Database Systems

Week 02_Lecture 04_05

Revision

Database Development Life Cycle

Database Design Models

Database Architectures

- 1-tier
- 2-tier (client->database server)
- 3-tier (client->application layer-> database layer)

Contents

Database Schema

- 3 Schema Architecture
  - External Schema
  - Logical Schema
  - Physical Schema

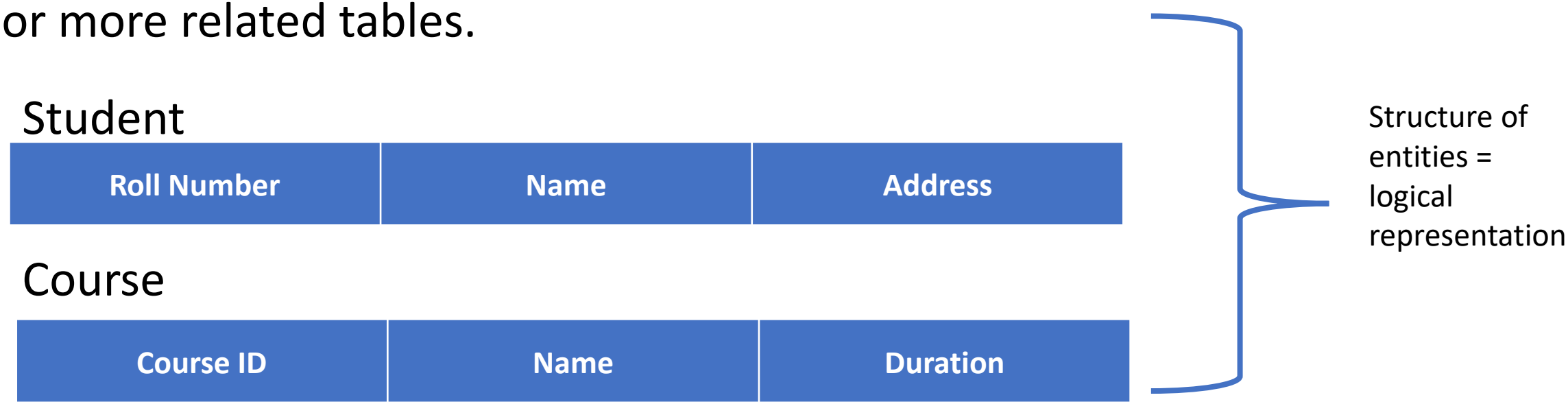
ER Model Basic Concepts

- Entity
- Attributes
  - Type of Attributes
- Entity Sets & Keys
- Relationships
  - Mapping Cardinalities

DATABASE SCHEMA

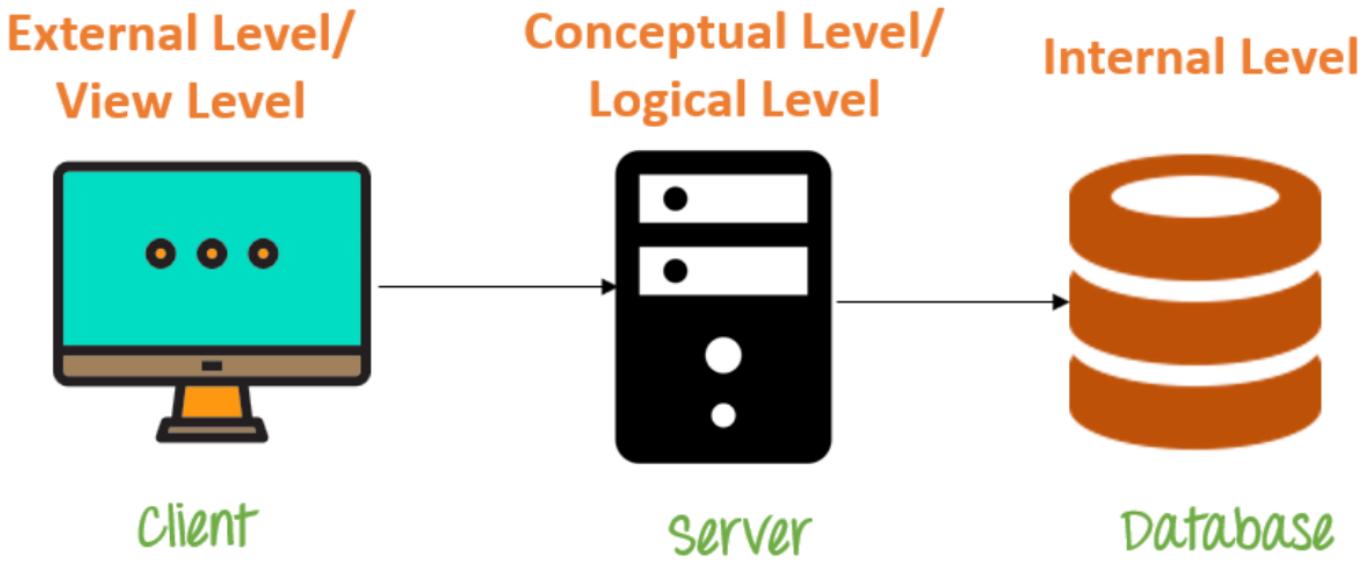
Database Schema

Schema: A logical representation of a database. A schema can have one or more related tables.



Implementation of Schema is through **SQL (data definition language)**

3 schema Architecture (Levels of Data Abstraction)



Internal Schema/Physical Level

The internal schema is the lowest level of data abstraction

It helps you to keeps information about the actual representation of the entire database. Like the actual storage of the data on the disk in the form of records

The internal view tells us what data is stored in the database and how

Conceptual Level/Logical Schema

This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, and integrity constraints.

This schema hides information about the physical storage structures and focuses on describing data types, entities, relationships, etc.

In the conceptual level, the data available to a user must be contained in or derivable from the physical level

External Schema/View Level

An external schema describes the part of the database which specific user is interested in. It hides the unrelated details of the database from the user.

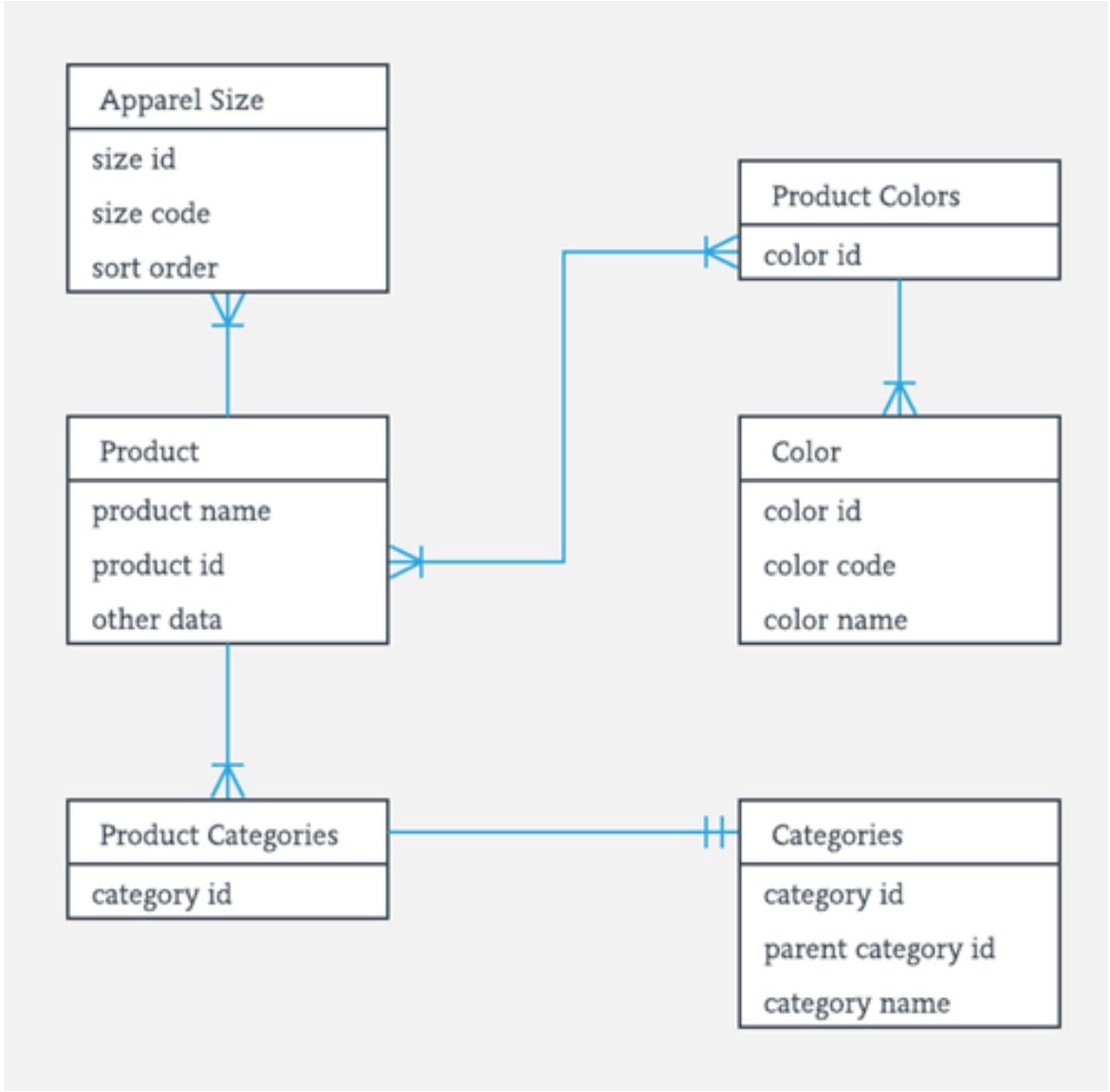
Each external view is defined using an external schema, which consists of definitions of various types of external record of that specific view.

An external view is just the content of the database as it is seen by some specific particular user. For example, a user from the sales department will see only sales related data.

Advantages of 3 Schema Database

- Every user should be able to access the same data but able to see a customized view of the data.
- The user need not to deal directly with physical database storage detail.
- DBMS Architecture allows you to make changes on the presentation level without affecting the other two layers
- As each tier is separate, it is possible to use different sets of developers
- It is more secure as the client doesn't have direct access to the database business logic
- In case of the failure of the one-tier no data loss as you are always secure by accessing the other tier

ER MODEL BASIC CONCEPTS



ER Model

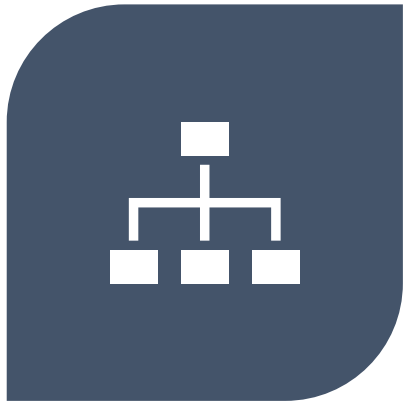
The ER model defines the conceptual view of a database.

It works around real-world entities and the associations among them. At view level, the ER model is considered a good option for designing databases.

ER Model Objectives

- 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

Components of the ER Model



ENTITIES

ATTRIBUTES

RELATIONSHIPS

Entity

An entity is a real-world object

- For example, in a school database, students, teachers, classes, and courses offered can be considered as entities. All these entities have some attributes or properties that give them their identity.

Entity Set: An entity set is a collection of similar types of entities. An entity set may contain entities with attribute sharing similar values.

- For example, a Students set may contain all the students of a school; likewise a Teachers set may contain all the teachers of a school from all faculties.

Example of Entities

Person: Employee, Student, Patient

Place: Store, Building

Object: Machine, product, and Car

Event: Sale, Registration, Renewal

Concept: Account, Course

Attributes

Course (name, code, course prerequisites)
Student (first_name, surname, address, age)
Book (title, ISBN, price, quantity in stock)

Entities are represented by means of their properties, called **attributes**. All attributes have values.

For example, a student entity may have name, class, and age as attributes.

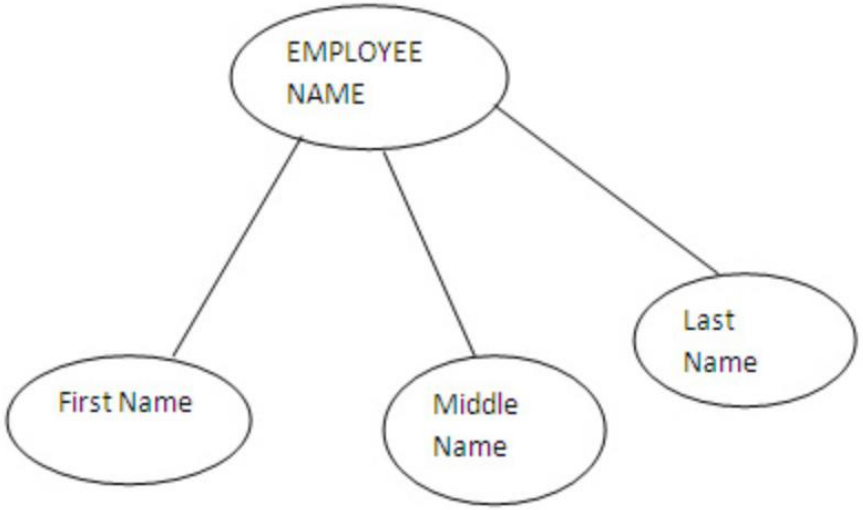
There exists a domain or range of values that can be assigned to attributes.

For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

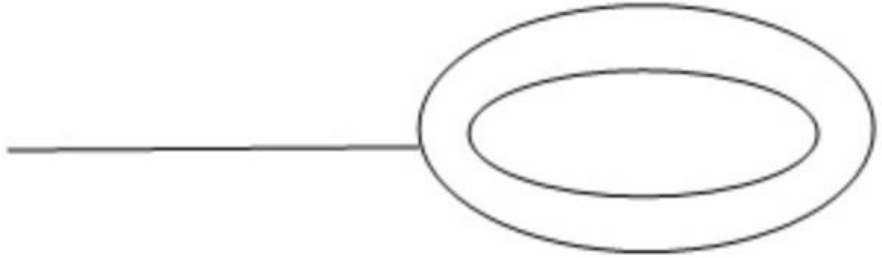
Types of Attributes

1. **Simple attribute** − Simple attributes are atomic values, which cannot be divided further. For example, a student's phone number is an atomic value of 10 digits.
2. **Composite attribute** − Composite attributes are made of more than one simple attribute. For example, a student's complete name may have first_name and last_name.
3. **Derived attribute** − Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database. For example, average_salary in a department should not be saved directly in the database, instead it can be derived. For another example, age can be derived from data_of_birth.
4. **Single-value attribute** − Single-value attributes contain single value. For example − Social_Security_Number.
5. **Multi-value attribute** − Multi-value attributes may contain more than one values. For example, a person can have more than one phone number, email_address, etc.

Types of Attributes



Composite Attribute



Multi Value Attribute

Derived Attribute

Entity-Set and Keys

- Key is an attribute or collection of attributes that uniquely identifies any two tuples in the table
- For example, the roll_number of a student makes him/her identifiable among students.

Candidate Key − A minimal super key is called a candidate key. An entity set may have more than one candidate key.

Primary Key − A primary key is one of the candidate keys chosen by the database designer to uniquely identify the entity set.

Key

- An attribute from the table
- **Use**: To uniquely identify any two or more tuples in the table

Roll #	S_Name	City	Age
1	Amna	Lahore	20
2	Mahwish	Karachi	21
3	Amna	Lahore	20

Student Table Unique Values

1. **NIC #**
2. **Roll #**
3. **Registration #**
4. **License #**
5. **Voter ID**
6. **Etc.**

Relationships

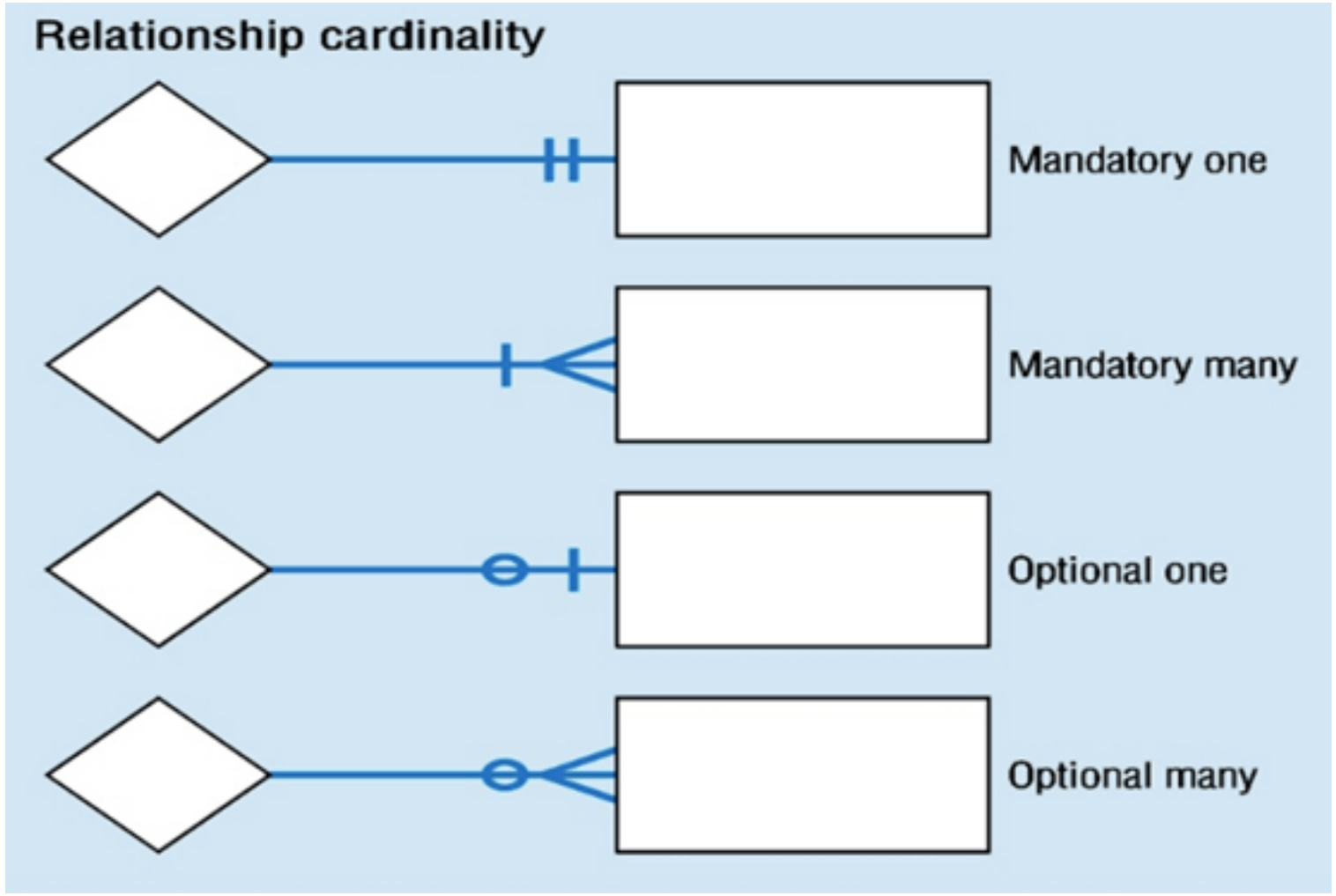
- The association among entities is called a relationship. For example, an employee **works_at** a department, a student **enrolls** in a course. Here, Works_at and Enrolls are called relationships.

Mapping Cardinalities

Cardinality defines the number of entities in one entity set, which can be associated with the number of entities of other set via relationship set.

Different types of cardinal relationships are:

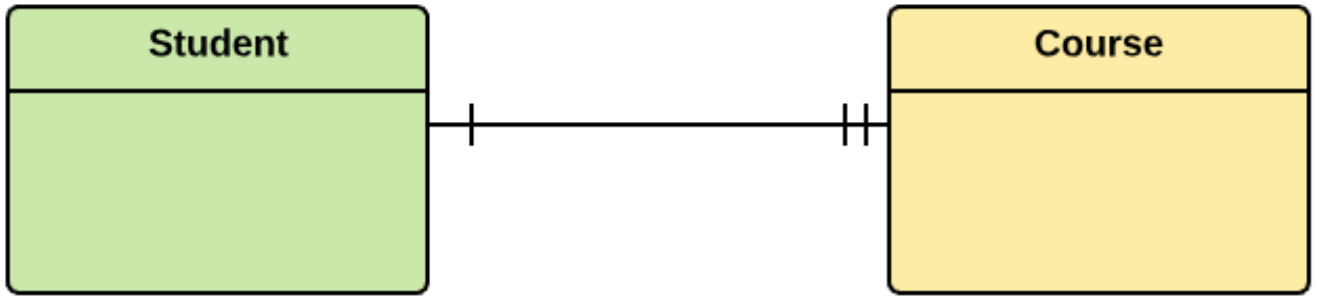
One-to-One Relationships	One-to-Many Relationships	May to One Relationships	Many-to-Many Relationships



Mapping Cardinalities

One to One

- One entity from entity set X can be associated with at most one entity of entity set Y and vice versa.
- Example: One student can register for numerous courses. However, all those courses have a single line back to that one student.



One to Many

- One entity from entity set X can be associated with multiple entities of entity set Y, but an entity from entity set Y can be associated with at least one entity.
- For example, one class is consisting of multiple students.

Many to One

- More than one entity from entity set X can be associated with at most one entity of entity set Y. However, an entity from entity set Y may or may not be associated with more than one entity from entity set X.
- For example, many students belong to the same class.

Many to Many

- One entity from X can be associated with more than one entity from Y and vice versa.
- For example, Students as a group are associated with multiple faculty members, and faculty members can be associated with multiple students.

End of Week 02

Week 03 Contents

ER Diagram Notations

Steps to Create an ERD

Case Study

Assignment