Conceptual Design

by Sophia

WHAT'S COVERED

This lesson explores the steps in designing a new relational database. This lesson explains the first step. Each of the other steps will be covered in upcoming lessons. Specifically, this lesson will cover:

1. Conceptual Model Design
    1a. Gathering Requirements
2. E-Commerce Example

1. Conceptual Model Design

Conceptual data models are high-level representations of a system's data requirements and structure, independent of its specific database management system or technical implementation. This document is typically created during the early stages of database design to capture the essential entities, attributes, and relationships that define the application's information needs.

During the conceptual design process, the primary focus is on understanding a company's or application's requirements and how data entities are related to each other. At this point in the development process, it is unnecessary to consider how data will be accessed or stored. Rather, it provides stakeholders, analysts, and designers with a clear and abstract view of the data, enabling them to communicate and agree on the fundamental data organization more effectively.

Conceptual data models are often represented using **entity-relationship (ER) diagrams** or other visual modeling techniques, enabling stakeholders to visualize and validate data designs before moving on to logical and physical designs. Conceptual design models guide the development of logical and physical database schemas, ensure alignment with business needs and objectives, and serve as a foundation for refining and developing database designs.

There are some basic design steps performed during the conceptual design phase.

The basic steps in this process are:

1. Gather Requirements	Understand and document the business or application requirements the database should fulfill. This involves consulting stakeholders, end users, and domain experts to identify the necessary data entities, attributes, and relationships.
2. Identify Entities	Identify the main data entities or objects relevant to the application or domain. Entities represent real-world objects or concepts that must be captured and stored in the database.
3. Define Attributes	Determine the attributes (properties) of each entity, describing the specific characteristics and data elements that need to be stored for each entity.
4. Establish Relationships	Analyze relationships between entities. Identify associations, dependencies, and connections between entities that describe how they relate to each other.
5. Create an Entity-Relationship Diagram (ERD)	Represent entities, attributes, and relationships using an entity-relationship diagram (ERD). This visual representation helps stakeholders and designers better understand the data structure and relationship between entities.
6. Normalize the Model	Apply normalization techniques to ensure data organization efficiently, minimizing redundancy and anomalies. Normalization helps achieve a more scalable and maintainable database design.
7. Review and Validate	Review the conceptual design model with stakeholders and subject matter experts to ensure that it accurately reflects the requirements and business rules. Make necessary adjustments and refinements based on feedback.
8. Document the Model	Document the conceptual design model, including descriptions of entities, attributes, relationships, and any critical design decisions. This documentation will serve as a reference for future database development.

TERMS TO KNOW

Conceptual Data Model

A high-level representation of a system's data requirements and structure that establishes the entities, their attributes, and relationships between entities in a relational database.

Entity Relationship (ER) Model

A visual representation that illustrates the structure and relationships within a database or data model.

1a. Gathering Requirements

The first step in conceptual model design is to discover the necessary characteristics of the data elements. This is the most consequential part of database design, as issues here will show up all the way through the design process and can impact performance, data translation, and database usage.

We start this process by posing some questions to help frame the type of data we will be gathering:

REFLECT

- Where is the data going to come from?
- Is the data structured or unstructured (SQL, NoSQL)?
- How will the data be used? Will it be mostly read for reports, will it be updated frequently (for example, using sensors to measure values every few minutes), or will it be a combination of the two?
- Who will use the data, how often will they access the data, and how will they use the data?
- What are the various end-user data views going to be?
- Where is the information to be found, and how is the information meant to be extracted?

Using questions like this, we can then start framing what the database should look like, how it will be used, who will use it, and where the data will come from. This will help us determine what data elements are needed to produce the information, along with what the data attributes should be. We can also define the relationships that exist in the data, how frequently the data is to be used, and what any data transformations may need to be to generate some of that information.

HINT

It can be useful to have the database designer and the end users create a description of what the end-user data views and reports are going to be. Designers and users working together will help identify what the main data elements are and how they will be used. It can also be useful to look at any existing manual process or application, if there is one, and if the database is going to be used to automate those manual processes. This is a big part of one of the latter steps in the process: “Review and Validate.”

If there’s already an automated system in place, we can also take a look at what the current reports look like, as well as determine what the designed reports would consist of for the new system. If an application is being developed in conjunction with the database design, it would also be useful to review the application designs for insight into the data elements.

Business rules can be of benefit when working out what the new system should do and where the data is coming from. Data from manufacturing is different from data coming from research, for example, or there may be other reasons why the system is being built. That knowledge is important for determining the entities, attributes, relationships, connectivity, and constraints in the database. Business rules need to be simple and precise and follow the business process from start to finish.

Later in the course you'll work with a Postgres database that has a number of tables about artists, music, and tracks. Here are some business rules that might be appropriate for that data:

- An artist can record more than one album.
- An album can list only one artist.
- An album can store multiple tracks.
- A track can belong to only a single genre.
- A track can be sold under only a single media type.
- An employee can be the main support representative of many customers.
- A customer can only have one main support representative.

Business rules help define what the entities are and what the key attributes and relationships will be. They also, in many cases, start to define the **cardinality** between the entities—in other words, the one-to-one, one-to-many, or many-to-many relationships. It is important that business rules are accurate and vetted by the business end users and stakeholders. Having incorrect business rules can create underlying problems in the database that will affect the applications that rely on it.

TERM TO KNOW

Cardinality

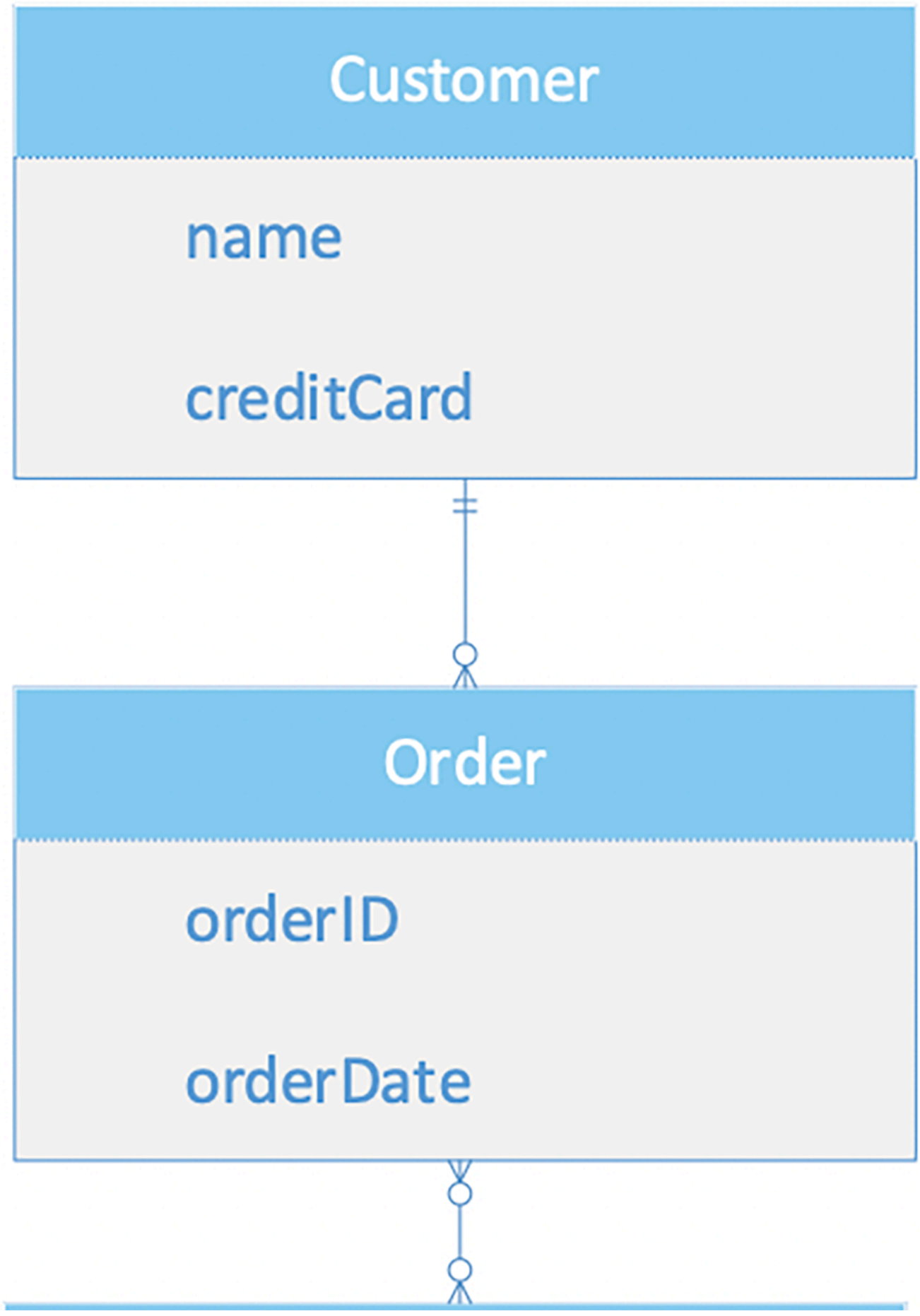
The number of instances of one entity (or table) that can be associated with a single instance of another entity (or table) through a relationship.

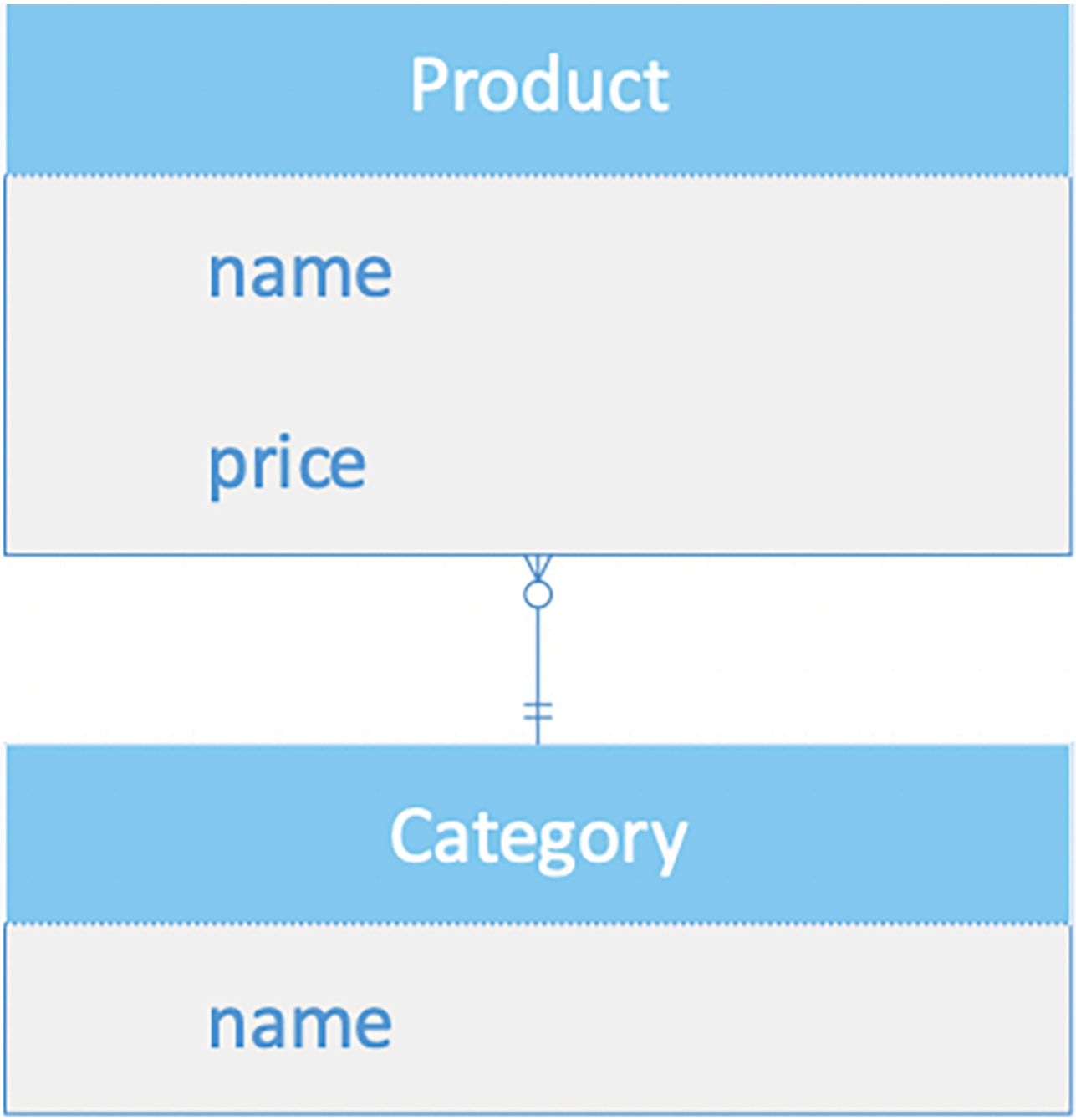
2. E-Commerce Example

Let's take a look at what a conceptual model may look like for a simple e-commerce site. The business rules are as follows:

- A customer can place many orders.
- A customer uses a credit card to pay for the orders.
- An order can only belong to a single customer.
- An order can consist of many products.
- A product can be purchased by many customers.
- A product can belong to a single category.
- A category can consist of many products.

A sample entity-relationship diagram would look like the following:





Based on these business rules, we see the following relationships:

- Between a customer and an order is a one-to-many relationship.
- Between an order and a product is a many-to-many relationship.
- A category to a product is a one-to-many relationship.

SUMMARY

In this lesson, you learned that **conceptual models** are high-level representations of a system's data **gathering requirements** and structure, independent of its specific database management system or

technical implementation. The conceptual design document is typically created during the early stages of database design to capture the essential entities, attributes, and relationships that define the application's information needs. Finally, you explored an **e-commerce example** model design that considered how business rules established specific 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

Cardinality

The cardinality of a relationship depends on the number of one-to-one, one-to-many, or many-to-many relationships.

Conceptual Data Model

A high-level representation of a system's data requirements and structure that establishes the entities, their attributes, and relationships between entities in a relational database.

Entity Relationship (ER) Model

A visual representation that illustrates the structure and relationships within a database or data model.