Databases
Lab 07: A ‘gentle’ Introduction to E-R Modelling.

Andrés Oswaldo Calderón Romero, Ph.D.

September 23, 2025

1 Introduction

In modern software systems, data is at the core of nearly every application —and designing a solid data structure from the beginning is crucial. The Entity-Relationship Diagram (ERD) is a powerful conceptual tool that allows developers, analysts, and database architects to visually map the structure and relationships within a database before it’s built.

In this lab, you will explore the fundamentals of the E-R modelling, learning how to identify key components such as entities, relationships, attributes, and cardinalities. You’ll also gain hands-on experience in designing an ERD based on real-world scenarios, setting the foundation for creating robust and scalable relational databases. This tutorial draws inspiration from the article *“Entity-Relationship Model of a Database: What It Is, Elements, and Examples”* by iLerna[1], providing a clear and practical guide for understanding and applying E-R modelling techniques.

2 What is the Entity-Relationship Model?

The entity-relationship model, entity-relationship diagram, or ERD, is a tool used in the second phase of database design.

This diagram, created by computer scientist Peter Chen, visually and simplistically represents how people, objects, or concepts within a system are connected.

The E-R model allows organizing and visualizing the information structure before creating the final database, providing a clear and easy-to-understand schema for designing how data will be stored and related.

3 Elements of the Entity-Relationship Model

The E-R diagram consists of four basic elements (three of them can be seen in figure 1):

- Entities
- Relationships
- Attributes
- Cardinalities

[1] Available at `https://www.ilerna.es/blog/modelo-entidad-relacion-base-datos`

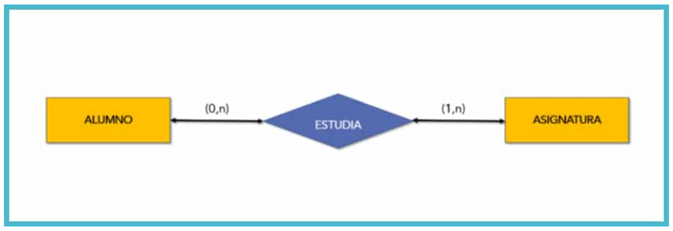

Figure 1: Simple schema of a relationship between two entities and their cardinalities.

3.1 Entity

Entity sets are elements about which information is collected to represent in the database. There are two types of entities, represented graphically with a rectangle:

- **Strong entities**: Do not depend on other entities to exist. For example, a student in a school's database is a strong entity because they exist independently.
- **Weak entities**: Depend on a strong entity to make sense. For instance, an order in an online store is a weak entity because it is always linked to a customer.

3.2 Relationship

A relationship indicates how two or more entities interact or connect. It is represented with a diamond and is usually expressed as a verb. There are different types of relationships according to their degree:

- **Degree 1 or unary relationship**: Relates an entity to itself. For example, a class delegate who is also a student of the course.
- **Degree 2 or binary relationship**: Connects two different entities. For example, a supplier that provides items.
- **Degree 3 or ternary relationship**: Links three or more entities. For example, a bank customer who has several accounts, each in a different branch.

Relationships between entities can have different types of correspondence, indicating how they connect:

- **1:1 (One to one)**: Each element of the first entity corresponds to one and only one of the second entity, and vice versa. For example, a hotel guest occupies a single room, or a class of students is assigned to a single classroom, and only that group attends that classroom.
- **1:N (One to many)**: Each element of the first entity corresponds to one or more elements of the second entity, but each element of the second entity corresponds to only one of the first entity. An example would be a supplier that provides several items, but each item has a single supplier.
- **N:M (Many to many)**: Several occurrences of one entity can be associated with several of the other. For example, a student can enroll in several courses, and each course can have several enrolled students.

3.3 Attributes

Attributes are the characteristics or details that describe an entity. They are like the properties that define an element within existing databases.

For example, if we have an entity called Person, some of its attributes could be:

- Name
- Age
- ID (which would be unique for each person)

The set of attributes helps provide more information about an entity, such as a student's name or a customer's date of birth. In a diagram, attributes are represented with circles.

3.4 Cardinality

Cardinality defines how many times an entity can be related to another. Basically, it shows how many elements of one entity can be associated with those of another entity.

For example, in a 1:1 relationship, like a car and its license plate, each car has only one license plate, and each license plate belongs to a single car. In a 1:N relationship, like a customer and their orders, a customer can place several orders, but each order belongs to only one customer.

In an N:M relationship, like that of students and courses, several students can be enrolled in several courses, and each course can have several students.

4 How to Design a Conceptual Database Schema

The conceptual schema of a database provides an overview of how data will be organized and what relationships will exist between them. It is the first step to correctly structure the database, and the steps to follow to design it are as follows.

4.1 Gather Client Information

It is essential to gather all the necessary information from the client to understand their needs. This allows designing a database that truly meets the user's requirements and aligns with their objectives.

4.2 Design the Entity-Relationship Model

In this step, the entity-relationship (E-R) diagram is created, incorporating the collected data. Standard graphical rules are followed: entities are represented with rectangles, relationships with diamonds, and attributes with circles.

This model helps visualize how the different elements within the database are connected.

4.3 Create the Relational Model

Once we have the conceptual diagram, it is transformed into a relational model. In this stage, entities and relationships are organized into tables containing the data, and the relationships between them are defined. This facilitates implementation in a real system.

4.4 Decide on the Database Management System (DBMS)

With the model clear, it is necessary to choose the Database Management System (DBMS) that best suits the project's needs. This system is the software that will allow creating, managing, and querying the database.

4.5 Implement the Model in the Program

Finally, the relational model must be implemented in the selected DBMS. In this phase, the conceptual model becomes a physical data model that will be managed by the system, and the data will be ready to be used in the application.

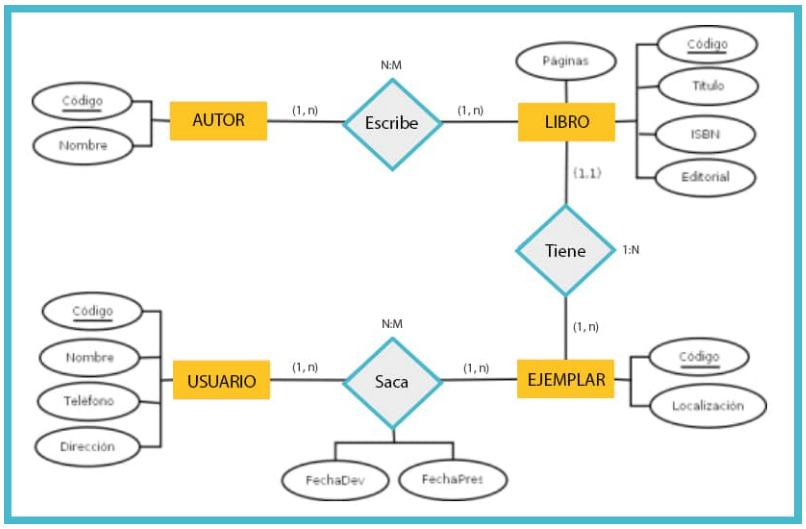

Figure 2: Example of a representation of a Entity-Relationship model.

5 Examples of Entity-Relationship

Below is an example of how a database would be graphically represented with the following entities: author, book, copy, and user (words in parenthesis are how they are referred in figure 2):

- **Author (Autor)**: The attributes of the author entity are code and name. The code is the main attribute, as it is unique for each author.
- **Book (Libro)**: The attributes of the book entity are pages, code, title, ISBN, and publisher. The code is the main attribute and is unique for each book.
- **Copy (Ejemplar)**: The attributes of the copy entity are code and location. The code is the main attribute and is unique for each copy.
- **User (Usuario)**: The attributes of the user entity are code, name, phone number, and address. The code is the main attribute and is unique for each user.

In this model, there are three main relationships:

- **Author - Book (Escribe)**: An author can write several books. This is a relationship between the `author` and `book` entities.
- **Book - Copy (Tiene)**: A book can have multiple copies. This relationship connects the `book` and `copy` entities.
- **User - Copy (Saca)**: A user can have several borrowed copies. This is the relationship between the `user` and `copy` entities.

6 The Data Dictionary

Harvard defines a **Data Dictionary** (DD) as a document that provides detailed descriptions of all elements in a dataset (including variable names, definitions, permissible values, units of measurement, and relationships) to ensure that the data remain interpretable, comparable, and reusable. A well-maintained, up-to-date DD enhances reproducibility, facilitates data sharing, and promotes consistent understanding across users. Moreover, it should evolve alongside the project (e.g., through version crosswalks).

For further guidance, consult this page, which discusses the construction of a DD in greater detail. The page also offers templates, examples, and additional resources to support the creation of effective data dictionaries. It provides practical examples and recommendations, with the *Resources* section being particularly useful for completing your independent work.

7 Independent Work

You will read the following exercises and select **one** of them. Based on your choice, create an initial attempt at an E–R model and a Data Dictionary by identifying the main entities, relationships, attributes, and cardinalities.

Follow a similar approach to the previous example (Section 5) and apply the concepts introduced in Section 6. For this stage, draw your diagram by hand, scan it, and include it in a well-structured report where you discuss the key aspects of your analysis, the design decisions made, and provide a simple yet sufficient version of the corresponding data dictionary.

Submit your report in **PDF format** via Brightspace™ no later than **October 8, 2025**.

Happy Hacking! 😎