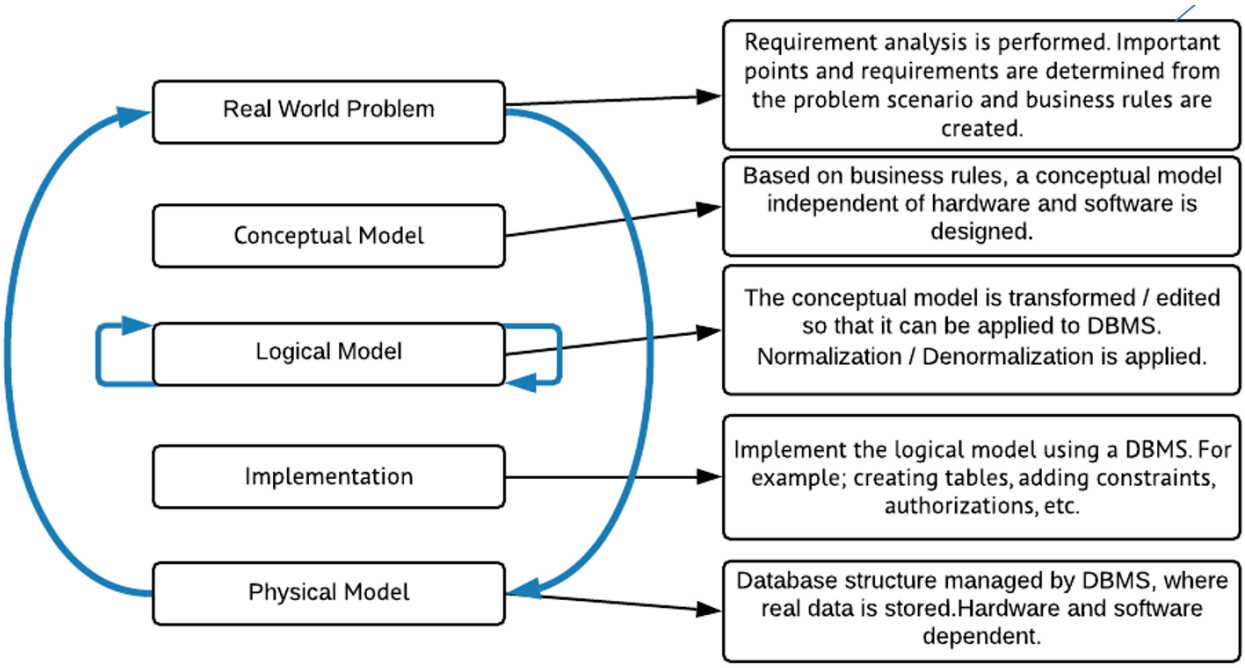
Module 2: Entity-Relationship (ER) Model

1. Database Development Lifecycle

When developing databases, we need to adhere to the **Database Development Lifecycle.** It consists of several stages to build a well-structured database:



- **Requirements Analysis**: Understanding business needs and defining **business rules**.
- **Conceptual Design**: Designing high-level Entity-Relationship (ER) models.
- **Logical Design**: Translating ER models into relational schemas.
- **Implementation**: Building and configuring the database using SQL

2. Business Rules

Business rules can be seen as a summarized version of the requirements list, focusing on data.

Statements that define or constrain aspects of a business, often used to guide database design.

The sources of business rules can include end users, managers, policymakers, and written documents (such as standards and regulations).

Directly consulting end users is an effective approach to defining business rules.

Examples include:

- An **employee** can belong to only one **department**.
- A **customer** must provide a **valid email**.
- A **student** can enroll in many **courses**, and a **course** can have many **students**
- An **order** must have at least **one product**.
- A **customer** can place multiple **orders**, but each **order** is placed by only one **customer**.

Business rules help identify **entities, attributes, relationships, and constraints** in the ER model.

Once the business rules are established, the next step is modeling the database to implement them.

3. Data Model

A **data model** is a tool used to represent complex real-world data structures in a simplified (often graphical) form.

Forming a data model has two main advantages:

1. **Better Communication** – It facilitates communication between database designers, application developers, and end users. It helps database designers, developers, and end users understand and agree on the database structure before implementation.
2. **Facilitated database implementation** – Thanks to data models, implementing the database becomes easier.

Data modeling is an **iterative process**:

1. A basic model is designed first.
2. Details are gradually added.
3. Eventually, a **blueprint** for database design is obtained.

Key Components of a Data Model

1. **Entity** – Represents real-world objects about which data is collected and stored (e.g., Student, Course, Employee). Entities must be distinguishable from one another.

2. **Attribute** – Characteristics or properties of an entity (e.g., Student Name, Course Code, Salary).
3. **Relationship** – Defines the association between entities. There are three relationship types or cardinalities:
    - **One-to-Many (1:M)** – A customer can place multiple orders, but each order is associated with only one customer.
    - **Many-to-Many (M:N)** – A student can enroll in multiple courses, and each course can have multiple students.
    - **One-to-One (1:1)** – A store is managed by one employee, and each employee manages only one store.
4. **Constraints** – Rules that maintain data integrity, such as:
    - A student's grade must be between **0 and 100**.
    - A national ID number must be **11 characters long**.
    - The same product **cannot be recorded multiple times**.

Evolution of Data Models

While the **File System, Hierarchical Model, and Network Model** was influential in the early days of database systems, it is rarely used in modern software development. Today, relational databases and other models like NoSQL are more commonly employed due to their flexibility, scalability, and ease of use.

- **File System** – Data stored in files without structured relationships.
- **Hierarchical Model** – Data is organized in a tree-like structure.
- **Network Model** – Uses graph structures for complex relationships.

—----------

- **Relational Model** – Stores data in tables with structured relationships.
- **Entity-Relationship (ER) Model** – Conceptual representation of data using entities, attributes, and relationships.
- **Object-Oriented Model** – Integrates object-oriented programming concepts with databases.
- **New Data Models** – Includes NoSQL, graph databases, and other emerging models.

4. Entity-Relationship Model

The **ER Model** is a conceptual representation of data using:

- **Entities** – Represents real-world objects about which data is collected and stored (e.g., Student, Course, Employee).

- **Attribute** – Characteristics or properties of an entity (e.g., Name, Product Code, Unit Price).
- **Relationship** – Defines the association between entities. There are three relationship types or cardinalities:
    - **One-to-Many (1:M)**
    - **Many-to-Many (M:N)**
    - **One-to-One (1:1)**
- **Constraints** – Rules that maintain data integrity:
    - An employee's age must be between 18 and 65.
    - A phone number must follow the format (XXX) XXX-XXXX.
    - Each email address must be unique in the system.

Two main benefits of an ER model are**: Better Communication** and **Facilitated database implementation**

Different notations for conceptual modeling include:

- **Chen Notation** – Focuses on conceptual modeling, clearly representing entities, attributes, and relationships.
- **Crow's Foot Notation** – More implementation-oriented, widely used in relational database design.
- **UML Notation** – Can be used for both **conceptual and implementation modeling**, making it flexible for different database design approaches.

Transforming Business Rules into an Entity-Relationship Model

In general, **nouns/noun phrases** in business rules represent **entity candidates or attributes**. If two entities appear in the same sentence, there is a **relationship** between them.

- **Nouns or noun phrases** that contain properties are entity candidates.
- **Nouns without specific stored data** are attribute candidates.

This is a **basic guideline**, but through **practical exercises**, you can develop a deeper understanding and gain experience in identifying **entities, attributes, and relationships** more effectively.

Examples of Business Rules and Their ER Model Interpretation:

1. Entity, Attribute -> A **Customer** entity has the following attributes:
    - First Name
    - Last Name

        - Customer Number
        - Address
2. **Relationship ->** A **Customer** can generate many **Invoices (1:M).**
3. **Types of relationships**

   Relationships are bidirectional. We can employ this feature.

    - One **Instructor** can teach **up to 4(many) Courses (1:M).**
    - Each **Course** is taught by **only one Instructor (1:1).**

    - One **Person** can be the **Manager of one Department (1:1).**
    - Each **Department** has **only one Manager (1:1).**

    - One **Student** can enroll in **multiple Courses (M:N).**
    - Each **Course** can have **multiple Students (M:N).**

4. **Attributes**: Properties of an entity.
    - **Simple**: Single-valued (e.g., Name).
    - **Composite**: Divisible (e.g., Full Name → First Name + Last Name).
    - **Derived**: Computed from other attributes (e.g., Age from Date of Birth).
    - **Multivalued**: Can have multiple values (e.g., Phone Numbers).

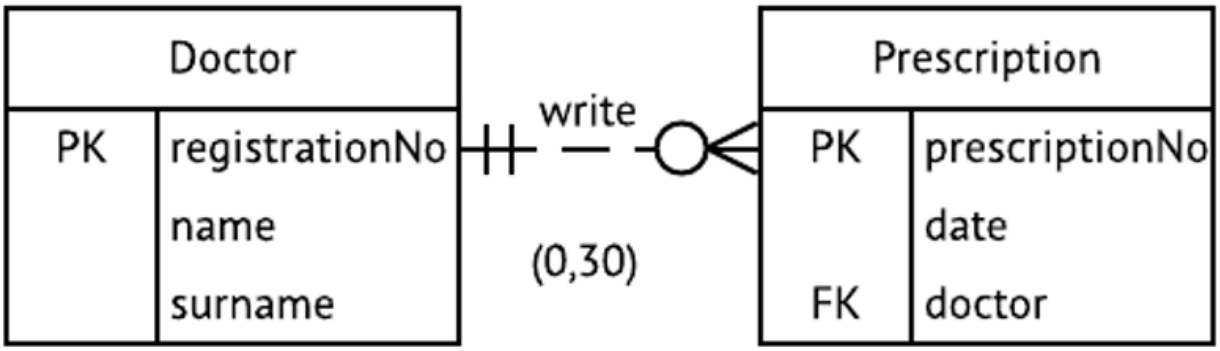
This structured transformation ensures proper **entity identification, attribute assignment, and relationship definition** in database modeling.

5. Designing ER Diagrams Using Crow's Foot Notation

One-to-Many (1:M) Relationship

A **doctor** can issue multiple **prescriptions**,

Each **prescription** is issued by only <u>one</u> **doctor**. *(1:M Relationship)*

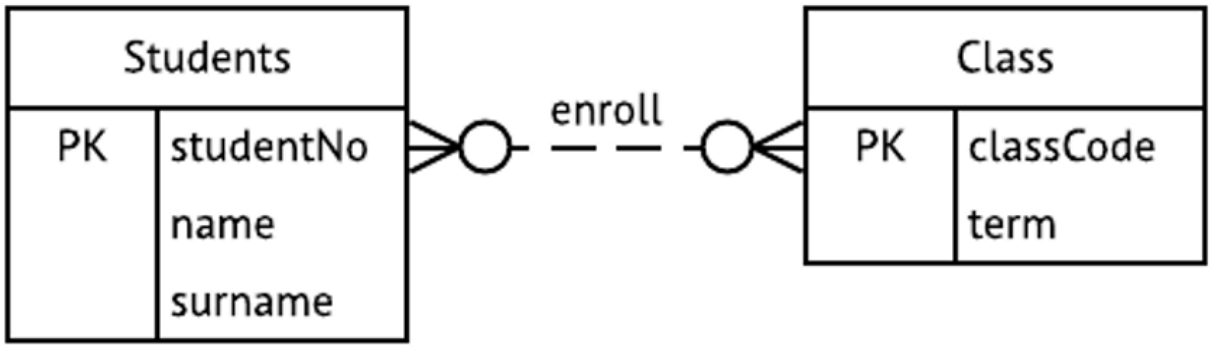


In a one-to-many relationship, the primary key (PK) of the entity on the 'one' side must be added as a foreign key (FK) to the entity on the 'many' side.

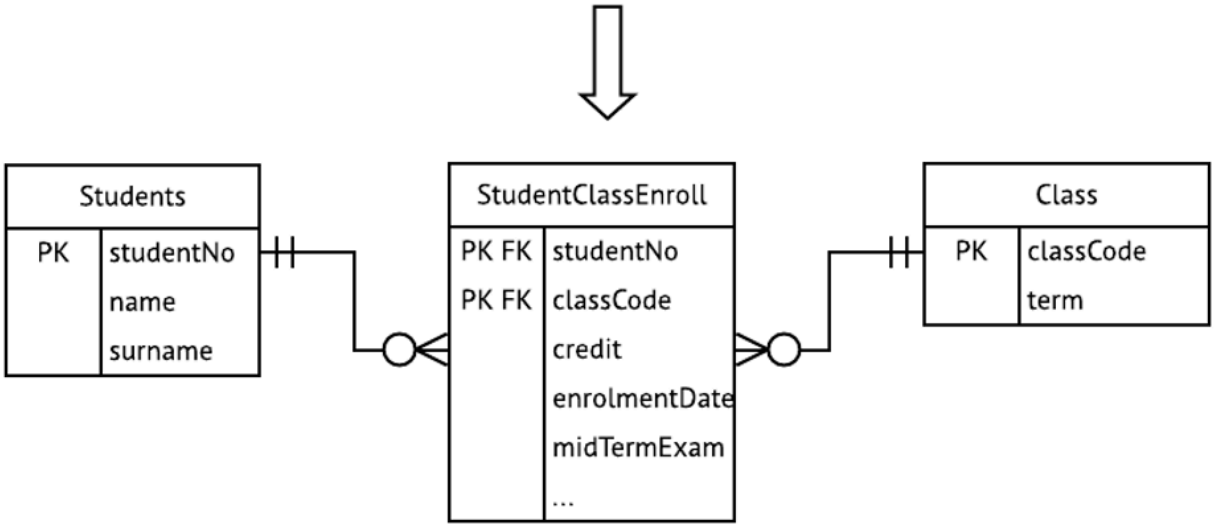
Many-to-Many (N:M) Relationship

A **student** can enroll in **<u>many</u> courses**

A **course** can have **<u>many</u> students**



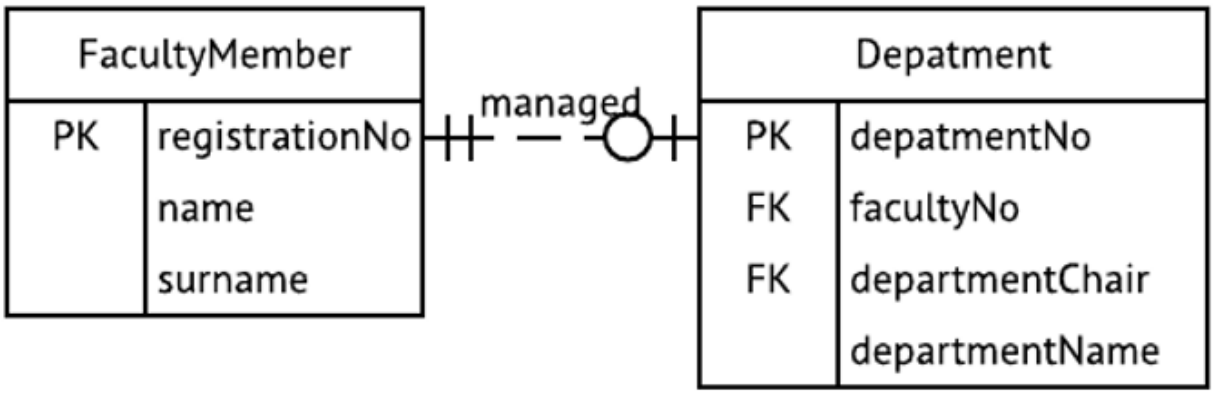
In a many-to-many relationship, a junction table is created to link the two entities. This table includes the primary keys of both entities as foreign keys, along with any additional attributes specific to the relationship(student's course related info).



in a junction table for a many-to-many relationship, instead of using a combined primary key (composite key) consisting of the foreign keys (FKs) from both tables, you can use a single auto-incremented primary key (surrogate key). This approach is often preferred for simplicity and performance reasons, especially in certain database systems or application scenarios.

One-to-One (1:1) Relationship

1 faculty member can manage 1 department.
1 department can be managed by 1 faculty member.



Add the PK of one table as an FK to the other table.

Hands-on Exercise 1: Employee and Project Management System - Business Rules to ER Model with Crow's Foot Notation

Objective In this exercise, you will practice transforming business rules into an Entity-Relationship (ER) model using Crow's Foot notation. You are required to identify entities, attributes, relationships, and constraints based on a given business scenario.

Task 1: Analyze Business Rules

Below are the business rules for a company's employee and project management system. Your task is to extract the necessary entities, define their attributes, determine relationships, and establish constraints.

1. A company has multiple departments, and each department has a unique Department ID, a Name, and a Budget.
2. Each department is managed by one employee, but an employee can manage only one department.
3. Employees work in departments. Each employee has a unique Employee ID, Name, Age, Gender, and Job Title.
4. Employees can be assigned to multiple projects, and each project has multiple employees working on it.
5. Each project has a unique Project ID, Name, Start Date, and End Date.
6. Each employee working on a project has a specific Role in that project (e.g., Developer, Manager, Analyst).
7. An employee must be assigned to at least one project.
8. Some employees have supervisors. A supervisor is also an employee.
9. The company keeps track of employee education, which includes Degree Name, Institution, and Graduation Year. An employee can have multiple degrees.

Task 2: Construct the ER Model

Using the business rules provided, perform the following steps:

1. Identify and list all entities and their attributes.
2. Determine the relationships between the entities.
3. Identify the cardinalities (one-to-one, one-to-many, or many-to-many) between entities.
4. Use Crow's Foot notation to create an ER diagram representing the business rules.

Hands-on Exercise 2: E-Commerce System – Business Rules to ER Model with Crow's Foot Notation

Objective

In this exercise, you will practice database modeling by converting business rules into an Entity-Relationship (ER) model using Crow's Foot notation. You need to identify entities, attributes, and relationships within an e-commerce system.

Task 1: Analyze Business Rules

Below are several business rules describing the structure of an e-commerce database. Carefully analyze them and identify the necessary entities, attributes, and relationships:

1. A **customer** can place **many orders**, but each order is placed by **one customer**.
2. Each **customer** has a **unique customer ID**, **name**, **email**, **phone number**, and **shipping address**.
3. An **order** has a **unique order ID**, an **order date**, a **total amount**, and a **status** (e.g., pending, shipped, delivered).
4. An **order** can contain **multiple products**, and a **product** can be part of multiple orders. The quantity of each product in an order must be recorded.
5. Each **product** has a **unique product ID**, **name**, **description**, **price**, and **stock quantity**.
6. A **product** belongs to **one or more categories**, and each **category** can have multiple products.
7. The system tracks **payments** made by customers. A **payment** is associated with **one order**, and an order can have **only one payment**.
8. Each **payment** has a **unique payment ID**, **payment date**, **amount paid**, and **payment method** (e.g., credit card, PayPal).
9. Customers can leave **reviews** for products. A **review** belongs to **one customer** and **one product**.
10. Each **review** has a **review ID**, a **rating** (1 to 5), **review text**, and a **date**.

Task 2: Convert Business Rules into an ER Model

Using Crow's Foot notation, create an ER model that represents the above business rules. Your model should include:

- Entities and their attributes
- Relationships between entities
- Relationship types (cardinalities) (one-to-one, one-to-many, many-to-many)

Hands-on Exercise 4: Learning Management System – Business Rules to ER Model with Crow's Foot Notation

Objective

In this exercise, you will design an Entity-Relationship (ER) model for a **Learning Management System (LMS)** based on given business rules. You will identify entities, attributes, and relationships, then represent them using **Crow's Foot notation**.

Task 1: Analyze Business Rules

Below are the business rules defining a Learning Management System. Analyze them carefully and identify the required **entities, attributes, and relationships**:

1. The system manages **students**, each with a **unique student ID**, **name**, **email**, **phone number**, and **enrollment date**.
2. The system also tracks **instructors**, who have a **unique instructor ID**, **name**, **email**, **phone number**, and **specialization**.
3. **Instructors** can teach **multiple courses**, but each course is taught by **only one instructor**.
4. A **course** has a **unique course ID**, **title**, **description**, **credit hours**, and a **start and end date**.
5. **Students** can enroll in **multiple courses**, and each course can have **multiple students**.
6. The system keeps track of **enrollments**, recording the **enrollment date** and **status** (e.g., active, completed, dropped).
7. Each course consists of **multiple lessons**, where each lesson has a **unique lesson ID**, **title**, **content**, and an **associated course**.
8. **Assignments** are given to students within a course. Each assignment has a **unique assignment ID**, **title**, **description**, **due date**, and **maximum score**.

9. Students submit **assignments**, and each submission records the **submission date**, **file link**, and the **grade** awarded by the instructor.
10. The system allows **students to take quizzes**, where each quiz is associated with a **course** and contains **multiple questions**.
11. A **quiz** has a **unique quiz ID**, **title**, **total score**, and a **time limit**.
12. Each **question** in a quiz has a **unique question ID**, **question text**, **answer choices**, and a **correct answer**.
13. Students attempt **quizzes**, and the system records their **score** and **submission date**.
14. The system enables **discussion forums**, where students can post **questions** related to a course. Each post has a **post ID**, **content**, **timestamp**, and is linked to a **student** and a **course**.
15. **Students and instructors** can respond to discussion posts, and each response has a **response ID**, **content**, **timestamp**, and links to the original post.

Task 2: Convert Business Rules into an ER Model

Using Crow's Foot notation, create an ER model that represents the above business rules. Your model should include:

- **Entities** and their **attributes**
- **Relationships** between entities
- **Cardinalities** (one-to-one, one-to-many, many-to-many)

References

- https://github.com/celalceken/DatabaseManagementSystems
- Gözde Yolcu Öztel, Database Management Systems Lecture Notes.
- Carlos Coronel, Steven Morris, and Peter Rob, Database Systems: Design, Implementation, and Management, Cengage Learning.
- http://www.digitalinformationworld.com/2015/02/fascinating-social-networking-stats-2015.html

- https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/#230ac18d60ba
- Apache Spark Tutorial | Spark Tutorial for Beginners | Apache Spark Training | Edureka
    - https://www.youtube.com/watch?v=9mELEARcxJo