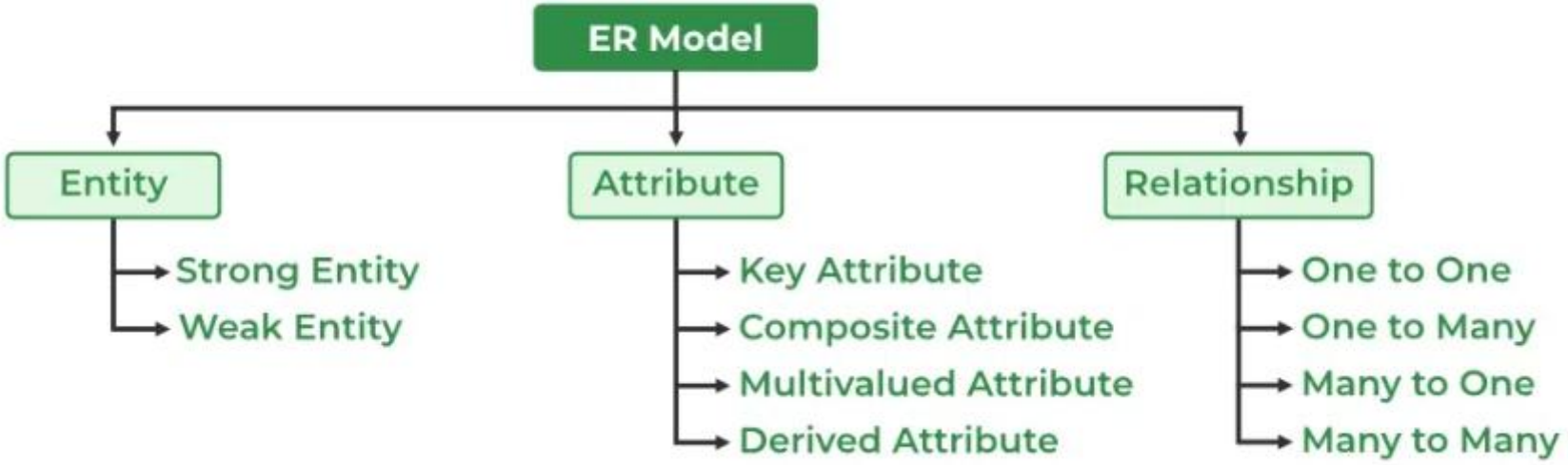
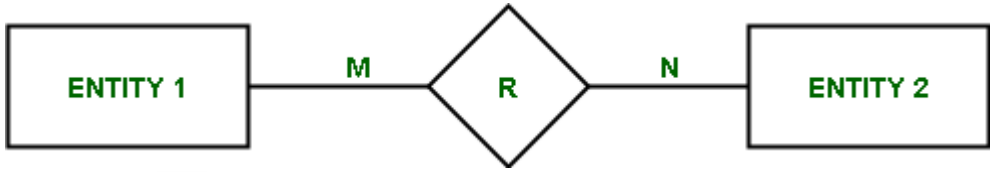
Lebanese University **Faculty of Media and Documentation** **- Section 1 -**		**الجامعة اللبنانية** **كلية الإعلام والتوثيق** **- الفرع الأول -**
Computer Science		
First Year - Semester 1		
Database 1		**Chapter 3: Relational model and ER integirty constraints**







1. Recall

Before creating a database, we must analyze and model the real-world information needs of an organization.
This step produces a conceptual design — independent of any specific DBMS.

1. Why Data Modeling?

- To understand what data needs to be stored.
- To define how entities are related.
- To ensure consistency and accuracy before implementation.

2. Entity–relationship model (ER model)

describes the structure of a database with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram). **An ER model** is a design or blueprint of a database that can later be implemented as a database. **The main components of E-R** model are: entity set and relationship set.

3. What is an Entity Relationship Diagram (ER Diagram)?

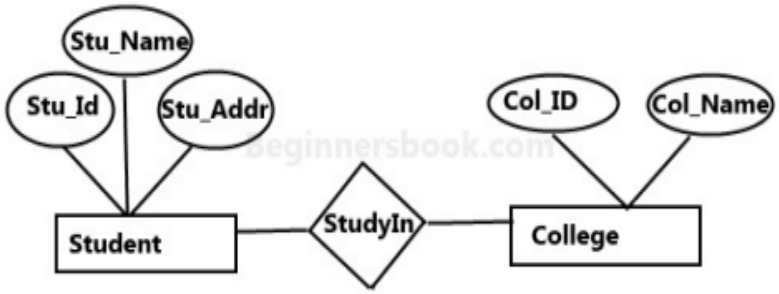
The **Entity–Relationship (ER) model** is a **conceptual data model** introduced by Peter Chen (1976). It visually represents:

- **Entities** (things in the real world), **Attributes** (their properties), and Relationships (how they are connected).

Output: An ER Diagram (ERD) **— a graphical representation of data**.

4. Facts about ER Diagram Model:

- ER model allows you to draw Database Design.
- It is an easy to use graphical tool for modeling data
- Widely used in Database Design
- It is a GUI representation of the logical structure of a Database
- It helps you to identifies the entities which exist in a system and the relationships between those entities



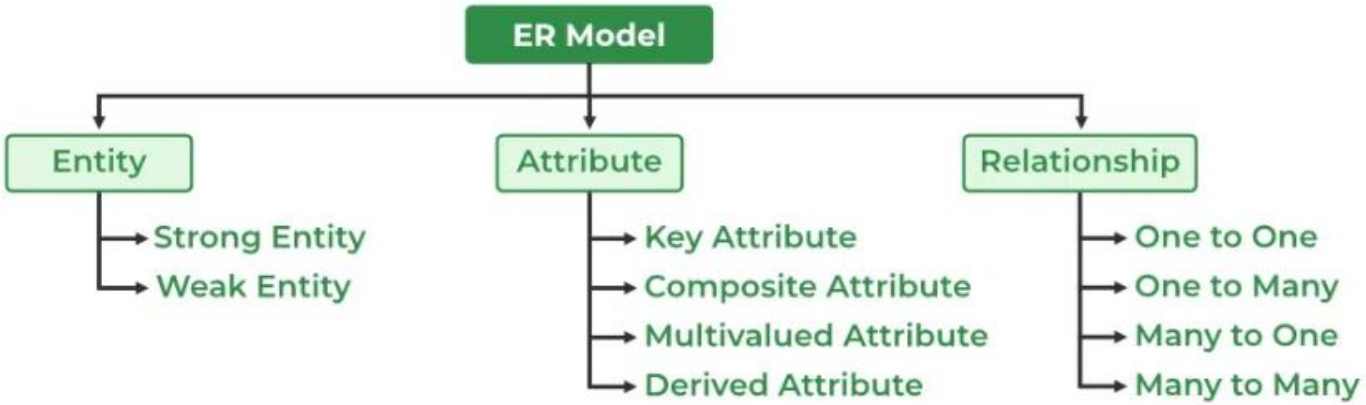
Sample E-R Diagram

5. Why use ER Diagrams?

- Helps you to define terms related to entity relationship modeling Provide a preview of how all your tables should connect, and what fields are going to be on each table
- Helps to describe entities, attributes, relationships
- ER diagrams are translatable into relational tables which allow you to build databases quickly
- ER diagrams can be used by database designers as a blueprint for implementing data in specific software applications

- **Rectangle**: Represents Entity sets.
- **Ellipses**: Attributes
- **Diamonds**: Relationship Set
- **Lines**: They link attributes to Entity Sets and Entity sets to Relationship Set
- **Double Ellipses**: Multivalued Attributes
- **Dashed Ellipses**: Derived Attributes
- **Double Rectangles**: Weak Entity Sets
- **Double Lines**: Total participation of an entity in a relationship set

6. Components of an ER Diagram



6.1.Entity?

- **An Entity** represents a real-world object, concept or thing about which data is stored in a database. It act as a building block of a database. Tables in relational database represent these entities.

Example of entities:

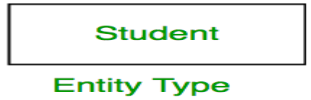
- Real-World Objects: Person, Car, Employee etc.
- Concepts: Course, Event, Reservation etc.
- Things: Product, Document, Device etc.

The entity type defines the structure of an entity, while individual instances of that type represent specific entities.

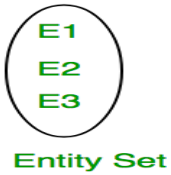
Entity Set?

- A collection of similar entities.

Example: All students → STUDENT entity set.



In the ER diagram below, the entity type is represented as:



- **Types of Entity: There are two main types of entities:**

1. Strong Entity (independent entity)

A Strong Entity is a type of entity that has a key Attribute that can uniquely identify each instance of the entity.

It has a primary key that ensures its uniqueness and is represented by a rectangle in an ER diagram.

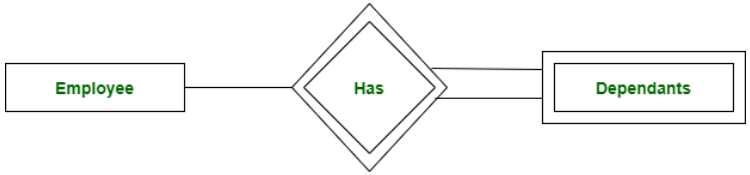
2. Weak Entity (dependent entity)

Exists only with a strong entity (depends on it).

A weak entity are represented by a double rectangle. The participation of weak entity types is always total.

Example:

A company may store the information of dependents (Parents, Children, Spouse) of an Employee. But the dependents can't exist without the employee. So dependent will be a Weak Entity Type and Employee will be identifying entity type for dependent, which means it is Strong Entity Type.



6.2. Attributes



Attributes are the properties that define the entity type. For example, for a Student entity Roll_No, Name, DOB, Age, Address, and Mobile_No are the attributes that define entity type Student. In ER diagram, the attribute is represented by an oval.

Types of Attributes

1.simple attribute.

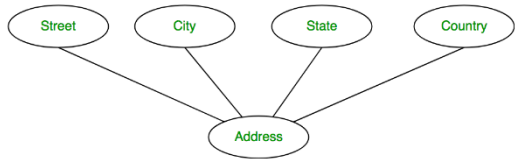
Cannot be divided further. An attribute that holds **only one value** for each entity instance. Examples: DateOfBirth (only one per person)- EmployeeID (each employee has a single ID).

2. Key Attribute



An attribute whose value is **unique across all entities** — it **identifies** an entity uniquely (similar to a key). For example, SSN (Social Security Number) — no two employees share it. **In ER diagram, the key attribute is represented by an oval with an underline.**

3. Composite Attribute



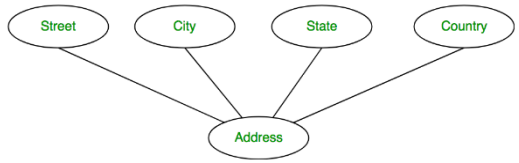
Can be split into smaller parts. For example, the Address attribute of the student Entity type consists of Street, City, State, and Country. **In ER diagram, the composite attribute is represented by an oval comprising of ovals.**

4. Multivalued Attribute



May contain multiple values. For example, Phone_No (can be more than one for a given student). **In ER diagram, a multivalued attribute is represented by a double oval.**

5. Complex Attribute



Composite + mutli-valued Example: address (street-city)

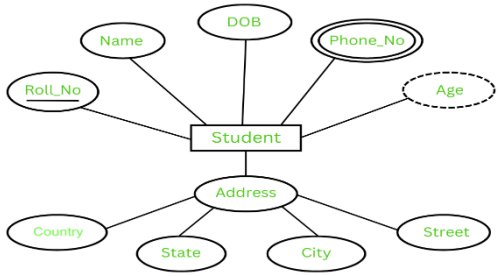
Composite + Multi-valued

6. Derived Attribute

Calculated from another value. e.g.; Age (can be derived from DOB). **In ER diagram, the derived attribute is represented by a dashed oval.**



The Complete Entity Type Student with its Attributes can be represented as:



Domain: The set of possible values for an attribute.

Schema: The structure or design of the database (names of tables, attributes, data types).

Instance: The actual content of the database at a particular moment.

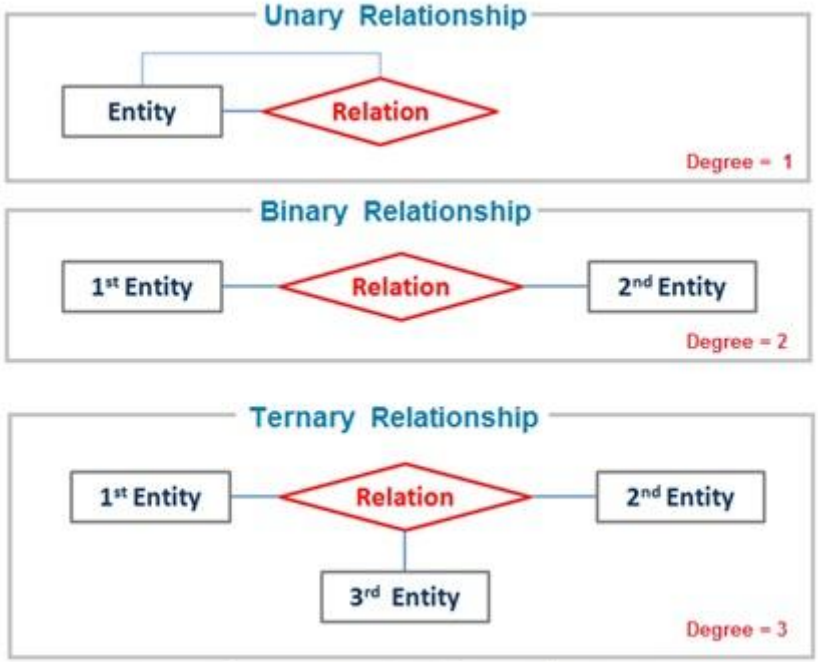
6.3. Relationships.

- **A relationship is an association among entities.** Examples: Student “enrolls in” Course - Employee “works for” Department



- **Relationship Degree: the number of entities types that participate inrelationship**

Degree	Description	Example
Unary (1 entity)	Entity related to itself	Employee supervises Employee
Binary (2 entities)	**Most common**	**Student–Course**
Ternary (3 entities)	3 entities involved	Doctor–Patient–Medicine



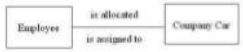
6.4. Relationship Cardinality and Constraints

- **Cardinality (Mapping Multiplicity):** Describes **how many instances** of one entity can be related to another.

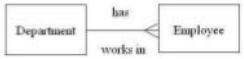
Type	Example	Represent
One-to-One (1:1)	Department → Manager	
One-to-Many (1:N)	Department → Employees	
Many-to-Many (M:N)	Student ↔ Course	

There are three degrees of Cardinality, known as:

- **One-to-One** (1:1):One occurrence of an entity relates to only one occurrence in another entity.

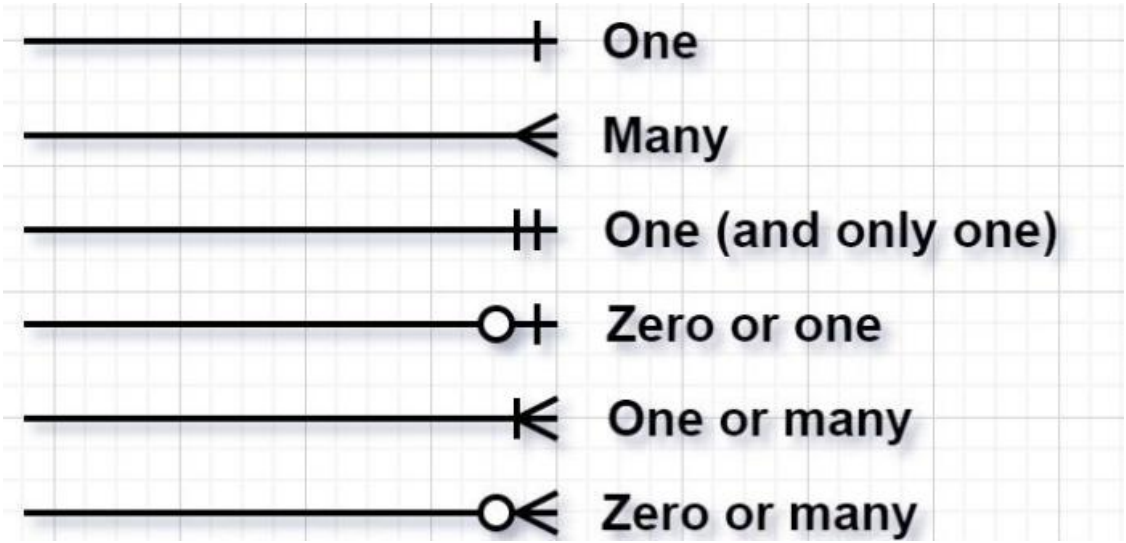


- **One-to-Many** (1:M): One occurrence in an entity relates to many occurrences in another entity.



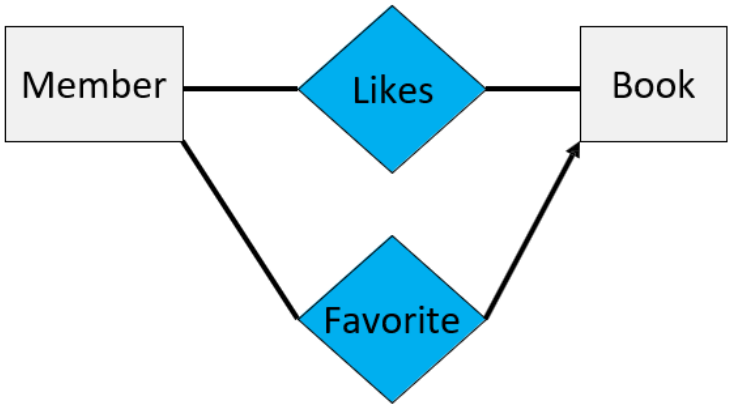
- **Many-to-Many** (M:N) Many occurrences in an entity relate to many occurrences in another entity.





Examples:

Many to one



A member likes many books and a book is liked by many members

A member selects at most one book as their favorite, while a book can be chosen as a favorite by multiple members.

One to one



A book is the best-seller for 0 or 1 publisher.

A publisher has 0 or 1 best seller.

- **Participation**

Type	Meaning
Total	Each entity must participate
Partial	Some may not participate



6.5. Keys

In the context of a relational database, keys are one of the basic requirements of a relational database model.

- Keys are fundamental components that ensure **data integrity**, **uniqueness** and **efficient access**. It is widely used to identify the tuples(rows) uniquely in the table.

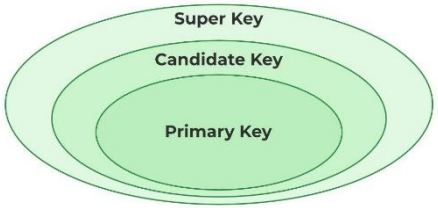
Keys are important in a Database Management System (DBMS) for several reasons:

- Uniqueness: Keys ensure that each record in a table is unique and can be identified distinctly.
- Data Integrity: Keys prevent data duplication and maintain the consistency of the data.
- Efficient Data Retrieval: By defining relationships between tables, keys enable faster querying and better data organization

Without keys, it would be extremely difficult to manage large datasets and queries would become inefficient and prone to errors.

Types of Keys

1. Super Key



A **Super Key** is a **set of one ore more attributes** that uniquely identify an entity in a table.
It can contain **extra attributes** that are not necessary for unique identification.
Example:
For an **Employee** entity: {EmployeeID}, {EmployeeID, Name}, and {EmployeeID, Email} are all **super keys** because EmployeeID alone is enough to identify each employee.

2. Candidate Key

A **Candidate Key** is a minimal **super key** — it has no unnecessary attributes.
There can be **multiple candidate keys** in an entity.
Example: In the **Student** entity:
Here both: {StudentID} -{Email}
are candidate keys because each can uniquely identify a student, and neither contains extra attributes.

StudentID	Email	Name
S01	s1@uni.com	Ali
S02	s2@uni.com	Sara

3. Primary Key

A Primary Key is the chosen candidate key that uniquely identifies each record. There can be only one primary key per entity.
Example: In the **Student** entity:: {StudentID}

4. Alternate Key

When there are multiple candidate keys, the ones not chosen as the primary key are called alternate keys.
Example:
From the Student example above: StudentID → Primary Key and email → Alternate Key

5. Foreign Key

A Foreign Key is an attribute in one entity that refers to the primary key of another entity.
It defines a relationship between two entities.

- Example:

Entities:
Department(DeptID, DeptName)
Employee(EmpID, Name, DeptID)

6. Composite Key

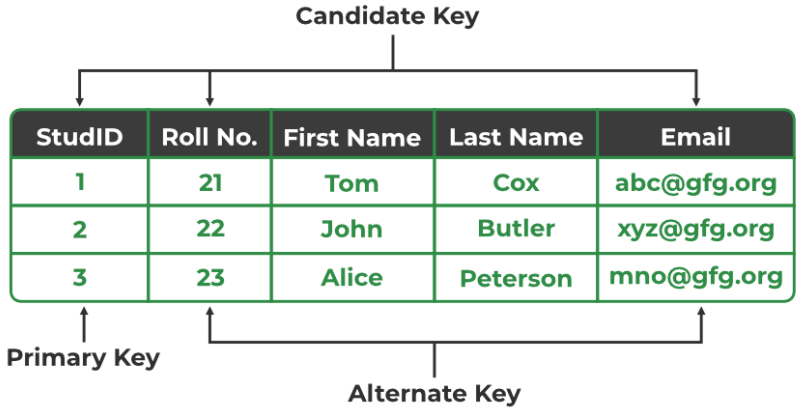
A Composite Key consists of two or more attributes together forming a unique identifier.
Used when a single attribute cannot uniquely identify a record.

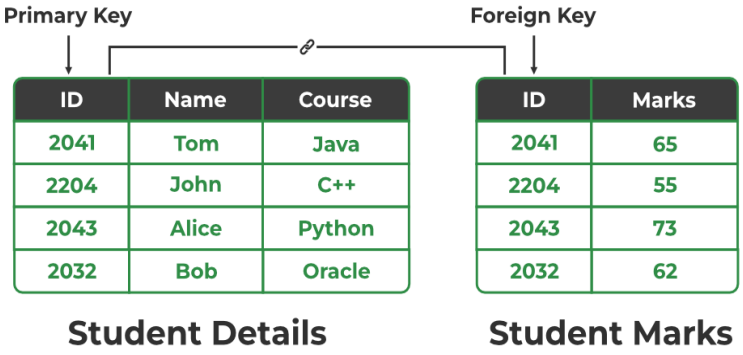
Because a student can enroll in many courses, but the pair (StudentID, CourseID) uniquely identifies each enrollment.

StudentID	CourseID	Grade
S01	C01	A
S01	C02	B

7. secondary Key

A **Secondary Key** is used **for searching or indexing**, not for unique identification.
Example: Searching for employees by Department or JobTitle — those fields are **secondary keys**.





Key Type	Definition	Example
Super Key	Uniquely identifies record (may include extra fields)	`{EmployeeID, Name}`
Candidate Key	Minimal super key	`{EmployeeID}` or `{Email}`
Primary Key	Chosen candidate key	`EmployeeID`
Alternate Key	Candidate key not chosen	`Email`
Foreign Key	Links two entities	`DeptID` in Employee
Composite Key	Multiple fields used as a key	`(StudentID, CourseID)`
Secondary Key	Used for search/sort	`Department`

Summary

The **ER model** is the **first step** in database design:

1. Identify entities and their attributes.
2. Define relationships and constraints.
3. Represent everything visually with an ERD.
4. Convert it to a **relational schema** (tables).

Relational Integrity Constraints

Integrity constraints ensure **data accuracy and consistency** in a relational database.

Integrity Constraints preserve data correctness:

- Domain Constraint
- Entity Integrity
- Referential Integrity
- Key Constraint
- Null Constraint

1. Domain Constraint

Each attribute must contain **only valid values** from its domain.

Age must be between 18 and 60.

2. Entity Integrity

Every table must have a **Primary Key**, and **no part of the primary key can be NULL**.

DeptID cannot be NULL because it's the primary key.

3.Referential Integrity

Ensures that **Foreign Key values** in one table **match existing Primary Key values** in another table (or are NULL).

DeptID in STUDENT must exist in the DEPARTMENT table.

4. Key Constraint

No two tuples in a relation can have the same value for a primary key.

5. Null Constraint

An attribute declared as **NOT NULL** must always contain a value.

Next Chapter: Enhanced ER (EER) Model — Optional Extension

Sometimes real-world concepts are more complex.
EER adds features like:

Concept	Description	Example
Generalization	Subclass inherits from superclass	`Employee` → `Manager`, `Engineer`
Specialization	Dividing an entity into subtypes	`Vehicle` → `Car`, `Truck`
Aggregation	A relationship treated as an entity	`Teacher` teaches `Course` → becomes entity `Teaching`

2.8 Mapping ER Model to Relational Schema

After designing the ER diagram, we convert it to **tables** (relations).

ER Concept	Relational Concept
Entity	Table
Attribute	Column
Relationship	Foreign key
Multivalued Attribute	Separate table
Weak Entity	Table with foreign key to strong entity

Example:

ER Relationship:
Student — enrolls in — Course (M:N)

→ Relational tables:

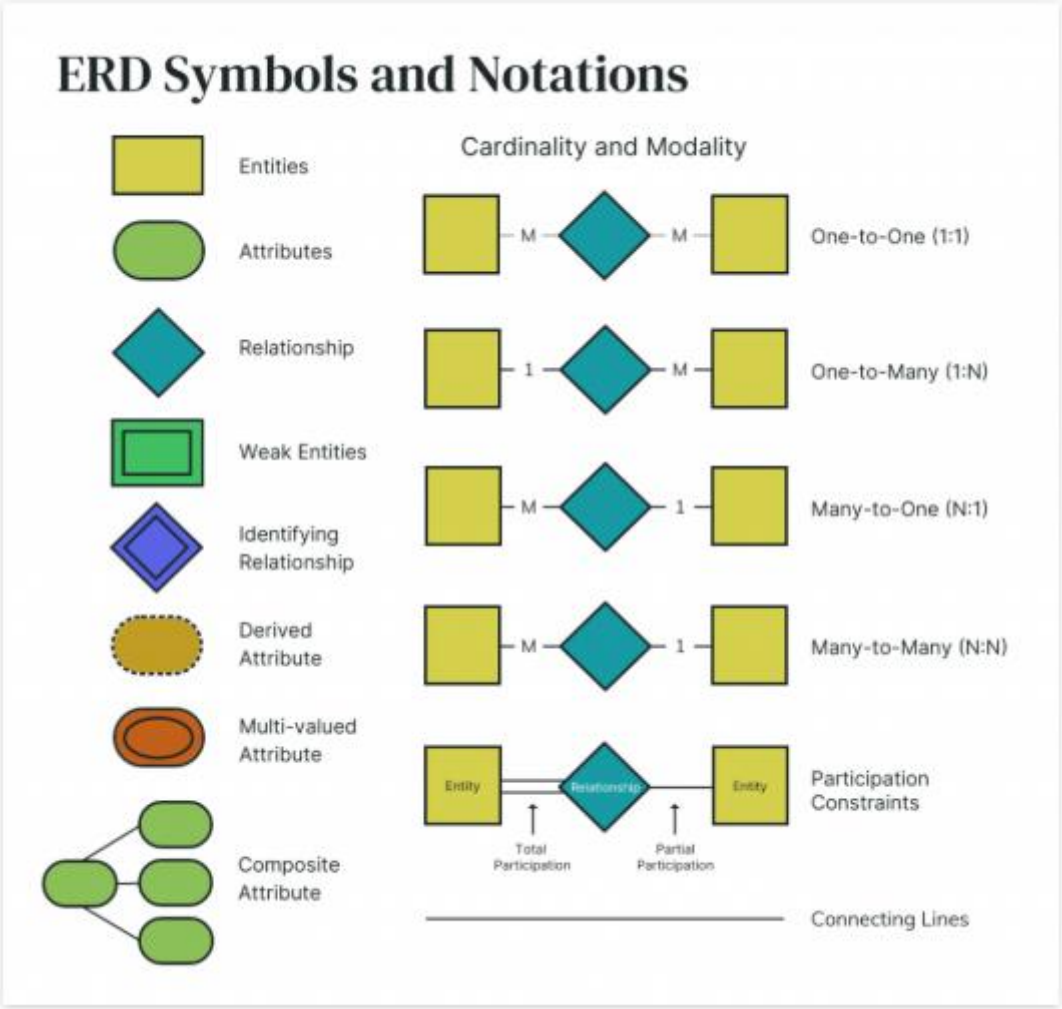
```
STUDENT(StudentID, Name)
COURSE(CourseID, Title)
ENROLLS(StudentID, CourseID, Grade)
```

1. Data Integrity

2. Entity constraints

Operation	Symbol / Keyword	Description
Selection (σ)	σ<condition>(R)	Extracts rows satisfying a condition
Projection (π)	π<columns>(R)	Extracts specific columns
Union (∪)	R ∪ S	Combines tuples from both relations (no duplicates)
Set Difference (-)	R - S	Tuples in R but not in S
Cartesian Product (×)	R × S	All combinations of tuples
Rename (ρ)	ρ<newname>(R)	Renames a relation
Join (⋈)	R ⋈ S	Combines related tuples based on a common attribute

ERD



symbols and notations

