Logical Design

by Sophia

WHAT'S COVERED

This lesson explores the next step in designing a new database. We will consider how to extend the conceptual model into a logical model design, in two parts. Specifically, this lesson will cover:

1. Moving From Concept to a Logical Model
2. Steps to Create a Logical Model

1. Moving From Concept to a Logical Model

Once the conceptual model has been created, the **logical model** is used for database design. In a logical model, the conceptual model is further refined by transforming it into a representation that is closer to the actual implementation of the database management system (DBMS). For example, a logical model is normalized and specifies primary and foreign keys for relationships.

BIG IDEA

Logical models are blueprints for how database data will be arranged and related. Data modeling allows designers to understand data requirements, entity relationships, and data manipulation operations needed to support an application. This model specifies technical implementation details for a specific DBMS between the conceptual and physical models.

An entity's logical model defines its structure, including its tables, attributes, and relationships with other entities. This ensures that data is stored and retrieved efficiently by accurately representing the application's requirements. Data integrity constraints are incorporated into the logical model to ensure data consistency and accuracy. In order to maintain data quality and prevent data anomalies, logical models define relationships between entities and ensure referential integrity.

Logical models facilitate query optimization by considering the types of queries that are likely to be run on the database. The logical model can be optimized by understanding the patterns of data access and query requirements in order to enhance query performance and speed up data retrieval. The designer can also eliminate redundant data and improve data efficiency by using a logical model to facilitate **normalization** techniques. Data normalization reduces duplication and ensures consistency by replacing tables that contain redundant or unrelated information with smaller, focused tables with relationships between them.

Logical models are flexible and extensible since they don't depend on any specific database management system (DBMS). This decoupling allows the database to be migrated with minimal changes to different DBMS platforms. Logical models facilitate communication and collaboration between stakeholders, designers, and developers regarding database requirements by visually representing the database design. The data model acts as a common reference point for discussions and ensures everyone is on the same page.

As logical models define entities, attributes, and relationships, designers can identify potential security vulnerabilities and make appropriate access control plans to secure sensitive information. Integrity and consistency of data are maintained by the logical model, which defines the rules and constraints for doing so. Database errors and inconsistencies can be prevented by ensuring data is entered and managed correctly.

HINT

The logical model bridges the gap between the conceptual and physical implementations in the database design process. In addition to improving data integrity and consistency, it provides a detailed and structured representation of data requirements, relationships, and constraints.

By the end of the logical model design process, all of the entities, attributes, and relationships are defined. The primary key for each entity and the foreign key that links the tables together are specified. Normalization—which you will learn more about in a future lesson—is also performed at this level. Finally, the characteristics such as data location, path, and format should all be included.

TERMS TO KNOW

Logical Model

A high-level structure and technical map for a database that specifies primary and foreign keys and has been normalized.

Normalization

The process of applying design rules to a database to ensure that its table structures minimize duplication and ensure efficiency and integrity.

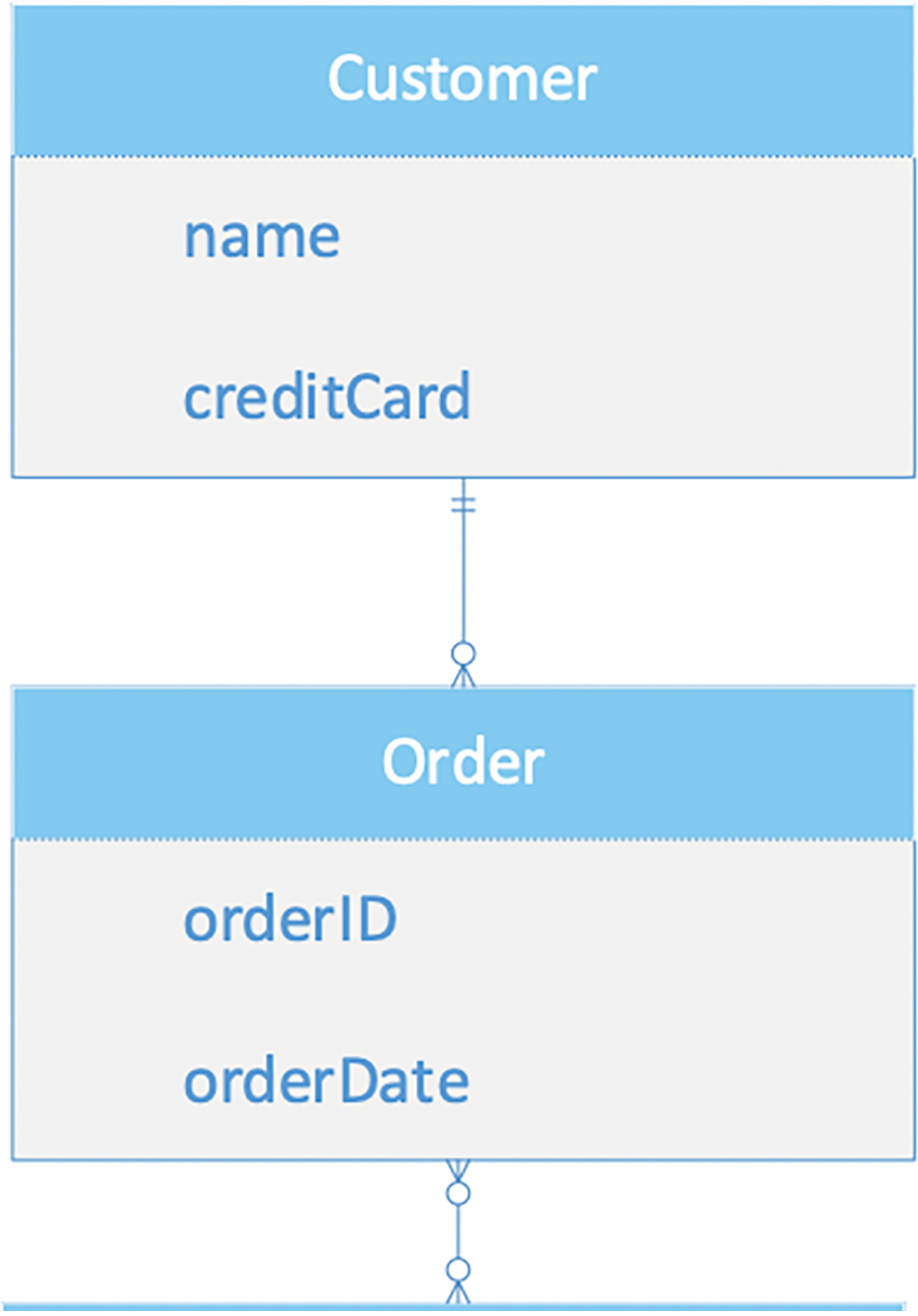
2. Steps to Create a Logical Model

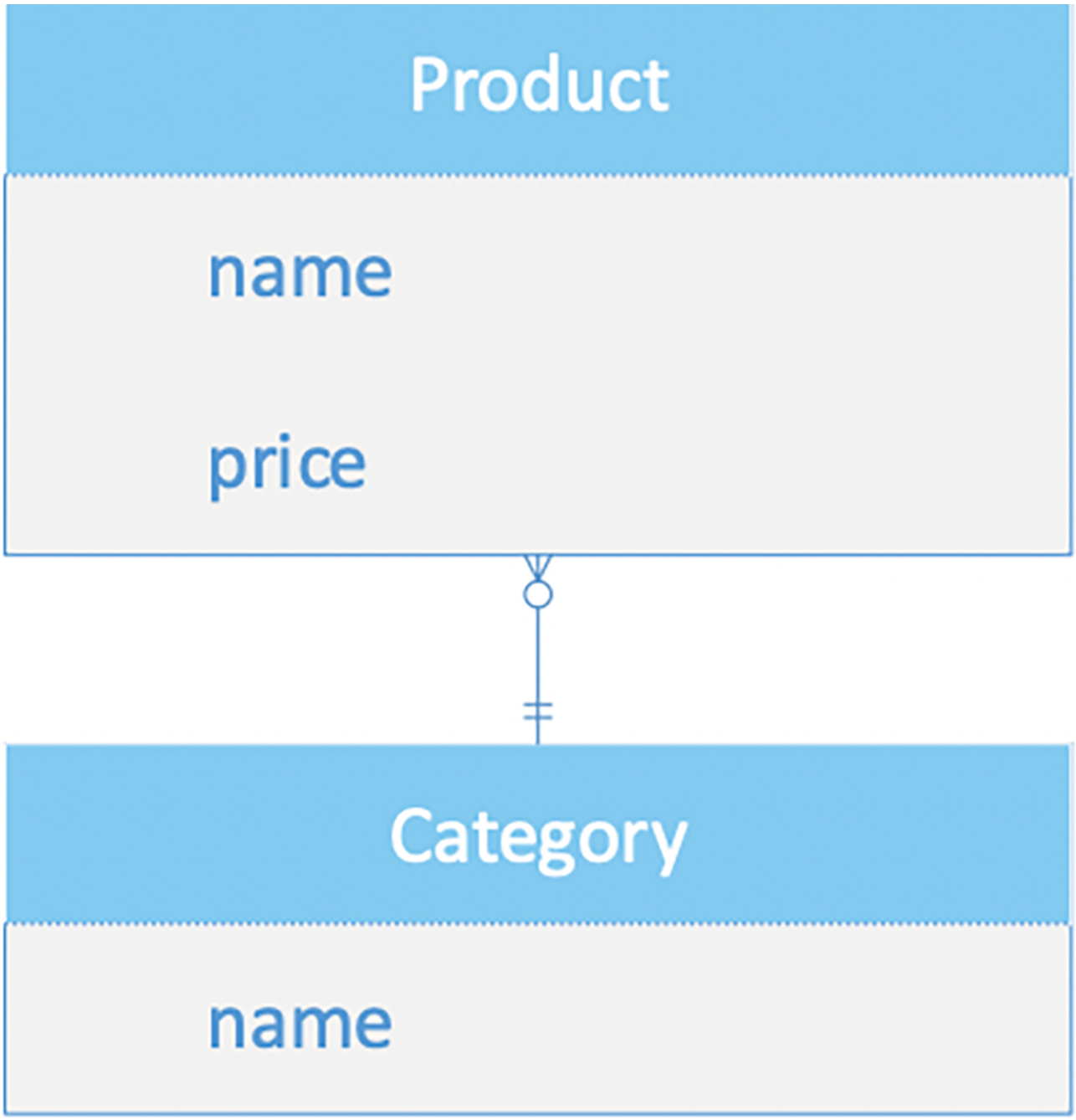
The first step in this process is to map the conceptual model to the logical model. This involves mapping out the strong entities, supertypes/subtypes, weak entities, binary relationships, and higher-degree relationships.

Logical Model Elements	
Strong Entities	An entity that is not dependent on any other entity in the schema. A strong entity will always have a primary key.
Weak Entities	An entity that doesn't have sufficient attributes to require its own primary key. It depends on a strong entity. One way to normalize a many-to-many relationship between two strong entities is to create a weak entity that has a one-to-many relationship between each of the strong tables involved.

Supertypes/Subtypes	A supertype entity has smaller groups (subtypes) that connect to it but add specialized attributes common to the specific subtype, while the supertype has attributes that are common to all of the subtypes. For example, a supertype might be “People,” with a subtype for “Employees,” “Vendors,” and “Customers.” A subtype is an entity that derives some of its attributes from a more general (supertype) entity.
Binary Relationships	The type of relationship between two separate entities. When a binary relationship exists, it can be one-to-one, one-to-many, many-to-one, or many-to-many.
Higher-Degree Relationships	A higher-degree relationship is one that occurs between more than two entities. For example, a ternary relationship can occur between three entities. Larger numbers of relationships are possible but have yet to be recommended.

The strong entities are ones that are on the “one” side of a binary relationship within the data model. Review the conceptual model. Which of the following would you consider strong entities?





In this example, the strong entities are the customer and category entities. These are indicated by the use of the two perpendicular crossed lines near the entity, which indicate a “one,” as opposed to the three prongs that indicate a “many” relationship. This example uses crow's foot notation.

Once you have the strong entities mapped out, the second step is to move on to the supertype/subtype relationships and the weak entities. In our example, we don’t have any supertype/subtype relationships.

IN CONTEXT

Imagine you have a Pet supertype entity. A subtype could be a Dog or a Cat entity with specific attributes associated with them.

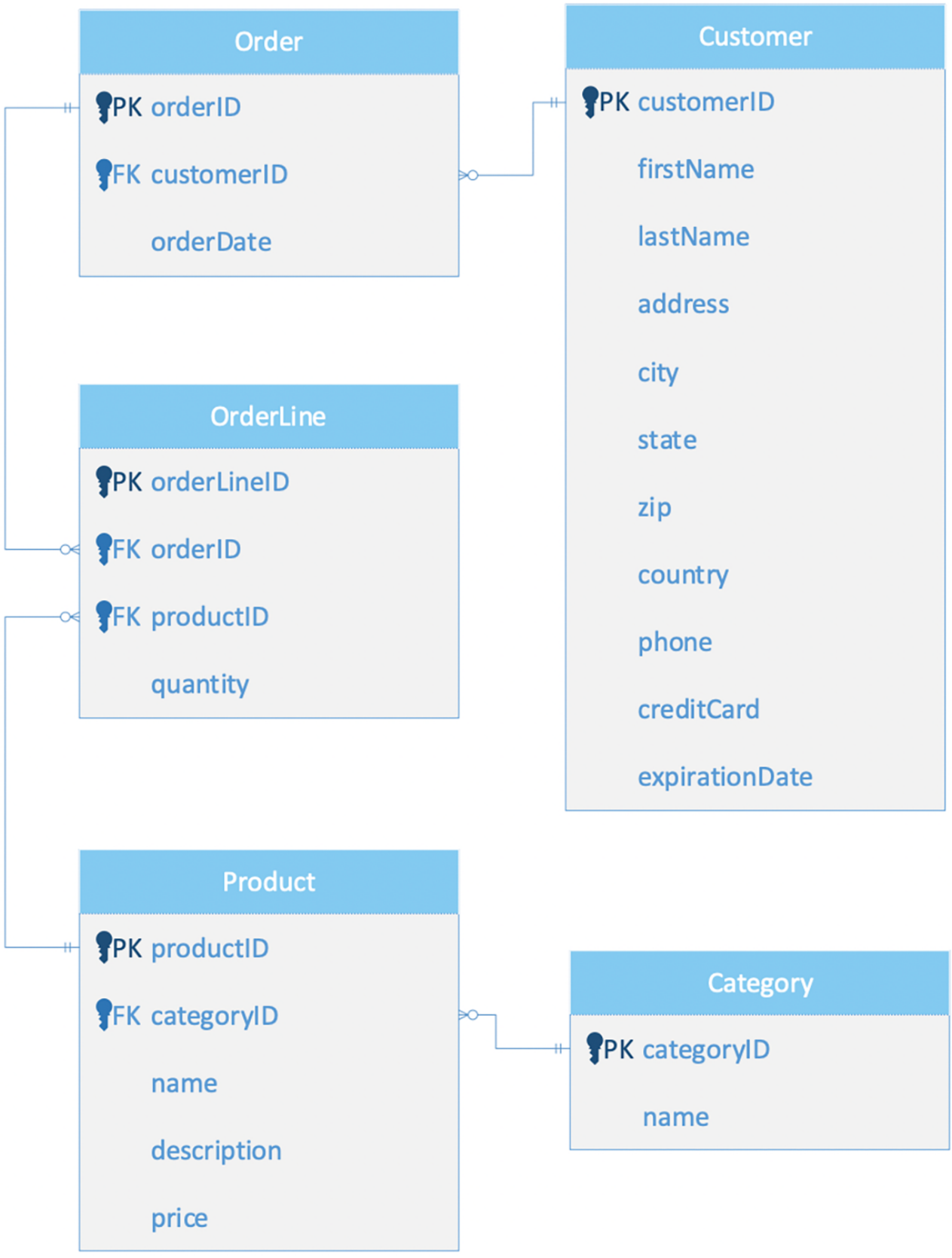
A weak entity is one whose existence depends on a strong entity. In the same example of a Dog or a Cat entity, both of those would be weak entities, as they would inherit the primary key from the Pet entity to use as a foreign key. They would also be dependent on the Pet entity to exist; that is, a record in the Dog table could not exist without a related record in the Pet table.

Next, you would work through all of the binary relationships between two entities. For example, you would map out the one-to-many relationships between the product and category tables, and the customer and order tables. Once you have those relationships mapped, you would move on to define the relationships between three or more entities until all of the relationships are defined. In our example, we had a many-to-many relationship between the order and product tables that needed to be resolved. We will get into that level of detail in a future lesson. For now, make note that any many-to-many relationships in a conceptual model must be resolved by having a bridge in the form of an intermediary table that creates two one-to-many relationships.

At the logical model level, we also define the primary keys and foreign keys as part of the relationships between entities and validate the model through normalization. We typically also define the integrity constraints of the data, if needed. For example, we may define that the quantity of a product in inventory must be an integer and must be greater than zero.

Typically, we won't define the data types and sizes in the logical model, as this database design stage is not meant to be database system specific. However, it is acceptable to generalize the data types in the logical model—for example, identifying which attributes are a "number" data type versus a "text" data type.

Here is an example of the logical data model for the e-commerce company that we have been using:



SUMMARY

In this lesson, you learned that the **logical model expands on the conceptual model** to include all entities and relationships between entities. This model includes all attributes for each entity, defines the primary key of each entity, and adds the foreign keys to identify the relationships between the entities. You also learned about the **steps to create a logical model**. Normalization is also performed on the data model during the logical design stage, and any many-to-many binary relationships are handled by splitting them into new one-to-many binary relationships.

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

Logical Model

A high-level structure and technical map for a database that specifies primary and foreign keys and has been normalized.

Normalization

The process of applying design rules to a database to ensure that its table structures minimize duplication and ensure efficiency and integrity.