# DATABASE SYSTEMS

Dr. Noha Nagy

## Database Design

# Database Design



- **Database design: Why do we need it?**
  - **Agree on structure** of the database before deciding on a particular implementation
- **Consider issues such as:**
  - What entities to model
  - How entities are related
  - What constraints exist in the domain
- **Several formalisms exist**
  - We discuss one flavor of E/R diagrams

# Database Design Process



1. Requirements Analysis
2. Conceptual Design
3. Logical, Physical, Security, etc.

1. **Requirements analysis**
   - What is going to be stored?
   - How is it going to be used?
   - What are we going to do with the data?
   - Who should access the data?

Technical and non-technical people are involved

# Database Design Process





## 2. Conceptual Design

- A high-level description of the database
- Sufficiently precise that technical people can understand it
- But, not so precise that non-technical people can't participate

This is where E/R fits in.

# Database Design Process



1. Requirements Analysis
2. Conceptual Design
3. Logical, Physical, Security, etc.

**3. More:**

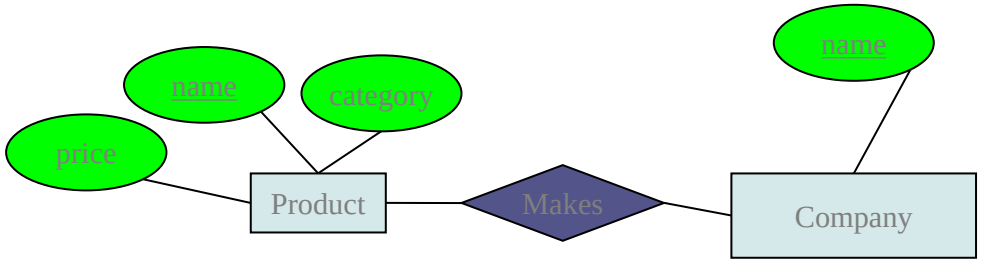
- Logical Database Design
- Physical Database Design
- Security Design

# Database Design Process



1. Requirements Analysis
2. Conceptual Design
3. Logical, Physical, Security, etc.

**E/R Model & Diagrams used**

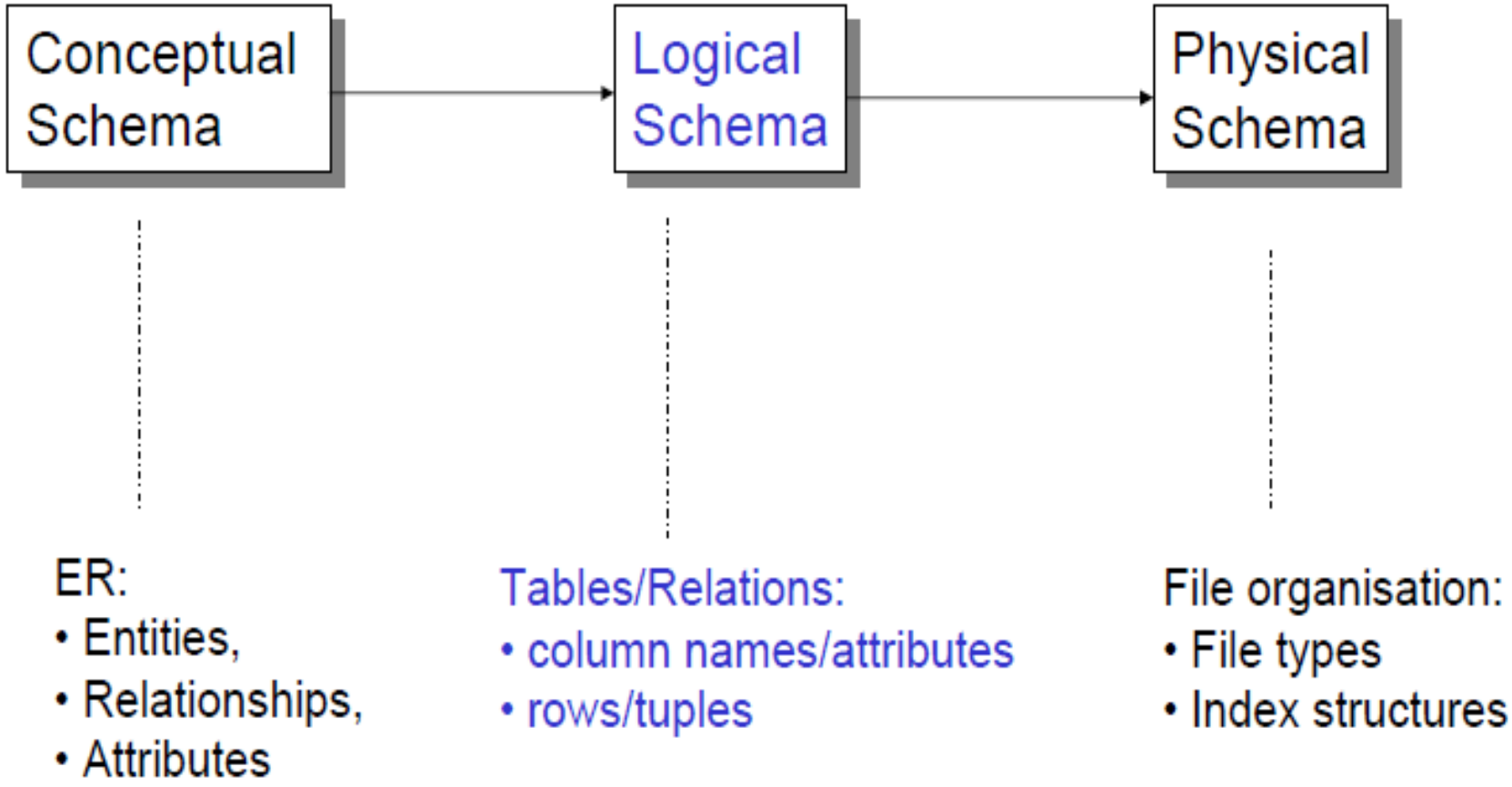



This process is iterated **many** times

E/R is a *visual syntax* for DB design which is ***precise enough*** for technical points, but ***abstracted enough*** for non-technical people

# Different Schemas





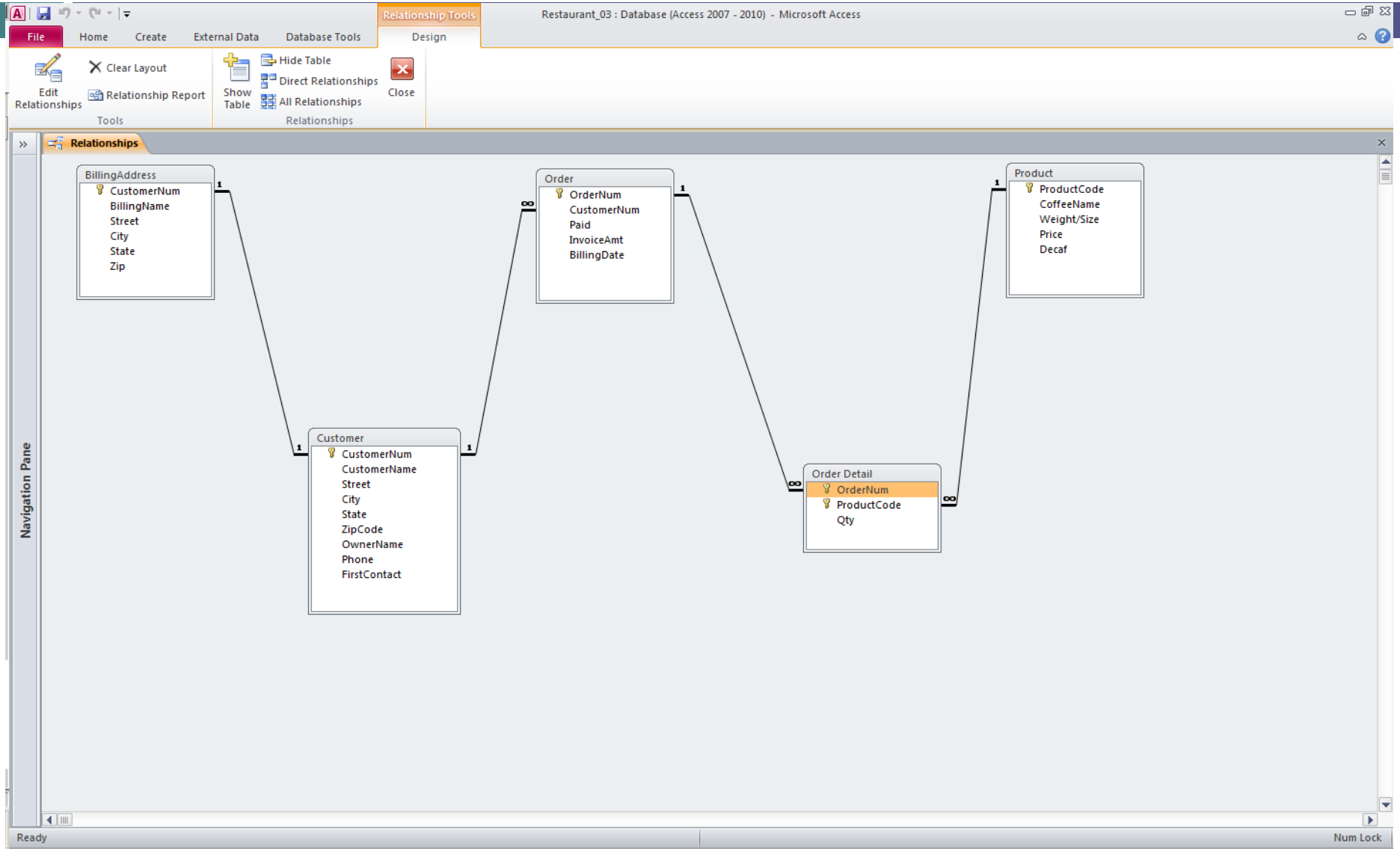
Database Design

# CONCEPTUAL DATA MODELS

# Data Model

- Model: an abstraction of a real-world object or event
  - Useful in understanding complexities of the real-world environment
- Data model
  - A diagram that displays a set of tables and the relationships between them

# Access Data Model using ERD



# Basic Modeling Concepts



- Database design is both art and science.
- A data model is the relatively simple representation, usually graphic, of complex real-world data structures. It represents data structures and their characteristics, relations, constraints, and transformations.
- The database designer usually employs data models as communications tools to facilitate the interaction among the designer, the applications programmer, and the end user.
- A good database is the foundation for good applications.

# Conceptual Model



- The conceptual model represents a global view of the data. It is an enterprise-wide representation of data as viewed by high-level managers.
- Entity-Relationship (E-R) model is the most widely used conceptual model.
- The conceptual model forms the basis for the conceptual schema.
- The conceptual schema is the visual representation of the conceptual model.
- The conceptual model is independent of both software (software independence) and hardware (hardware independence).

# Why Conceptual Design is Worthwhile?



- Allows users to influence design
- Independence from any particular DBMS
- Conceptual schema is a permanent description of database requirements
- Concepts are usually easy to understand
- Can be used as a tool to communicate with non-technical users

# Entity-Relationship Model



- Most popular conceptual model
- Describes data as **entities, attributes and relationships**
- Several tools for database design are based on this model
  - DeZign for Databases
  - ERwin/SQL
  - PowerDesigner
  - Designer 2000
  - SmartDraw

# Entity Relationship (E-R) Model



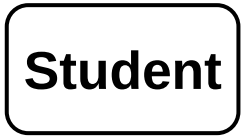
- E-R Model Components
  - **Entities**
  - **Attributes**
  - **Relationships**
- **Example**

In a University database we might have **entities** for **Students, Modules and Lecturers.** Students might have **attributes** such as their **ID, Name, and Course,** and could have **relationships** with Modules (enrolment) and Lecturers (tutor/tutee)

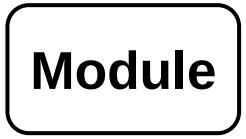
# E-R Model Components



- **Entities**
  - **In E-R models an entity refers to the entity set.**
  - **An entity is represented by a rectangle containing the entity's name.**
  - **Entity name should be noun.**


- **How to find Entities**




- Entity: "...anything (people, places, objects, events, etc.) about which we store information (e.g. supplier, machine tool, employee, utility pole, airline seat, etc.)."
- We look for **nouns**

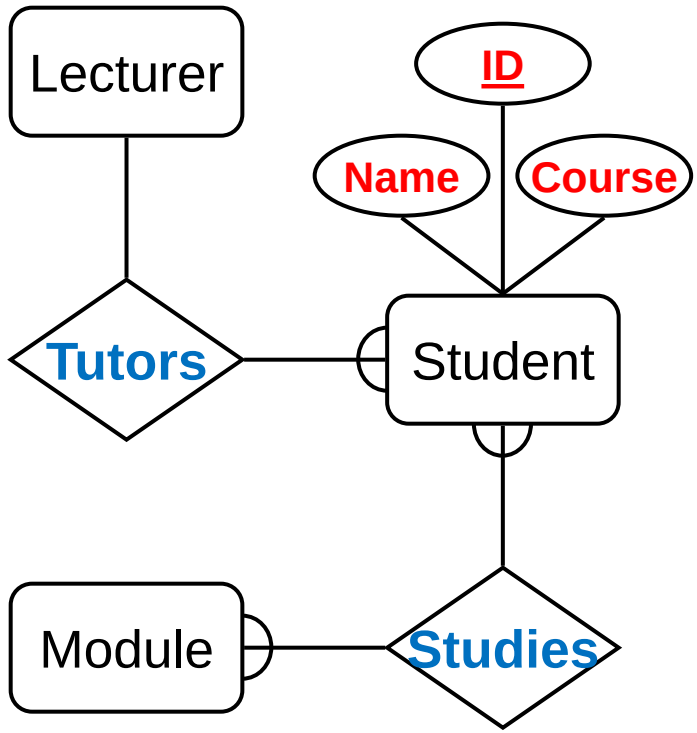
# Identify Entities

- **Identify the entities to support requirements**
- **Think of an entity as**
  - A template to store a record or object.
- **A potential entity should have:**
  - Multiple instances (i.e. the requirements should imply the need for more than one entity of that class)
  - NO SINGELTON ENTITIES ALLOWED IN THE DATABASE, so no entity to store a single record.
  - Entity Names should be Singular No PLURAL

# E-R Model Components Cont'

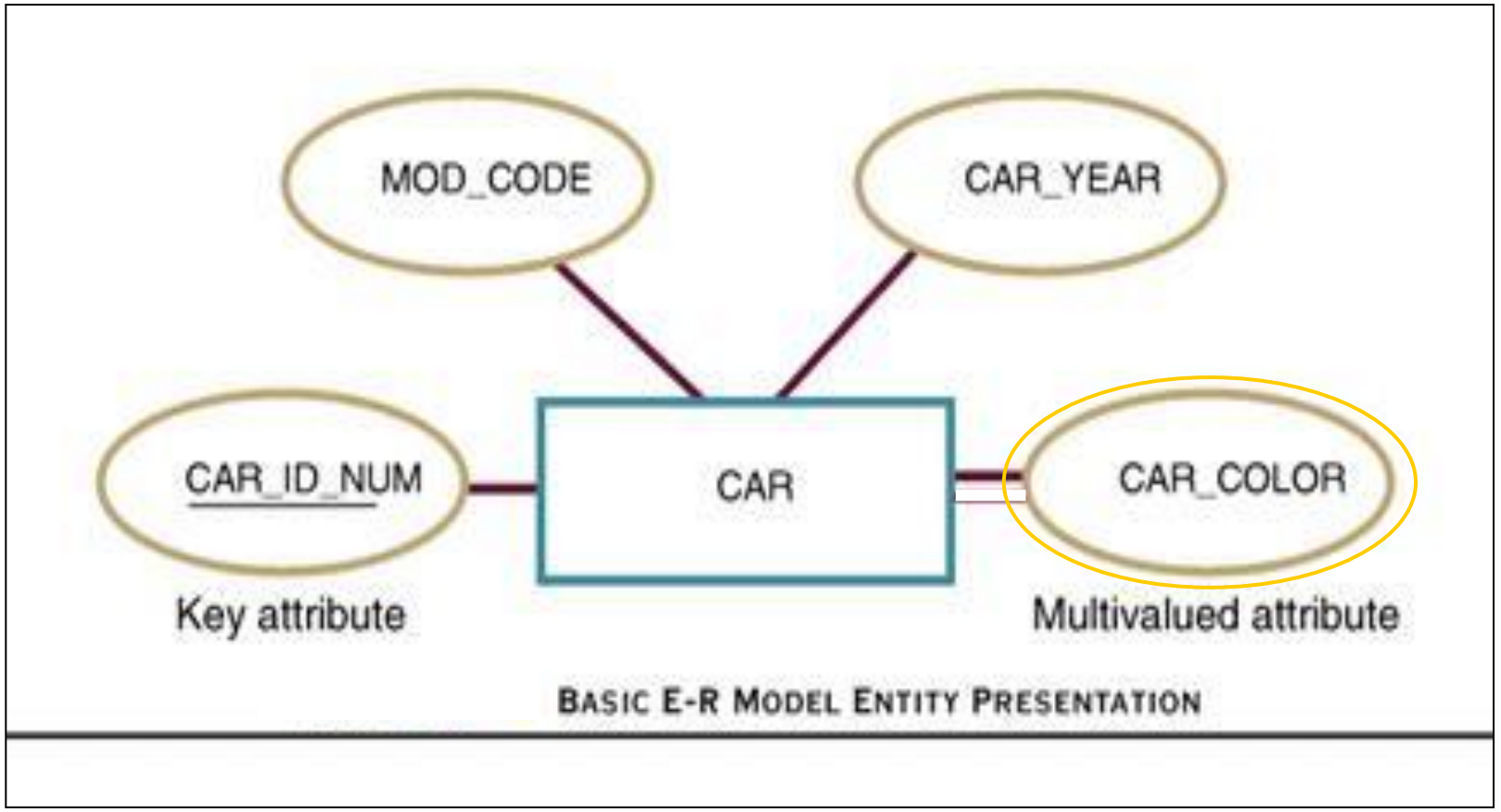


- **Attributes**
  - Attributes are represented by ovals and are connected to the entity with a line.
  - Each oval contains the name of the attribute it represents.
  - Attribute name should be noun.
  - Attributes have a domain -- the attribute's set of possible values.
  - Primary keys are underlined.
- **Relationships**
  - **Represented by lines between entities**



# Basic E-R Model Entity Presentation

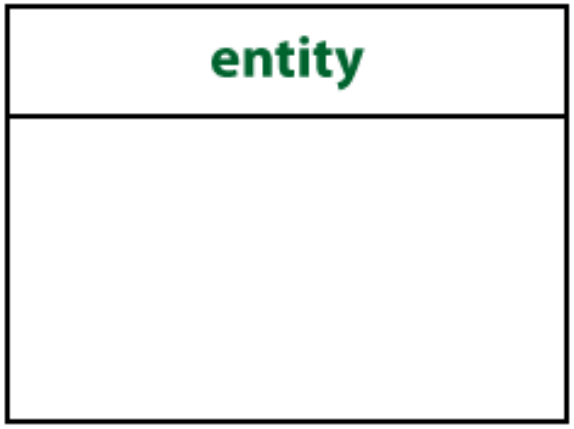




# Crow's Foot Notation



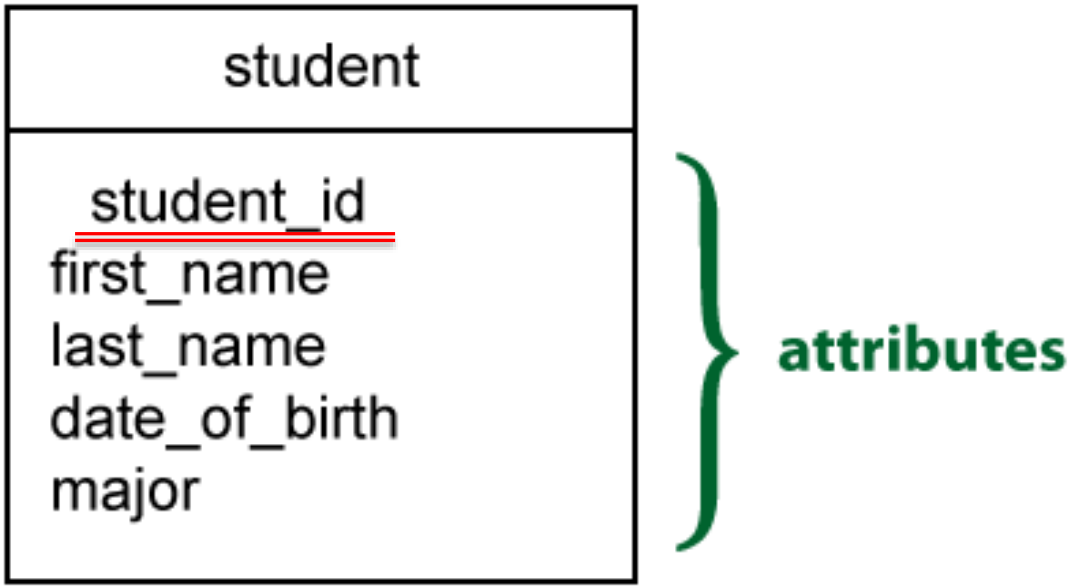
- Known as IE notation (most popular)
- Entity:
  - Represented by a rectangle, with its name on the top. The name is singular (entity) rather than plural (entities).



# Attributes



- Identifiers are represented by underlying the name of the attribute(s)



# Classes of Attributes



- A **simple attribute** cannot be subdivided.
  - **Examples: Age, Sex, and Marital status**
- A **composite attribute** can be further subdivided to yield additional attributes.
  - **Examples:ADDRESS [Street, City, State, Zip]**

    **PHONE NUMBER [Area code, number]**
- **Derived Attributes** is not physically stored within the database instead, it is derived by using an algorithm.
  - **Examples: Age [system_date – DOB]**

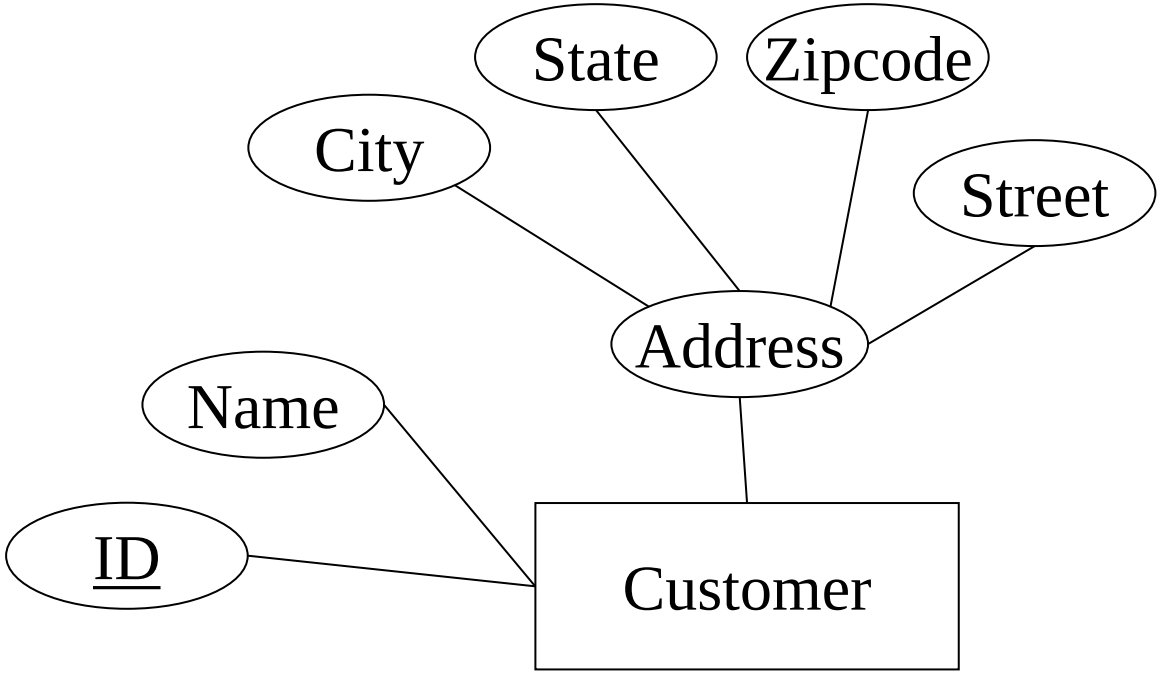
# Classes of Attributes



- A **single-valued attribute** can have only a single value.
  - **Examples: A person can have only one social security number.**
    - **A manufactured part can have only one serial number.**
- **Multivalued attributes** can have many values.
  - **Examples:**
    - **A person may have several college degrees.**
    - **A household may have several phones with different numbers**
  - **Multivalued attributes are shown by a double line connecting to the entity.**

# Composite Attributes



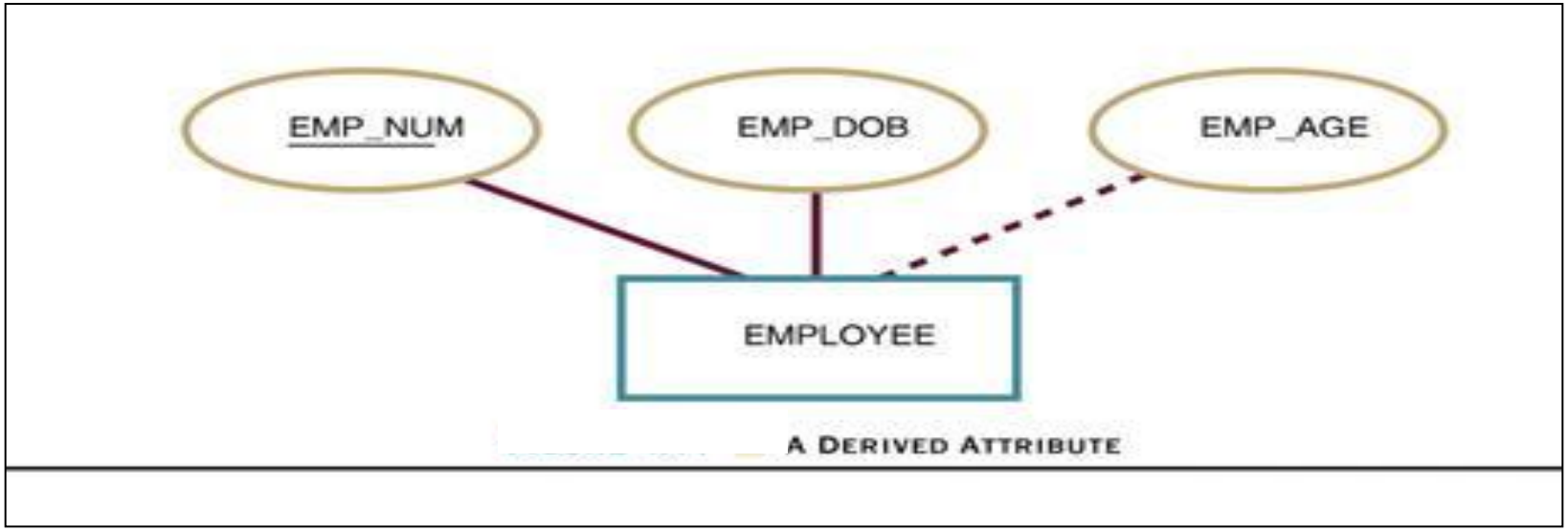


# Derived Attributes



**A derived attribute is not physically stored within the database; instead, it is derived by using an algorithm.**

**Example: AGE can be derived from the data of birth and the current date.**




A DERIVED ATTRIBUTE

# Choosing a good Identifier

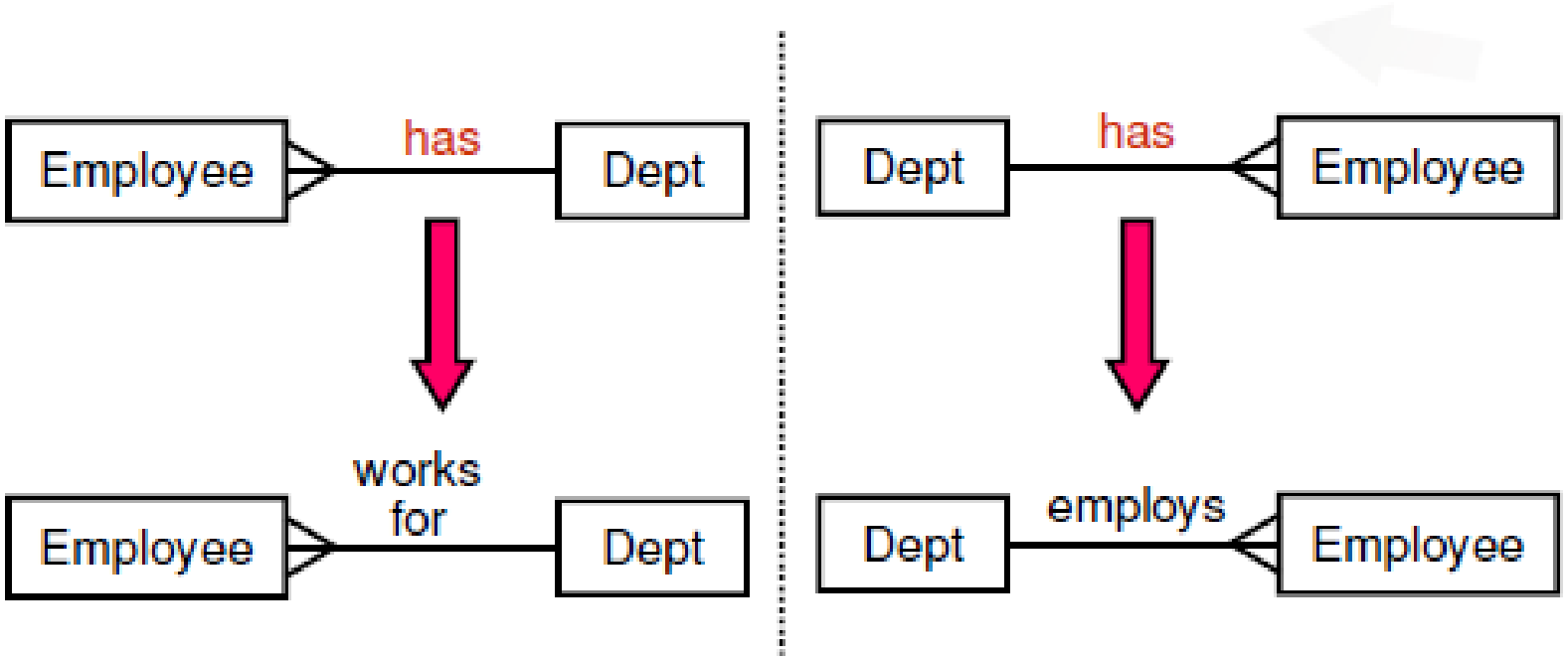


- Select a stable one
- That will not change its value over the life of the entity (e.g. the combination St_name & Tel# would be a poor choice)
- Select the less composite identifier
- Avoid concatenated (intelligent) identifiers
- Identifiers must be guaranteed to have values for all entities of the entity set.

# Relationships

- Relationships are an association between two or more entities
  - Each Student takes several Modules
  - Each Module is taught by a Lecturer
  - Each Employee works for a single Department
- Relationships have
  - A name
  - A set of entities that participate in them
  - A degree - the number of entities that participate (most have degree 2)
  - A cardinality ratio

# Choosing Relationship Characterizations

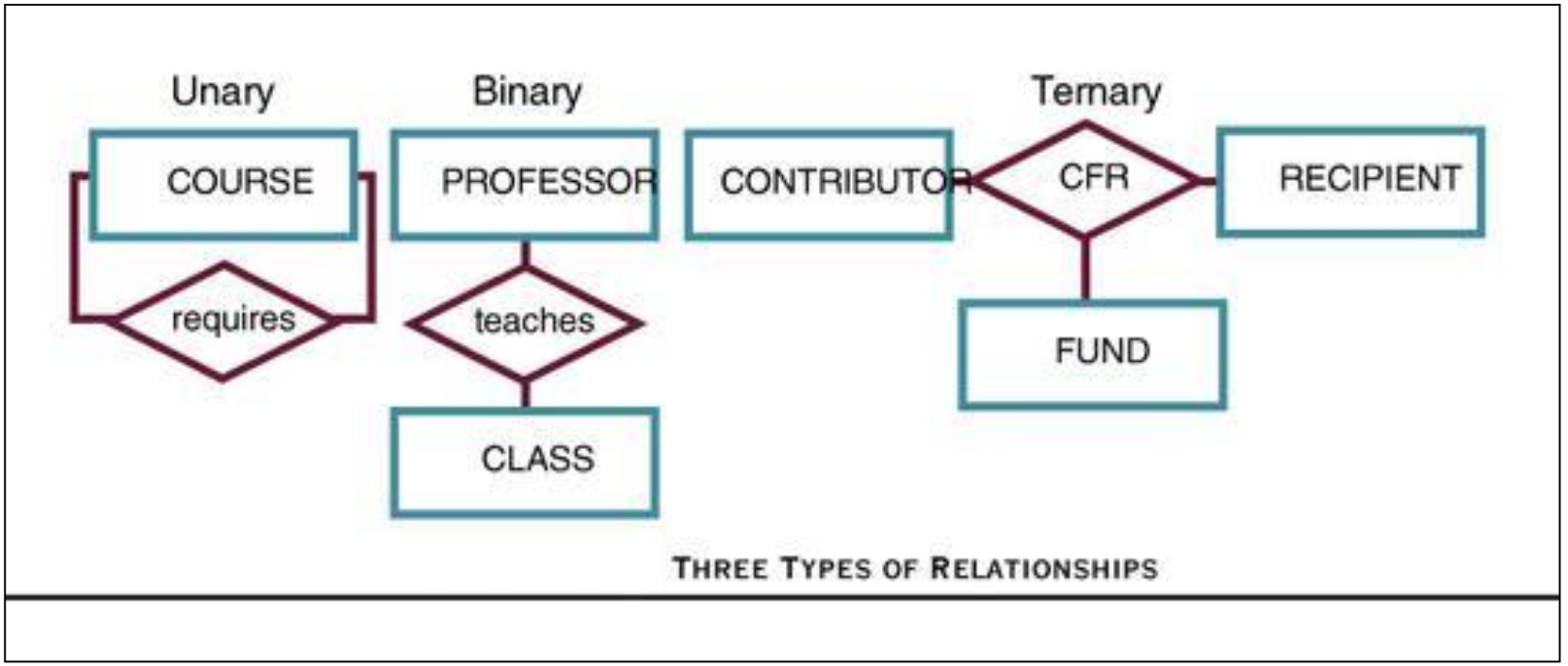



It is worth taking the time to choose the best possible relationship characterization.

# Relationships



- A relationship's **degree** indicates the number of associated entities or participants.
  - **A unary relationship exists when an association is maintained within a single entity.**
  - **A binary relationship exists when two entities are associated.**
  - **A ternary relationship exists when three entities are associated.**




THREE TYPES OF RELATIONSHIPS

# Relationships Types



## Connectivity

**The term connectivity is used to describe the relationship classification (Type)**

- **One-to-One**

Each participant in the relationship could be represented one time at most

- **One-to-Many**

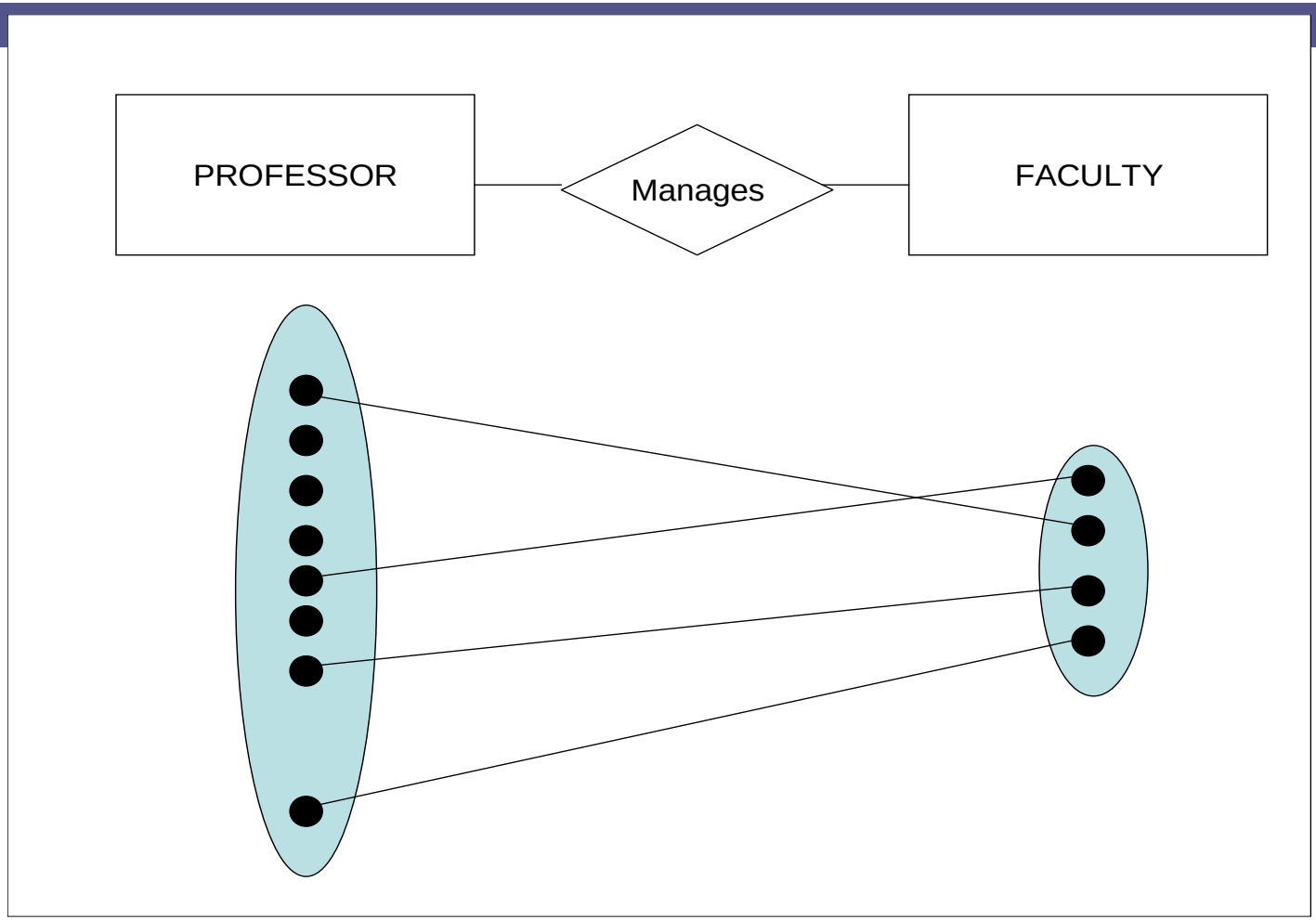
One participant in the relationship could be represented many times

- **Many-To-Many**

Each participant in the relationship could be represented many times

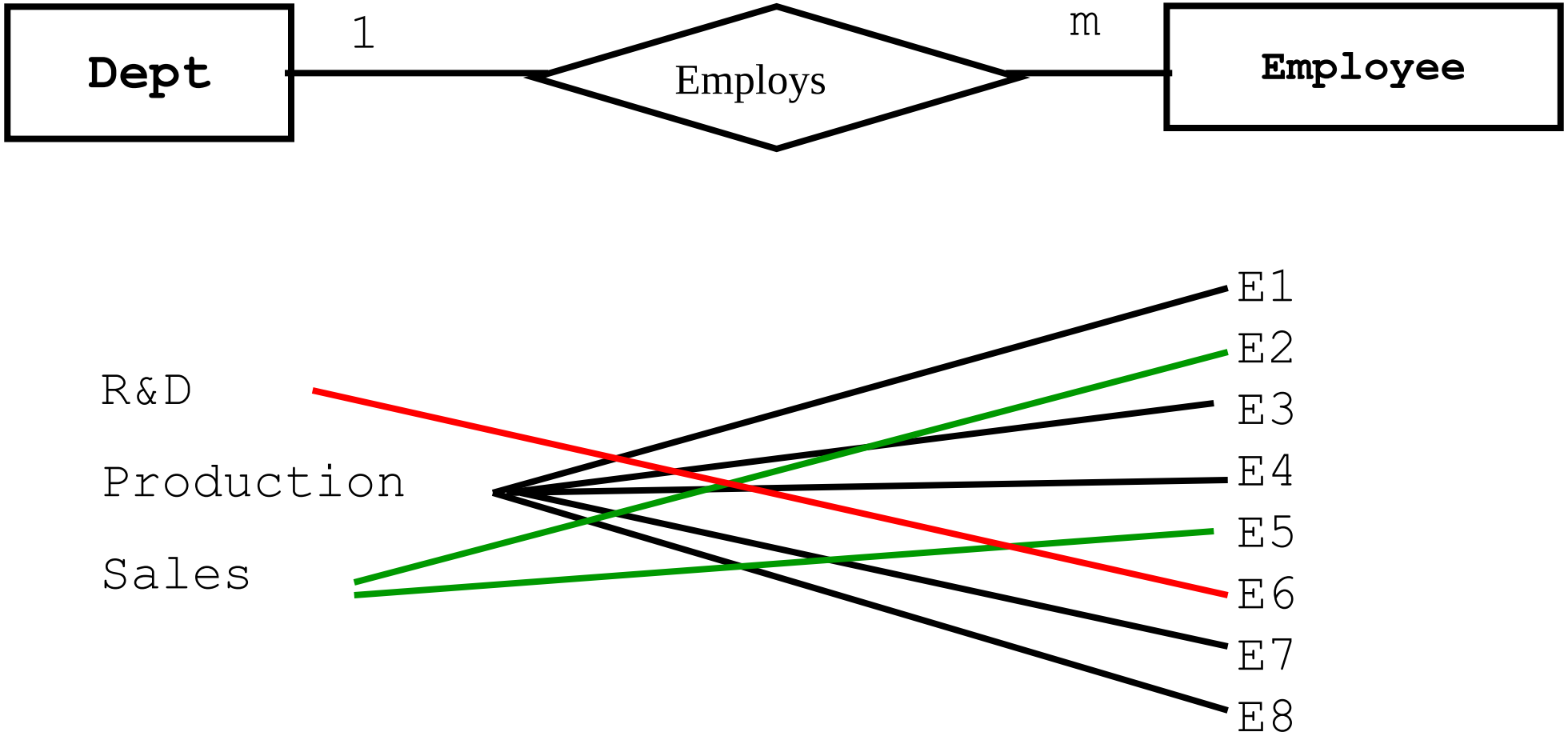
# One-to-One Connectivity





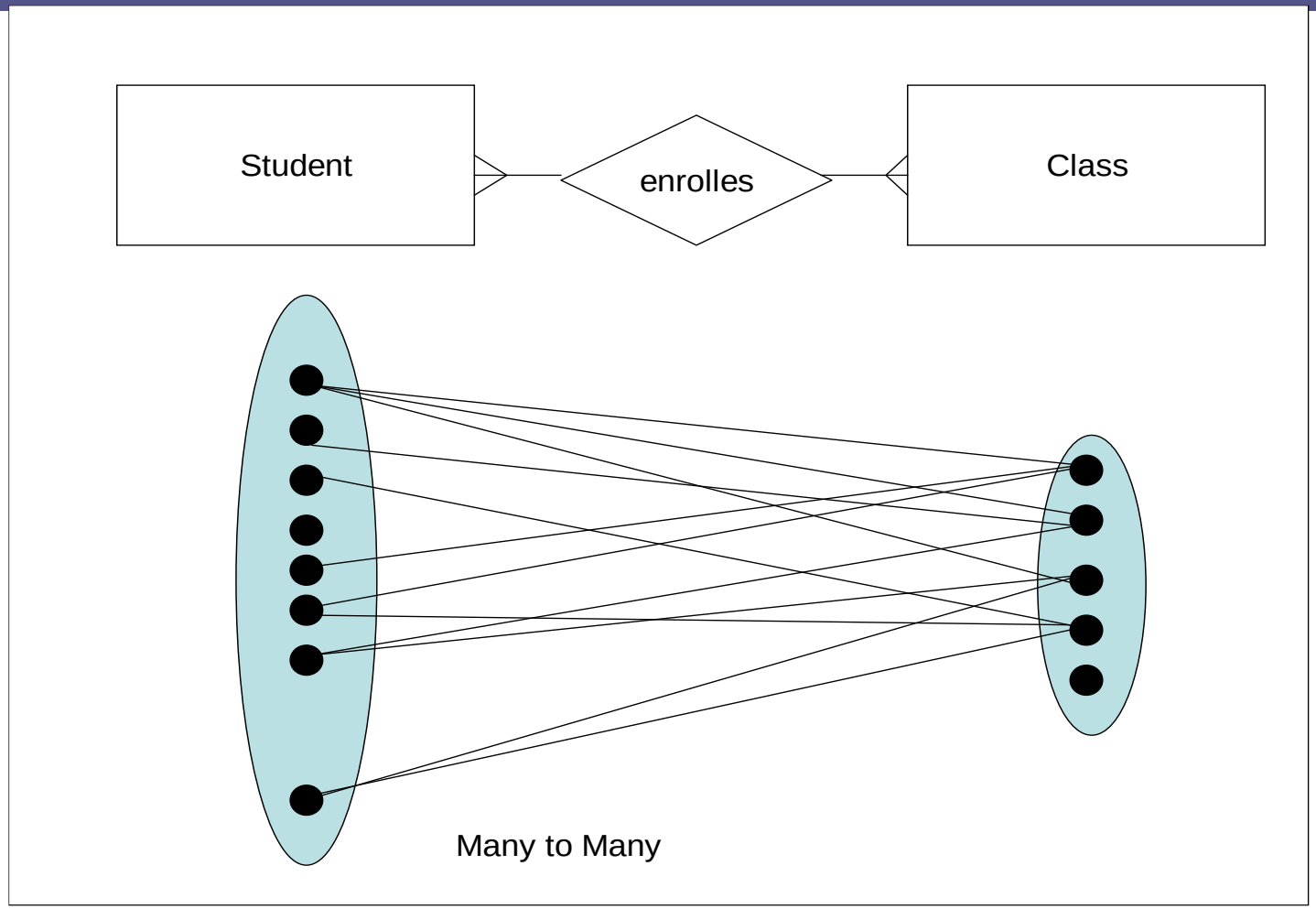
# One-to-Many connectivity





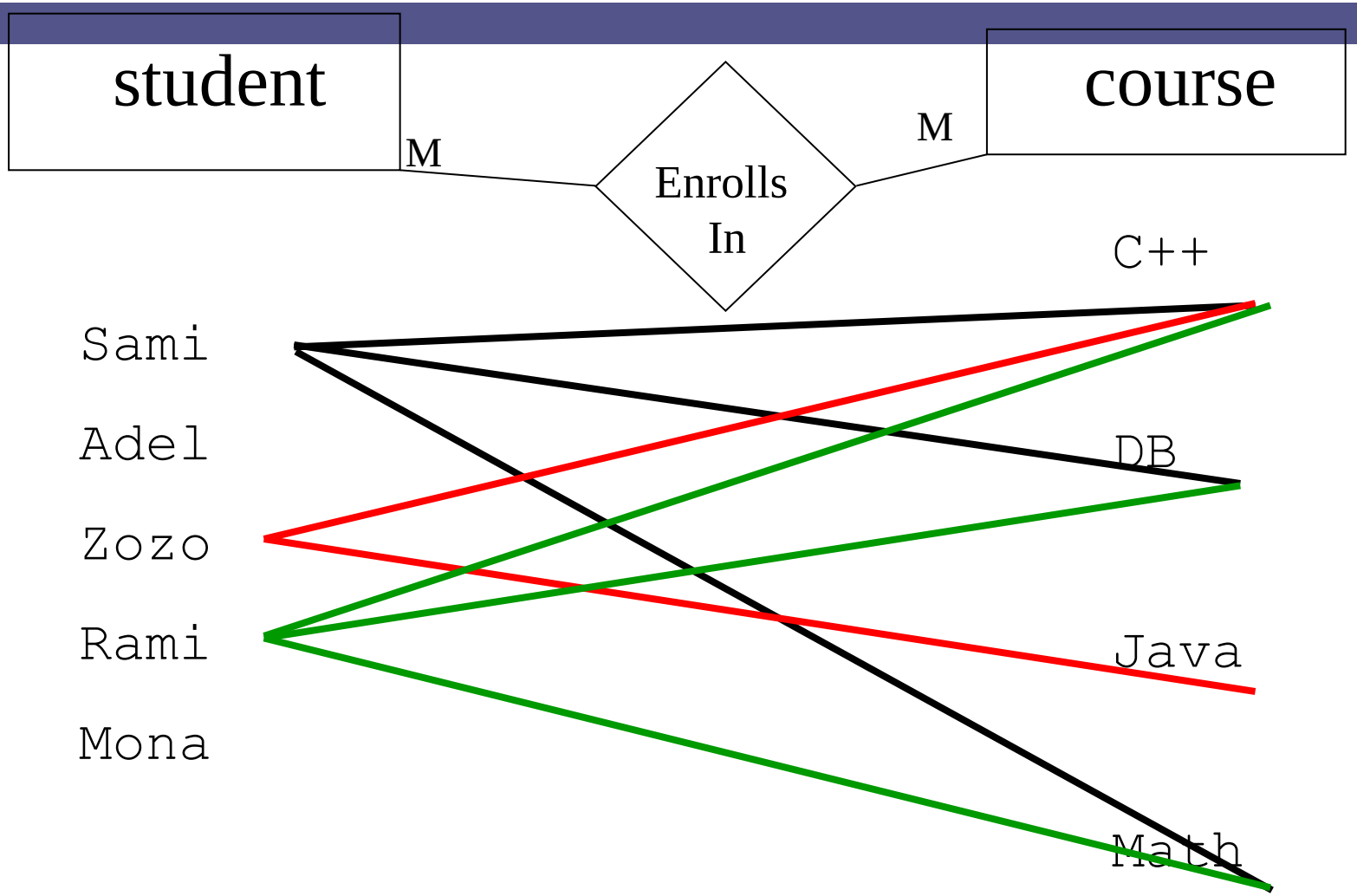
# Many-to-Many Connectivity





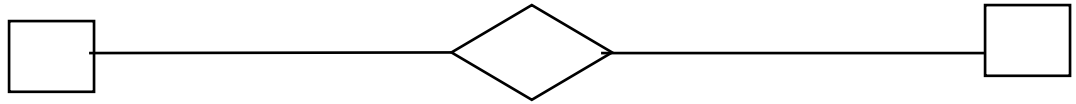
# Many-to-Many Connectivity





# Structural Constraints

- Participation
  - Do all entity instances participate in at least one relationship instance?
- Cardinality
  - How many relationship instances can an entity instance participate in?



(min,max) (min,max)

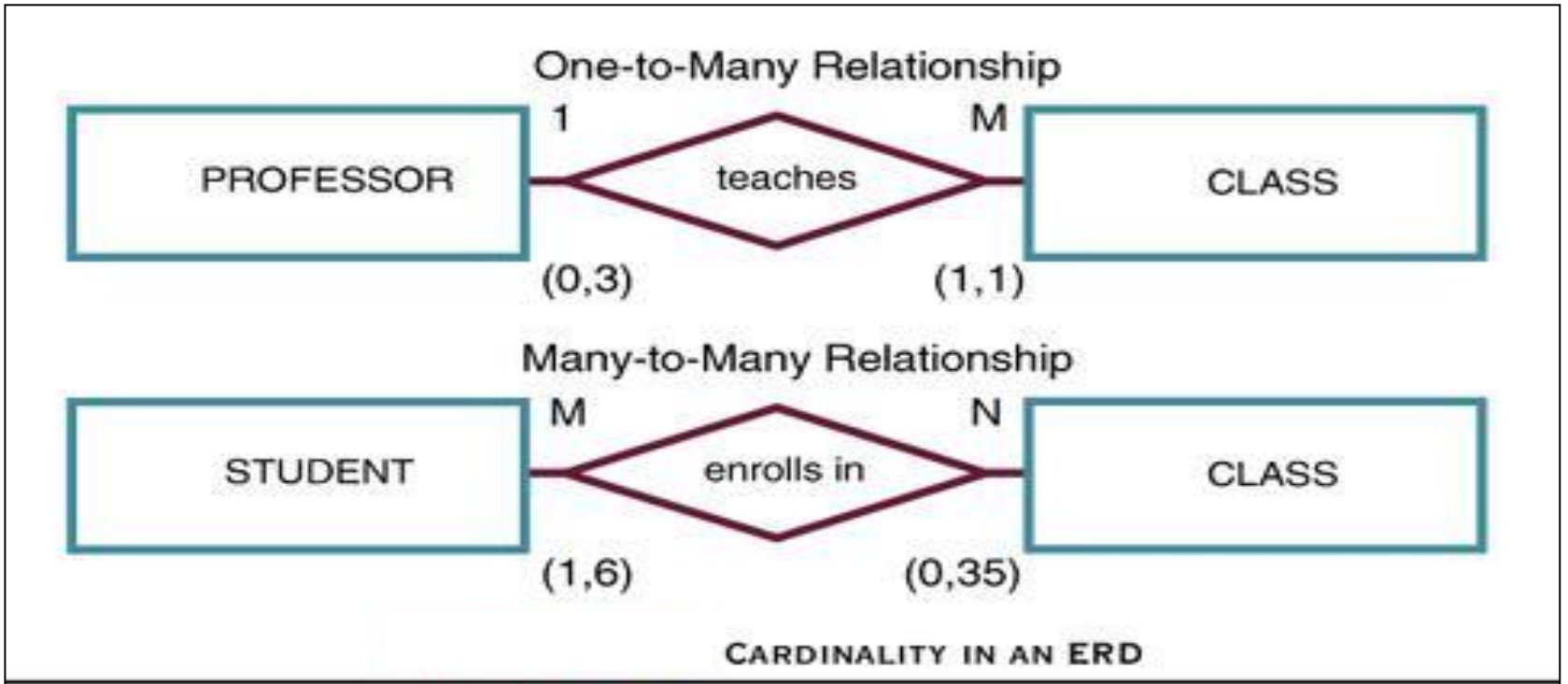
| Participation | Cardinality |
| --- | --- |
| 0 -- Partial | 1 -- one |
| 1 -- Total (Mandatory) | M -- more than one |

# Relationship



## Cardinality

- **Cardinality expresses the specific number of entity occurrences associated with one occurrence of the related entity.**



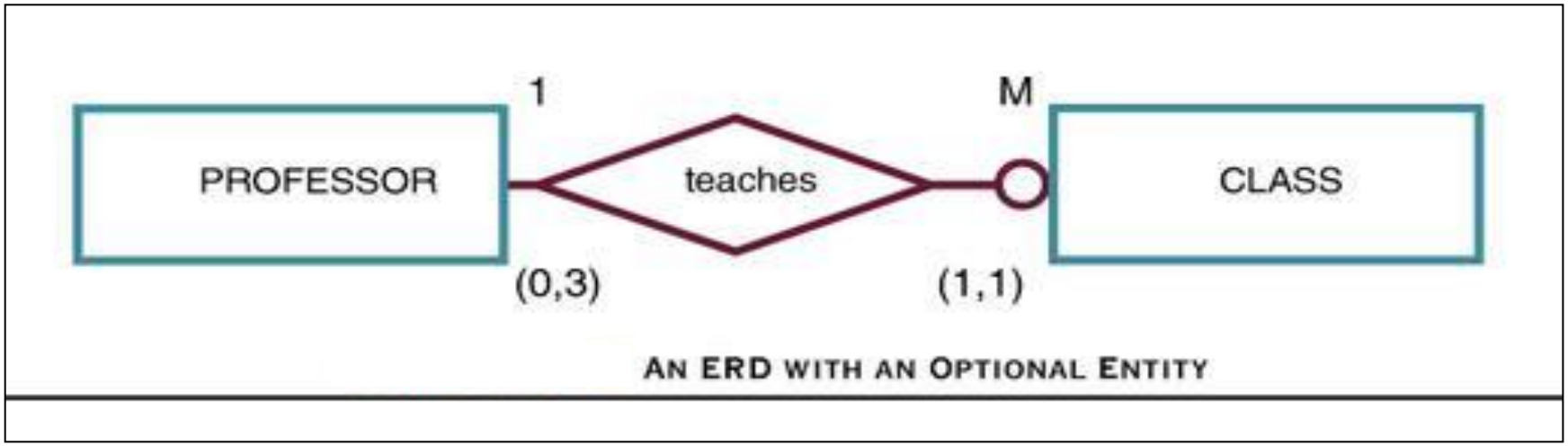

CARDINALITY IN AN ERD

# Relationship

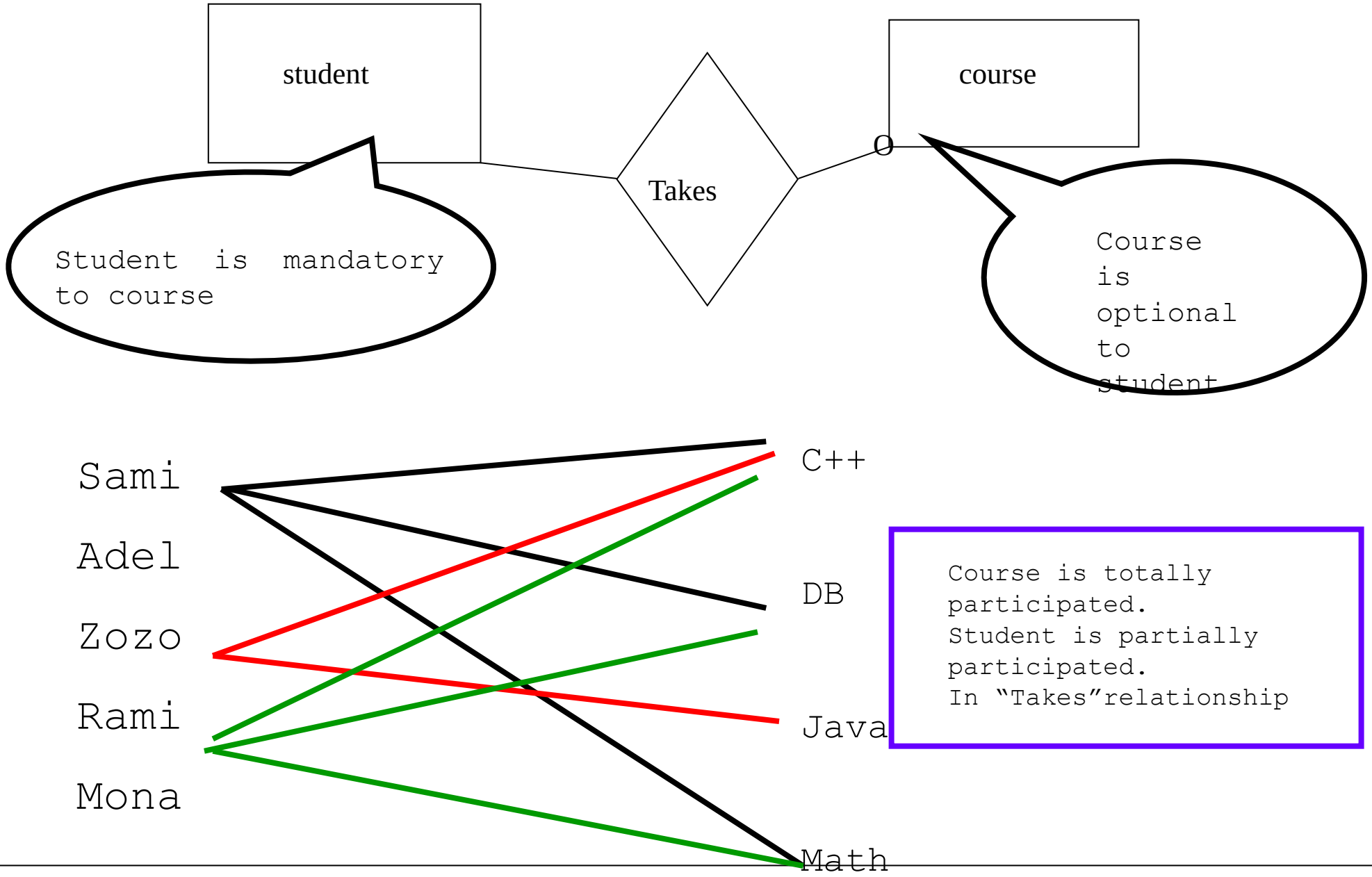


## Relationship Participation

- The participation is optional if one entity occurrence does not require a corresponding entity occurrence in a particular relationship.
- An optional entity is shown by a small circle on the side of the optional entity.




AN ERD WITH AN OPTIONAL ENTITY


student
Takes
course
O
Student is mandatory to course
Course is optional to student
Sami
Adel
Zozo
Rami
Mona
C++
DB
Java
Math
Course is totally participated.
Student is partially participated.
In "Takes"relationship