Entity-Relationship (E/R) Model

Design Phases

The initial phase of database design is to characterize fully the data needs of the prospective database users. Next, the designer chooses a data model and, by applying the concepts of the chosen data model, translates these requirements into a conceptual schema of the database.

A fully developed conceptual schema also indicates the functional requirements of the enterprise. In a "specification of functional requirements", users describe the kinds of operations (or transactions) that will be performed on the data.

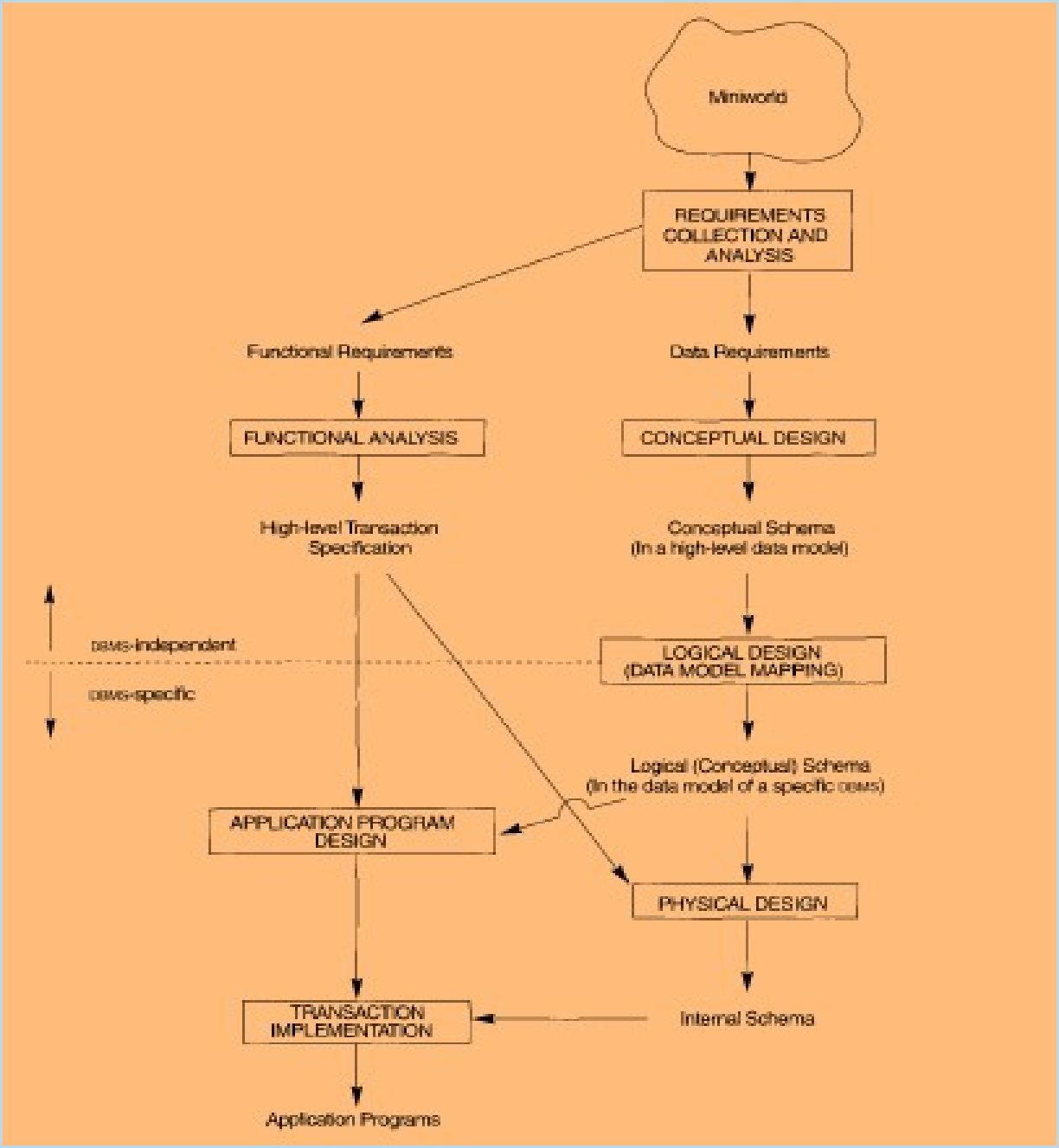
Design Phases (Cont.)

The process of moving from an abstract data model to the implementation of the database proceeds in two final design phases.

Logical Design – Deciding on the database schema. Database design requires that we find a "good" collection of relation schemas.

- Business decision – What attributes should we record in the database?
- Computer Science decision – What relation schemas should we have and how should the attributes be distributed among the various relation schemas?

Physical Design – Deciding on the physical layout of the database


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

Entity-Relationship (E/R) Model

- Widely used conceptual level data model
  - proposed by Peter P Chen in 1970s
- Data model to describe the database system at the requirements collection stage
  - high level description.
  - easy to understand for the enterprise managers.
  - rigorous enough to be used for system building.
- Concepts available in the model
  - entities and attributes of entities.
  - relationships between entities.
  - diagrammatic notation.

Entities

- *Entity* - a thing (animate or inanimate) of independent physical or conceptual existence and *distinguishable*.

  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.

Chosen set of attributes – amount of detail in modeling.

Types of Attributes (1/2)

- Simple Attributes
  - having atomic or indivisible values.

    `example`: *Dept* – a string

    PhoneNumber – an eight digit number
- Composite Attributes
  - having several components in the value.

    `example`: *Qualification* with components

    (*DegreeName, Year*, *UniversityName*)
- Derived Attributes
  - Attribute value is dependent on some other attribute.

    `example`: *Age* depends on *DateOf Birth*.

    So age is a derived attribute.

Types of Attributes (2/2)

- 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

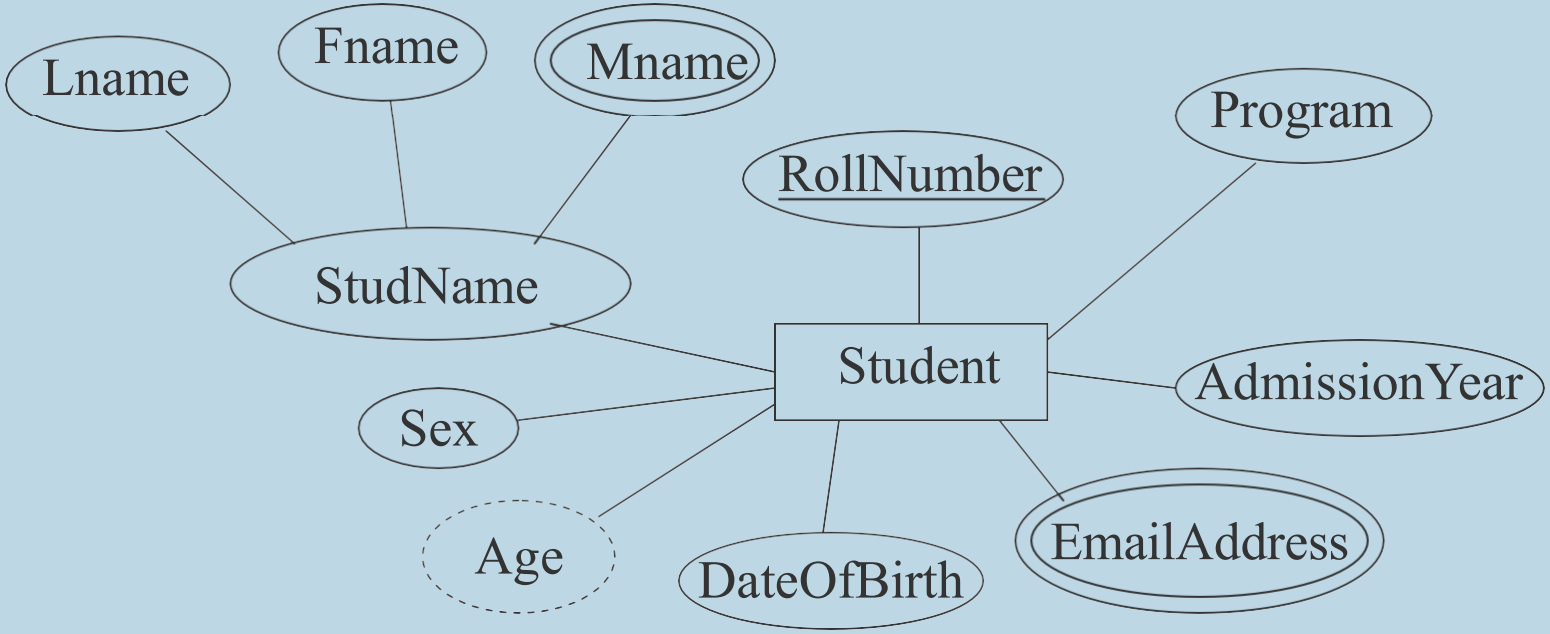
entity - rectangle

attribute - ellipse connected to rectangle

multi-valued attribute - double ellipse

composite attribute - ellipse connected to ellipse

derived attribute - dashed ellipse



Domains of Attributes

Each attribute takes values from a set called its *domain*

For instance, *studentAge* – {17,18, …, 55}

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.
- An entity set may have more than one key.
- Keys can be determined only from the meaning of the attributes in the entity type.
  - Determined by the designers

Relationships

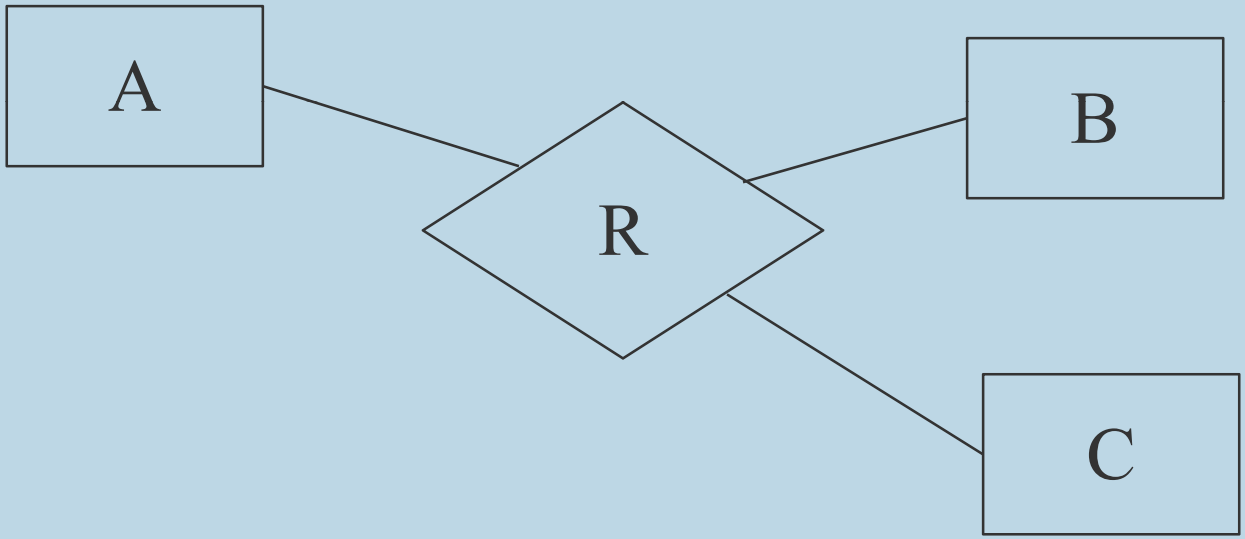
- When two or more entities are associated with each other, we have an instance of a *Relationship*.
- E.g.: *student* Ramesh *enrolls* in Discrete Mathematics *course*
- Relationship *enrolls* has *Student* and *Course* as the *participating* entity sets.
- Formally, *enrolls* $\subseteq$ *Student* $\times$ *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

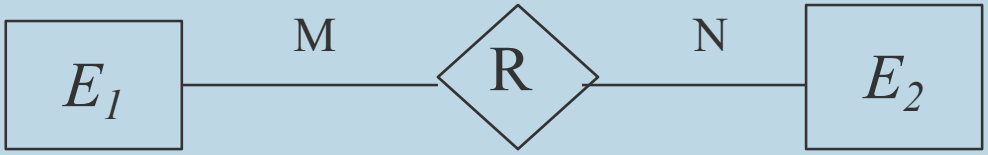
- *Degree* : the number of participating entities.
  - Degree 2: *binary*
  - Degree 3: *ternary*
  - Degree n: *n-ary*
- Binary relationships are very common and widely used.

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



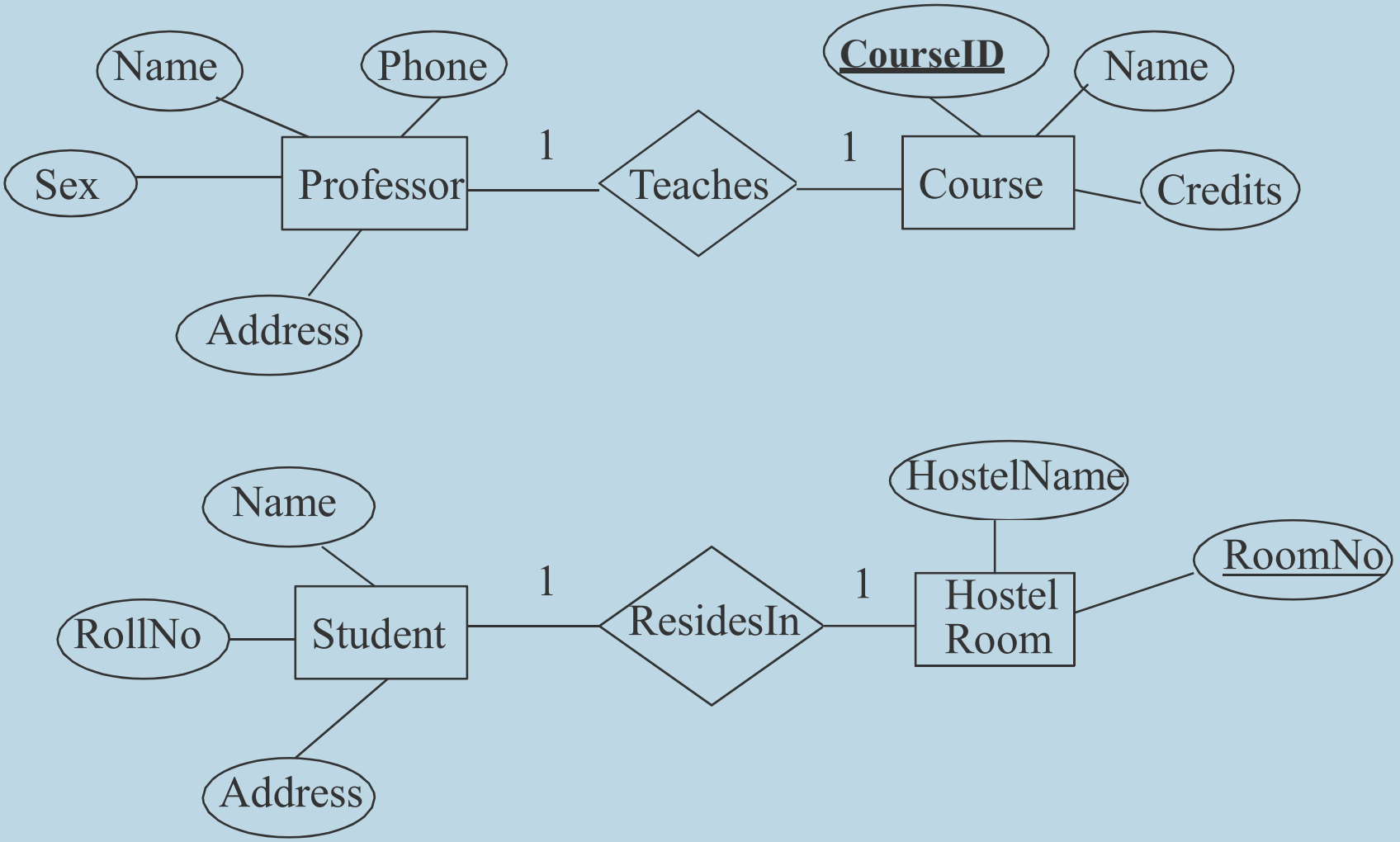
- The number of entities from E_2 that an entity from E_1 can possibly be associated thru R (and vice-versa) determines the *cardinality ratio* of R.

- 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)*

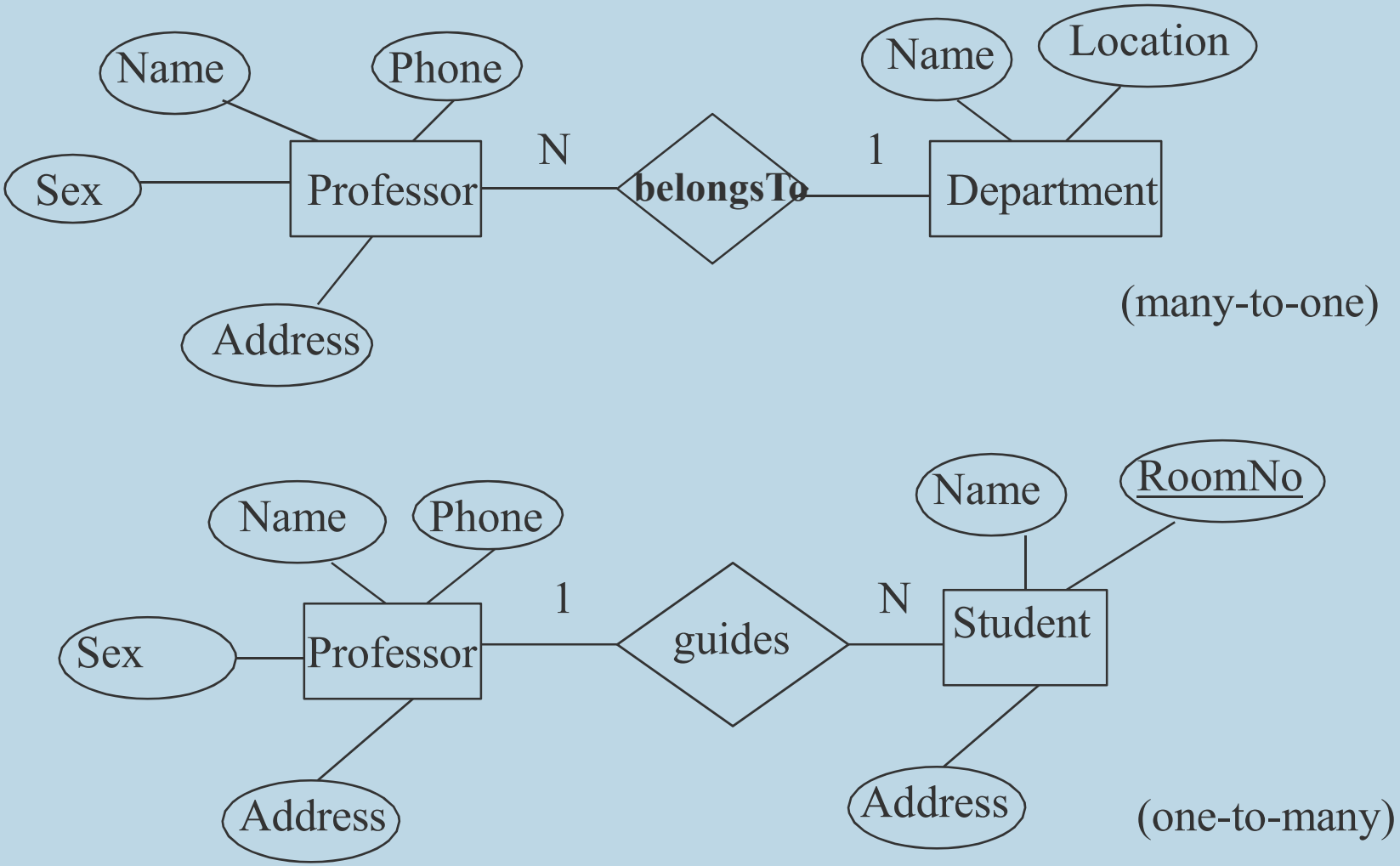
Cardinality Ratios

- *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)
- *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.

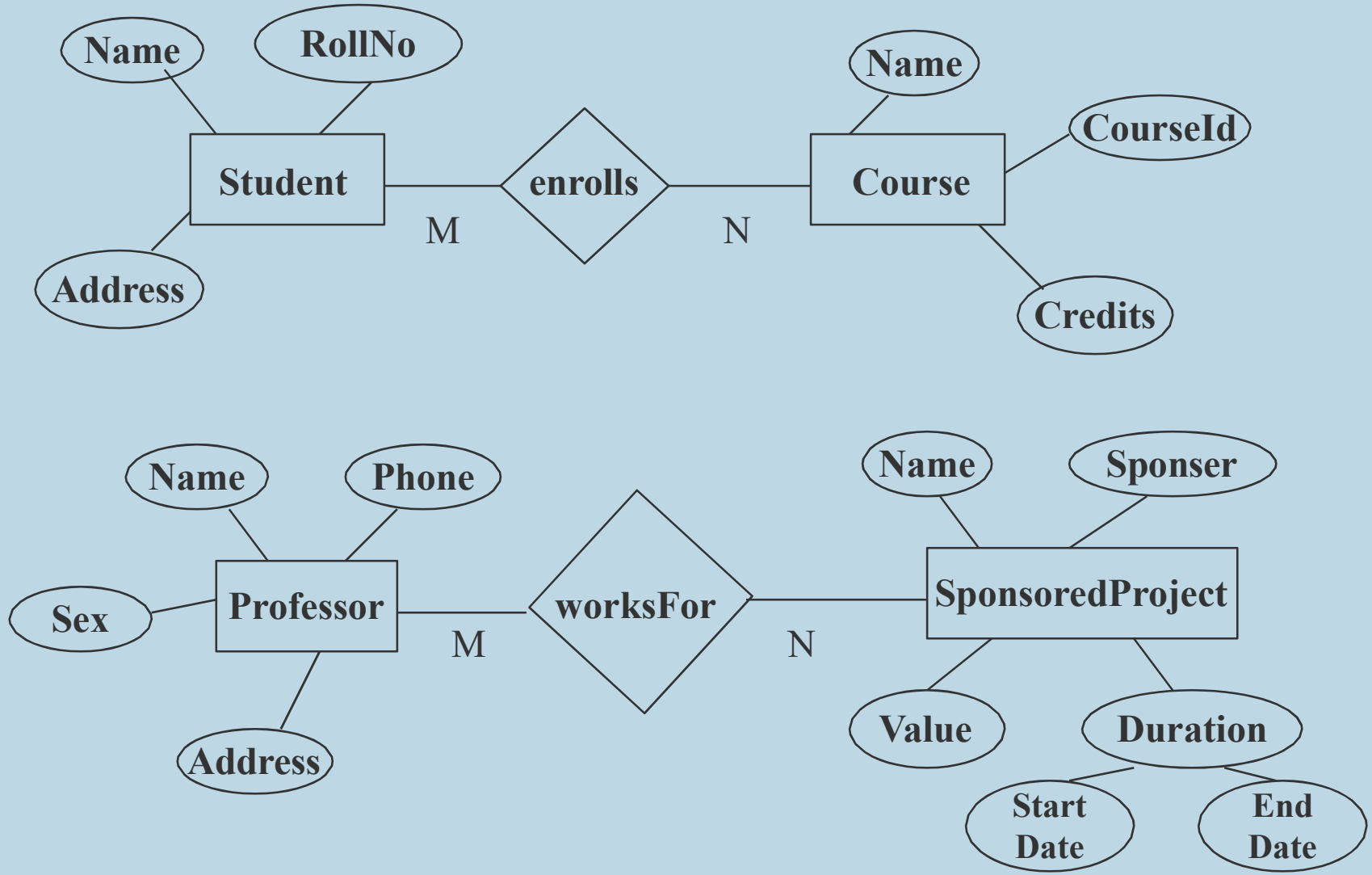
Cardinality Ratio – example (*one-to-one*)



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



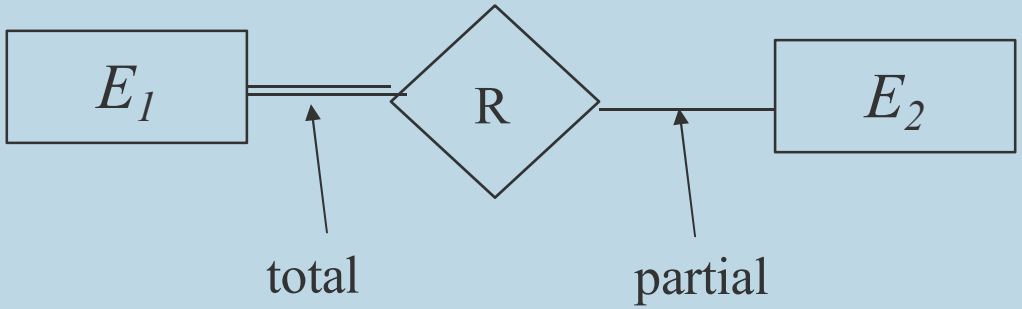
Cardinality Ratio – example (*many-to-many*)



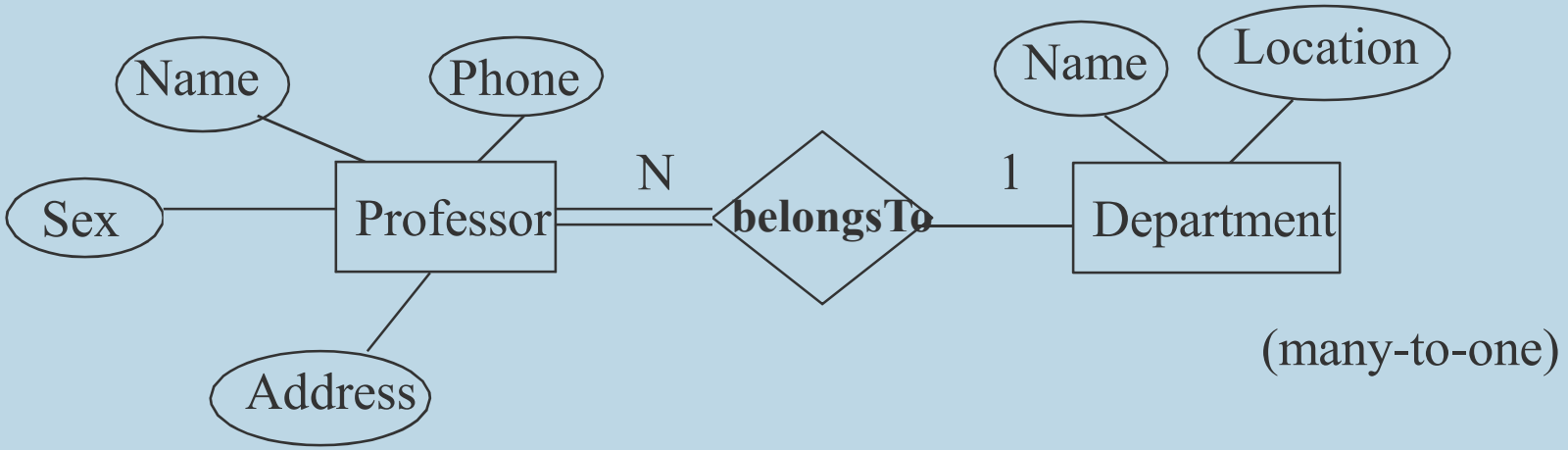
Participation Constraints

- 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.

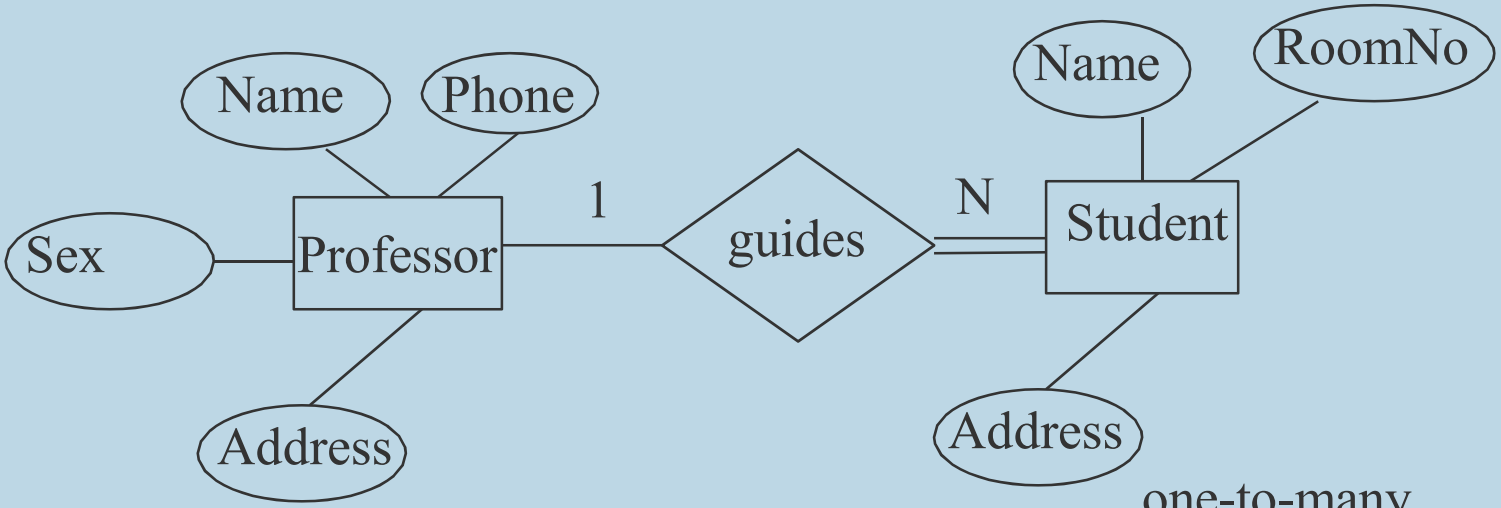
Notation:



Example of total/partial Participation



(many-to-one)

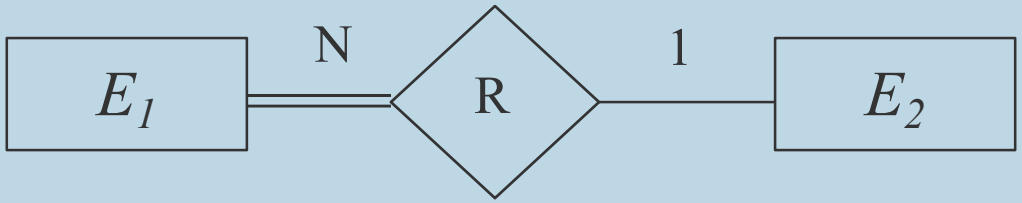


one-to-many

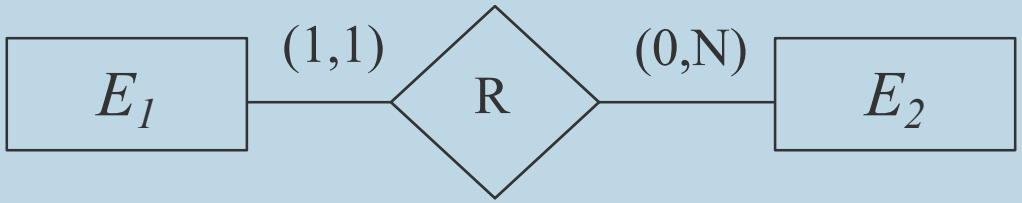
Structural Constraints

- 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*: similarly, the maximum number of times a particular entity *can appear* in the relationship tuples at any point of time

Comparing the Notations



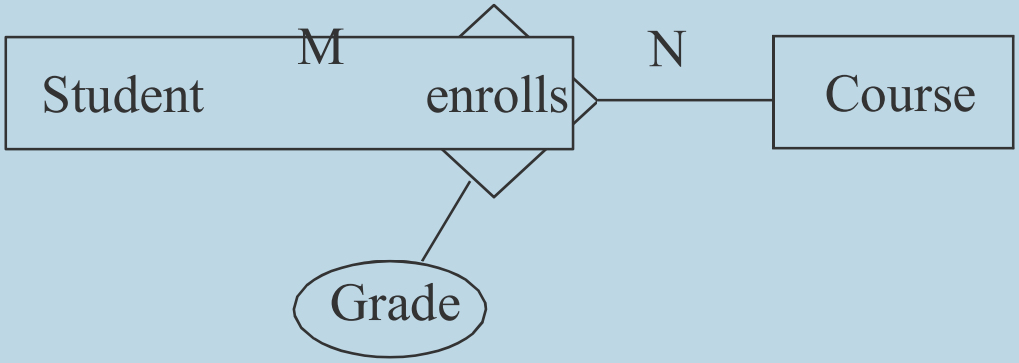
is equivalent to



Attributes for Relationship Types

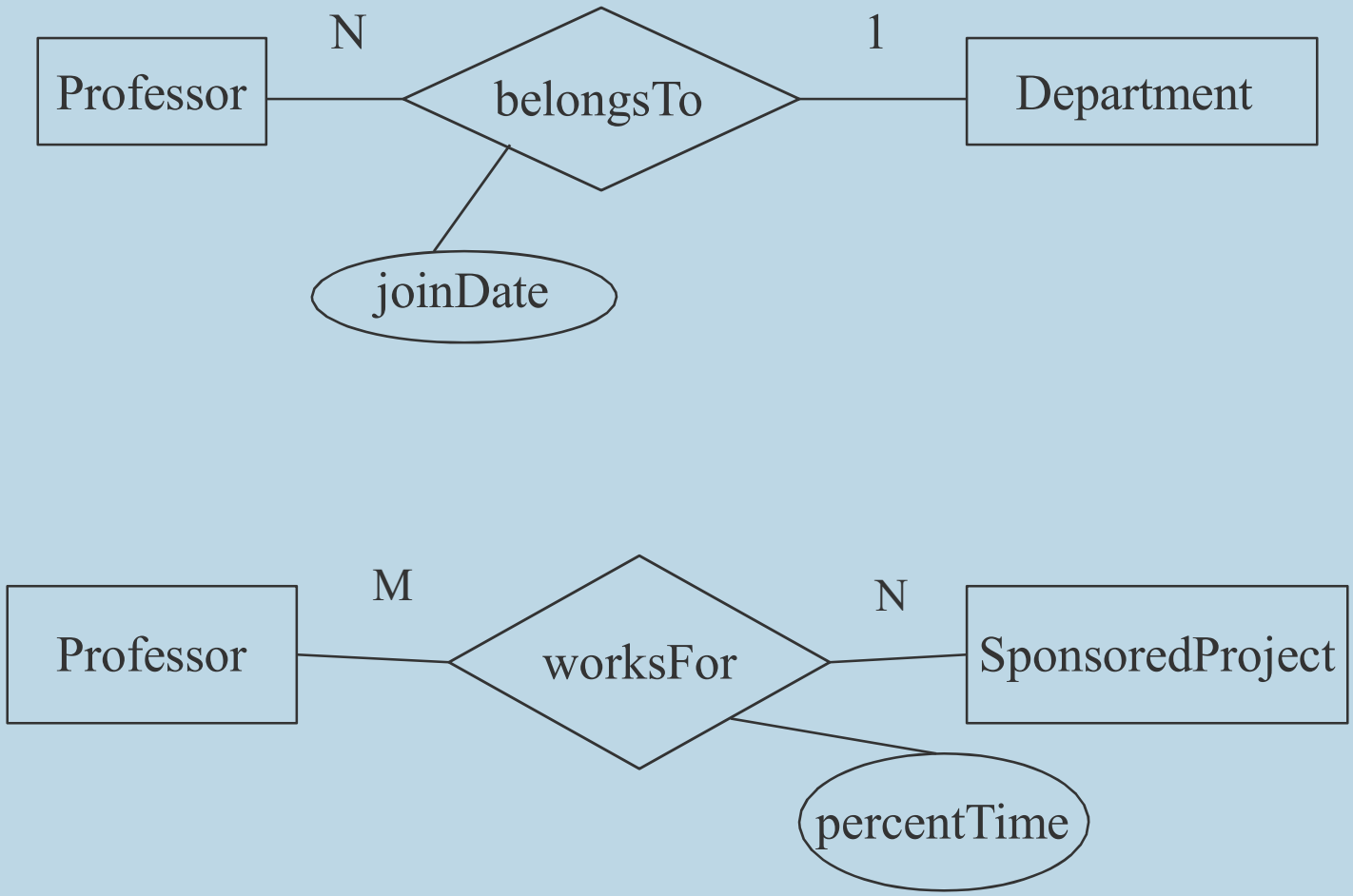
Relationship types can also have attributes.

- properties of the association of entities.



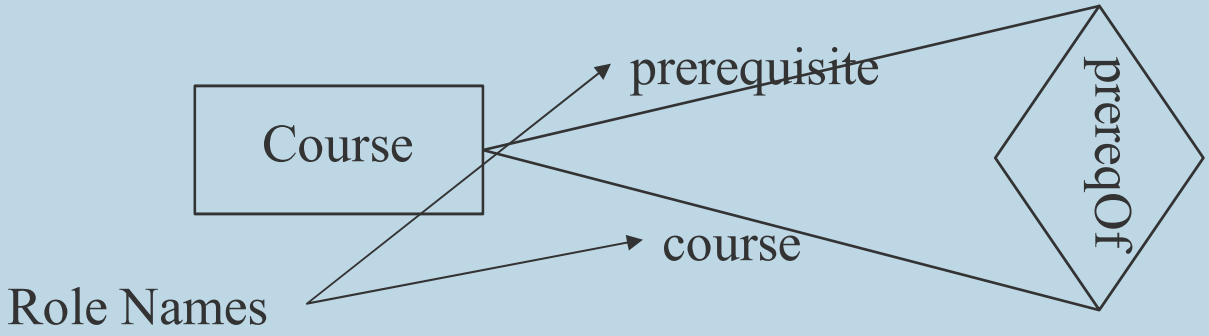
- *grade* gives the letter grade (S,A,B, etc.) earned by the student for a course.
  - neither an attribute of *student* nor that of *course*.

Attributes for Relationship Types – More Examples



Recursive Relationships and Role Names

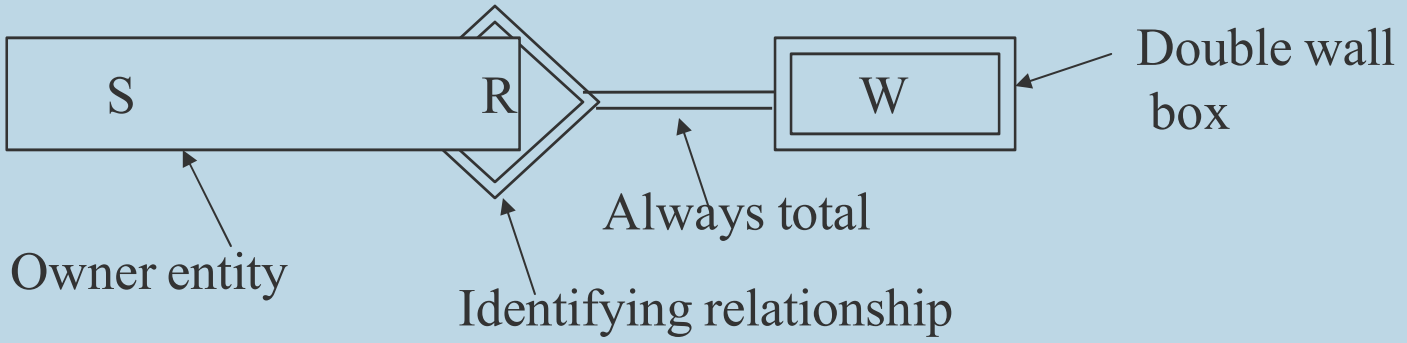
- Recursive relationship: An entity set relating to itself gives rise to a *recursive* relationship
- E.g., the relationship *prereqOf* is an example of a recursive relationship on the entity *Course*
- Role Names – used to specify the exact role in which the entity participates in the relationships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



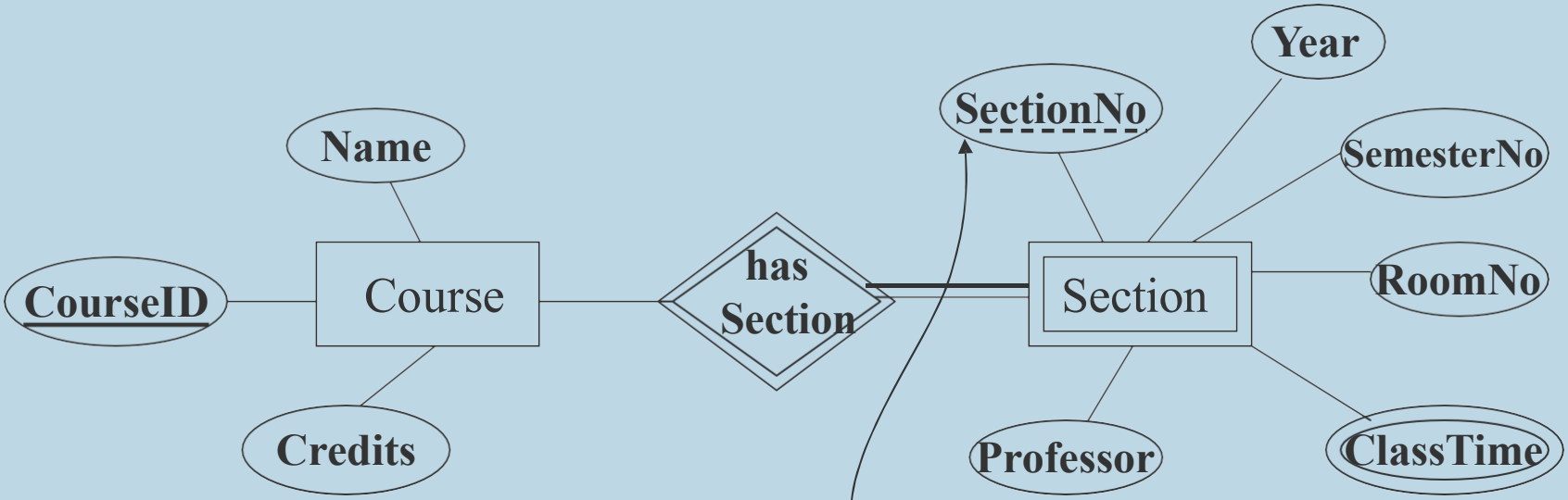
Weak Entity Sets

Weak Entity Set: An entity set whose members owe their existence to some entity in a *strong entity set.*

- entities are not of independent existence.
- each weak entity is associated with some entity of the *owner* entity set through a special relationship.
- weak entity set may not have a key attribute.



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

Complete Example for E/R schema: Specifications (1/2)

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.

We like to 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.

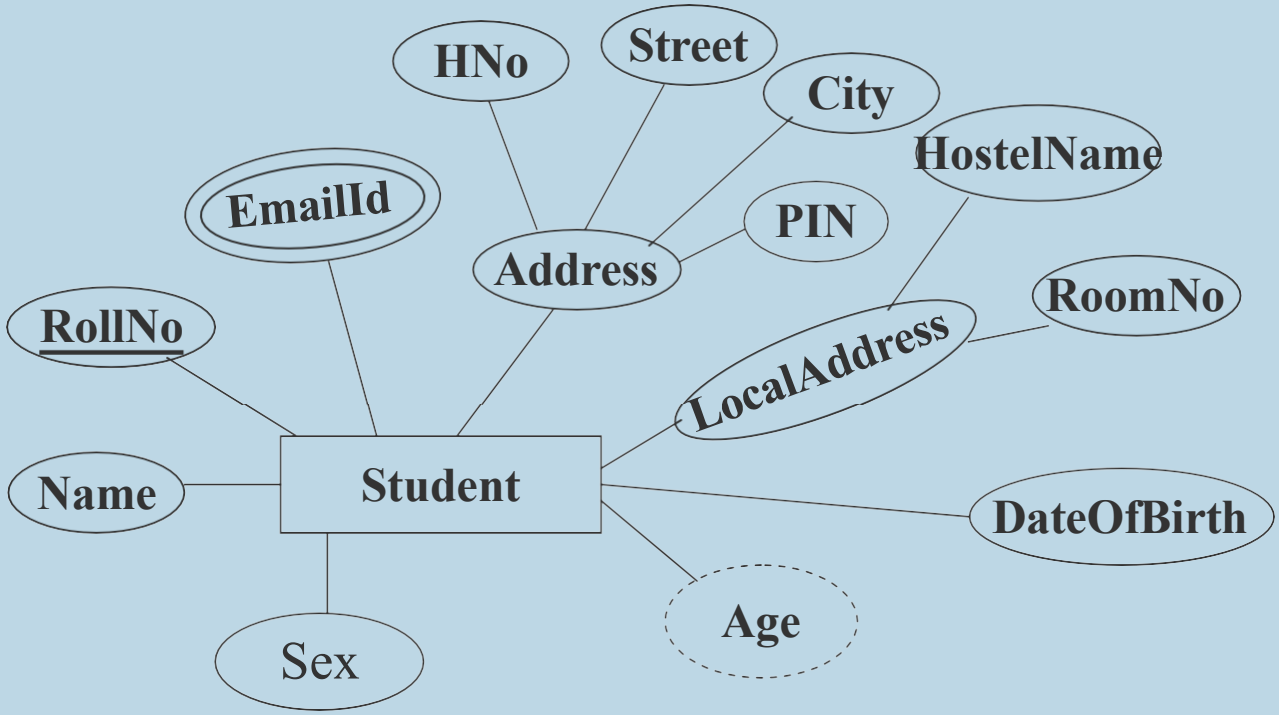
Complete Example for E/R schema: Specifications (2/2)

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.

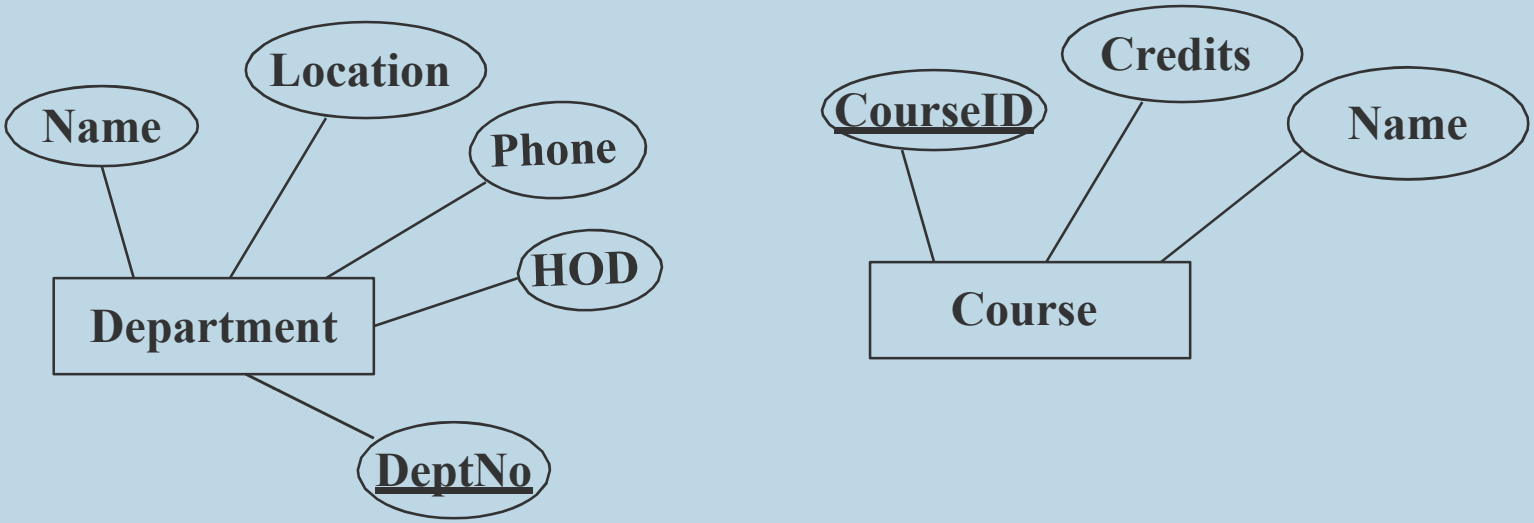
Each course has a name, number of credits and the department that offers it. A course may have other courses as pre-requisites i.e, courses to be completed before it can be enrolled in.

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*.

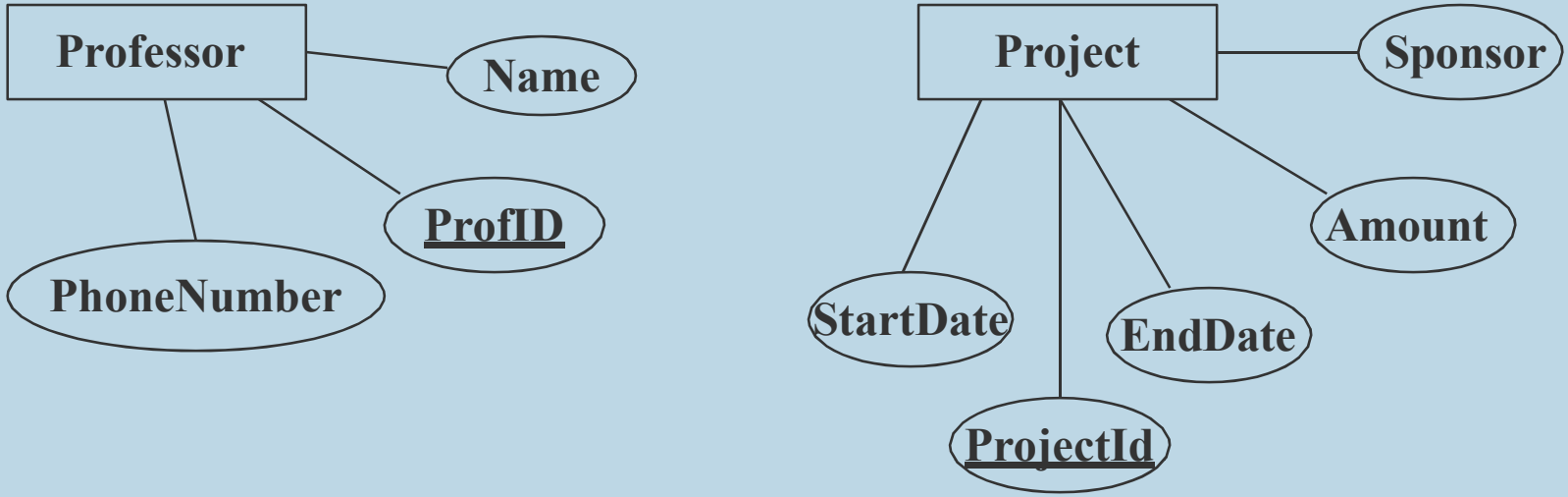
Entities - Student

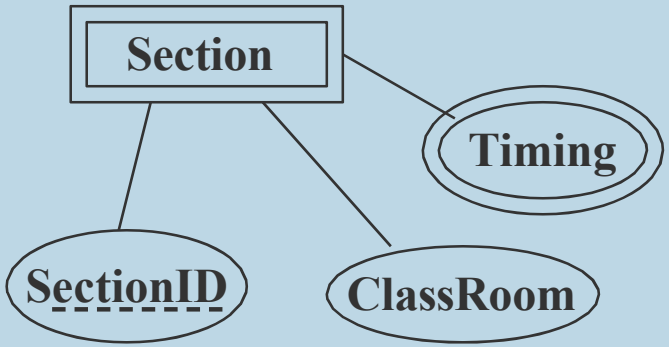


Entities – Department and Course

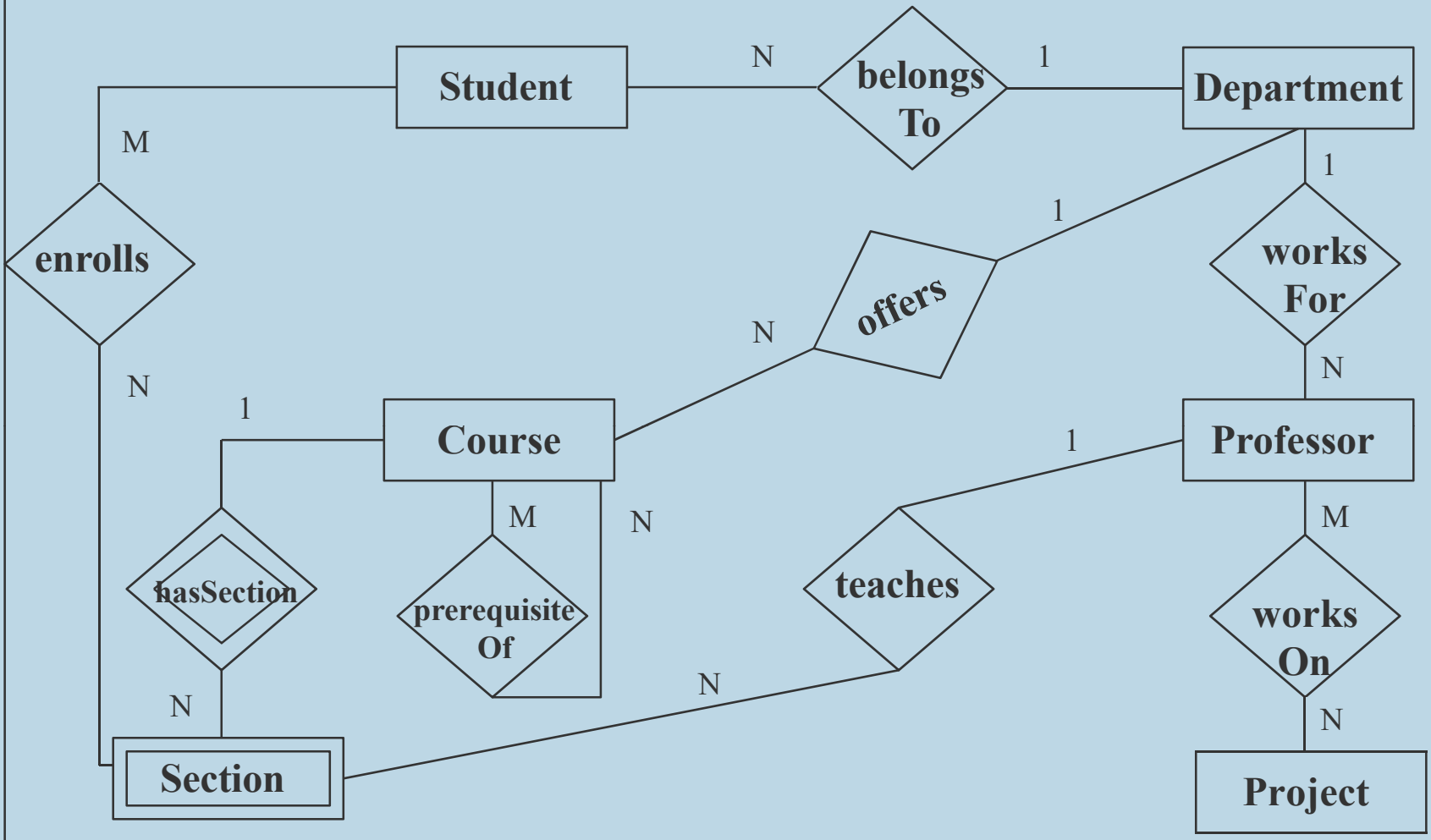


Entities – Professor, Project and Sections





E/R Diagram showing relationships



Design Choices: Attribute versus Relationship

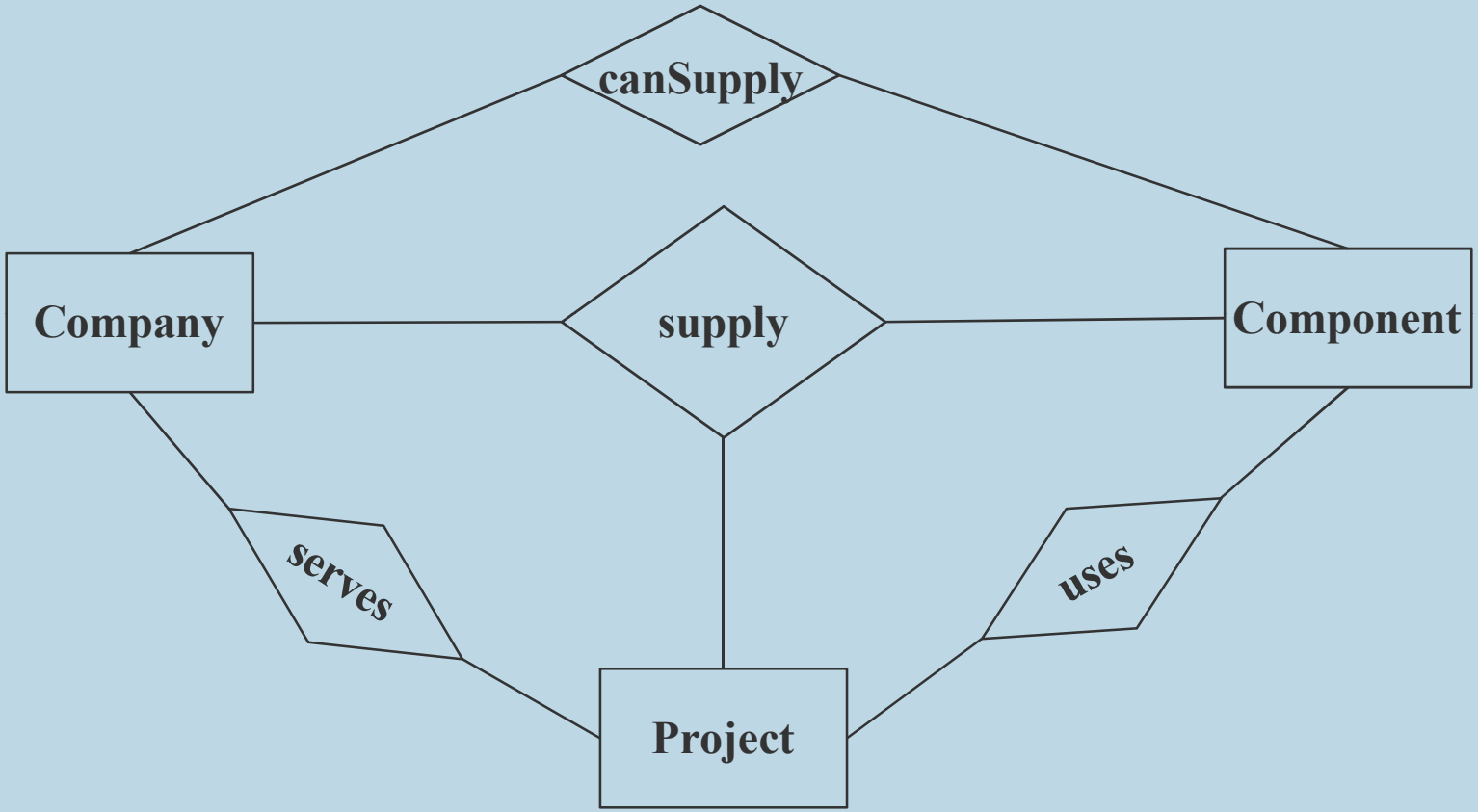
- Should *offering department* be an attribute of a course or should we create a relationship between Course and Dept entities called, say, *offers* ?
  - Later approach is preferable when the necessary entity, in this case the Department, already exists.
- Should *class room* be an attribute of Section or should we create an entity called ClassRoom and have a relationship, say, meetsIn, connecting Section and ClassRoom?
  - In this case, the option of making classRoom as an attribute of Section is better as we do not want to give a lot of importance to class room and make it a an *entity*.

Design Choices:
Weak entity versus composite multi-valued attributes

- Note that *section* could be a composite multi-valued *attribute* of Course entity.
  - However, if so, *section* can not participate in relationships, such as, *enrolls* with Student entity.
- In general, if a thing, even though not of independent existence, participates in other relationships on its own, it is best captured as a *weak entity.*
  - If the above is not the case, composite multi-valued attribute may be enough.

Ternary Relationships

Relationship instance (c, p, j) indicates that
company c supplies a component p that is made use of by the project j



Ternary Relationships

(c,p) in *canSupply*, (j,p) in *uses*, (c,j) in *serves* may not together imply (c,p,j) is in *supply*. Whereas the other way round is of course true.