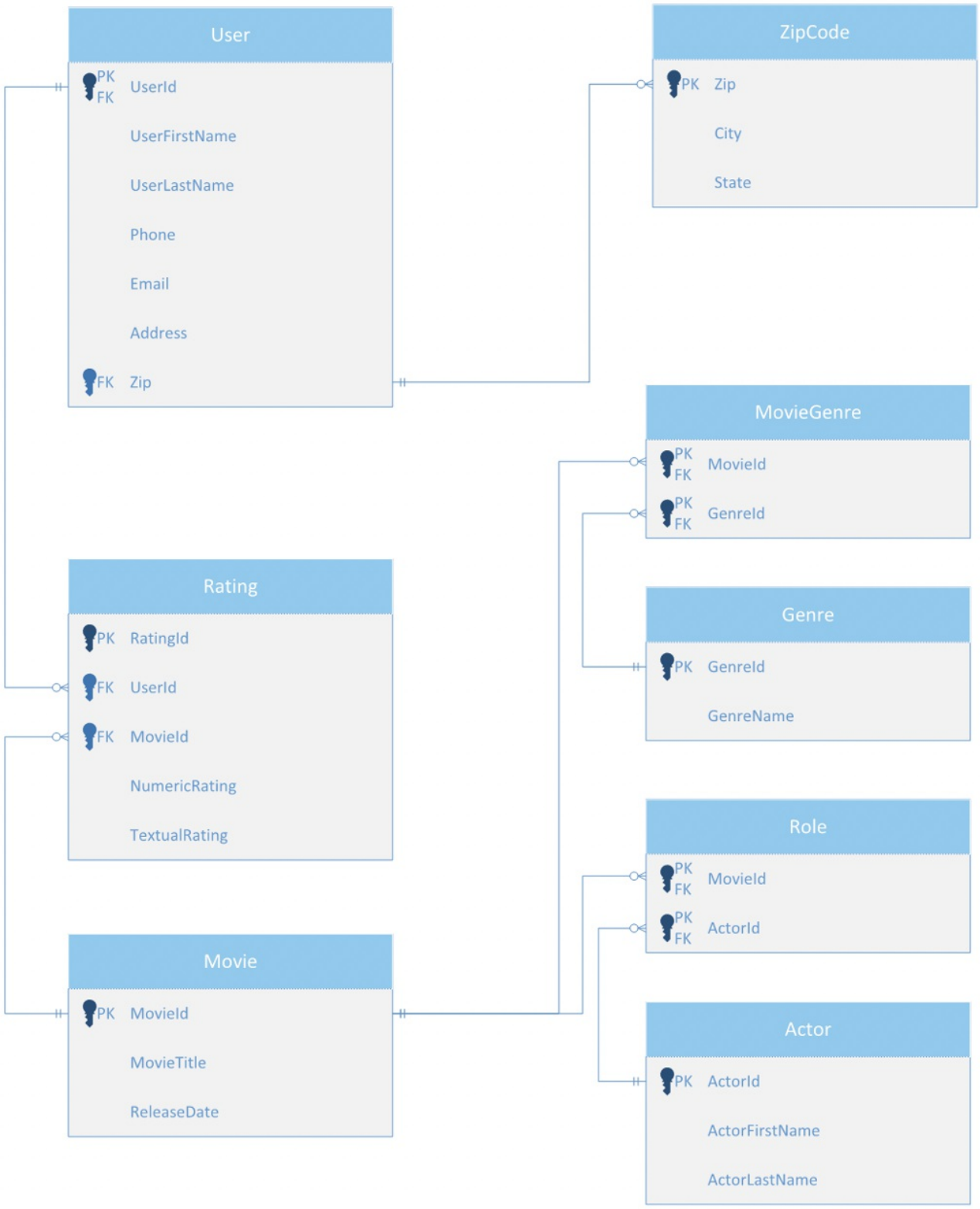
However, imagine you wanted to expand the User table to include some additional details about the user, such as the email, phone number, address, city, state, and zip code. Let's assume that all of the users are in the United States. By expanding it in such a way, our User table would now look like the following:

- User (UserId, UserFirstname, UserLastName, Email, Phone, Address, City, State, Zip)

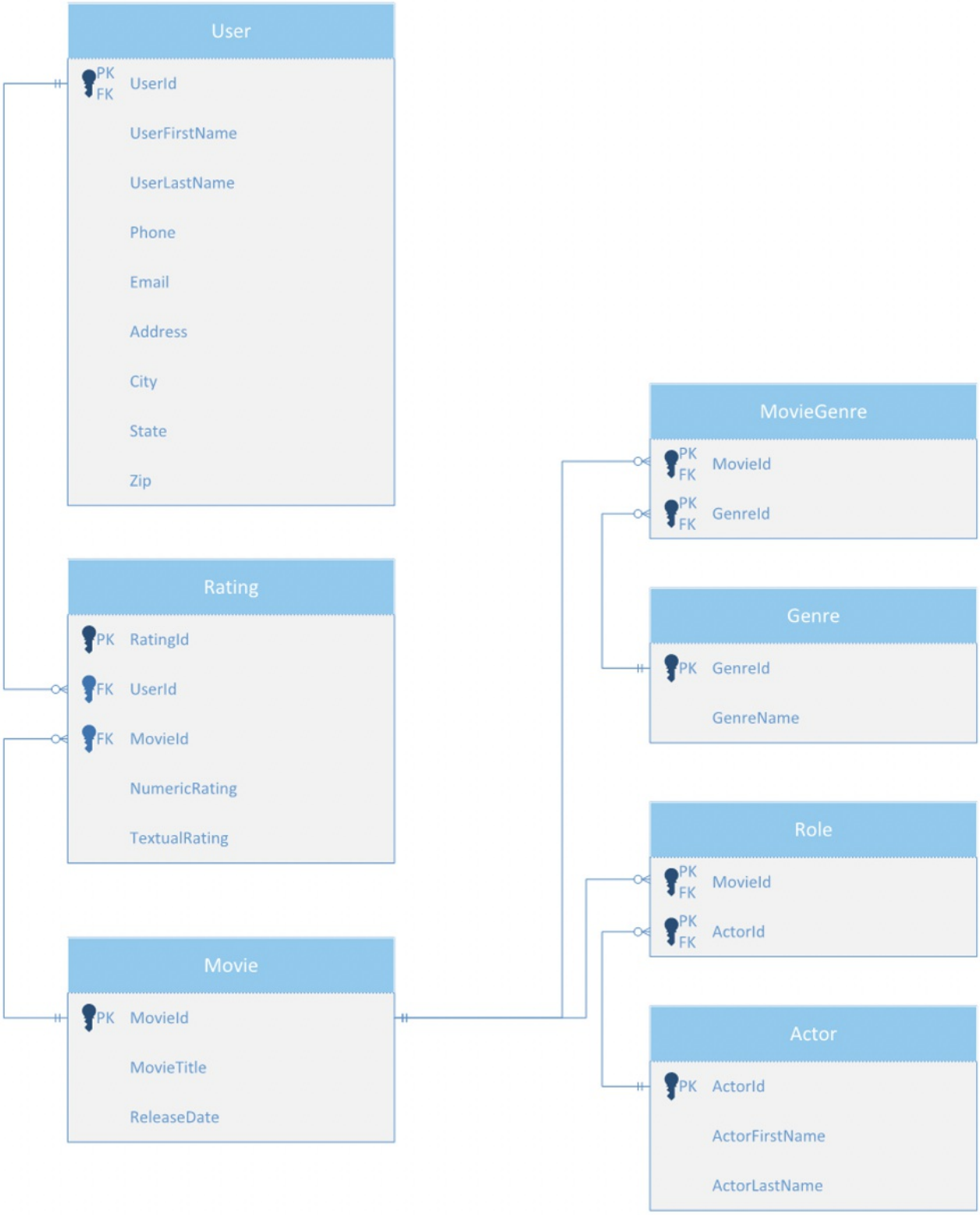
Looking at this table now, you can see that the City and State are determined by the Zip. This is a transitive functional dependency, because the Zip is dependent on the UserID, and the City and State are dependent on the Zip. To convert this table to 3NF, you would need to move this data into a separate table, like this:

- User (UserId, UserFirstname, UserLastName, Email, Phone, Address, Zip)
- ZipCode (Zip, City, State)

The Zip in the User table would link to the Zip in the ZipCode table. This way, any time that the zip code was entered in, the city and state would automatically be established. If you had to add an address in any other table later on, you could use the same ZipCode table as a lookup table. The resulting ERD in 3NF would look like the following:



Note, though, that the zip code itself is not commonly split up on its own even with it being a 3NF rule as there are instances where a zip code in the US can reflect different cities. If you plan to have this setup for any country, this may not apply either in which case the following ERD would make the most sense:



Depending on the requirements, this will be our final normalized database design in third normal form (3NF).

SUMMARY

In this tutorial, you learned the **requirements for the third normal form** (3NF) will ensure that the database fulfills all second normal form (2NF) rules and handle transitive functional dependency. You saw that after the normalization to the second normal form (2NF), our **movie ratings example** is

actually in the third normal (3NF) form already since none of the non-primary key fields depends on something else than the primary key. Finally, you learned how to **apply third normal form** (3NF) rules while expanding the User table to include additional details.

Source: Authored by Vincent Tran