

# Database Management System

(15ECSC208)

UNIT I: Chapter 1: **Introduction and ER Model**

# Lesson Schedule

---

Class No. - Portion covered per hour / per class
1. Introduction to DBMS with an example
2. Data models, Schemas and Instance, Three-schema architecture
3. Database languages, Using High-Level Conceptual Data Models for Database Design
4. Entity Types, Entity Sets, Attributes and Keys, Relationship types, Relationship Sets
5. Constraints, Keys, Constraint Types, Weak Entity Sets
6. Design Principles, ER Diagram and ER Models - Examples

# Topic Learning Outcomes

---

Topic Learning Outcomes	COs	BL	CA Code
Compare advantages of DBMS approach over traditional approach.	CO1	L2	1.4
Describe layered architecture for database design.	CO1	L2	1.4
Analyze the cardinality ratios, relationship types, instances for given problem specification.	CO1	L3	2.2
Explain the main phases of database design.	CO1	L2	1.4
Design an E-R diagram for given mini-world description.	CO1	L3	3.4

# Introduction to Databases

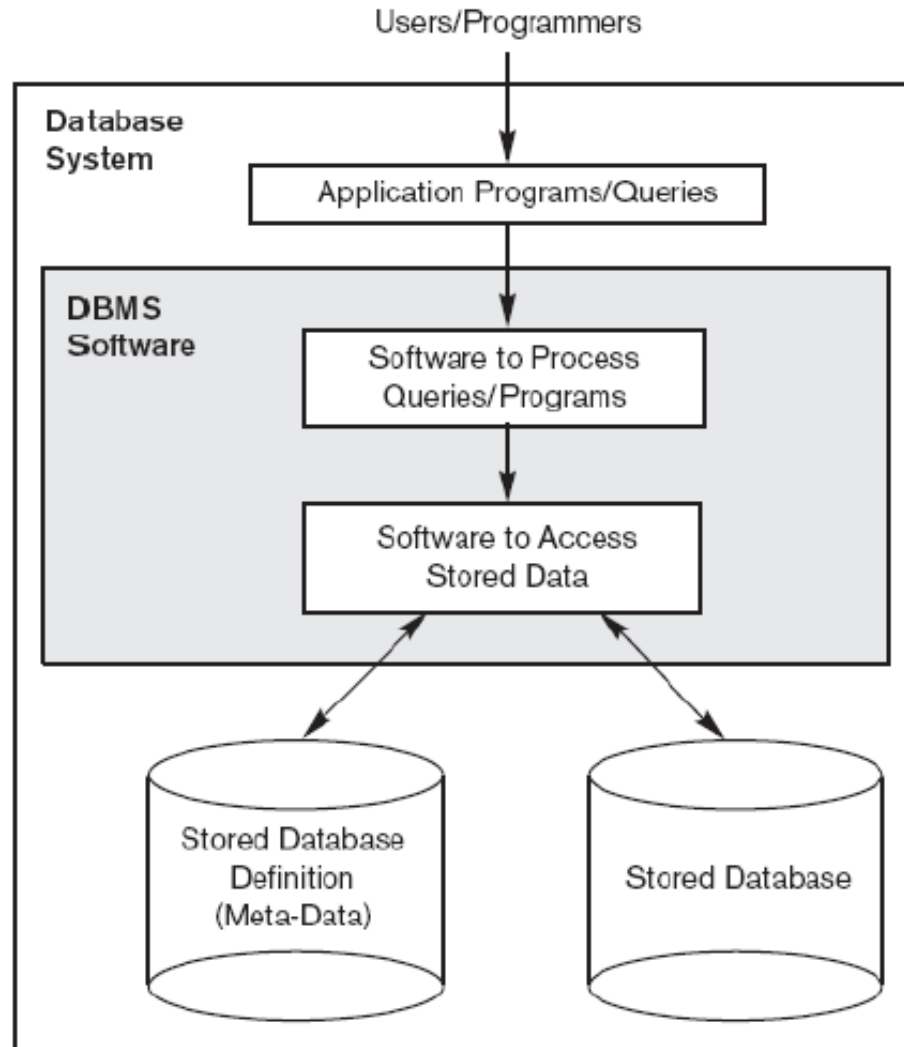
# Introduction

---

- ▶ **Database:** collection of related data.
- ▶ **Data:** know facts that can be recorded and that have implicit meaning.
- ▶ **DBMS:** collection of program that enables user to create and maintain a database.
- ▶ DBMS = data + set of programs to access/manipulate data

# Figure: A simplified DB system environment

---



# Formal Definition

---

- ▶ **DBMS:** is a general purpose software system that facilitates the processes of ***defining, constructing, manipulating, and sharing*** databases among various users and applications.
- ▶ **Defining** – a database involves specifying the datatypes, structures, and constraints of the data to be stored in the database.
- ▶ **Constructing** – the database is the process of storing the data itself on some storage medium that is controlled by the DBMS.
- ▶ **Manipulating** – a database includes such functions as querying the database to retrieve specific data, updating the database to reflect changes in the miniworld, and generating reports from the data.
- ▶ **Sharing** – a database allows multiple users and programs to access the database simultaneously.

# Example: University database

## STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

## COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

## SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

## GRADE\_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

## PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310



# Example: University database

---

- ▶ Queries:

- ▶ List of all courses and grades of 'Smith'.
- ▶ List the names of students who took the section of the 'Database' course offered in fall 2008 and their grades in that section.
- ▶ List the prerequisites of the 'Database' course.

- ▶ Updates:

- ▶ Change the class of 'Smith' to sophomore.
- ▶ Create a new section for the 'Database' course for this semester.

# Characteristics of DB Approach

---

- ▶ **File processing approach:**

- ▶ Each user defines and implements the files needed for a specific software application

- ▶ **Database approach:**

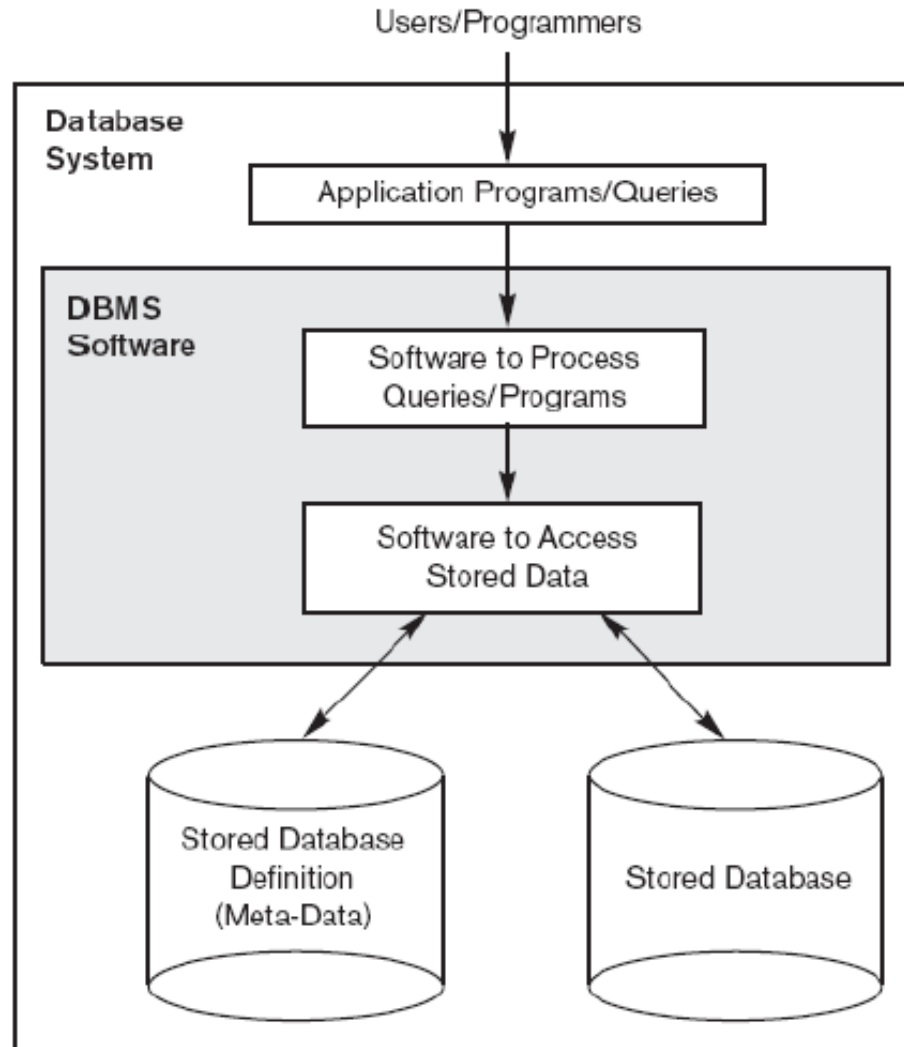
- ▶ Single repository maintains data that is defined once and then accessed by various users

# 1. Self-Describing Nature of a Database System

---

- ▶ Database system contains complete definition of structure and constraints.
- ▶ Information stored in the system **catalog** is **meta-data**
  - ▶ Describes structure of the database
- ▶ Database **catalog** used by:
  - ▶ DBMS software
  - ▶ Database users who need information about database structure.

# Figure: A simplified db system environment



# Figure: Database catalog

---

## RELATIONS

Relation_name	No_of_columns
STUDENT	4
COURSE	4
SECTION	5
GRADE_REPORT	3
PREREQUISITE	2

## COLUMNS

Column_name	Data_type	Belongs_to_relation
Name	Character (30)	STUDENT
Student_number	Character (4)	STUDENT
Class	Integer (1)	STUDENT
Major	Major_type	STUDENT
Course_name	Character (10)	COURSE
Course_number	XXXXNNNN	COURSE
....	....	....
....	....	....
....	....	....
Prerequisite_number	XXXXNNNN	PREREQUISITE

## 2. Insulation Between Programs and Data

---

- ▶ **Program-data independence**

- ▶ Structure of data files is stored in DBMS catalog separately from access programs

- ▶ **Program-operation independence**

- ▶ Operations specified in two parts:

- ▶ Interface includes operation name and data types of its arguments
- ▶ Implementation can be changed without affecting the interface

- ▶ User application programs can operate on the data by invoking these operations.

# Data Abstraction

---

## ▶ Data abstraction

- ▶ Allows program-data independence and program-operation independence

## ▶ Conceptual representation of data

- ▶ Does not include details of how data is stored or how operations are implemented

## ▶ Data model

- ▶ Type of data abstraction used to provide conceptual representation

### 3. Support of Multiple Views of the Data

---

#### ▶ **View**

- ▶ Subset of the database
- ▶ Contains **virtual data** derived from the database files but is not explicitly stored

#### ▶ **Multiuser DBMS**

- ▶ Users have a variety of distinct applications
- ▶ Must provide facilities for defining multiple views



## 4. Sharing of Data and Multiuser Transaction Processing

---

- ▶ Allow multiple users to access the database at the same time
- ▶ **Concurrency control software**
  - ▶ Ensure that several users trying to update the same data do so in a controlled manner
    - ▶ Result of the updates is correct (**OLTP**)

# Advantages of Using a DBMS

---

## 1. Controlling Redundancy

- ▶ Redundancy problems three fold:
  - ▶ (i) entering new data multiple times [duplication of effort]
  - ▶ (ii) same data stored repeatedly [storage space is wasted]
  - ▶ (iii) update applied to some but not to others [inconsistency]
- ▶ Data normalization
- ▶ Denormalization
  - ▶ Sometimes necessary to use **controlled redundancy to improve the performance of queries**

# Advantages of Using a DBMS

---

## **2. Restricting unauthorized access**

- ▶ Type of access operation – retrieval or update controlled
- ▶ Security and authorization subsystem
- ▶ Privileged software

## **3. Providing storage structures and search techniques for efficient query processing**

- ▶ Indexes
- ▶ Buffering and caching
- ▶ Query processing and optimization

# Advantages of Using a DBMS

---

## **4. Providing backup and recovery**

- ▶ Backup and recovery subsystem of the DBMS is responsible for recovery (crash recovery)

## **5. Providing multiple user interfaces**

- ▶ Graphical user interfaces (GUIs)

## **6. Representing complex relationships among data**

- ▶ May include numerous varieties of data that are interrelated in many ways

# Advantages of Using a DBMS

---

## 7. Enforcing integrity constraints

### ▶ **Referential integrity constraint**

- ▶ Eg: Every section record must be related to a course record

### ▶ **Key or uniqueness constraint**

- ▶ Eg: Every course record must have a unique value for Course\_number

# Example: University database

## STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

## COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

## SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

## GRADE\_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

## PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

# Advantages of Using a DBMS

---

## **Additional implications of using the database approach**

- ▶ Reduced application development time
- ▶ Flexibility
- ▶ Availability of up-to-date information
- ▶ Economies of scale
- ▶ Potential for Enforcing Standards

# Disadvantages of Database Systems

---

- ▶ Database systems are complex, difficult, and time-consuming to design.
- ▶ Substantial hardware and software start-up costs.
- ▶ Damage to database affects virtually all applications programs.
- ▶ Extensive conversion costs in moving from a file-based system to a database system.
- ▶ Initial training required for all programmers and users.



---

# When not to use a DBMS ?

# DBMS and Relational DBMS

DBMS	RDBMS
<p>Introduced in 1960s.</p> <p>It followed the navigational method for data storage and fetching.</p> <p>Data fetching is slower for complex and large amount of data.</p> <p>Used for applications using small amount of data.</p> <p>Data Redundancy is common in this model</p> <p>Example systems are dBase, Microsoft Acces, LibreOffice Base, FoxPro.</p>	<p>Introduced in 1970s.</p> <p>This model uses relationship between tables using primary keys, foreign keys and indexes.</p> <p>Comparatively faster because of its relational model.</p> <p>Used for complex and large amount of data.</p> <p>Keys and indexes are used in the tables to avoid redundancy.</p> <p>Example systems are SQL Server, Oracle , MySQL, MariaDB, SQLite.</p>

# Database System Concepts and Architecture

# Data Models

---

- ▶ Collection of concepts that describe the structure of a database
- ▶ Provides means to achieve data abstraction
- ▶ **Basic operations**
  - ▶ Specify retrievals and updates on the database
- ▶ **Dynamic aspect or behavior** of a database application

# Categories of Data Models

---

- ▶ **High-level or conceptual data models**
  - ▶ Close to the way many users perceive data
- ▶ **Low-level or physical data models**
  - ▶ Describe the details of how data is stored on computer storage media
- ▶ **Representational (implementation) data models**
  - ▶ Easily understood by end users
  - ▶ Also similar to how data organized in computer storage

# Conceptual Data Model

---

- ▶ **Entity**

- ▶ Represents a real-world object or concept

- ▶ **Attribute**

- ▶ Represents some property of interest
- ▶ Further describes an entity

- ▶ **Relationship among two or more entities**

- ▶ Represents an association among the entities
- ▶ **Entity-Relationship model**

# Schemas, Instances, and Database State

---

**Distinguish between desc. of db and db itself**

- ▶ **Database schema**
  - ▶ Description of a database
- ▶ **Schema diagram**
  - ▶ Displays selected aspects of schema
- ▶ **Schema construct**
  - ▶ Each object in the schema
- ▶ **Database state or snapshot**
  - ▶ Data in database at a particular moment in time

# Example: Database schema diagram for University database

---

## STUDENT

Name	Student_number	Class	Major
------	----------------	-------	-------

## COURSE

Course_name	Course_number	Credit_hours	Department
-------------	---------------	--------------	------------

## PREREQUISITE

Course_number	Prerequisite_number
---------------	---------------------

## SECTION

Section_identifier	Course_number	Semester	Year	Instructor
--------------------	---------------	----------	------	------------

## GRADE\_REPORT

Student_number	Section_identifier	Grade
----------------	--------------------	-------



# Example: Database state for University database

## STUDENT

Name	Student_number	Class	Major
Smith	17	1	CS
Brown	8	2	CS

## COURSE

Course_name	Course_number	Credit_hours	Department
Intro to Computer Science	CS1310	4	CS
Data Structures	CS3320	4	CS
Discrete Mathematics	MATH2410	3	MATH
Database	CS3380	3	CS

## SECTION

Section_identifier	Course_number	Semester	Year	Instructor
85	MATH2410	Fall	07	King
92	CS1310	Fall	07	Anderson
102	CS3320	Spring	08	Knuth
112	MATH2410	Fall	08	Chang
119	CS1310	Fall	08	Anderson
135	CS3380	Fall	08	Stone

## GRADE\_REPORT

Student_number	Section_identifier	Grade
17	112	B
17	119	C
8	85	A
8	92	A
8	102	B
8	135	A

## PREREQUISITE

Course_number	Prerequisite_number
CS3380	CS3320
CS3380	MATH2410
CS3320	CS1310

# Database Schema Vs Database State

---

- ▶ **Define a new database**
  - ▶ Specify database schema to the DBMS
- ▶ **Database state – *empty state***, with no data
- ▶ **Initial state**
  - ▶ **Populated or loaded with the initial data**
- ▶ **Database state – *current state***, with every update operation on the database
- ▶ **Valid state**
  - ▶ Satisfies the structure and constraints specified in the schema
- ▶ Database schema – **intension**, and Database state – **extension**
- ▶ **Schema evolution**
  - ▶ Changes applied to schema as application requirements change

# Three-Schema Architecture

---

- ▶ **Internal level**

- ▶ Describes physical storage structure of the database
- ▶ Describes complete details of data storage and access paths for the database

- ▶ **Conceptual level**

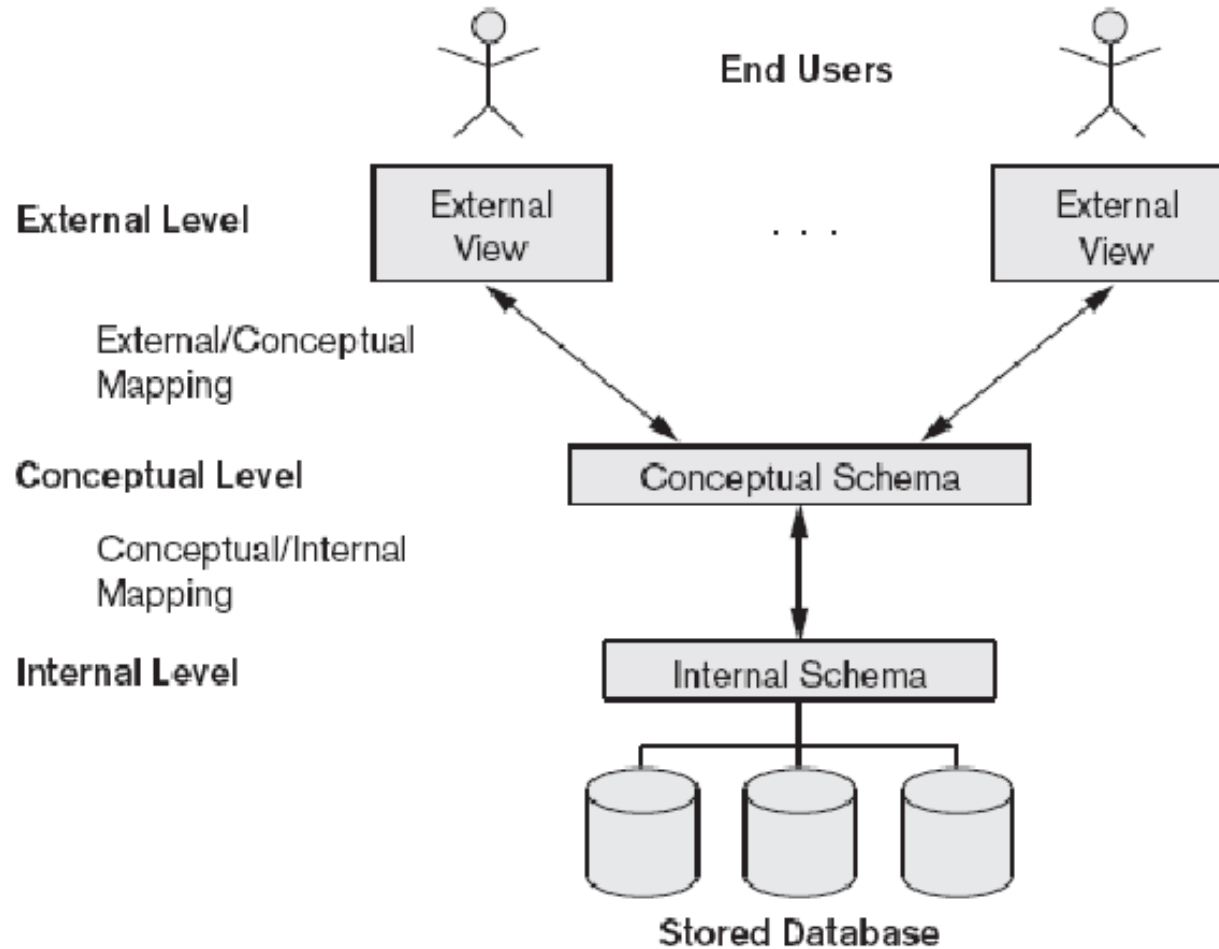
- ▶ Describes structure of the whole database for a community of users
- ▶ Hides details of physical storage structure

- ▶ **External or view level**

- ▶ Describes part of the database that a particular user group is interested in

# Three-Schema Architecture

---



# Data Independence

---

- ▶ Capacity to change the schema at one level of a database system without having to change the schema at the next higher level.

## 1. Logical data independence

- ▶ The capacity to change the conceptual schema without affecting the external schemas or application programs
- ▶ Conceptual schema could be changed:
  - ▶ to expand the database – by adding a new attribute to some relation
  - ▶ to reduce the database – by deleting an attribute

# Data Independence

---

## 2. Physical data independence

- ▶ The capacity to modify physical schema without affecting the logical or external schema.
- ▶ Performance tuning (retrieval or update) – modification at physical level or creating a new index etc.

# DBMS Languages

---

- ▶ **Data definition language (DDL)**
  - ▶ Defines schemas
- ▶ **Storage definition language (SDL)**
  - ▶ Specifies the internal schema
- ▶ **View definition language (VDL)**
  - ▶ Specifies user views/mappings to conceptual schema
- ▶ **Data manipulation language (DML)**
  - ▶ Allows retrieval, insertion, deletion, modification

# DBMS Languages

---

- ▶ **High-level or nonprocedural DML**
  - ▶ Can be used on its own to specify complex database operations concisely
  - ▶ **Set-at-a-time or set-oriented**
- ▶ **Low-level or procedural DML**
  - ▶ Must be embedded in a general-purpose programming language
  - ▶ **Record-at-a-time**



# DBMS Interfaces

---

- ▶ Menu-based interfaces for Web clients or browsing
- ▶ Forms-based interfaces
- ▶ Graphical user interfaces
- ▶ Natural language interfaces
- ▶ Interfaces for parametric users
- ▶ Interfaces for the DBA

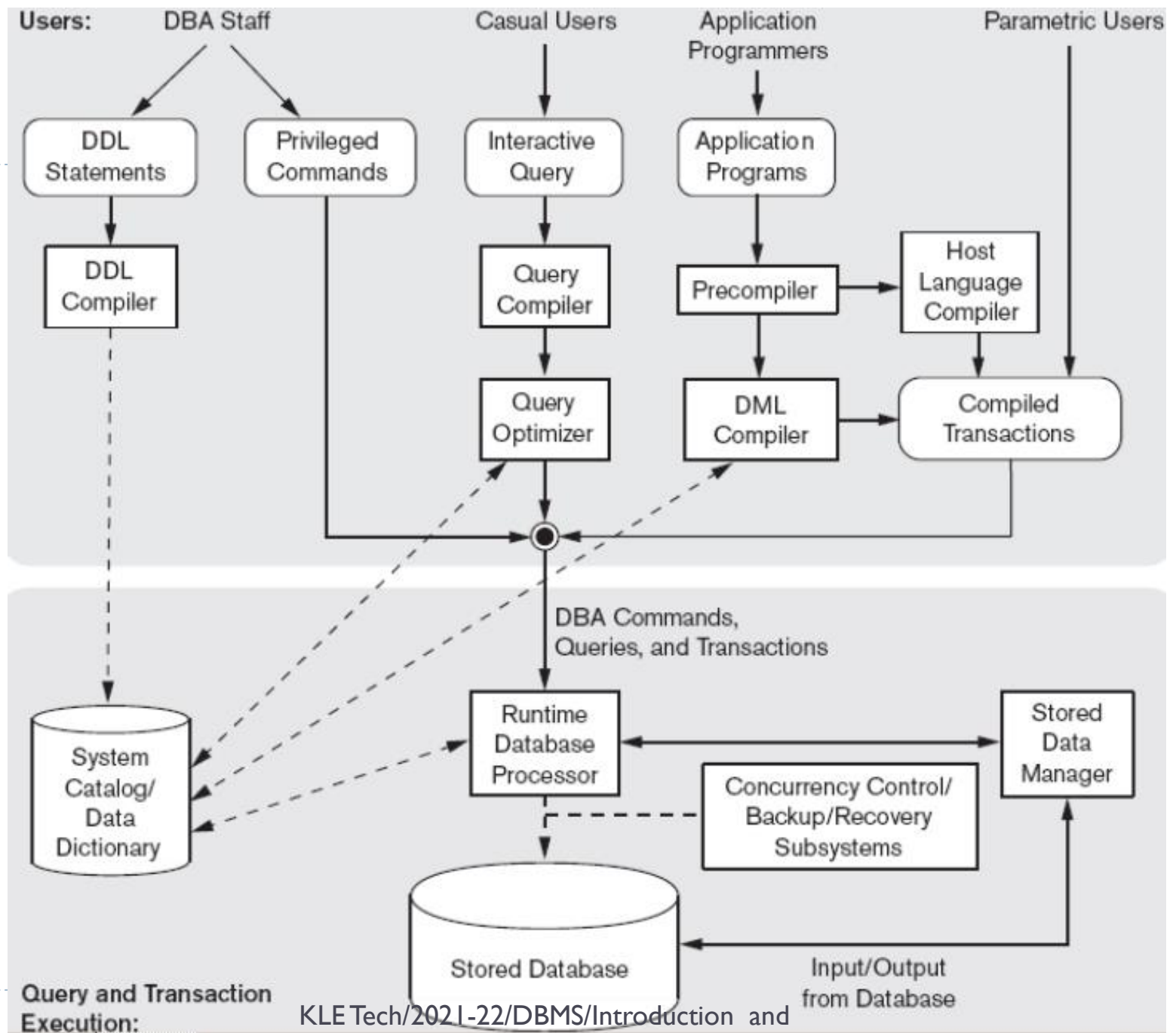
# Database System Environment

---

## ▶ **DBMS component modules**

- ▶ Stored data manager
- ▶ DDL compiler
- ▶ Runtime database processor
- ▶ Interactive query interface
  - ▶ Query compiler
  - ▶ Query optimizer
- ▶ Precompiler
- ▶ System catalog
- ▶ Concurrency control system
- ▶ Backup and recovery system

# Component modules of a DBMS and their interactions



# Database System Utilities

---

- ▶ **Loading**

- ▶ Load existing data files into the database

- ▶ **Backup**

- ▶ Creates a backup copy of the database

- ▶ **Database storage reorganization**

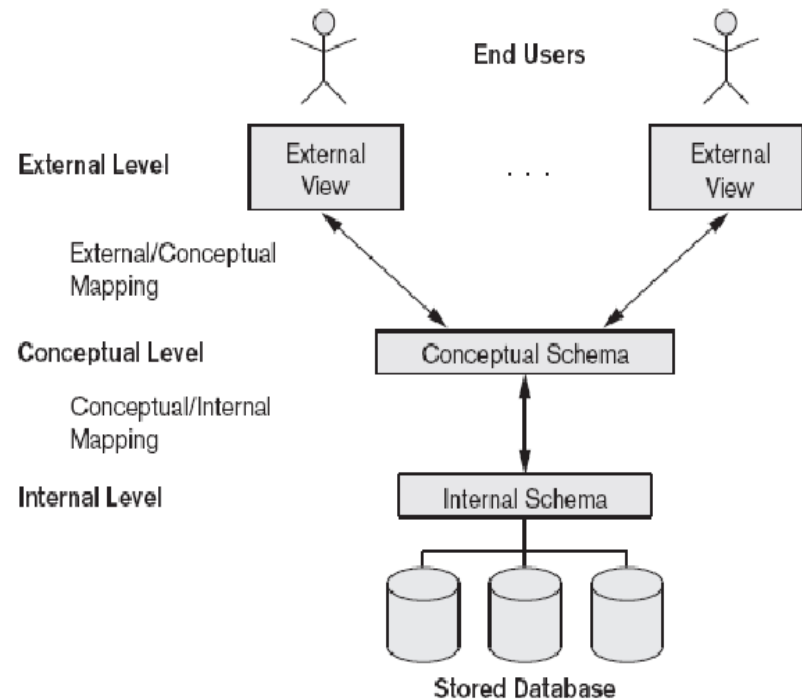
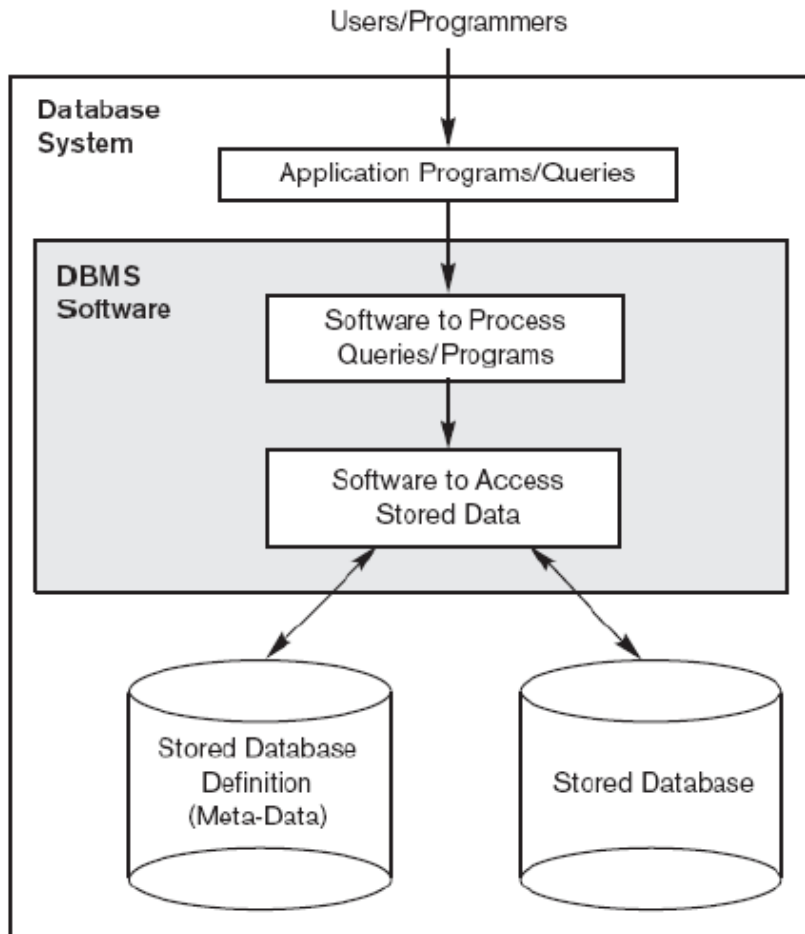
- ▶ Reorganize a set of database files into different file organizations

- ▶ **Performance monitoring**

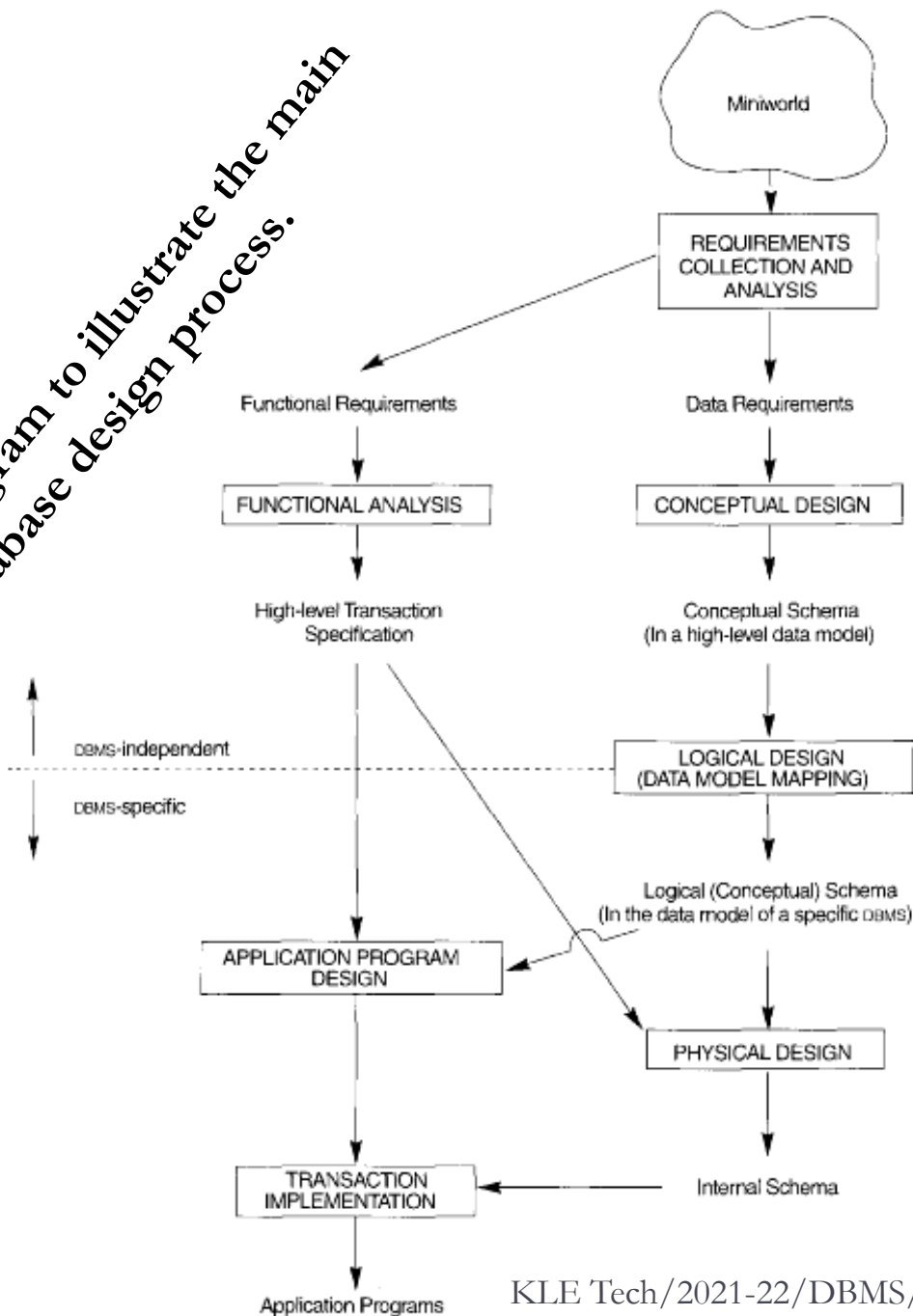
- ▶ Monitors database usage and provides statistics to the DBA

# Data Modeling Using the Entity – Relationship Model

# Review



*A simplified diagram to illustrate the main phases of database design process.*



# Conceptual Schema

---

- ▶ Outcome of the high-level conceptual design
- ▶ Collective description of data requirements of the users
- ▶ Includes description of entity types, relationships and constraints
- ▶ No implementation details
- ▶ Ease of understanding. Used to communicate with non-technical users.



# Entities

---

- ▶ **Entity** – represents an object of the real world that has an independent (physical or conceptual) existence.
- ▶ In the University database context, an individual student, faculty member, a class room, a course are entities.
- ▶ **Entity Set or Entity Type** – Collection of entities all having the same properties.
- ▶ Student entity set – collection of all student entities. Course entity set – collection of all course entities.

# Attributes

---

- ▶ Each entity is described by a set of attributes/properties.
- ▶ student entity
  - ▶ StudName—name of the student.
  - ▶ RollNumber—the roll number of the student.
  - ▶ Sex—the gender of the student etc.
- ▶ All entities in an Entity set/type have the same set of attributes.

# Types of Attributes

---

- ▶ **Simple Attributes** – having atomic or indivisible values.
  - ▶ Eg: Dept – a string; PhoneNumber – an eight digit number
- ▶ **Composite Attributes** – having several components in the value.
  - ▶ Eg: Qualification with components (DegreeName, Year, UniversityName)
- ▶ **Derived Attributes** – Attribute value is dependent on some other attribute.
  - ▶ Eg: Age depends on DateOfBirth. So age is a derived attribute. Birthdate is a **stored attribute**.

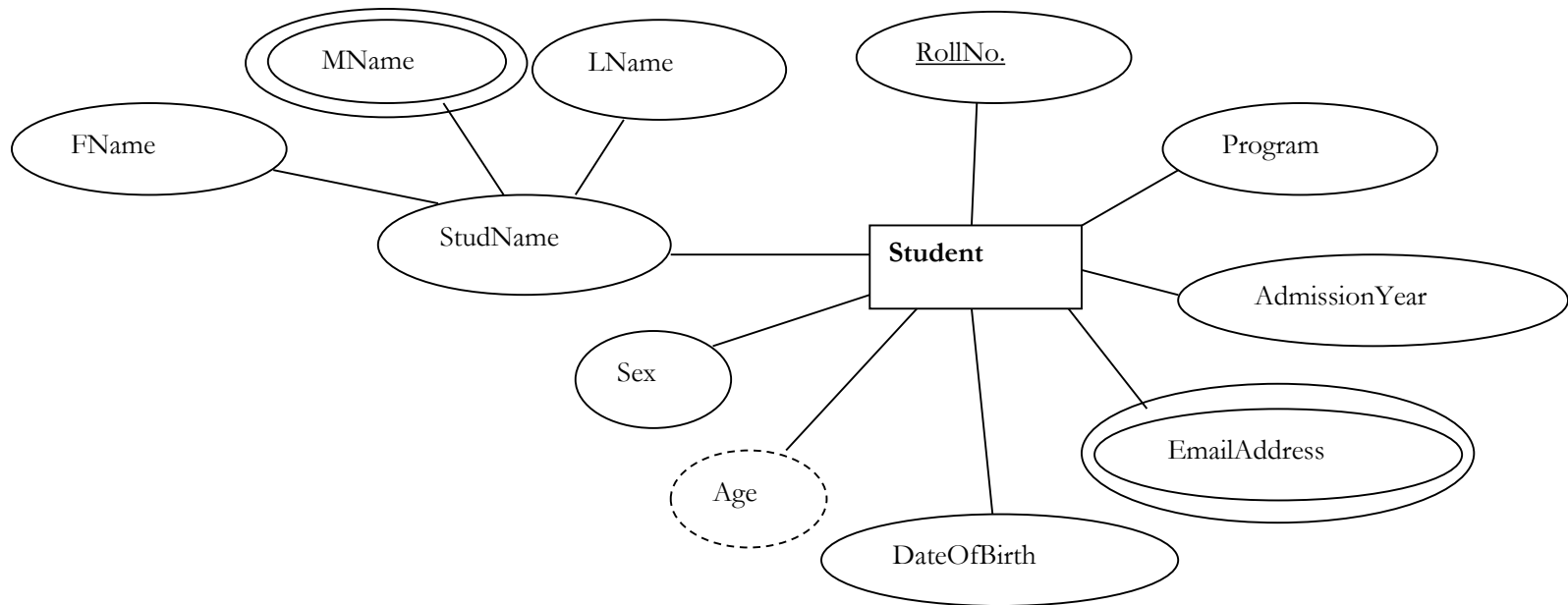
# Types of Attributes

---

- ▶ **Single-valued** – having only one value rather than a set of values.
  - ▶ For instance, `PlaceOfBirth` – single string value.
- ▶ **Multi-valued** – having a set of values rather than a single value.
  - ▶ For instance, `CoursesEnrolled` attribute for student;  
`EmailAddress` attribute for student;  
`PreviousDegree` attribute for student.
- ▶ Attributes can be:
  - ▶ simple single-valued, simple multi-valued,
  - ▶ composite single-valued or composite multi-valued.

# Diagrammatic Notation for Entities

---



**entity** – rectangle

**attribute** – ellipse connected to rectangle

**multi-valued attribute** – double ellipse

**composite attribute** – ellipse connected to ellipse

**derived attribute** – dashed ellipse

# Types of Attributes

---

- ▶ **Complex Attributes:** nesting of composite and multivalued attributes arbitrarily.
- ▶ **Eg:** {Address\_phone({Phone(Area\_code,Phone\_number)}), Address(Street\_address(Number,Street,Apartment\_number),City,State,Zip) )}
- ▶ **Representation:** grouping components of a composite attribute between parentheses ( ) and separating the components with commas, and by displaying multivalued attributes between braces { }

# Domains (Value Sets) of Attributes

---

- ▶ Each attribute takes values from a set called its domain
  - ▶ For instance, `studentAge` –  $\{17, 18, \dots, 60\}$ ;  
`HomeAddress` – character strings of length 35;
- ▶ Domain of composite attributes – cross product of domains of component attributes
- ▶ Domain of multi-valued attributes – set of subsets of values from the basic domain

# Entity Sets and Key Attributes

---

- ▶ **Key** – an attribute or a collection of attributes whose value(s) uniquely identify an entity in the entity set.
  - ▶ For instance,
    - RollNumber - Key for Student entity set;
    - EmpID - Key for Faculty entity set;
    - HostelName, RoomNo - Key for Student entity set  
(assuming that each student gets to stay in a single room)
- ▶ A key for an entity set may have more than one attribute.
- ▶ Keys can be determined only from the meaning of the attributes in the entity type. (Determined by designers)



# Relationships

---

- ▶ When two or more entities are associated with each other, we have an instance of a **Relationship**.
- ▶ Eg: student James *enrolls* in DBMS course
- ▶ Relationship *enrolls* has Student and Course as the **participating entity sets**.
- ▶ Formally,  $enrolls \subseteq \text{Student} \times \text{Course}$
- ▶  $(s,c) \in enrolls \Leftrightarrow$  Student 's' has enrolled in Course 'c'
- ▶ Tuples in *enrolls* – **relationship instances**
- ▶ *enrolls* is called a **relationship Type/Set**.

# Degree of a relationship

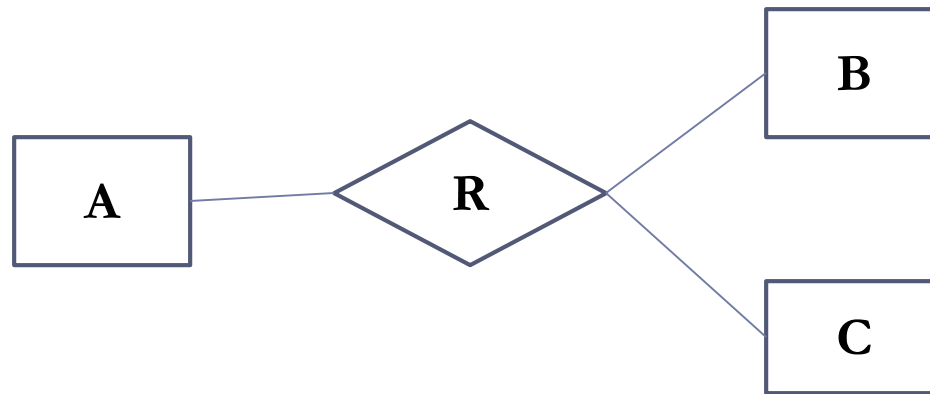
---

- ▶ **Degree:** the number of participating entities.
  - ▶ Degree 2: binary
  - ▶ Degree 3: ternary
  - ▶ Degree n: n-ary
- ▶ Binary relationships are very common and widely used.

# Diagrammatic Notation for Relationships

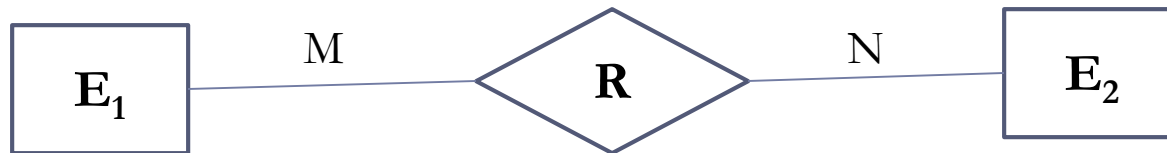
---

- ▶ **Relationship** –diamond shaped box
  - ▶ Rectangle of each participating entity is connected by a line to this diamond. Name of the relationship is written in the box.



# Binary Relationships and Cardinality Ratio

---



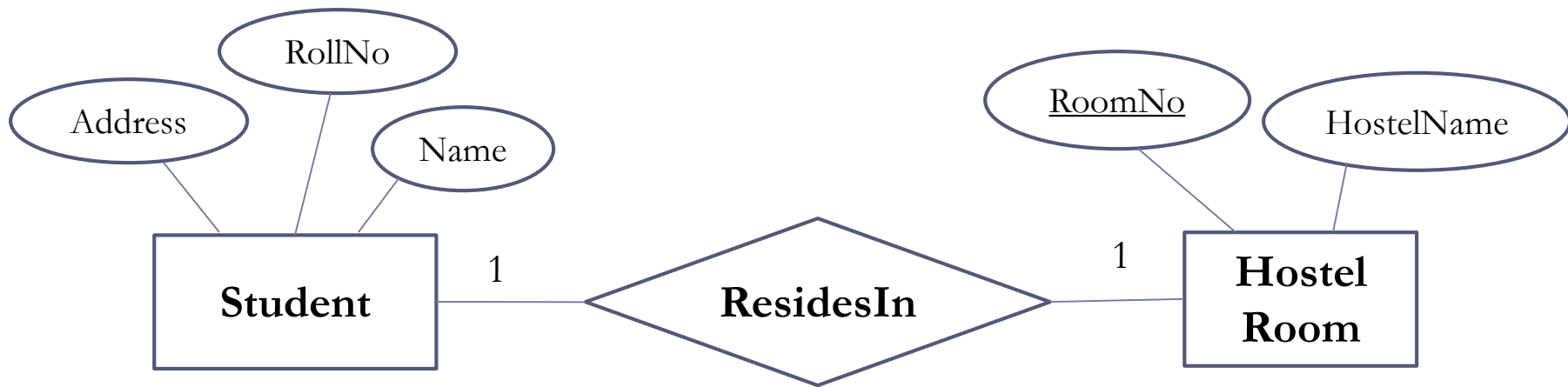
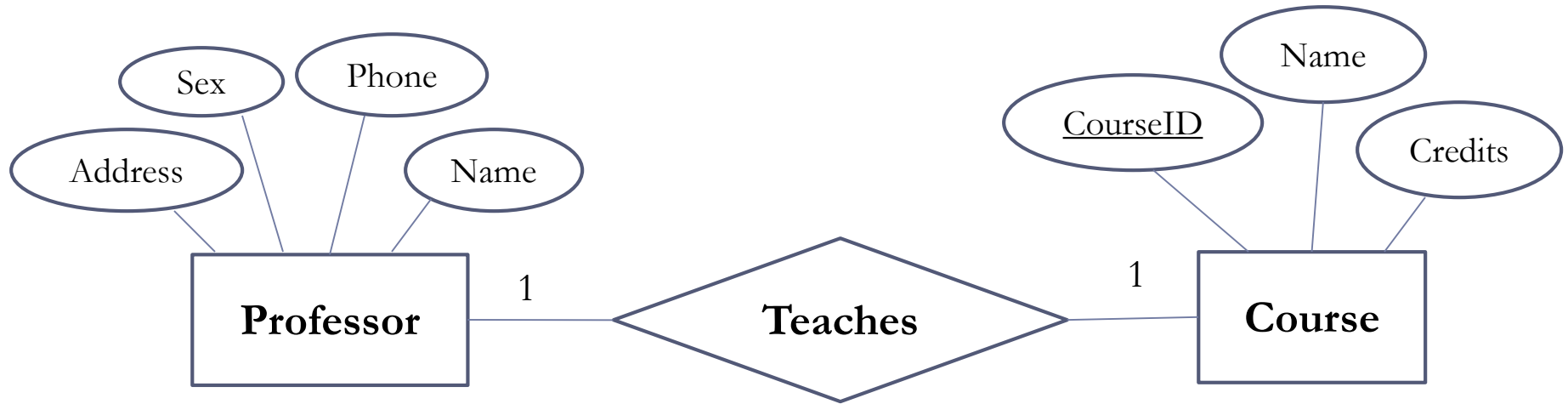
- ▶ The number of entities from  $E_2$  that an entity from  $E_1$  can possibly be associated through  $R$  (and vice-versa) determines the **cardinality ratio** of  $R$ .
- ▶ Four possibilities are usually specified:
  - ▶ one-to-one (1:1)
  - ▶ one-to-many (1:N)
  - ▶ many-to-one (N:1)
  - ▶ many-to-many (M:N)

# Cardinality Ratios

---

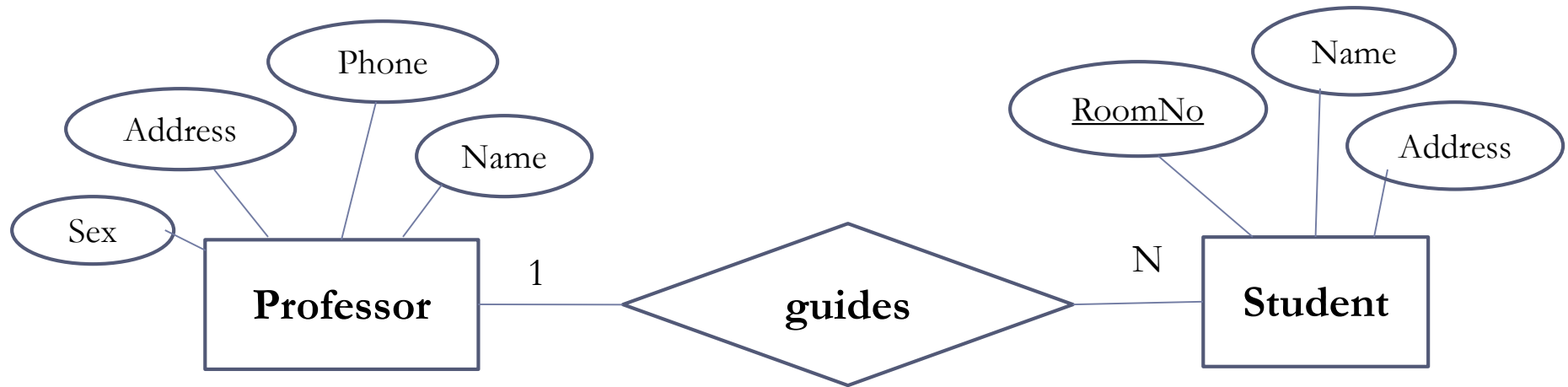
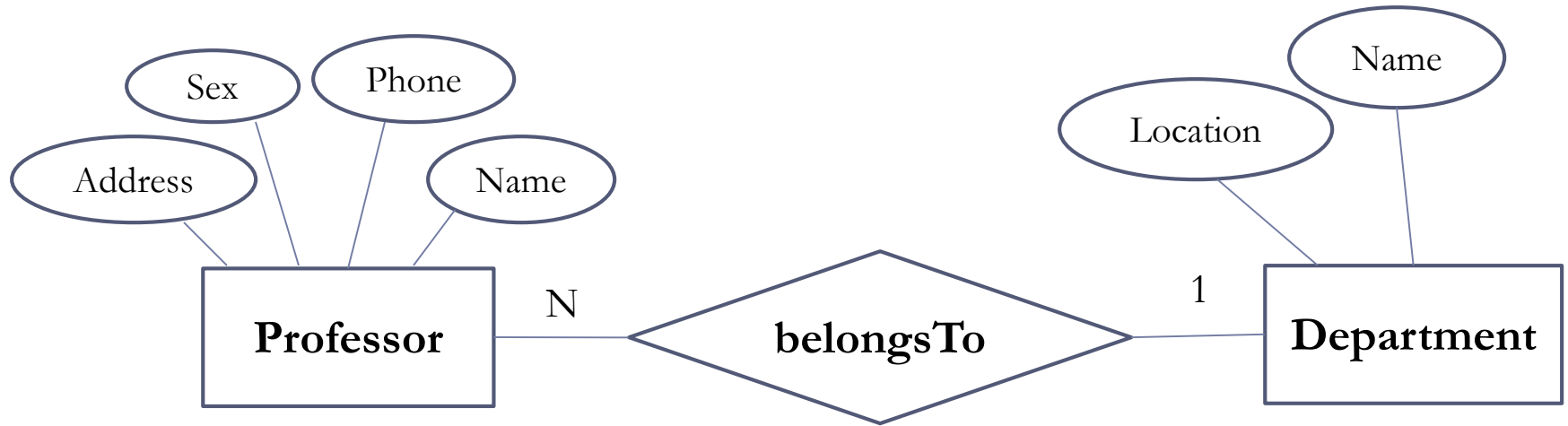
- ▶ **One-to-one:** An  $E_1$  entity may be associated with at most one  $E_2$  entity and similarly an  $E_2$  entity may be associated with at most one  $E_1$  entity.
- ▶ **One-to-many:** An  $E_1$  entity may be associated with many  $E_2$  entities whereas an  $E_2$  entity may be associated with at most one  $E_1$  entity.
- ▶ **Many-to-one:** (similar to above but vice-versa)
- ▶ **Many-to-many:** Many  $E_1$  entities may be associated with a single  $E_2$  entity and a single  $E_1$  entity may be associated with many  $E_2$  entities.

# Cardinality Ratio – example (*one-to-one*)



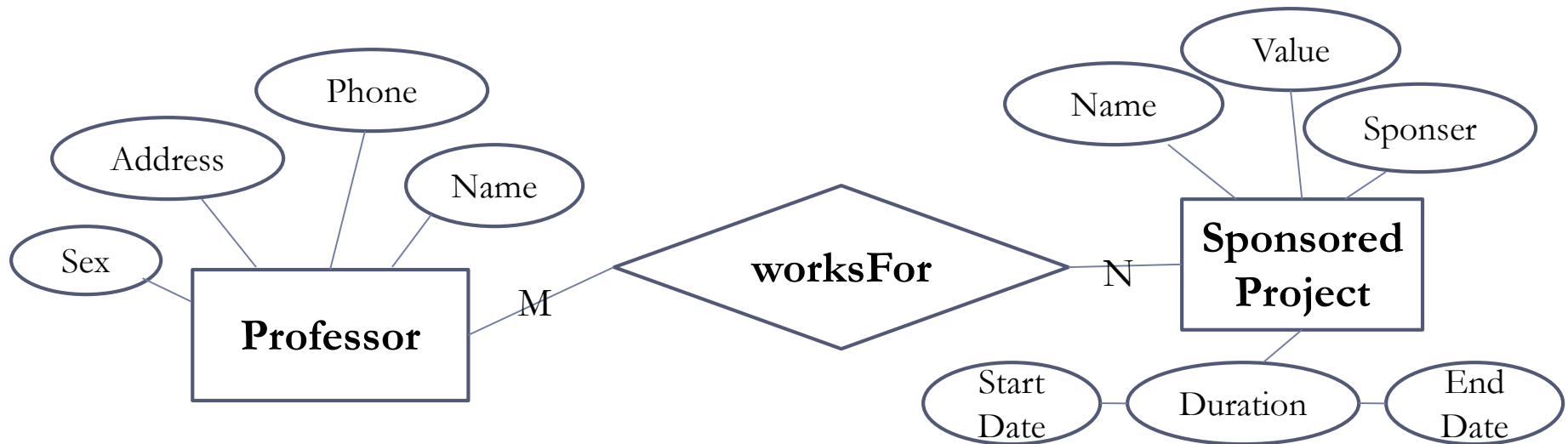
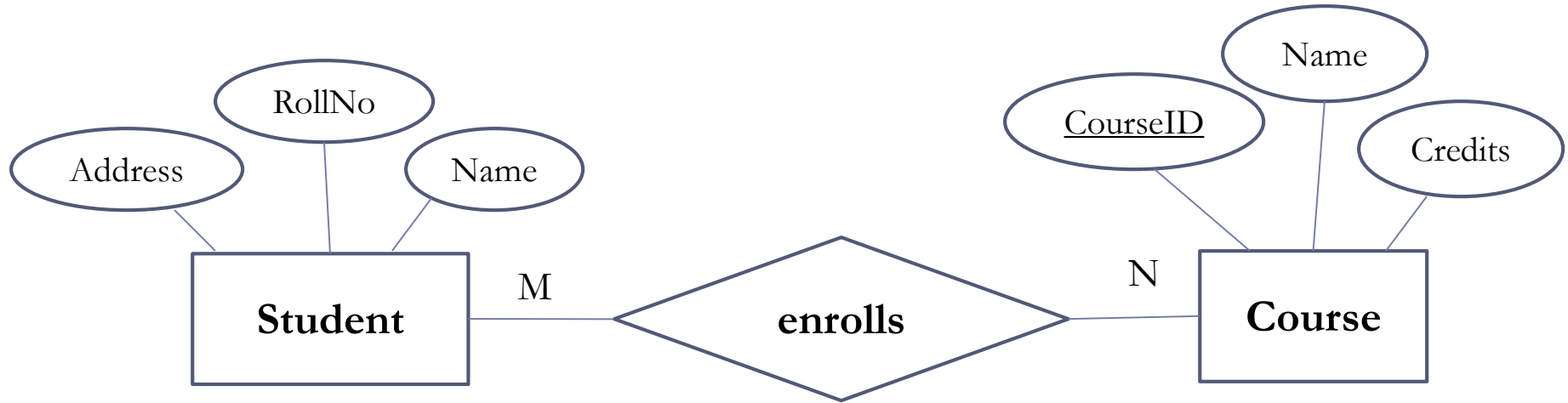
# Cardinality Ratio – example (*many-to-one* / *one-to-many*)

---



# Cardinality Ratio – example (*many-to-many*)

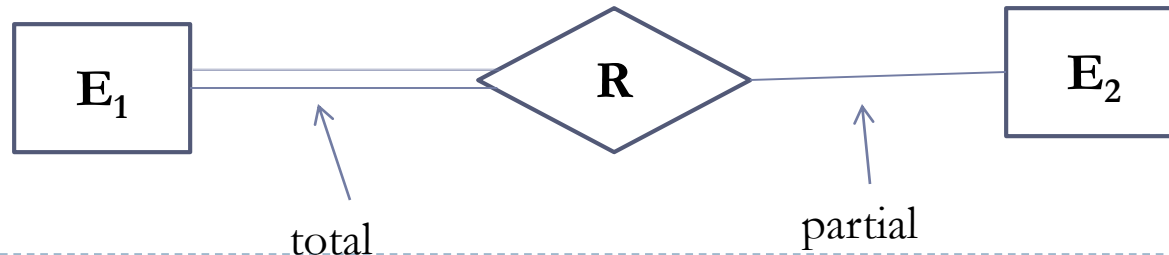
---



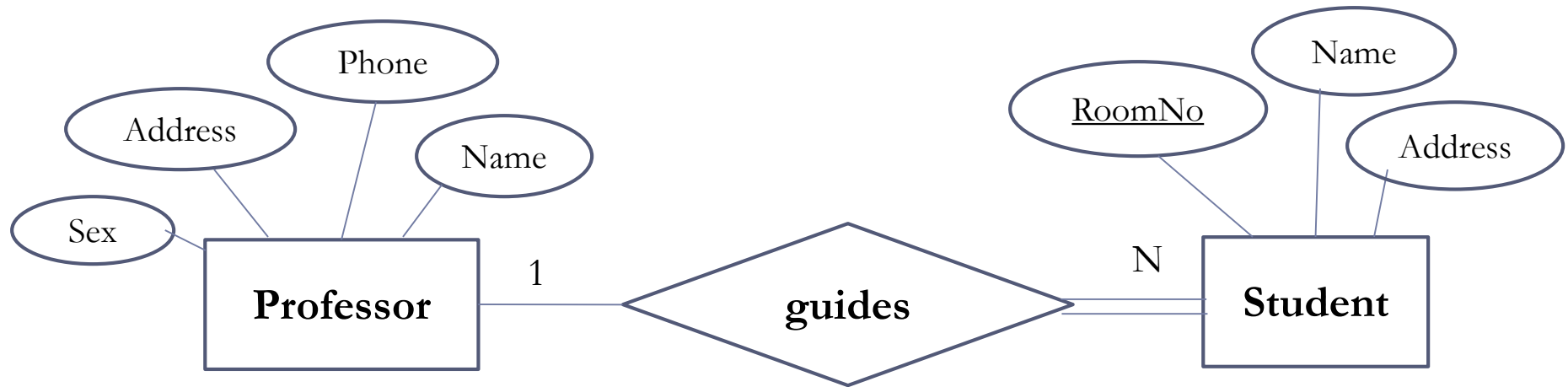
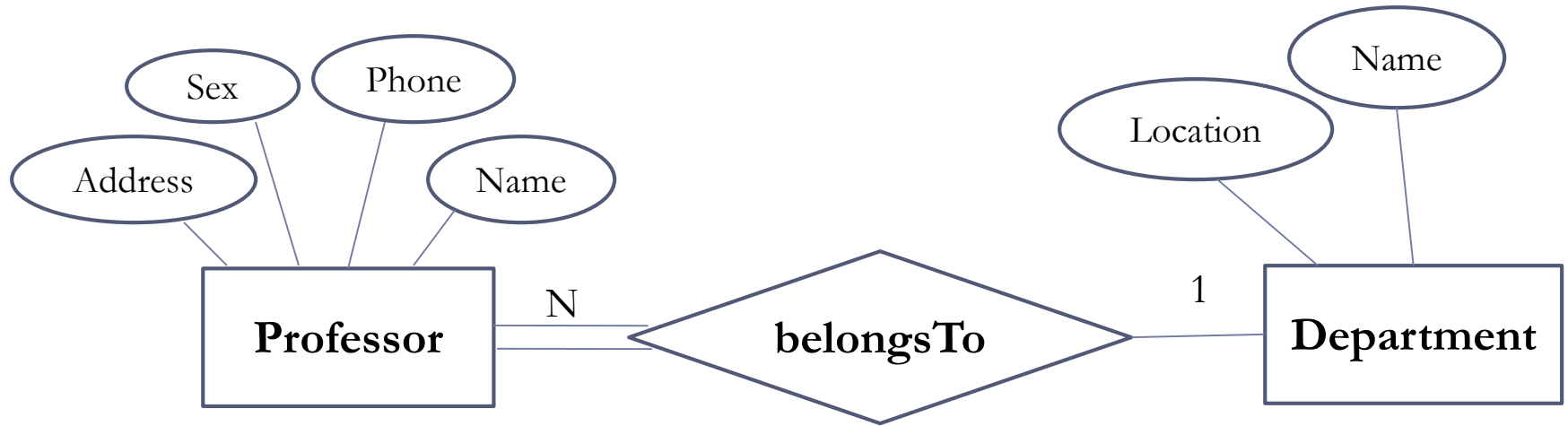


# Participation Constraints

- ▶ The **participation constraint** specifies whether the existence of an entity depends on its being related to another entity via the relationship type.
- ▶ An entity set may participate in a relation either **totally** or **partially**.
- ▶ **Total participation:** Every entity in the set is involved in some association (or tuple) of the relationship.
- ▶ **Partial participation:** Not all entities in the set are involved in association (or tuples) of the relationship.



# Example of total/partial Participation



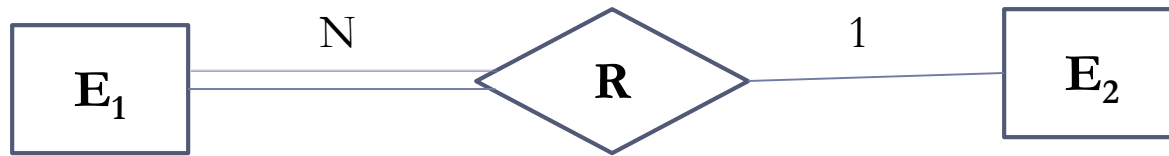
# Structural Constraints

---

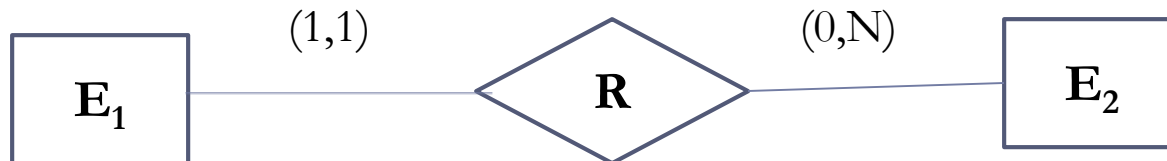
- ▶ Cardinality Ratio and Participation Constraints are together called **Structural Constraints**.
- ▶ They are called constraints as the data must satisfy them to be consistent with the requirements.
- ▶ **Min-Max notation:** pair of numbers  $(m, n)$  placed on the line connecting an entity to the relationship.
- ▶ **m:** the minimum number of times a particular entity must appear in the relationship tuples at any point of time
  - ▶ 0 – partial participation
  - ▶  $\geq 1$  – total participation
- ▶ **n:** similarly, the maximum number of times a particular entity can appear in the relationship tuples at any point of time

# Comparing the Notations

---



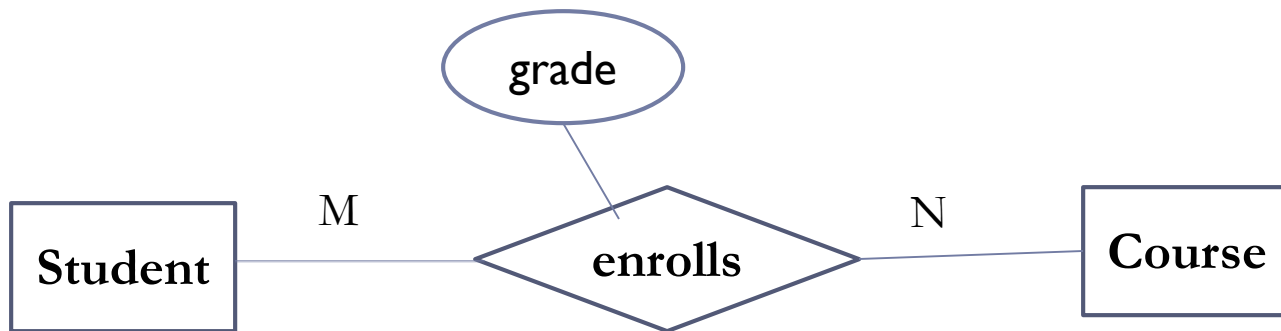
is equivalent to



# Attributes for Relationship Types

---

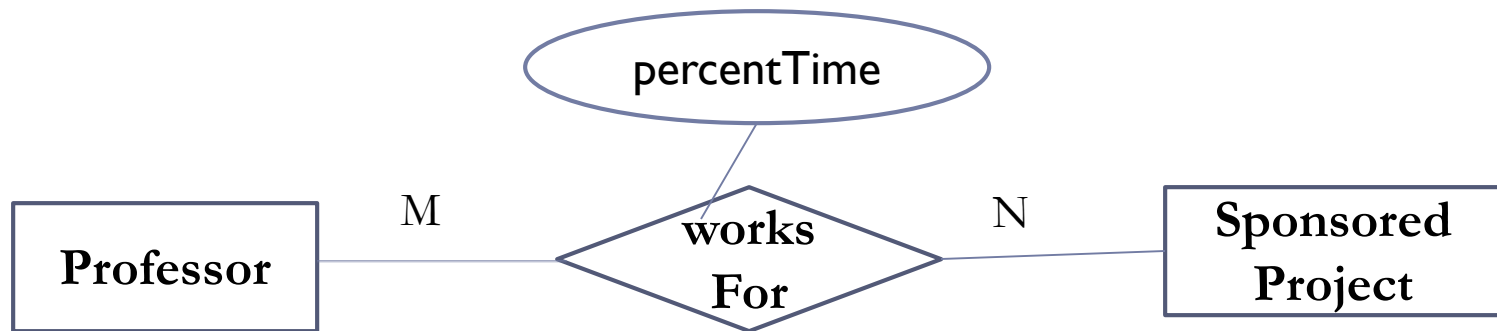
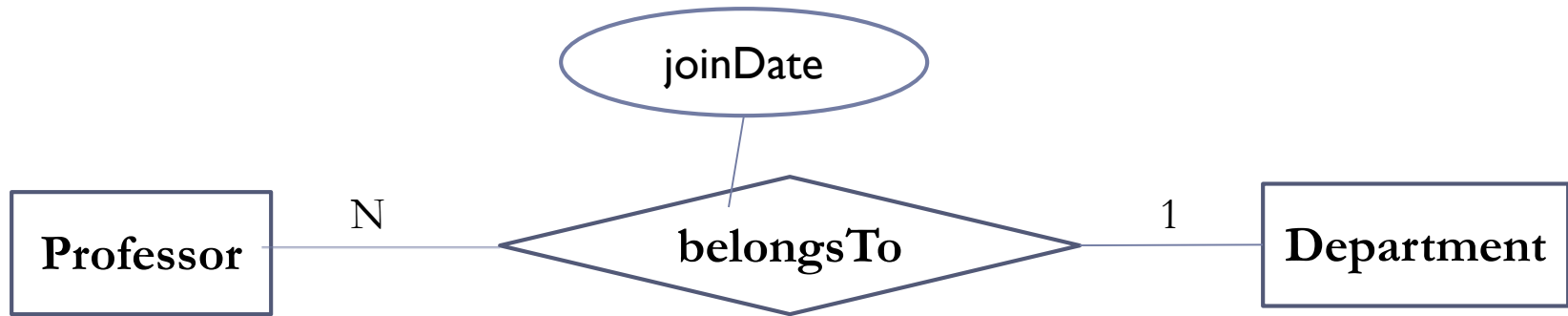
- ▶ Relationship types can also have attributes.
  - ▶ properties of the association of entities.



- ▶ grade gives the letter grade (S, A, B, etc.) earned by the student for a course.
  - ▶ neither an attribute of student nor that of course.

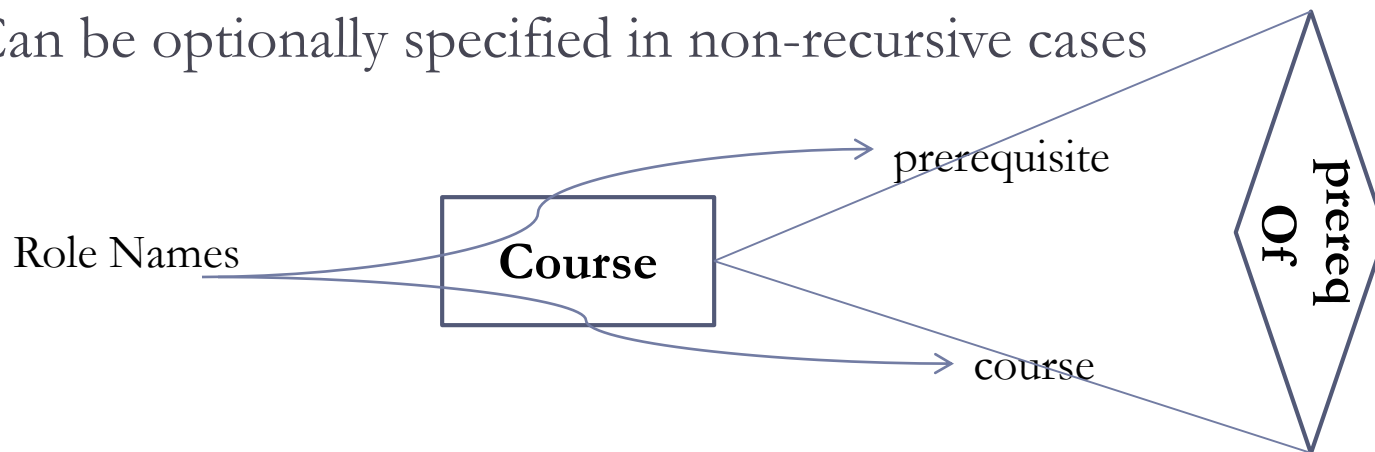
# Attributes for Relationship Types – More Examples

---



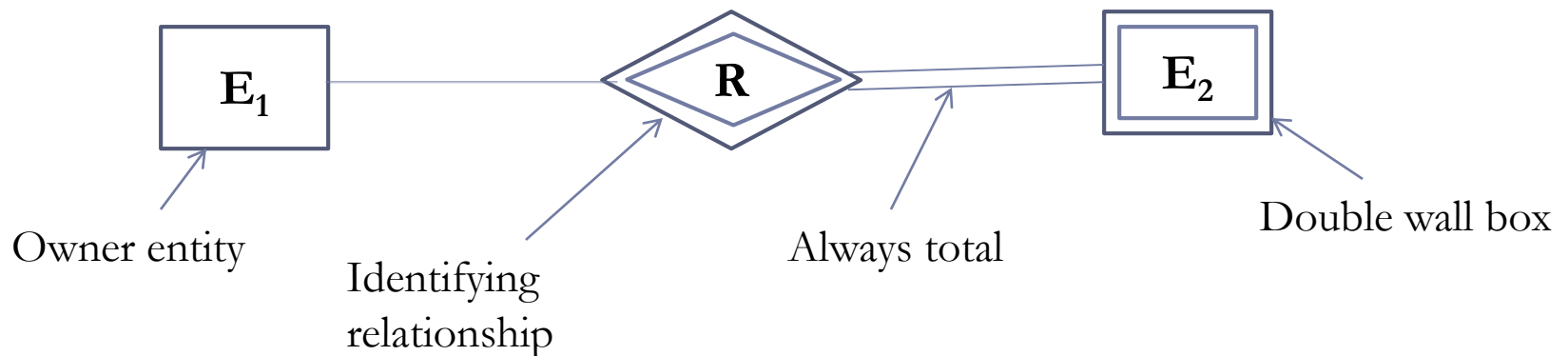
# Recursive Relationships and Role Names

- ▶ **Recursive relationship:** An entity set relating to itself gives rise to a recursive relationship
  - ▶ E.g., the relationship `prereqOf` is an example of a recursive relationship on the entity `Course`
- ▶ **Role Names** – used to specify the exact role in which the entity participates in the relationships
  - ▶ Essential in case of recursive relationships
  - ▶ Can be optionally specified in non-recursive cases



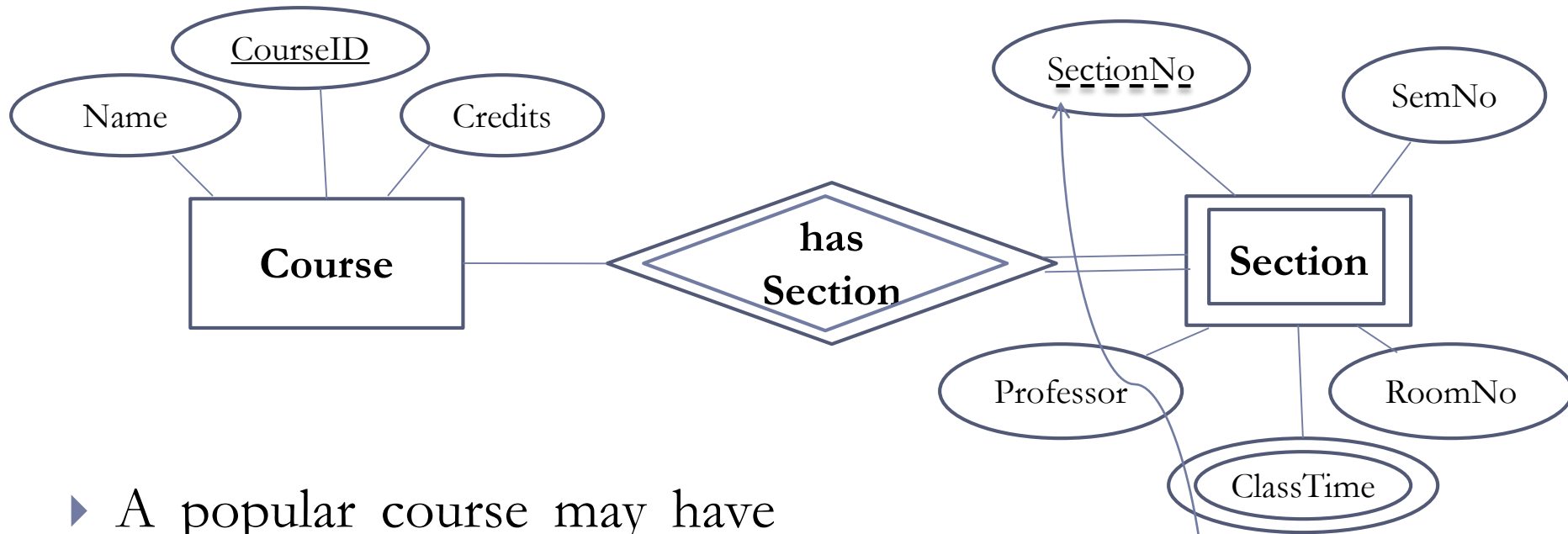
# Weak Entity Sets

- ▶ **Weak Entity Set:** Entity types that do not have key attributes of their own.
- ▶ each weak entity is associated with some entity of the owner entity set through a special relationship.
- ▶ Each weak entity normally has a **partial key** – which is the set of attributes that can uniquely identify weak entities that are related to the same owner entity.





# Weak Entity Sets - Example

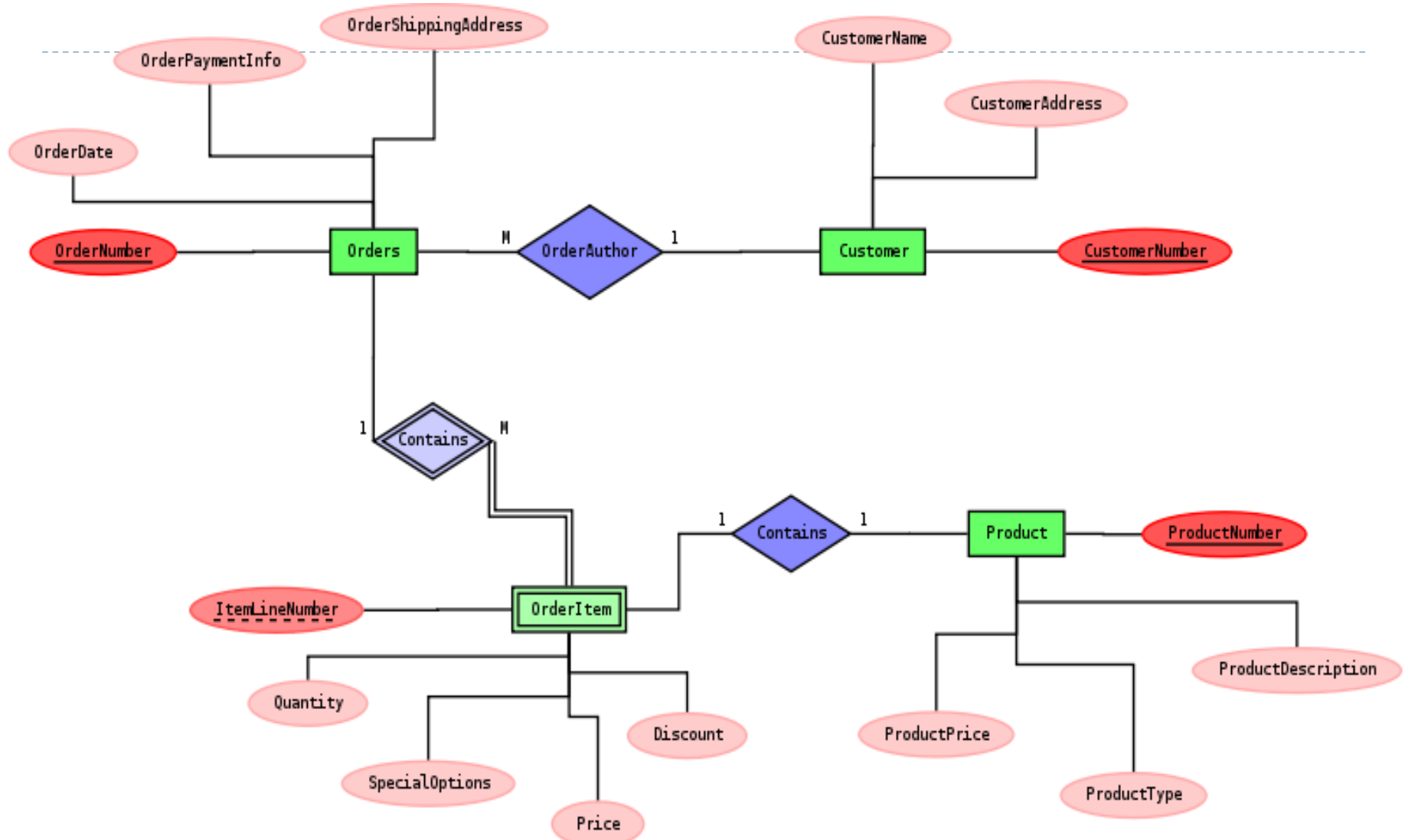


- ▶ A popular course may have several sections each taught by a different professor and having its own class room and meeting times.

**Partial key:**

Uniquely identifies a section among the set of sections of a particular course

# Weak Entity Sets - Example

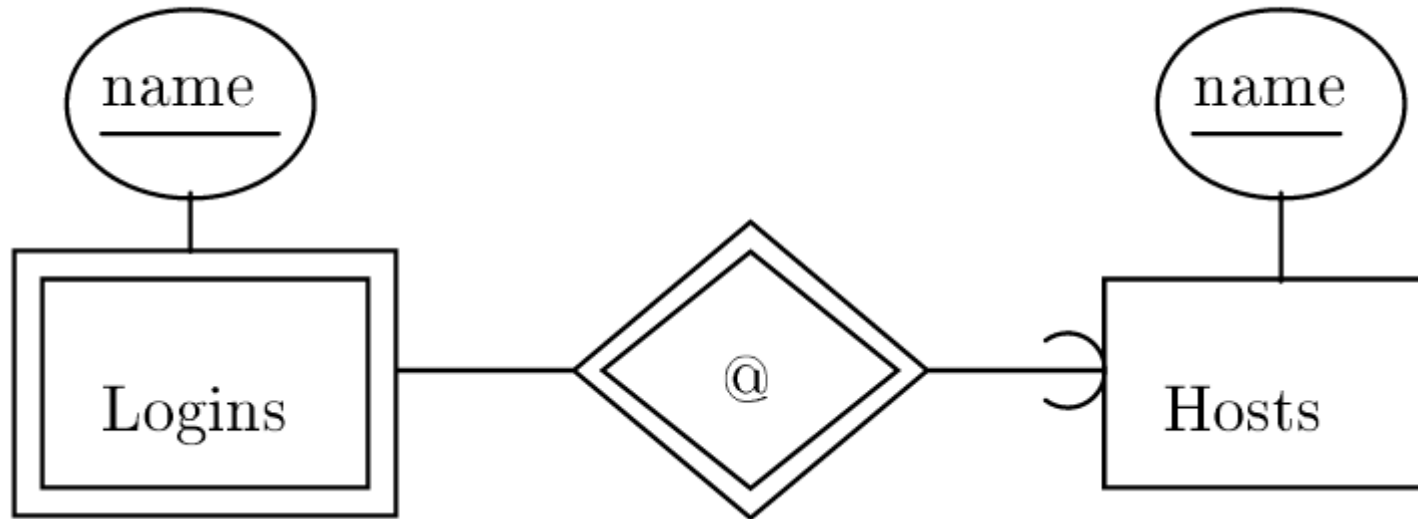


Courtesy: [http://en.wikipedia.org/wiki/Weak\\_entity](http://en.wikipedia.org/wiki/Weak_entity)

# Weak Entity Sets - Example

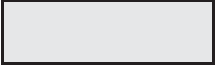

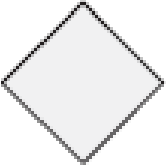
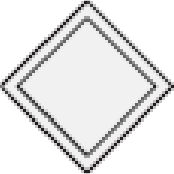



---

- ▶ Note: Login name = username + hostname

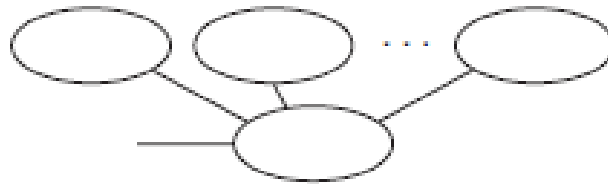


- ▶ Explanation: Logins entity corresponds to username on a particular host.
- ▶ Partial Key for Logins: name, because username @ that host.

# Summary of ER Diagram Notations

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Identifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute

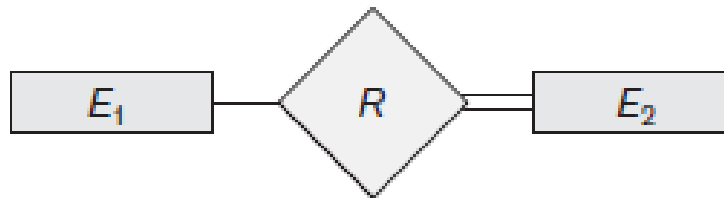
# Summary of ER Diagram Notations



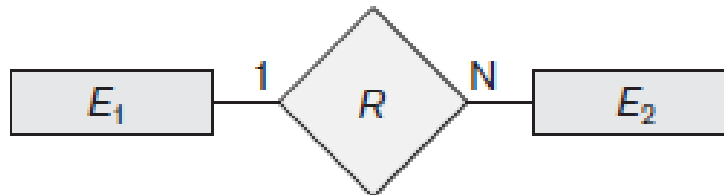
Composite Attribute



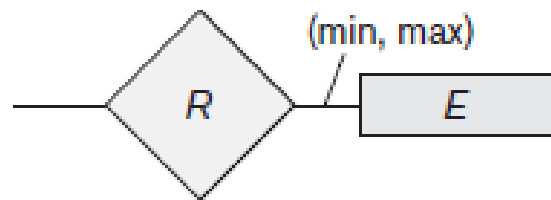
Derived Attribute



Total Participation of  $E_2$  in  $R$



Cardinality Ratio 1: N for  $E_1:E_2$  in  $R$



Structural Constraint (min, max)  
on Participation of  $E$  in  $R$

# Example1 for E/R schema: Specifications

---

- ▶ In an educational institute, there are several departments and students belong to one of them. Each department has a unique department number, a name, a location, phone number and is headed by a professor. Professors have a unique employee Id, name, phone number.
- ▶ Keep track of the following details regarding students: name, unique roll number, sex, phone number, date of birth, age and one or more email addresses. Students have a local address consisting of the hostel name and the room number. They also have home address consisting of house number, street, city and PIN. It is assumed that all students reside in the hostels.

## Example1 for E/R schema: Specifications cont.

---

- ▶ A course taught in a semester of the year is called a section. There can be several sections of the same course in a semester; these are identified by the section number. Each section is taught by a different professor and has its own timings and a room to meet. Students enroll for several sections in a semester.
- ▶ Each course has a name, number of credits and the department that offers it. A course may have other courses as pre-requisites i.e, courses to be completed before it can be enrolled in.

## Example1 for E/R schema: Specifications cont.

---

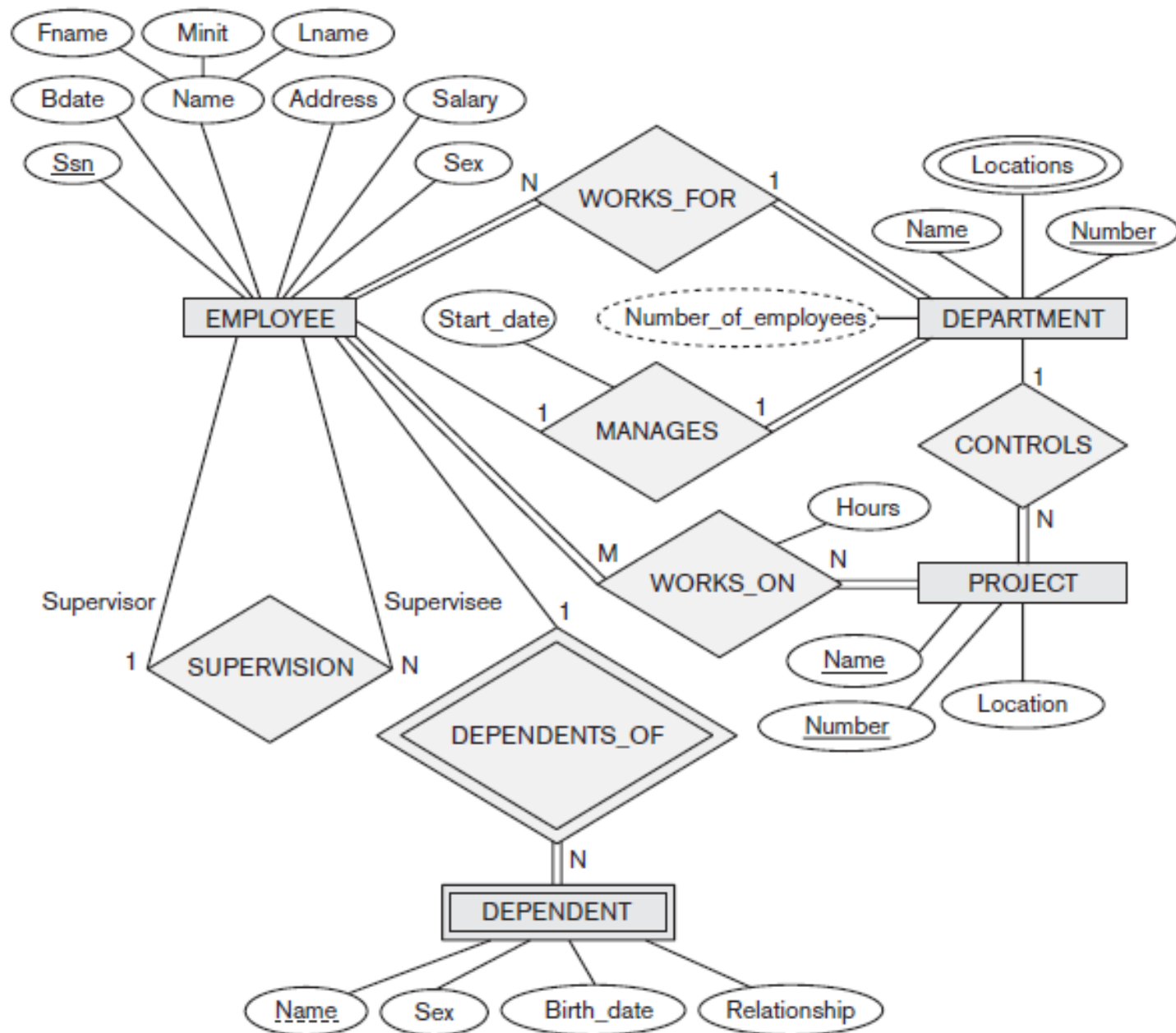
- ▶ Professors also undertake research projects. These are sponsored by funding agencies and have a specific start date, end date and amount of money given. More than one professor can be involved in a project. Also a professor may be simultaneously working on several projects. A project has a unique projectId.



## Example 2 – Specifications

---

- ▶ Let us design a database for a bank, including information about customers and their accounts. Information about a customer includes their name, address, phone, and Social Security number. Accounts have numbers, types (e.g., savings, checking) and balances. We also need to record the customer(s) who own an account. Draw the E/R diagram for this DB.



# Resources

---

- ▶ Chapter 7 of Fundamentals of Database Systems (FODS), 6<sup>th</sup> Edition.
- ▶ Ramakrishnan et al., Database Management Systems
- ▶ Silberschatz A. et al., Database System Concepts
- ▶ Internet Surf