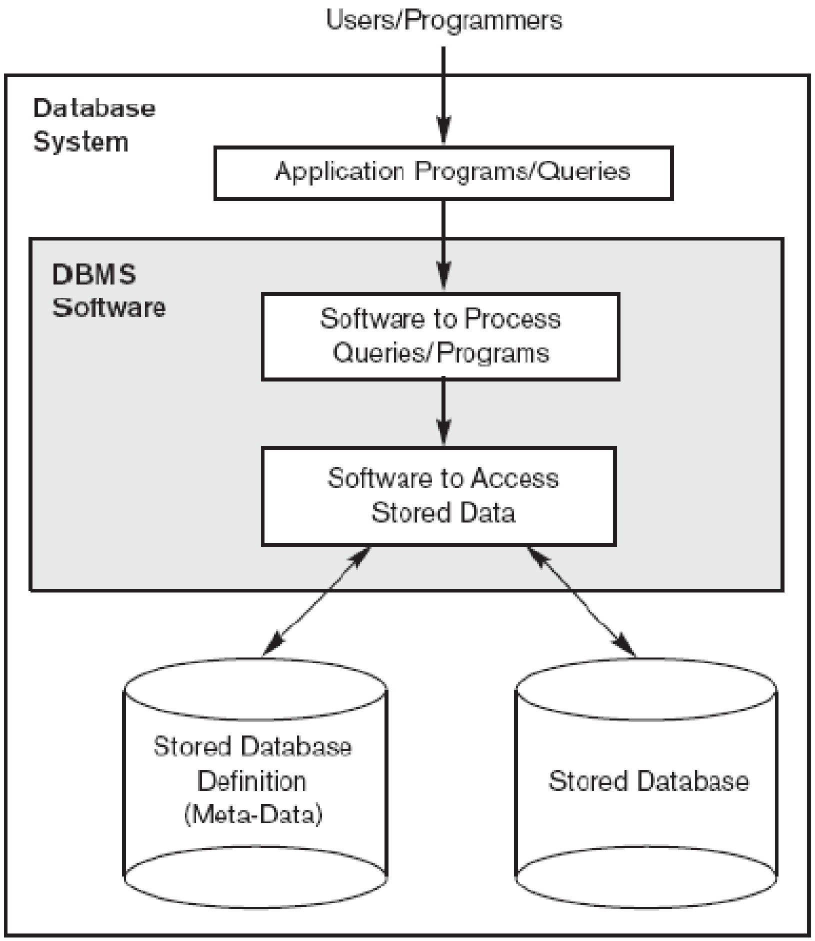
Database Management System (15ECSC208)

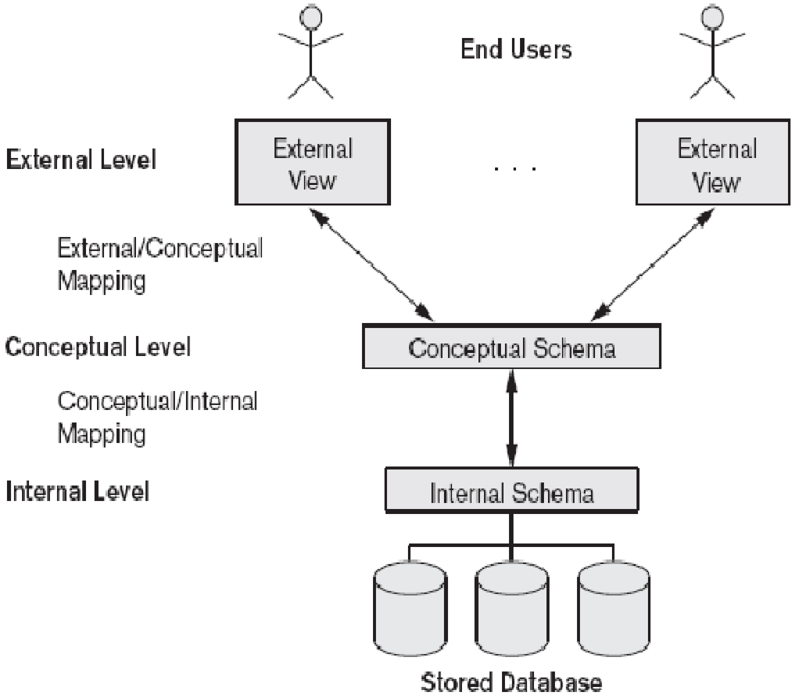
UNIT I: Chapter 1: Introduction to DBMS and ER-Model

Data Modeling Using the Entity – Relationship Model

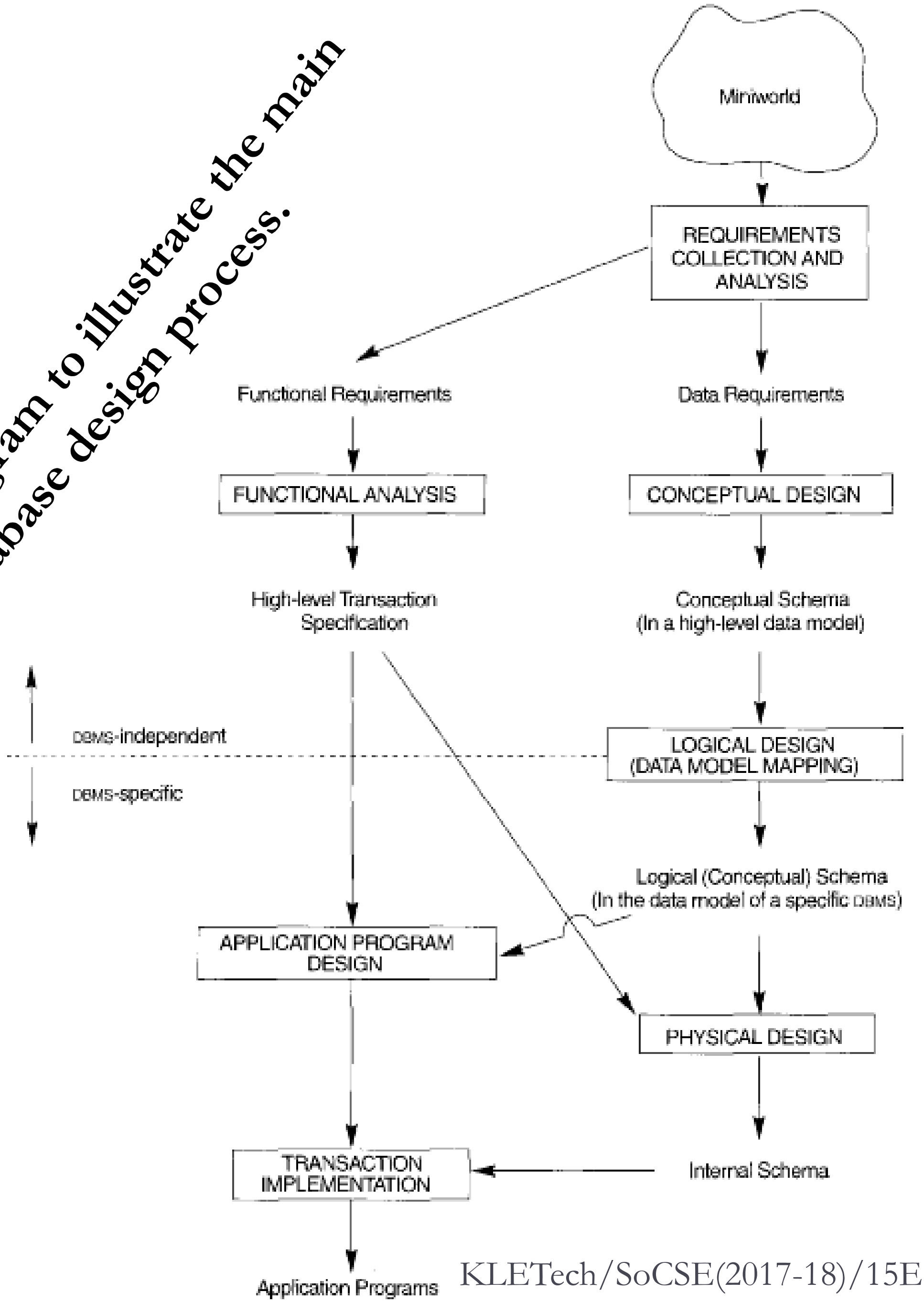
Part 3

Review


Users/Programmers
Database System
Application Programs/Queries
DBMS Software
Software to Process Queries/ Programs
Software to Access Stored Data
Stored Database Definition (Meta-Data)
Stored Database


End Users
External Level
External View
. . .
External View
External/Conceptual Mapping
Conceptual Level
Conceptual Schema
Conceptual/Internal Mapping
Internal Level
Internal Schema
Stored Database

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


Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

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.
  - `Gender`–the gender of the student etc.

- All entities in an Entity set/type have the same set of attributes.

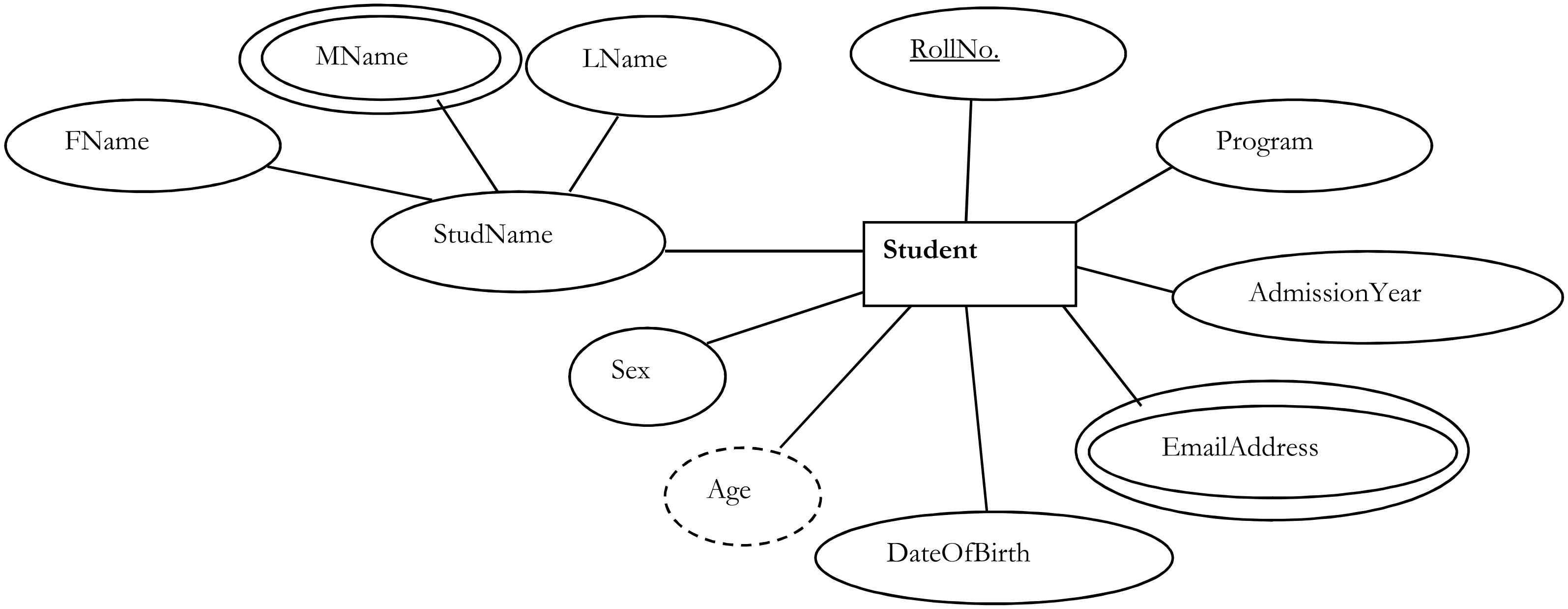
Types of Attributes

- **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, …, 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 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, $\mathit{enrolls} \subseteq \texttt{Student} \times \texttt{Course}$
- $(s,c) \in \mathit{enrolls} \Leftrightarrow$ `Student` ‘s’ has enrolled in `Course` ‘c’
- Tuples in *enrolls* – **relationship instances**
- *enrolls* is called a **relationship Type/Set**.

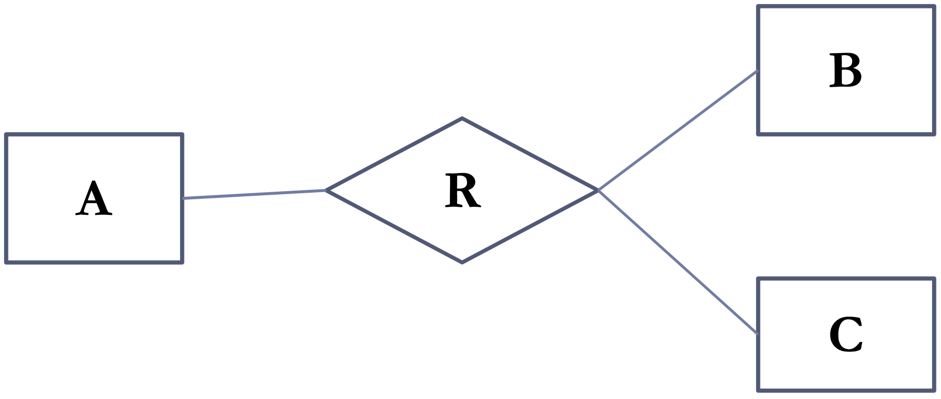
Degree of a relationship

- **Degree**: the number of participating entities.
  - Degree 2: binary
  - Degree 3: ternary
  - Degree n: n-ary

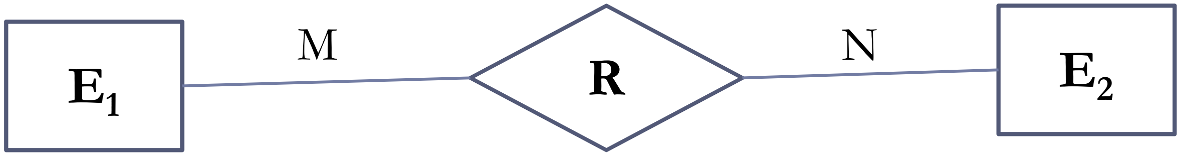
- Binary relationships are very common and widely used.

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

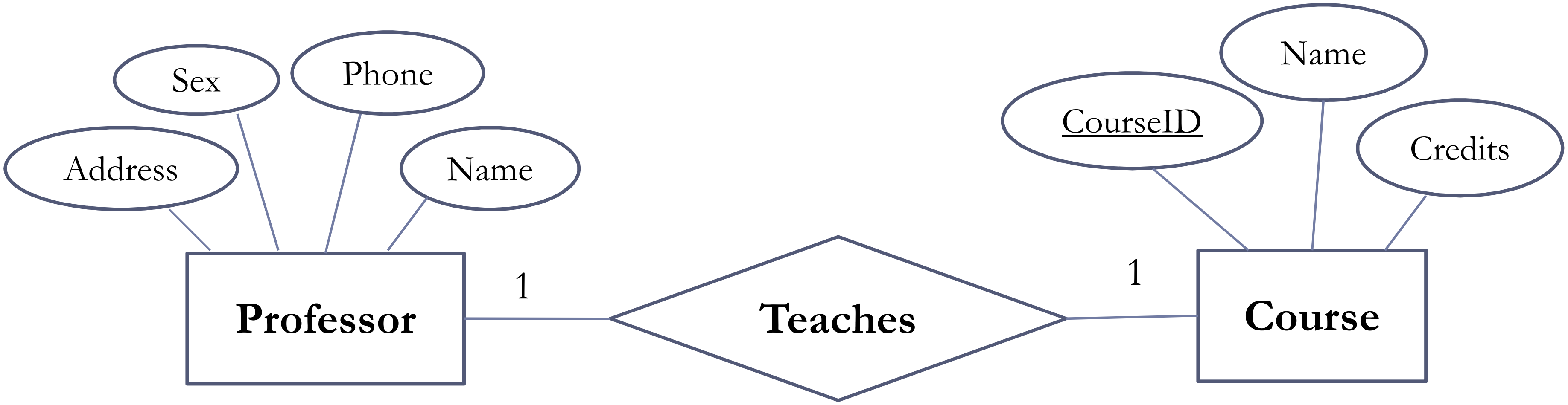


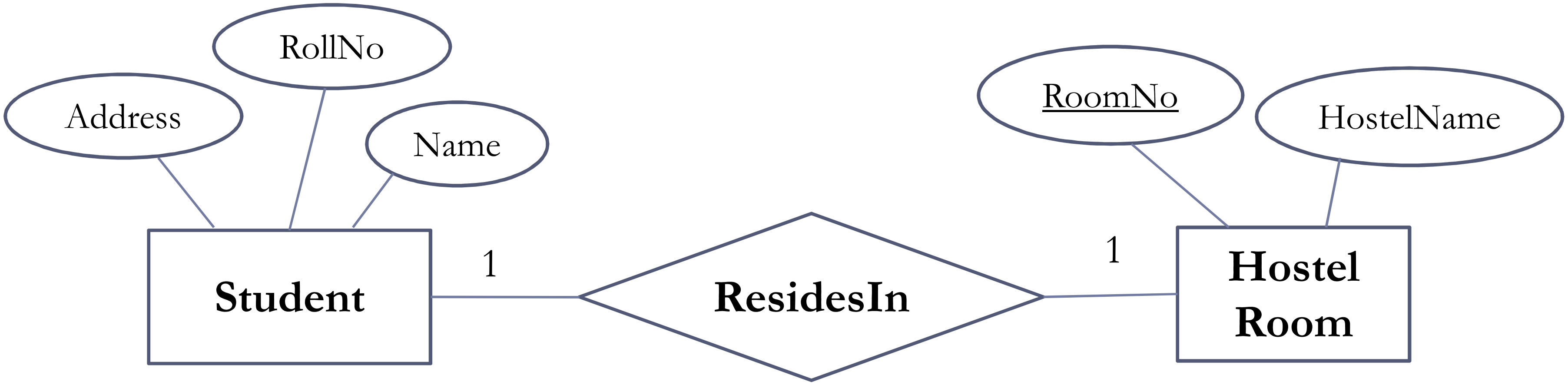
- The number of entities from E_2 that an entity from E_1 can possibly be associated through R (and vice-versa) determines the **cardinality ratio** of R.
- Four possibilities are usually specified:
  - one-to-one (1:1)
  - one-to-many (1:N)
  - many-to-one (N:1)
  - many-to-many (M:N)

Cardinality Ratios

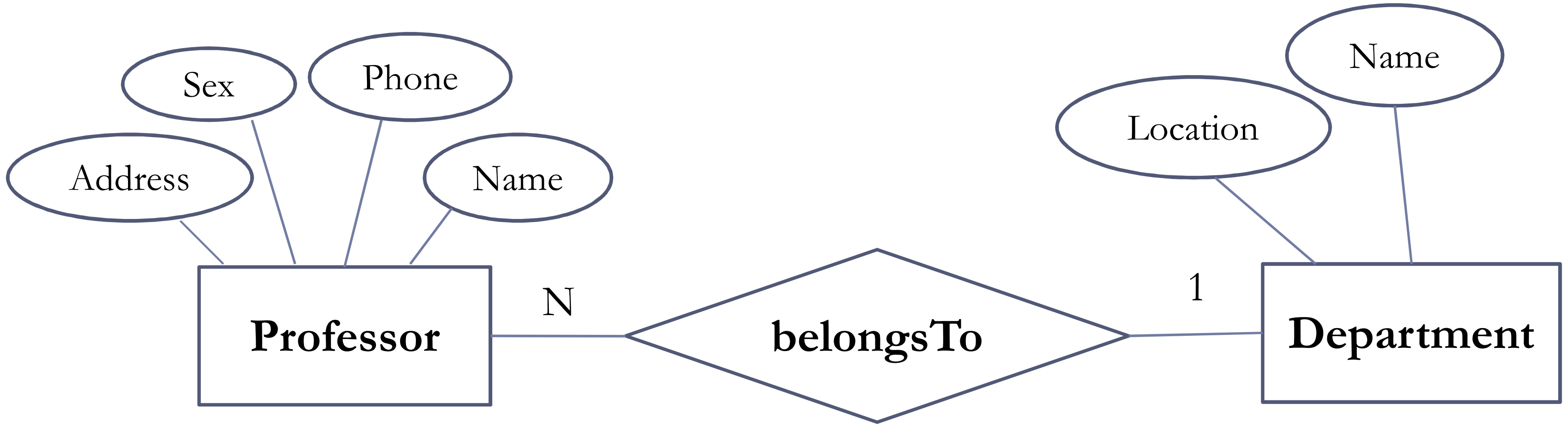
- **One-to-one:** An $\mathbf{E_1}$ entity may be associated with at most one $\mathbf{E_2}$ entity and similarly an $\mathbf{E_2}$ entity may be associated with at most one $\mathbf{E_1}$ entity.
- **One-to-many:** An $\mathbf{E_1}$ entity may be associated with many $\mathbf{E_2}$ entities whereas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

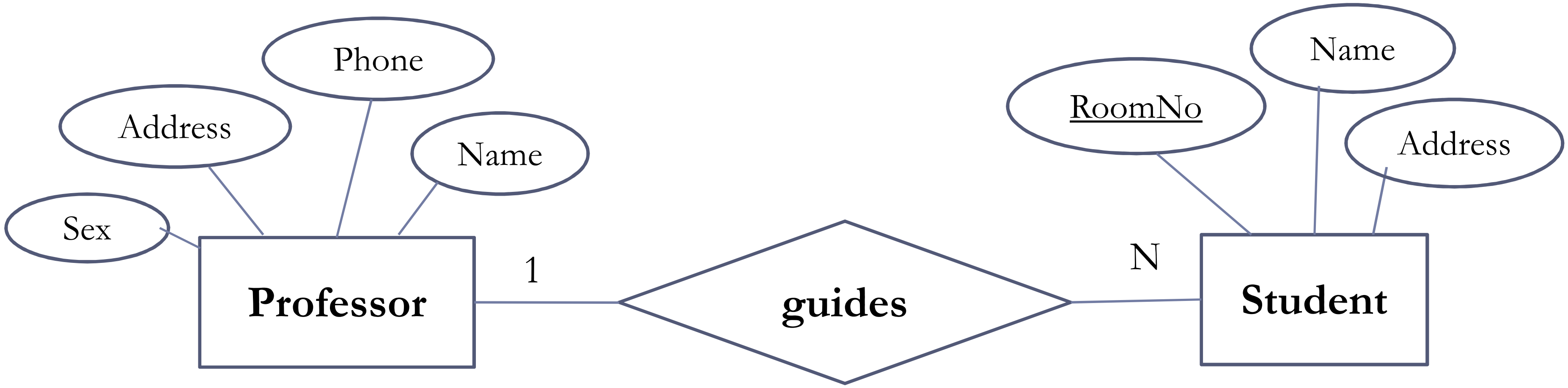
Cardinality Ratio – example (*one-to-one*)



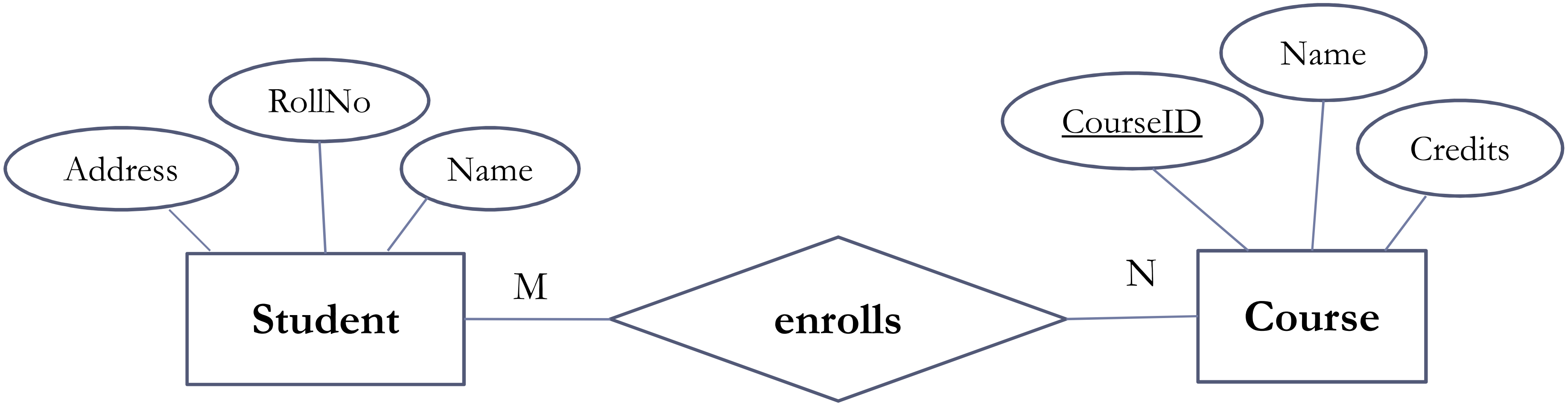


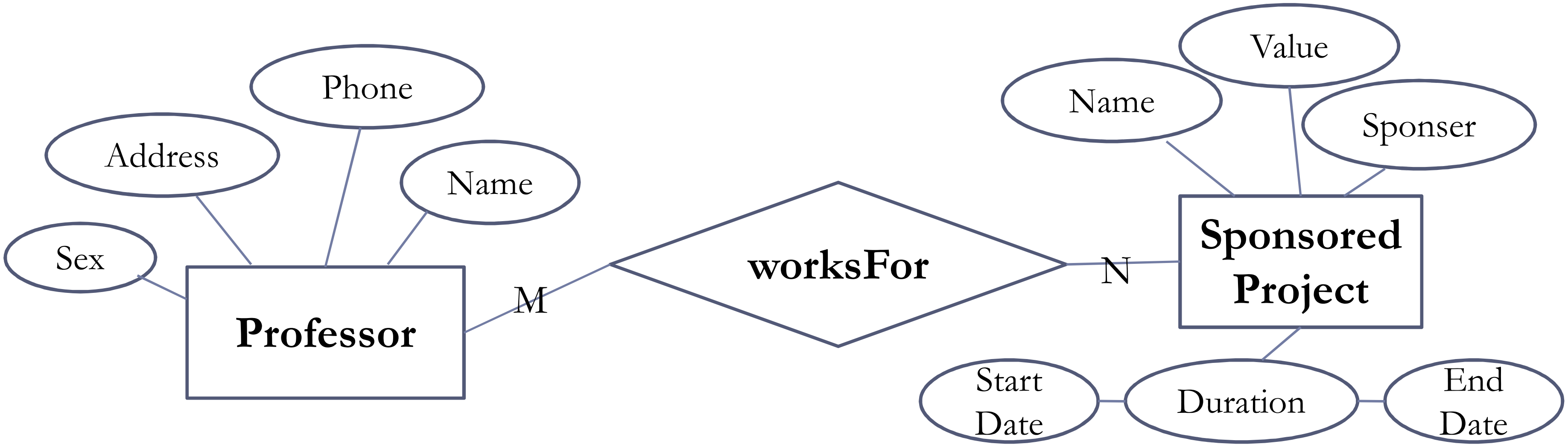
Cardinality Ratio – example (*many-to-one /one-to-many)*





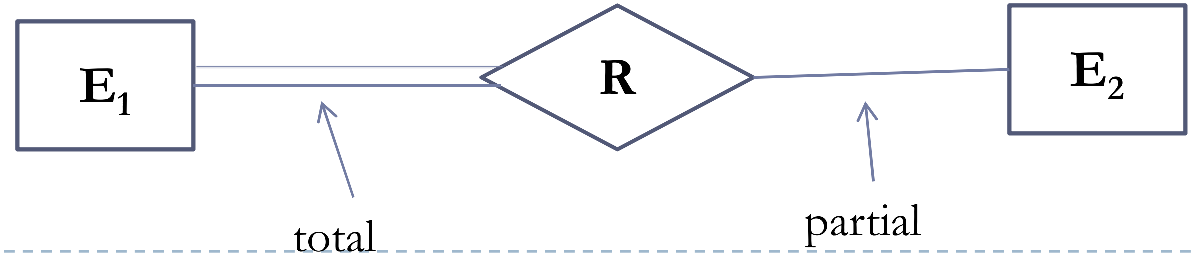
Cardinality Ratio – example (many-*to-many)*



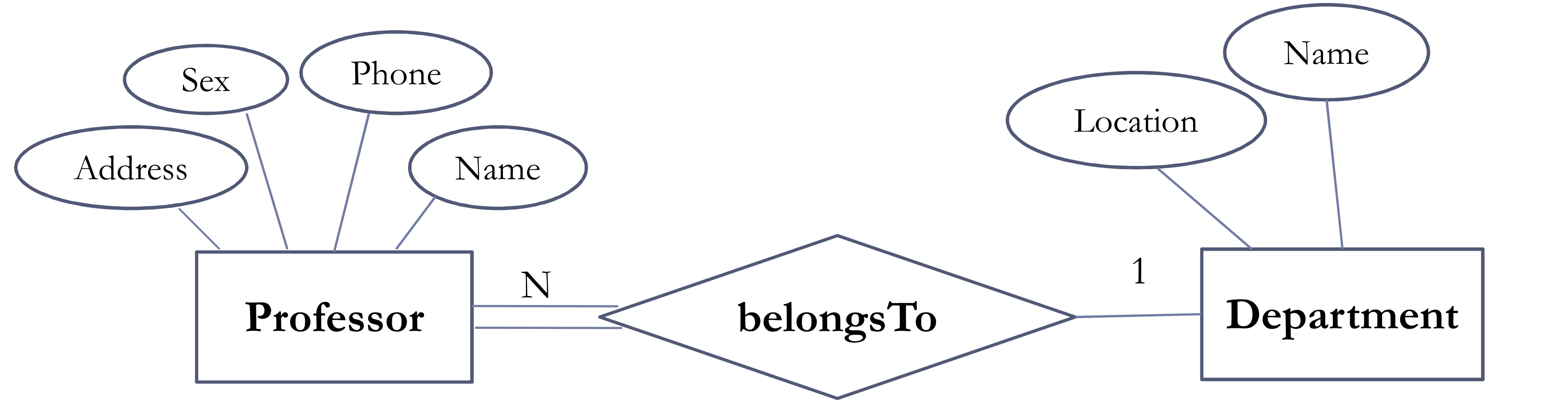


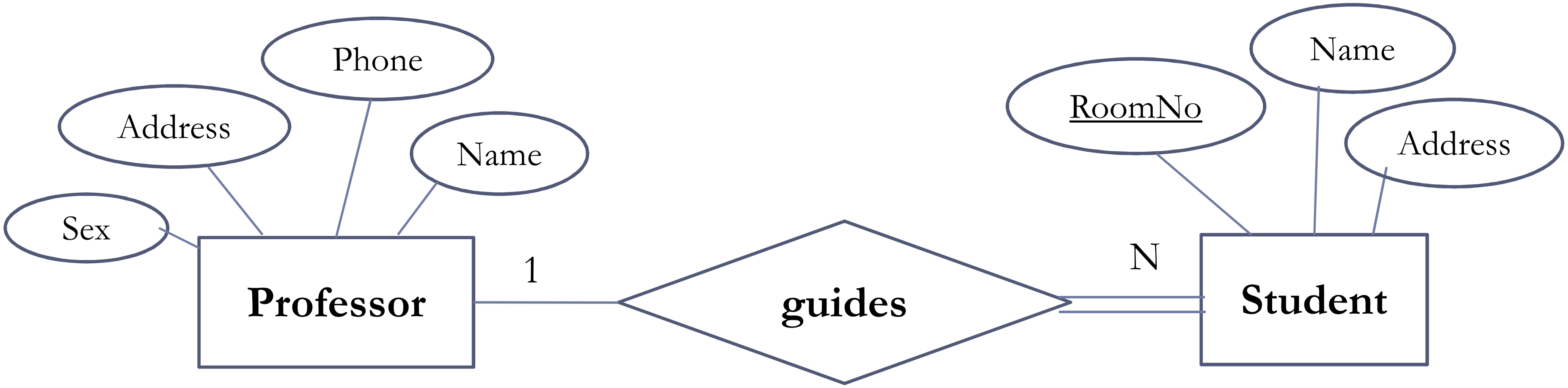
Participation Constraints

- 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

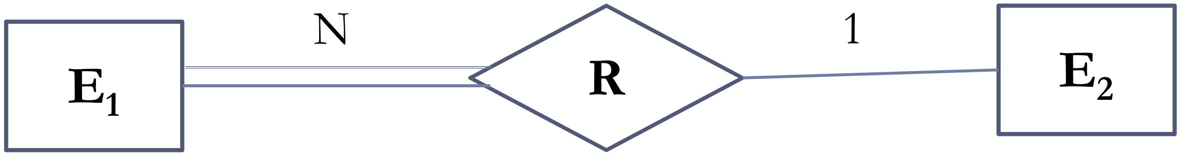




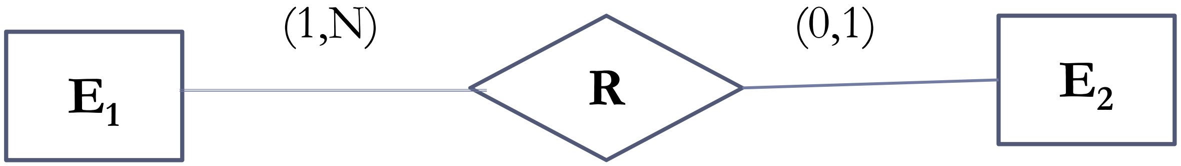
Structural Constraints

- 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
  - ≥1 – total participation
- **n:** the maximum number of times a particular entity can appear in the relationship tuples at any point of time

Comparing the Notations

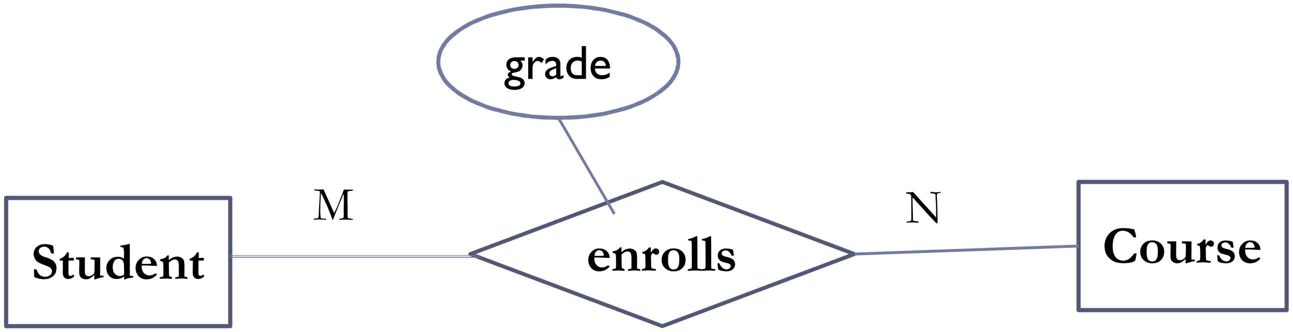


is equivalent to



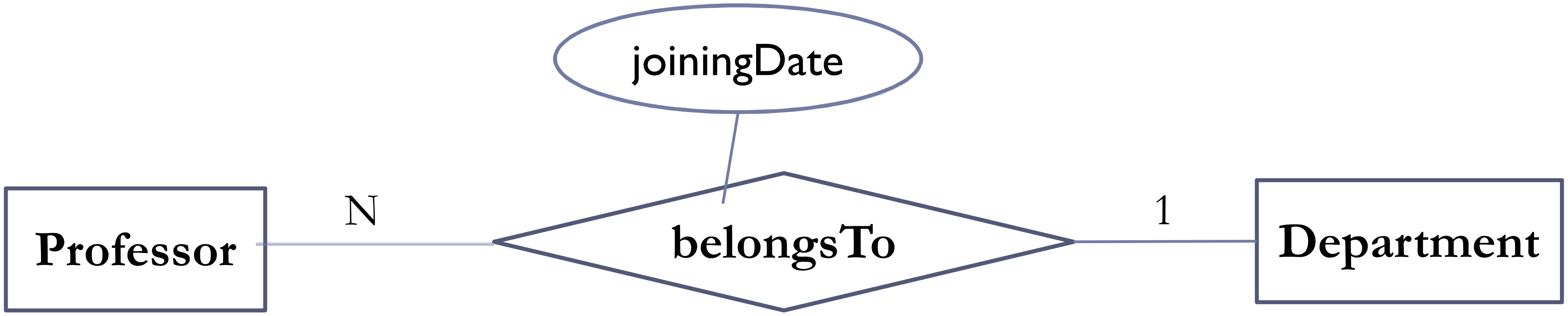
Attributes for Relationship Types

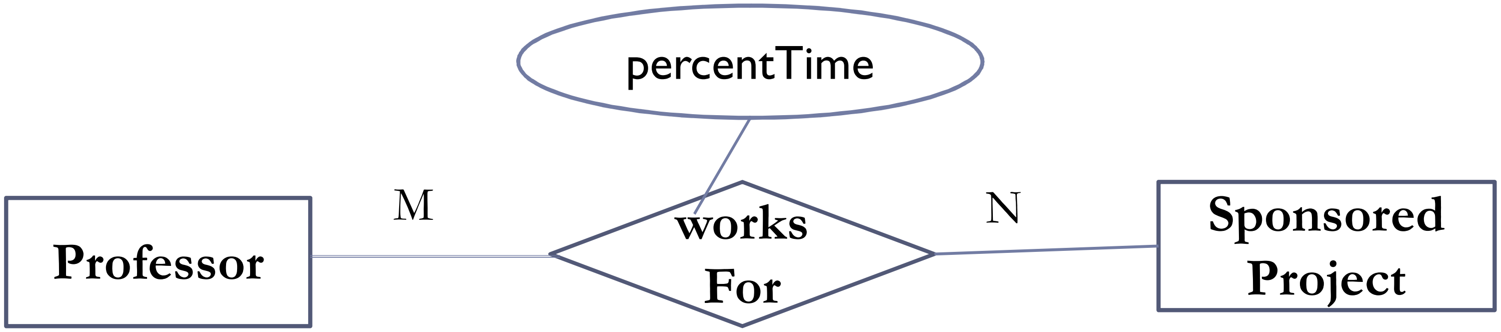
- Relationship types can also have attributes.
  - properties of the association of entities.



- 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.

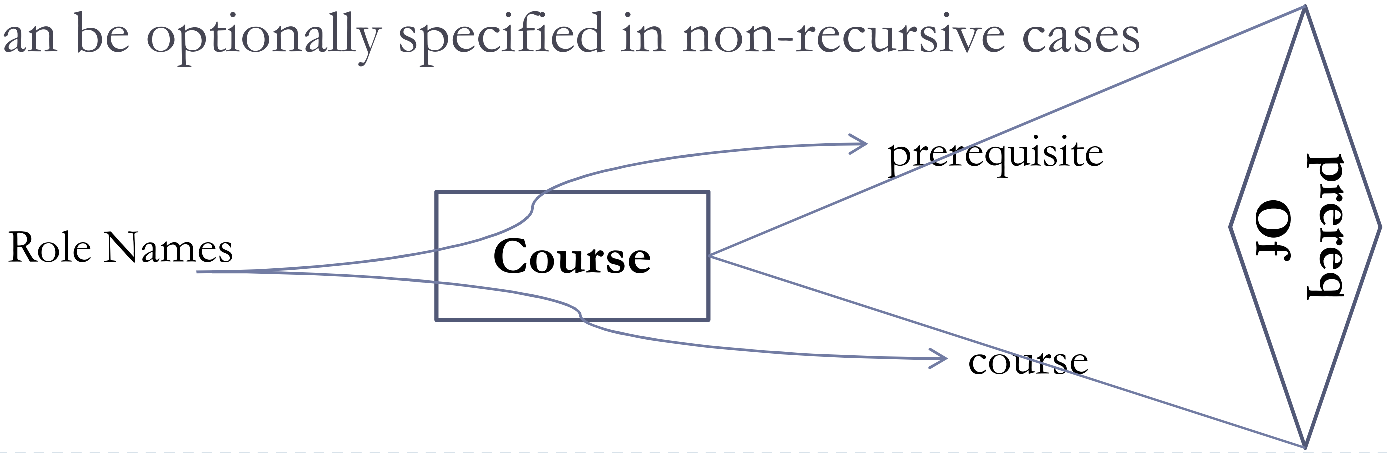
Attributes for Relationship Types – More Examples





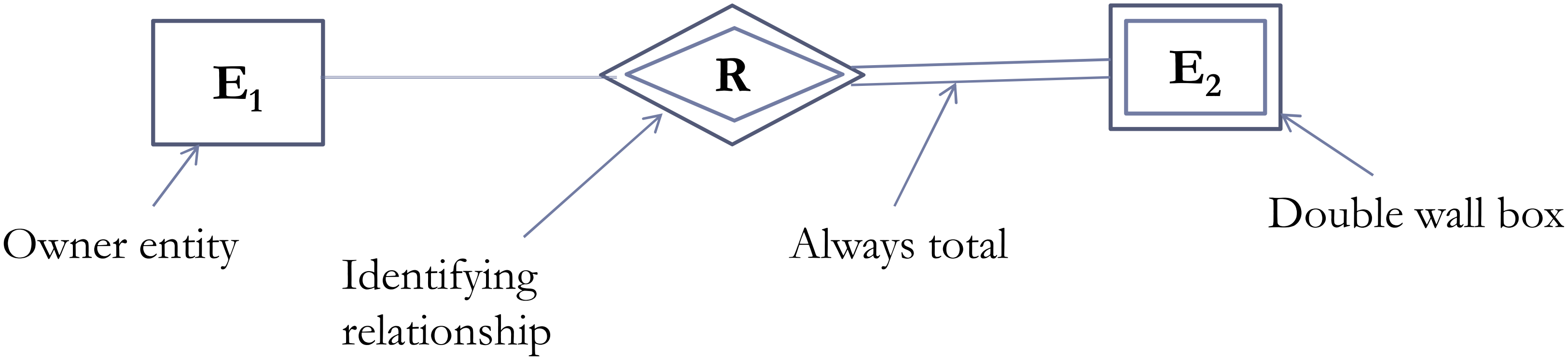
Recursive Relationships and Role Names

- **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

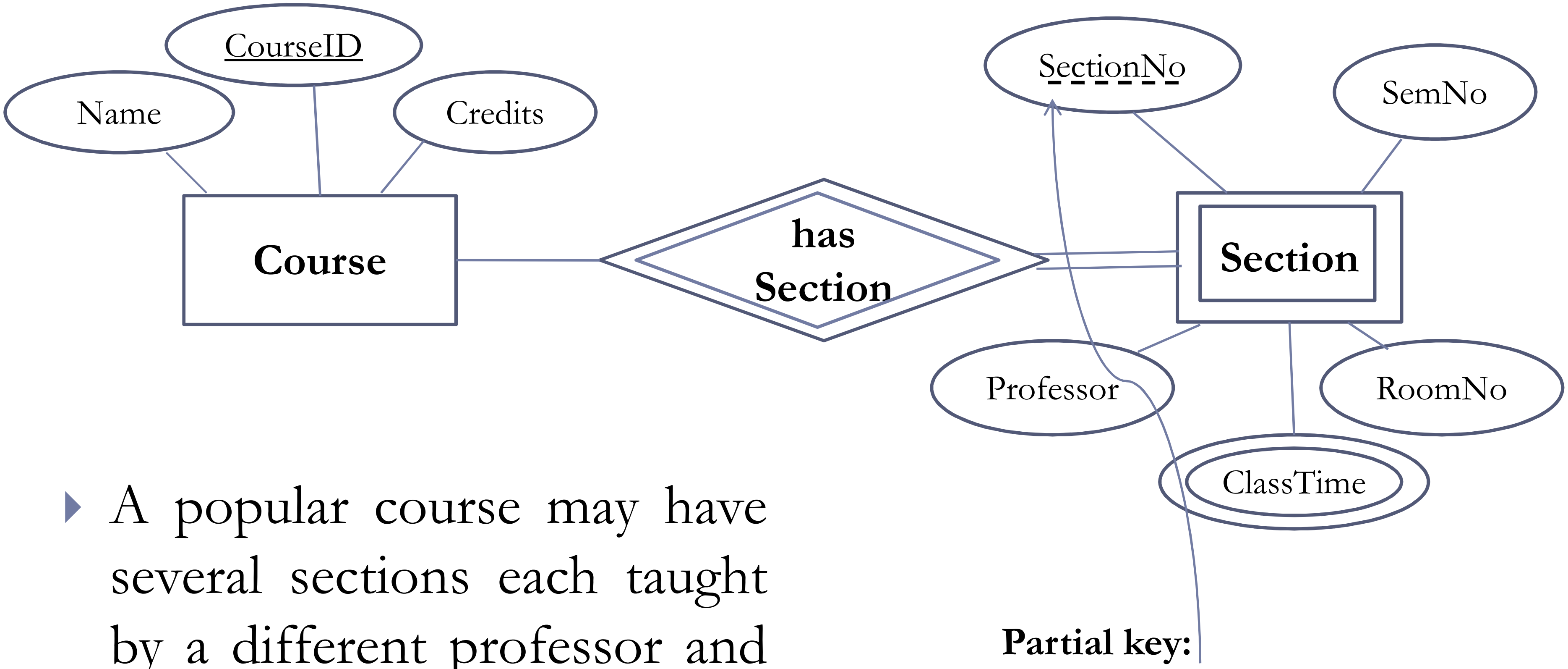


Weak Entity Sets

- **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.



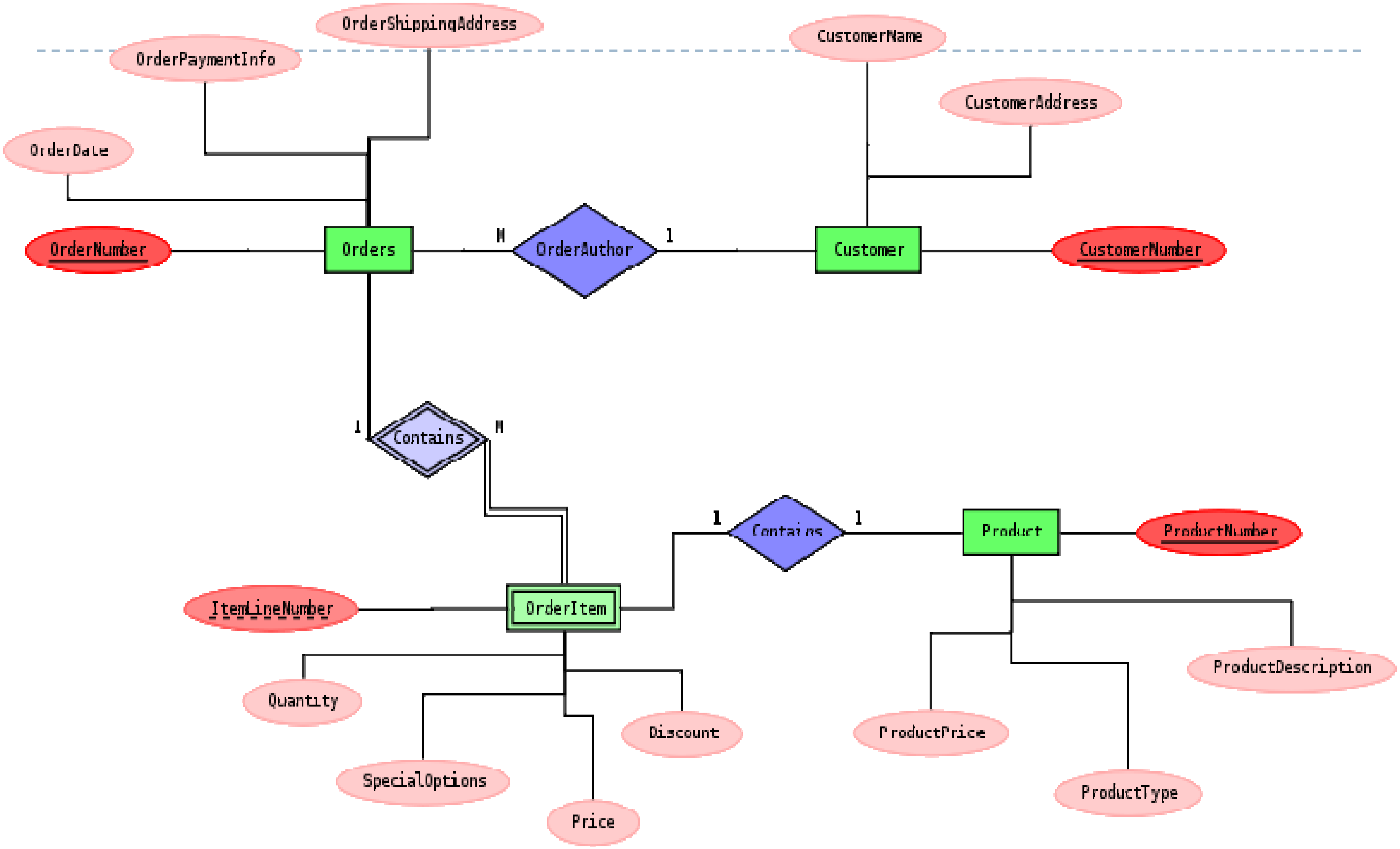
Weak Entity Sets - Example



- 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

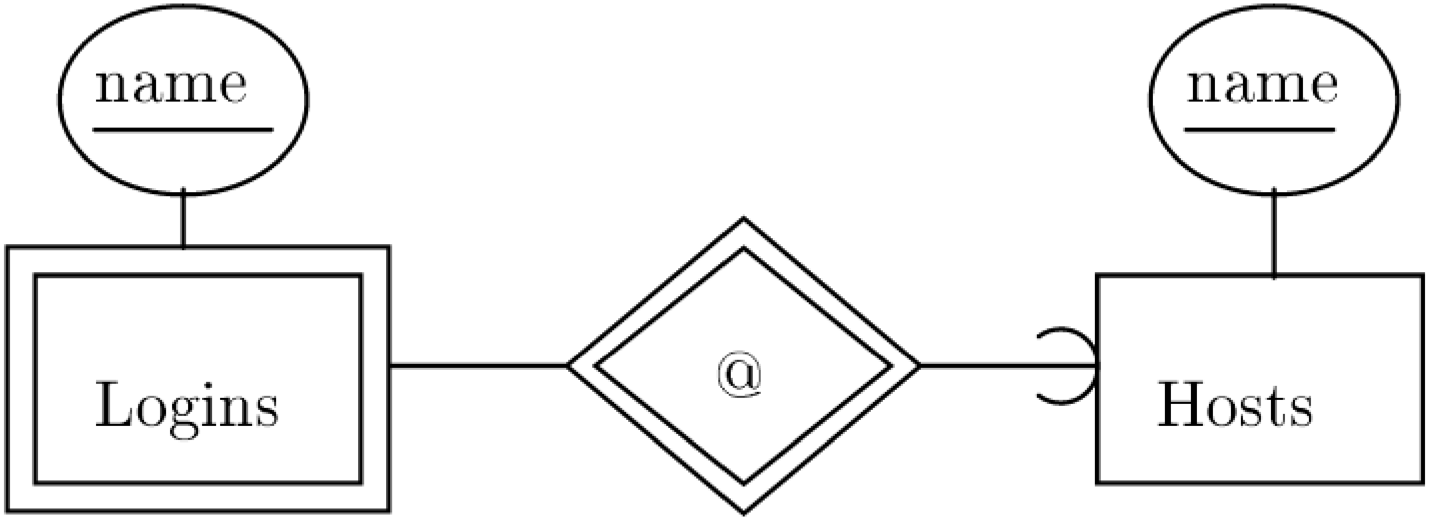
Weak Entity Sets - Example



Courtesy: http://en.wikipedia.org/wiki/Weak_entity

Weak Entity Sets - Example

- Note: Login name = username + hostname

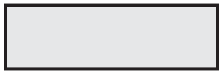
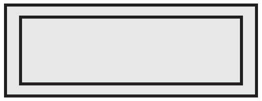
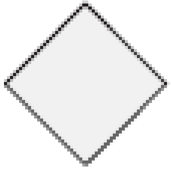
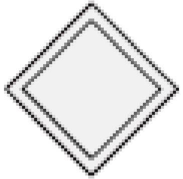
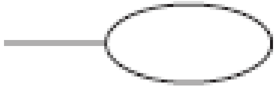
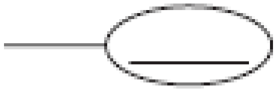
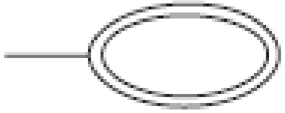


- Explanation: Logins entity corresponds to username on a particular host.
- Partial Key for Logins: name, because username @ that host.

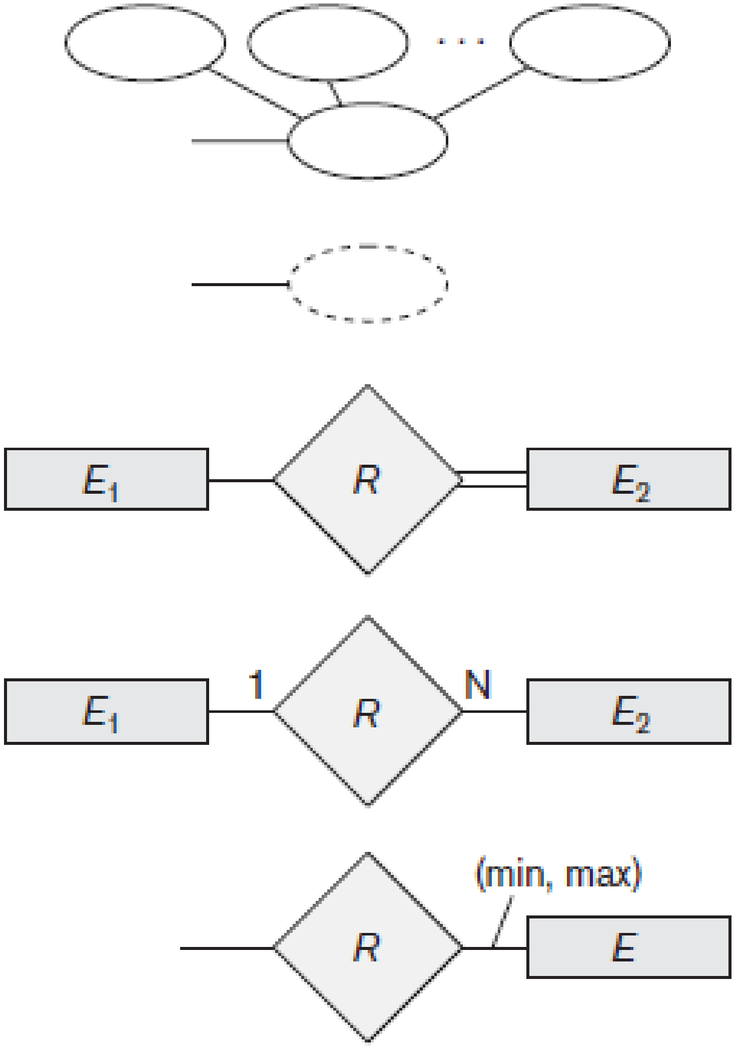
Weak Entity Sets - Example

- **Scenario:** A movie studio might have several film crews. The crews might be designated by a given studio as crew 1, crew 2, and so on. However, other studios might use the same designations for crews, so the attribute *number* is not a key for crews. Rather, to name a crew uniquely, we need to give both the name of the studio to which it belongs and the number of the crew.

Summary of ER Diagram Notations

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Indentifying 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

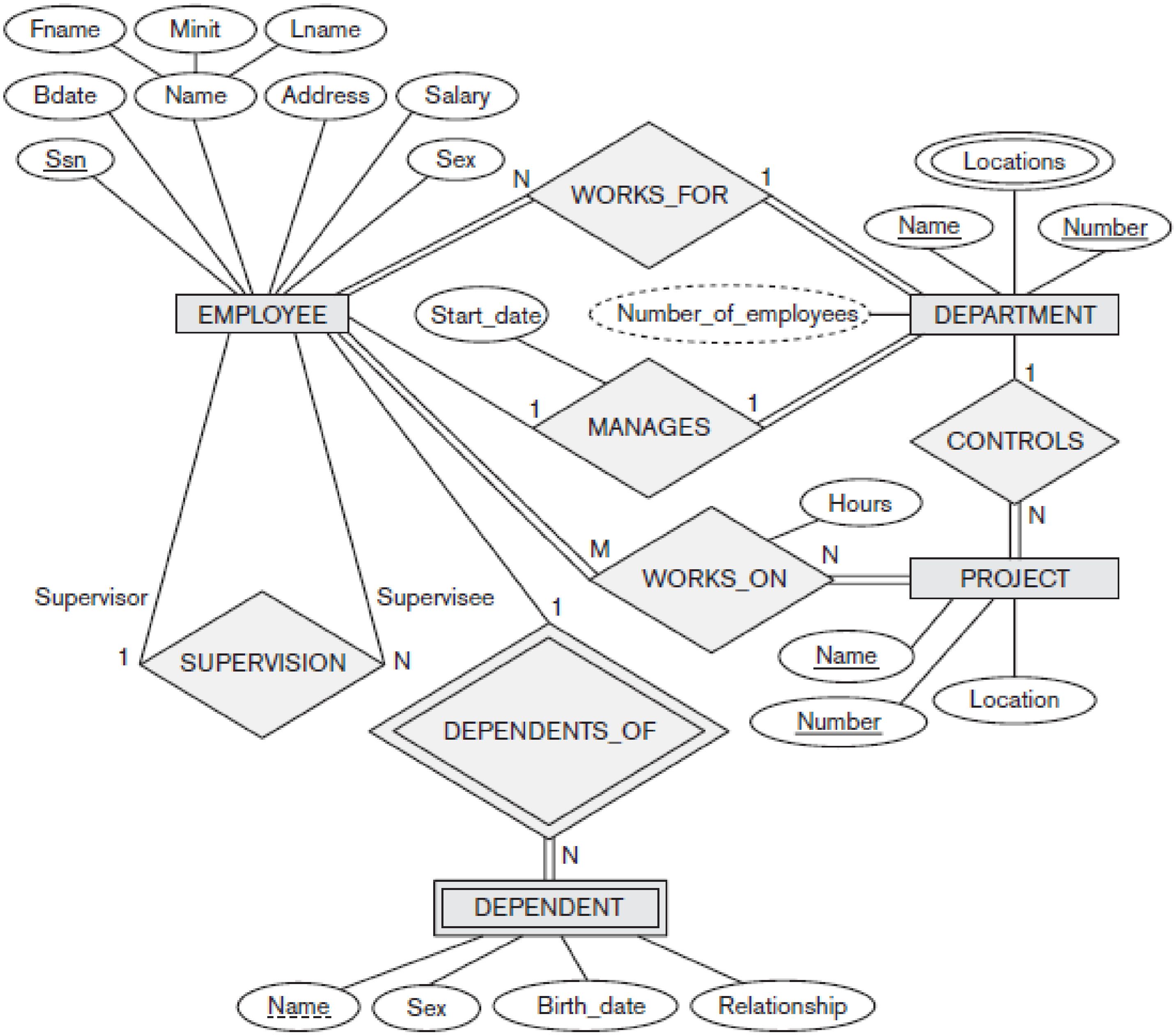
Example 0 – Specifications

- The **company** is organized into departments. Each department has a unique name, a unique number, and a particular employee who manages the department. We keep track of the start date when that employee began managing the department. A department may have several locations.
- A department controls a number of projects, each of which has a unique name, a unique number, and a single location.
- We store each employee's name, social security number. address, salary, sex, and birth date. An employee is assigned to one department but may work on several projects, which are not necessarily controlled by the same department. We keep track of the number of hours per week that an employee works on each project. We also keep track of the direct supervisor of each employee.
- We want to keep track of the dependents of each employee for insurance purposes. We keep each dependent's first name, sex, birth date, and relationship to the employee.

Example 0 – Specifications

- A company database needs to store information about employees (identified by *ssn, with salary and phone as attributes), departments (identified by dno,* with *dname and budget as attributes), and children of employees (with name and age* as attributes). Employees *work in departments; each department is managed by an* employee; a child must be identified uniquely by *name when the parent (who is an* employee; assume that only one parent works for the company) is known. We are not interested in information about a child once the parent leaves the company.

- Draw an ER diagram that captures this information.


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
WORKS_ON
N
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
DEPENDENTS_OF
Name
Location
Number
N
DEPENDENT
Name
Sex
Birth_date
Relationship

Resources

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