ERD Example: Complexity

by Sophia

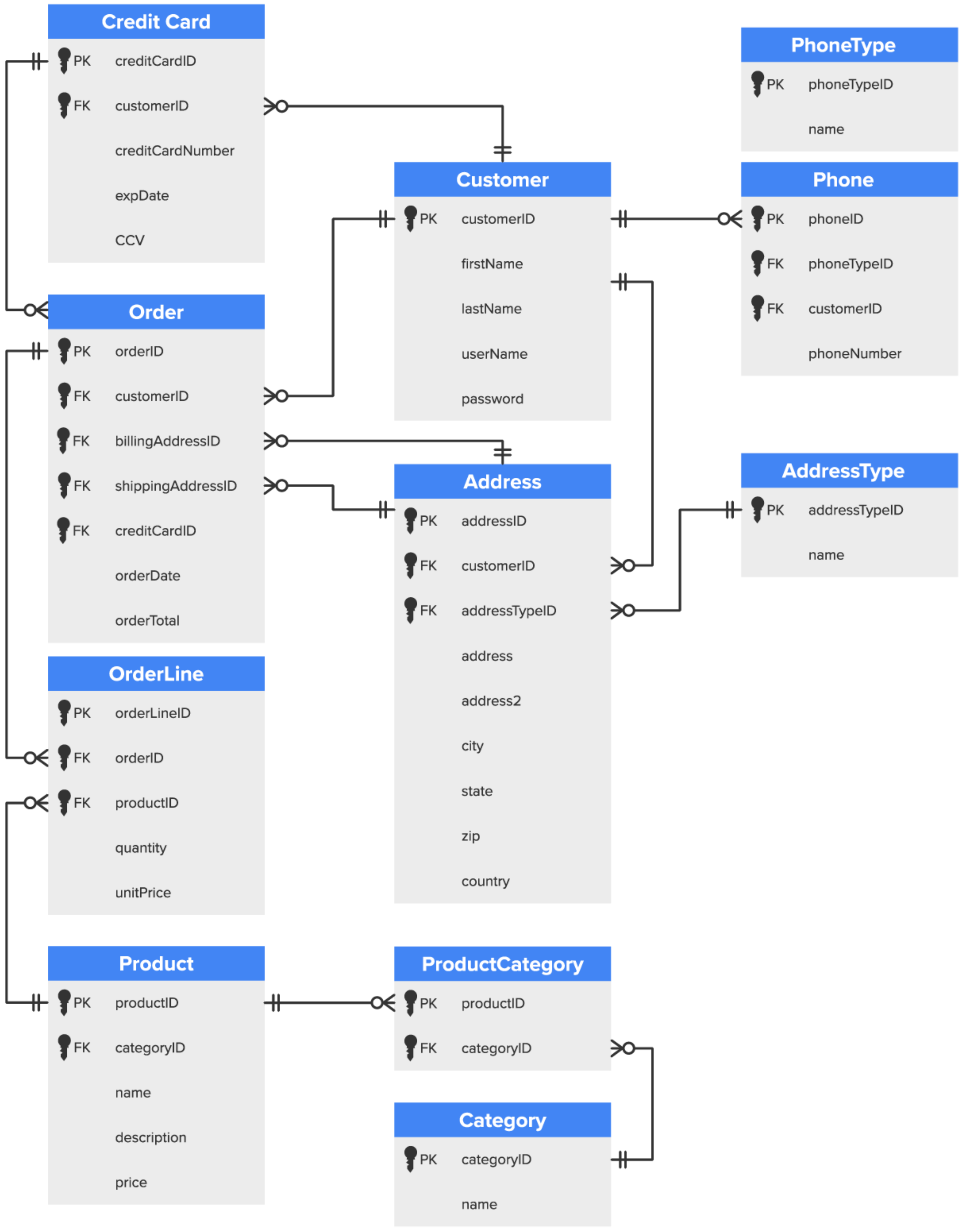
WHAT'S COVERED

In this lesson, you will explore the issues with an overly complex ERD, in two parts. Specifically, this lesson covers:

1. Flexibility Makes Complexity

In the previous lesson, you learned about a simple ERD but then made it fairly complex to the point where it might have problems with performance. In this lesson, you will learn how to manage an overly complex ERD, with performance in mind.

As a reminder, here is the ERD from the last lesson after the changes were made to it:



Some of the key issues that were identified and fixed in the previous lesson included:

- Initially, there was no way to have separate billing and shipping addresses. This was corrected by adding separate billing and shipping addresses to the Order table and having an AddressType table to create a list of address types.
- Initially, there was no way to enter multiple phone numbers for the customer. This was corrected by adding a PhoneType table for a type lookup and linking each phoneID to a customer.
- Initially, a customer could have only one stored credit card. A CreditCard table was added that enables multiple cards to be stored.
- Initially, there was no way to tell what price a customer paid for a product because the product price was stored separately from the order. This was corrected by storing the orderTotal in the Order table as a discount that could be used later.
- Initially, there was a many-to-many relationship between Product and Category. This was resolved by creating a ProductCategory table that mapped the relationship as two one-to-many relationships.

KEY CONCEPT

When considering the ERD, when you have a simple business, or one that relies on the shipping and billing addresses being the same, you have to ask yourself if you really need a more complex design.

EXAMPLE The database should be structured to optimally support the business's processes. A **business process** is a set of activities performed to achieve a specific business goal, including the inputs, outputs, and interactions involved. For example, say you only have one customer, like some suppliers for automotive parts who are wholly owned by the manufacturer. In this situation, billing and shipping might always be the same. It might also make sense that there is only one phone number as a contact number. When we start going into the business process, we can start determining if the database design is optimized for the business workflow. It also means that it might not make sense for the organization to have the level of complex database design that we introduced in the previous lesson.

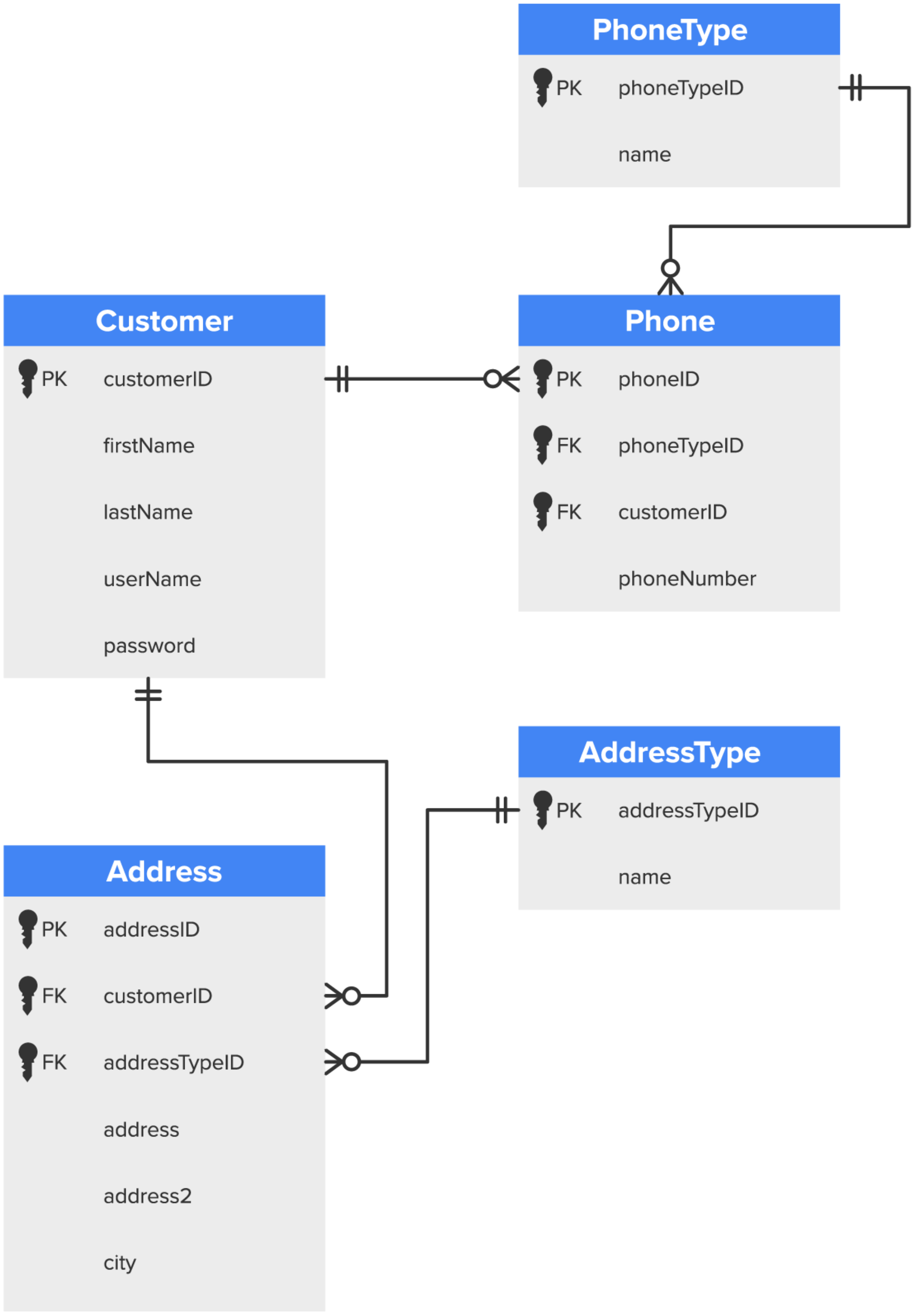
You must make such determinations early on by understanding the business process as much as possible. Remember that the database is meant to support the business and information flows between business process segments.

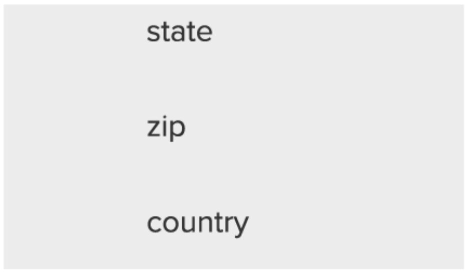
HINT

The more complex the database design, the more work it is not only to join the data but to create the underlying application code to make use of these tables.

Let us look at the customer table to compare a complex and simple data model, given the business rules you have defined.

You will see that in the complex design, the customer table is simple, with separate tables to list phone numbers and addresses, allowing as many values as possible. In reality, most customers only store a single shipping address, billing address, and phone number for a vendor. Some vendors will have multiple suppliers, but once the order goes to the vendor, it is a vendor process to ship from the closest warehouse, making one shipping and billing address as an answer viable in this design.


PhoneType
PK phoneTypeID
name
Customer
PK customerID
firstName
lastName
userName
password
Phone
PK phoneID
FK phoneTypeID
FK customerID
phoneNumber
AddressType
PK addressTypeID
name
Address
PK addressID
FK customerID
FK addressTypeID
address
address2
city



state

zip

country

The database design should reflect the way the business actually operates and the **business use cases** likely to be encountered. The more flexible the database is from the start, the less likely it is that you will need to restructure it later, but it's also important for performance reasons to avoid needlessly adding complexity. One way to find the optimal balance is to think about the use cases of different customers, orders, and payment methods that the company has had in the past—or may have in the future—and make sure that the database schema can accommodate all the information you might want to store about each of those use cases.

TERMS TO KNOW

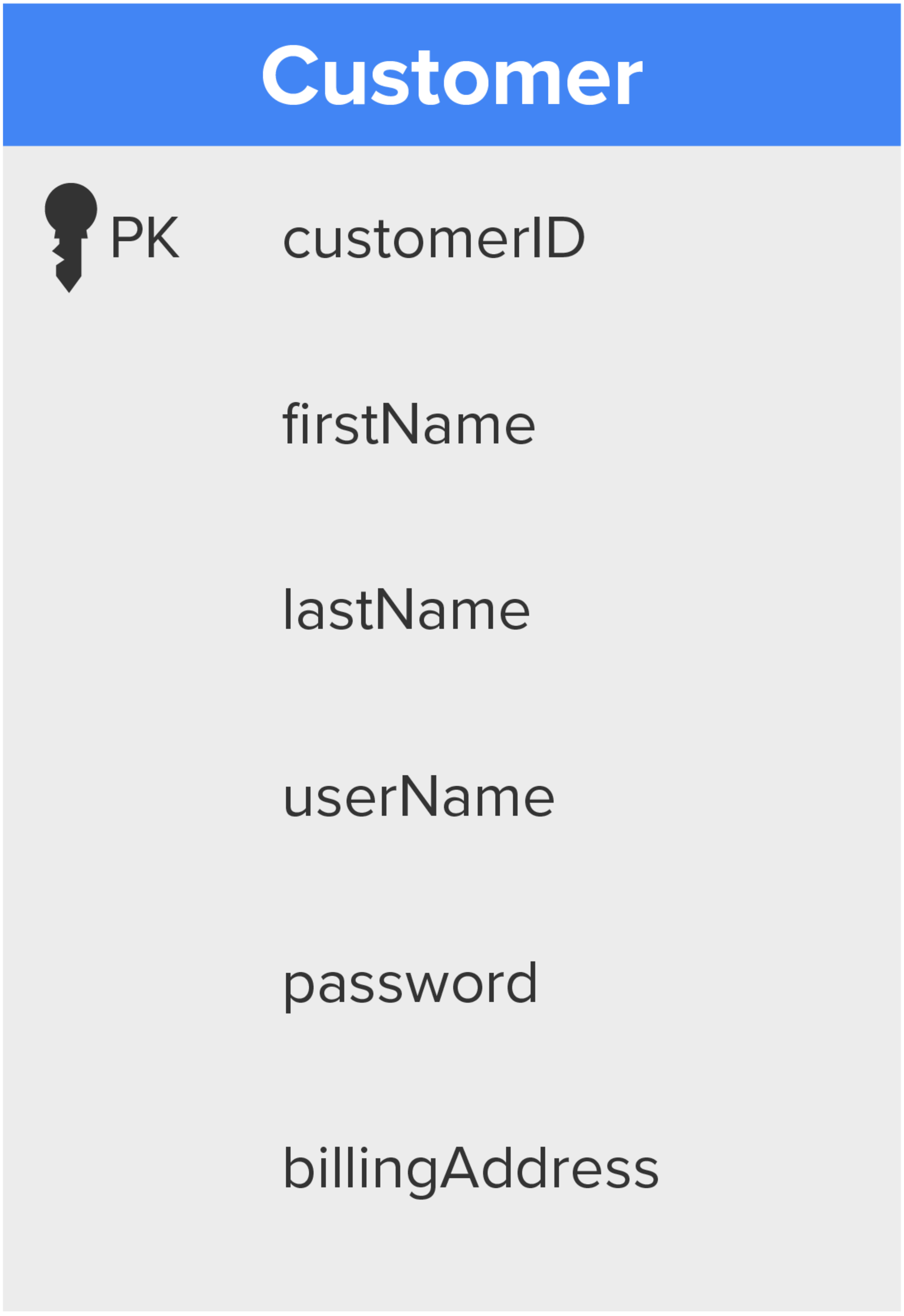
Business Process
A set of activities or tasks that are performed in a coordinated manner to achieve a specific business goal.

Business Use Case
The business use case describes the sequence of actions that the business must take to deliver a meaningful, observable result to the end user.

---

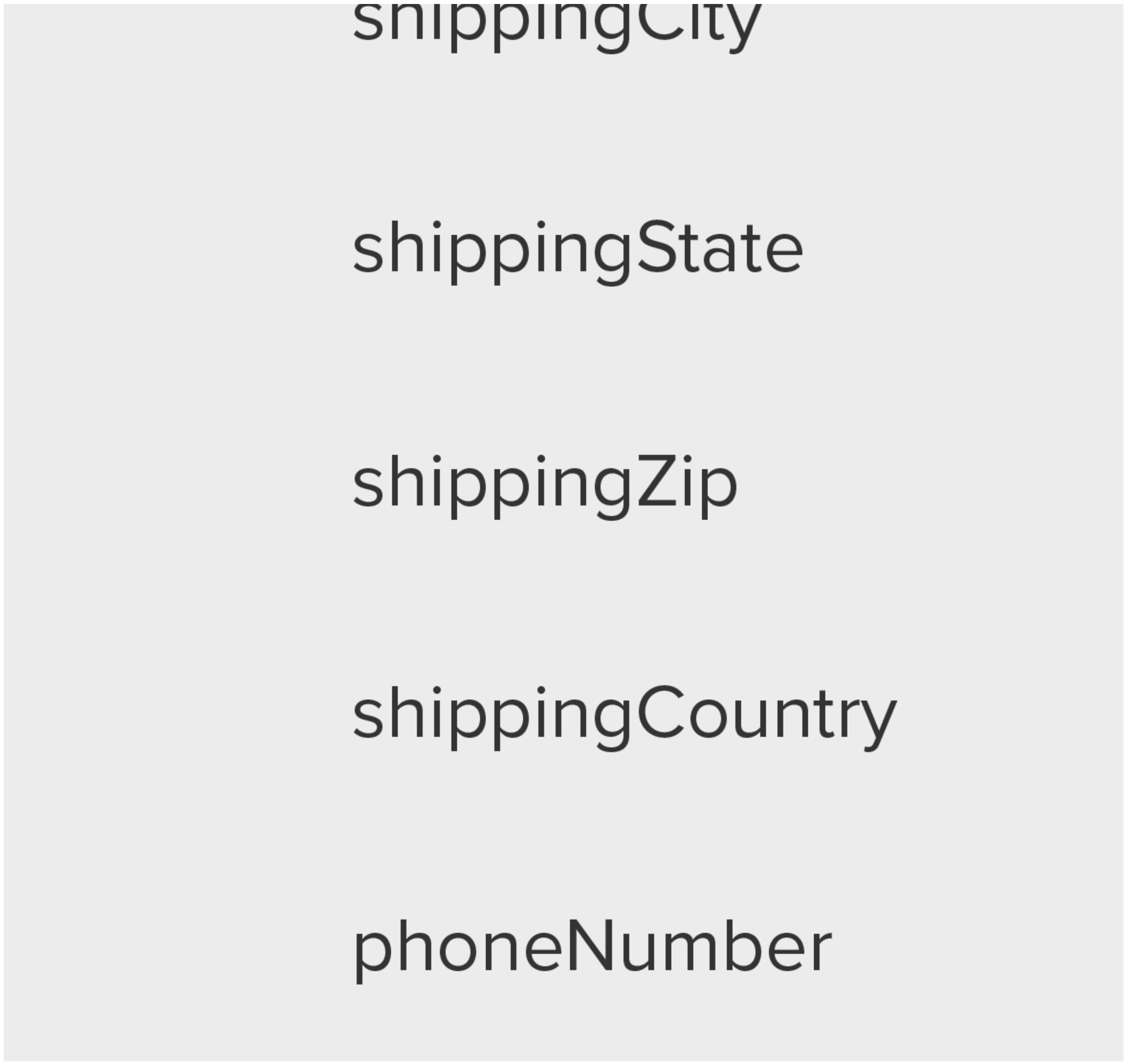
2. When to Keep It Simple

Depending on the use case, the business process might call for a simpler scenario; where there is not as much repeat business, a single table could suffice for the customer. For example, the following table might be more appropriate. It has more attributes, but they're all contained in a single table.



Customer	
PK	customerID
	firstName
	lastName
	userName
	password
	billingAddress

```
billingAddress2

billingCity

billingState

billingZip

billingCountry

shippingAddress

shippingAddress2

shippingCity
```


shippingCity

shippingState

shippingZip

shippingCountry

phoneNumber

The organization's business processes will determine whether to use a simple or complex design. If you focus on having full flexibility, you will introduce a lot more complexity in the database design and all other components that link to the database. You will also have processing, speed, and storage requirements that you need to factor into your database design. The data model may adapt more easily for scalability than a simpler model would, especially if business rules change, and it will reduce redundancy.

A simpler data model that reduces complexity will make queries, reports, and application code simpler to create. It allows for faster application development, simpler application code to maintain, and simpler SQL queries. You can use this type of structure to simplify the data model. For example, although in 3NF, you would typically split off the ZIP code, city, and state in its own table, it's common for simplicity with reporting and functionality to include those attributes in the same table as the rest of the address.

SUMMARY

In this lesson, you learned that **flexibility creates complexity** since more complex designs add more work, not only to join the data but to create the underlying application code to use all tables included. You also learned that it sometimes makes sense to **keep it simple**. Based on the business need, for example, a simple single table would suffice if not much repeat business was expected. Having overly complex database designs can create a lot more work, so it is best to design to the business need.

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

TERMS TO KNOW

Business Process
A set of activities or tasks that are performed in a coordinated manner to achieve a specific business goal.

Business Use Case
A detailed description of how a system will be used to achieve a specific business goal in a certain scenario.