Databases and Information Systems

Designing an ER Diagram – E-Commerce Platform

Prof. Ing. Lelio Campanile, PhD

Data Analytics Bachelor
Universita degli Studi della Campania Luigi Vanvitelli

Guided classroom exercise

Learning goals

- Apply the ER design method to a **more complex, realistic domain**
- Model **multiple entities and relationships** from a textual description
- Handle **relationship attributes** and **many-to-many** relationships

Quick review

Entity

Definition

An **entity** is a real-world object or concept we want to represent in the database. Entities are represented by **rectangles**.

Cardinality – reminder

BCN notation

Each arc carries a **(min,Max)** pair. **min** = minimum participation **Max** = maximum participation.

Guided example

Case study description

An e-commerce company wants to build a database to manage its online sales platform.

The platform serves **customers**, each identified by a **fiscal code** and characterised by a name, email address, and phone number. Customers can place **orders**: each order has a unique **order number**, a date, and a status. Every order must be placed by exactly one customer, while a customer may have placed any number of orders.

Each order is settled by exactly one **payment**, identified by a **transaction code** and storing the amount, the payment date, and the payment method. A payment corresponds to exactly one order.

Orders contain one or more **products**. Each product is identified by a **SKU** and has a name, a description, and a price. A product may appear in any number of orders; for each product in an order the **quantity** purchased is recorded. Every product belongs to exactly one **category**, identified by its name.

Products are supplied by one or more **suppliers**, each identified by a **VAT number** and characterised by a name and country. The same product may be sourced from multiple suppliers; for each supplier–product pair the **unit cost** is stored.

Customers can write **reviews** about products. Each review is identified by a **review ID** and records a rating, a comment, and a date. A customer may write any number of reviews; each review is written by exactly one customer and refers to exactly one product.

Problem description

An e-commerce company wants to manage its platform.

- **Customers**: fiscal code, name, email, phone
- **Products**: SKU, name, description, price; each belongs to one **Category** (category name)
- Customers place **Orders**: order number, date, status; each order contains one or more products with a quantity
- Each order is settled by exactly one **Payment**: transaction code, amount, date, method
- Products are supplied by one or more **Suppliers**: VAT number, name, country; each supplier–product pair has a unit cost
- Customers write **Reviews**: review ID, rating, comment, date

Step 1 - Identify the entities

Question

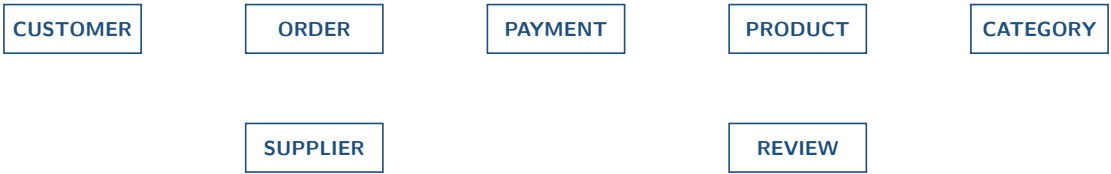
Which **main objects** of the domain do we need to store?

Think for a moment before looking at the solution.

Step 1 - Identify the entities

Answer — seven entities

- CUSTOMER
- PRODUCT
- CATEGORY
- ORDER
- PAYMENT
- SUPPLIER
- REVIEW



Step 2 - Add the attributes

Question

Which properties should we store for each entity?

List the attributes for each of the seven entities.

Step 2 - Add the attributes

CUSTOMER
- fiscal_code
- name, email, phone

PRODUCT
- SKU
- name, description, price

REVIEW
- review_id
- rating
- comment
- review_date

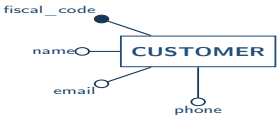
ORDER
- order_number
- order_date, status

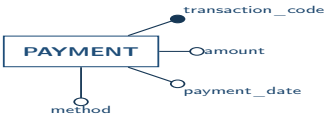
CATEGORY
- category_name, name

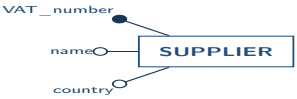
SUPPLIER
- VAT_number
- name, country

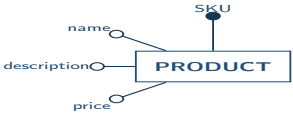
PAYMENT
- transaction_code
- amount, payment_date, method

Step 2 - Attributes on the diagram









Step 3 - Identify the relationships

Question

How are the entities connected?

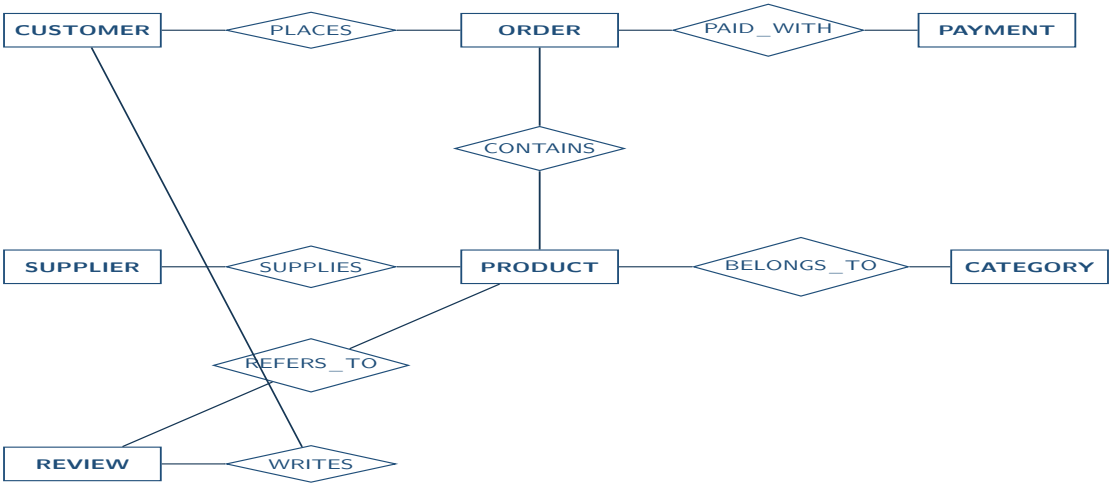
Look for the verbs and business rules in the description.

Step 3 - Identify the relationships

Answer — seven relationships

PLACES	CUSTOMER → ORDER	a customer places orders
PAID_WITH	ORDER → PAYMENT	each order has a payment
CONTAINS	ORDER ↔ PRODUCT	an order lists products
BELONGS_TO	PRODUCT → CATEGORY	a product has a category
SUPPLIES	SUPPLIER ↔ PRODUCT	suppliers provide products
WRITES	CUSTOMER → REVIEW	a customer authors reviews
REFERS_TO	REVIEW → PRODUCT	a review targets a product

Step 3 - Relationships on the diagram



Step 4 - Determine the cardinalities

Question

For each relationship, how many instances can participate on each side?

- Can a customer have zero orders? Many orders?
- Can an order contain more than one product?
- Can a product belong to more than one category?
- Can a product be supplied by more than one supplier?
- Can an order have more than one payment?
- Can a customer write more than one review?

Step 4 - Cardinalities

PLACES 1:N
Customer $(0, N)$ — Order $(1, 1)$

PAID_WITH 1:1
Order $(1, 1)$ — Payment $(1, 1)$

CONTAINS N:M
Order $(1, N)$ — Product $(0, N)$

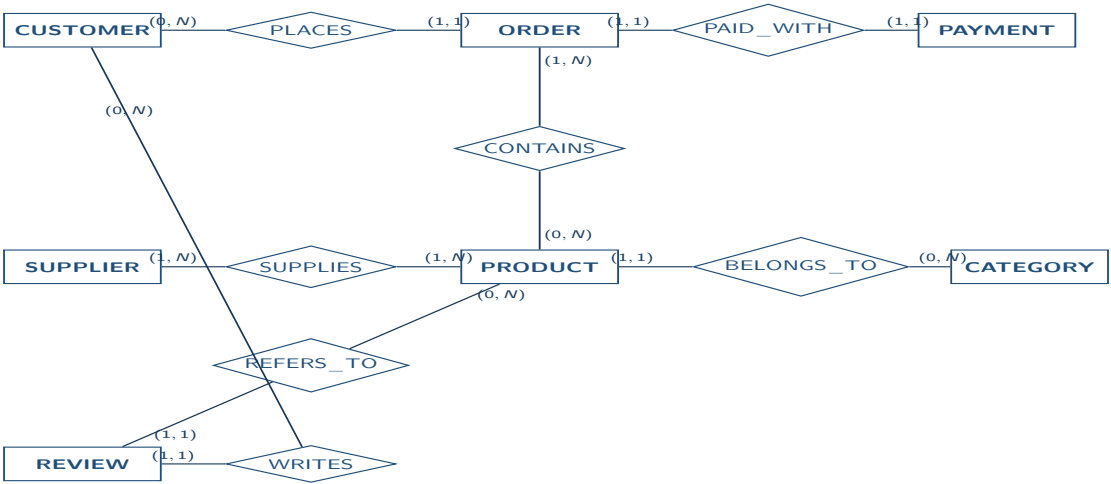
BELONGS_TO N:1
Product $(1, 1)$ — Category $(0, N)$

SUPPLIES N:M
Supplier $(1, N)$ — Product $(1, N)$

WRITES 1:N
Customer $(0, N)$ — Review $(1, 1)$

REFERS_TO N:1
Review $(1, 1)$ — Product $(0, N)$

Step 4 - Cardinalities on the diagram



Step 5 - Attributes of relationships

Question

Where should we store **quantity** and **unit_cost**?

- Is **quantity** a property of ORDER alone? Of PRODUCT alone?
- Is **unit_cost** a property of SUPPLIER alone? Of PRODUCT alone?
- Or does each depend on a *specific pair* of entities?

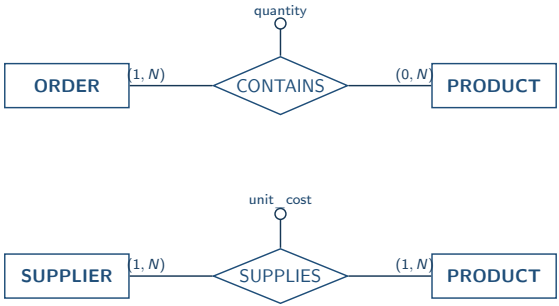
Step 5 - Attributes of relationships

CONTAINS — attribute: *quantity*

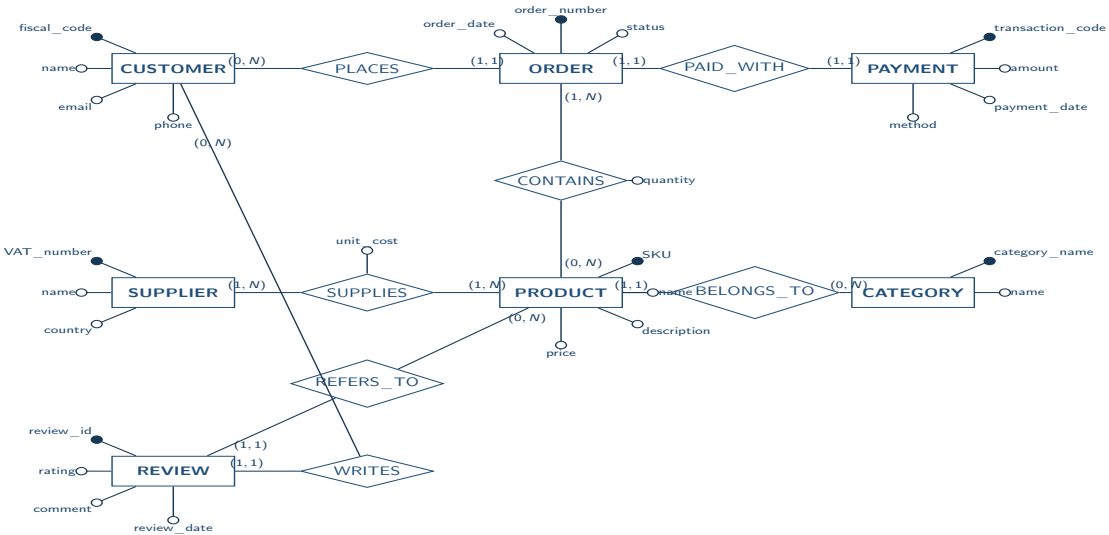
The quantity of a product in an order depends on the specific *(Order, Product)* pair. It is **not** a property of Order alone, nor of Product alone.

SUPPLIES — attribute: *unit_cost*

The cost depends on the specific *(Supplier, Product)* pair. Different suppliers may charge different prices for the same product.



Final ER diagram



How to reason during design

1. Read the text and highlight the **main nouns** – possible entities
2. Search for **properties** – possible attributes
3. Search for **verbs** – possible relationships
4. For each relationship, ask: *how many?* – cardinality (min, Max)
5. Check whether some attributes belong to a **relationship** instead of an entity

Questions for the class

- Why is **PAID_WITH** a 1:1 relationship? Could it ever become 1:N?
- Why is **quantity** an attribute of CONTAINS and not of PRODUCT?
- If a supplier could supply a product at different prices depending on the country, how would you modify the schema?
- Where would you store a **discount code** applied to a specific order?

In-class extension

How would you extend this model to support:

- **Shipping**: each order has a tracking number and delivery date
- **Wishlist**: customers can save products they want to buy later
- **Discount codes**: a code can apply a percentage off to one or more orders
- **Product variants**: same product in different sizes or colours, each with its own stock level

For each extension: identify the new entity or attribute, decide where it belongs, and determine the cardinality.

Summary

- We designed a complete ER schema for an **e-commerce platform**
- We modelled **7 entities** and **7 relationships**
- We used **relationship attributes**: quantity, unit_cost
- We introduced **all three cardinality types**: 1:1, 1:N, N:M
- All diagrams follow **BCN notation** (Batini-Ceri-Navathe)

Questions?