Database Management System (MDS 505)

Jagdish Bhatta

Unit-1

Fundamental Concept of DBMS:ER Model

High-Level Conceptual Data Models for Database Design

- **Precise Database:**
- Need to come up with a methodology to ensure that each of the relations in the database is “good”, “complete” and “precise”.
- Two ways of doing so:
  - Entity Relationship Model
    - Models an enterprise as a collection of *entities* and *relationships*
    - Represented diagrammatically by an *entity-relationship diagram:*
  - Normalization Theory
    - Formalize what designs are bad, and test for them

High-Level Conceptual Data Models for Database Design

- **Database Design Considerations:**
  - Data Constraints and Relational Database Design
  - Usage Requirements: Queries, Performance
    - Throughput, Response Time
  - Authorization Requirements
  - Data Flow

Database Design Considerations

- **Be aware about avoiding:**
  - Redundancy
  - Incompleteness

Entities Relationship Model

- The ***entity-relationship (E-R) model*** is a high level data model based on a perception of a real world that consists of collection of basic objects, called ***entities***, and of ***relationships*** among these entities. An ***entity*** is a thing or object in the real world that is distinguishable from other objects.

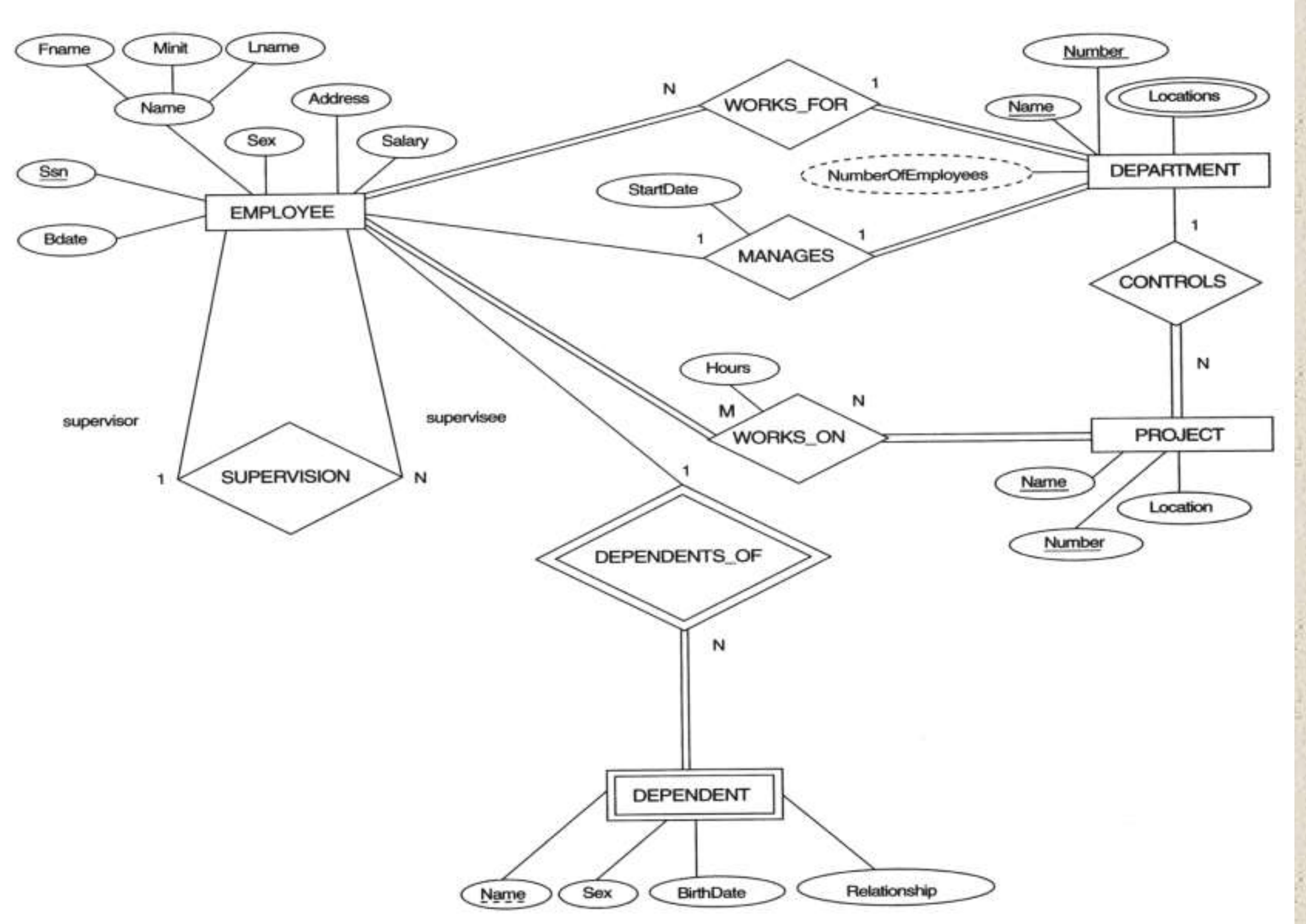
  Entities are described in a database by a set of ***attributes***. A ***relationship*** is an association among several entities. The set of all entities of the same type is called an ***entity set*** and the set of all relationships of the same type is called a ***relationship set***.

- ***ER Model is logical modeling of a database.***

Entities and Attributes

- ***Entity*** is a thing in real world with and independent existence that is distinguishable from all other objects. An Entity may be an object with physical existence like person, vehicle etc. or may be an object with conceptual existence like account, course etc.
- Each entities have certain kind of properties that gives proper identification to each of entity. Thus, ***attributes*** are the properties that describe the entity. Each entity has value for each of its attributes. Thus, for each attribute there is a set of permitted values known as ***Domain***. Hence attribute of an entity set can be defined as a function that maps from the entity set to a domain. Example:
  - **Entity:- Employee**
  - **Attributes:- Employee_id, Name, Salary, Age, Phone**
  - **An instance of entity Employee will be defined once certain values will be assigned to the given attributes.**

ER Diagram: An Example



Attribute Types…..

- Attributes can be of following types;
  - Simple vs. Composite
  - Single Valued vs. Multi-valued
  - Stored vs. Derived
  - Null Valued
  - Complex
- Simple Vs. Composite:-
  - Simple are those that are not divisible. They are also known as “atomic attributes”.
  - Composite attributes are those that are made up of one or more simple or composite attributes & hence are divisible to smaller subparts. Composite attributes can form a hierarchy.
  - Eg: Address can be the composite attribute as it can be subdivided into Streetaddress, City, State.

    Similarly, Name can be divided as Fname, Lname. But Fname, Lname themselves are simple.

Attribute Types…..

- **Single Valued vs. Multi-valued:-**
  - Single valued are those having just a single value, for a particular entity, from the permitted domain.
  - Multi-valued are those that can have a set of values for same entity.
  - Eg: Roll_no, age can be considered as single valued. While Phone_no, College_degree can have multiple values, so are multivalued.
- **Stored vs. Derived:-**
  - An attribute whose value can be derived from or inferred from the value of another attribute, then it is derived otherwise it is stored. For example, Age attribute is derived as it can be inferred from the Date_of_birth attribute which is stored.
- **Null Valued:-**
  - Are those attributes that do not have any applicable values. The null values can also be used if we do not know the value of an attribute for a particular entity.

Attribute Types…..

- **Complex:-**
  - Composite and multivalued attributes can be nested arbitrarily. We can represent arbitrary nesting by grouping components of a composite attribute between parentheses ( ) and separating the components with commas, and by displaying multivalued attributes between braces { }. Such attributes are called **complex attributes**. For example, if a person can have more than one residence and each residence can have a single address and multiple phones, an attribute Address_phone for a person. Both phone and address themselves are composite attributes.

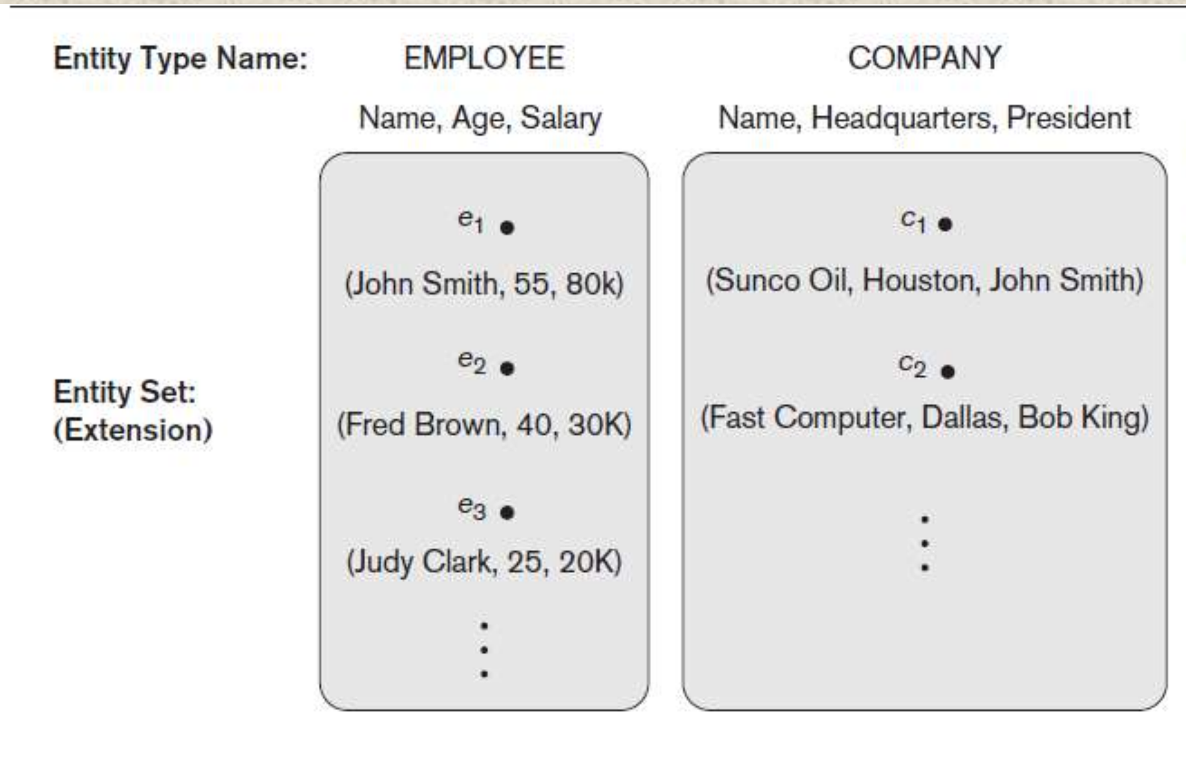
{Address_phone ({Phone(Area_code,Phone_number)},
Address(Street_address (Number,Street,Apartment_number),City,State,Zip))}

Entities types and Entity sets

- **Entity Types:** Entity type defines a collection of entities that have same attributes, but each entity has its own value for each attributes. An entity type describes the schema for a set of entities that share same structure. Each entity type in the database is described by its name and attributes. An entity type describes the **schema** or **intension** for a *set of entities* that share the same structure.
- **Entity Sets:** An entity set is a set of entities of the same type that share the same properties, or attributes i.e. the collection of all the entities of particular type in the database at any instance is called the **entity set or entity collection**. The entity set is usually referred to using the same name as entity type.
- In the process of modeling, we often use the term *entity set in the abstract*, without referring to a particular set of individual entities. We use the term **extension** of the entity set to refer to the actual collection of entities belonging to the entity set. The actual instances of employee form extension of the entity employee.

Entities types and Entity sets

- **Example:**



Entity Type Name: EMPLOYEE COMPANY

Name, Age, Salary

Name, Headquarters, President

Entity Set: (Extension)

e_1 ●

(John Smith, 55, 80k)

e_2 ●

(Fred Brown, 40, 30K)

e_3 ●

(Judy Clark, 25, 20K)

⋮

c_1 ●

(Sunco Oil, Houston, John Smith)

c_2 ●

(Fast Computer, Dallas, Bob King)

⋮

Figure 3.6
Two entity types, EMPLOYEE and COMPANY, and some member entities of each.

e4

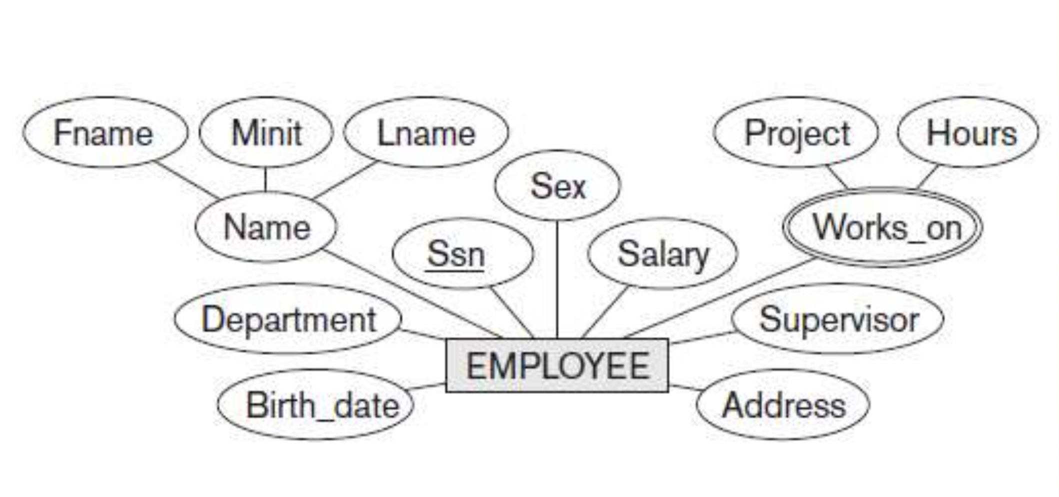
- **(John Smith, 55, 80k}**

Key Attributes of Entity Types

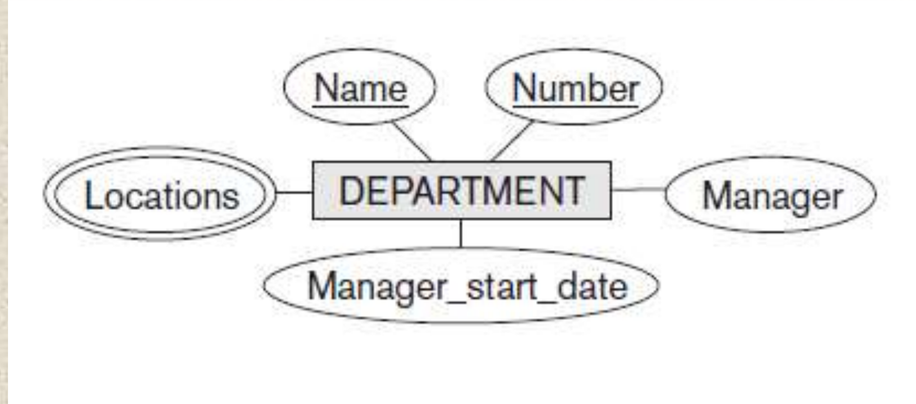
- An important constraint on the entities of an entity type is the **key** or **uniqueness constraint** on attributes. An entity type usually has one or more attributes whose values are distinct for each individual entity in the entity set. Such an attribute is called a **key attribute**, and its values can be used to identify each entity uniquely.
- Sometimes several attributes together form a key, meaning that the *combination* of the attribute values must be distinct for each entity. If a set of attributes possesses this property, the proper way to represent this in the ER model that we describe here is to define a *composite attribute* and designate it as a key attribute of the entity type. Notice that such a composite key must be *minimal*; that is, all component attributes must be included in the composite attribute to have the uniqueness property.

Keys Attributes of Entity Types

- In the entity employee, Ssn is the primary key attribute.

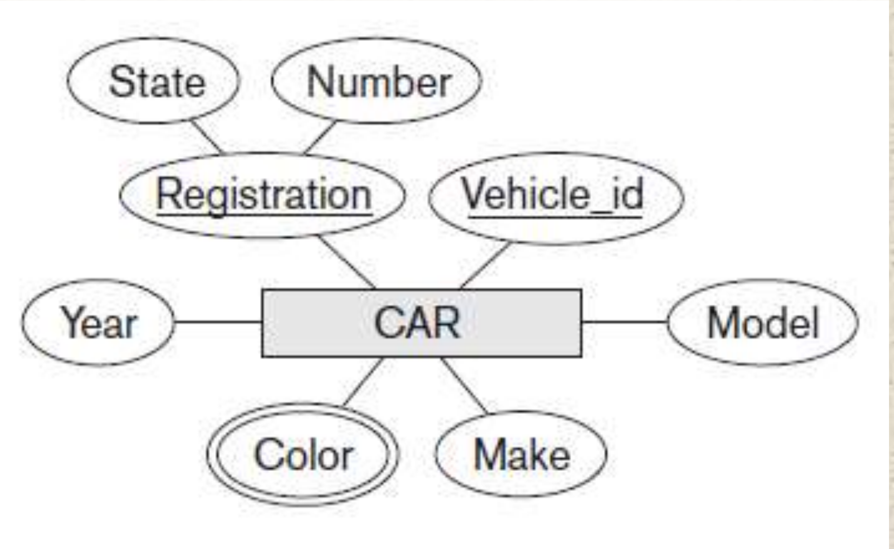


- In the entity Department, both Name and Number are the primary key attributes.

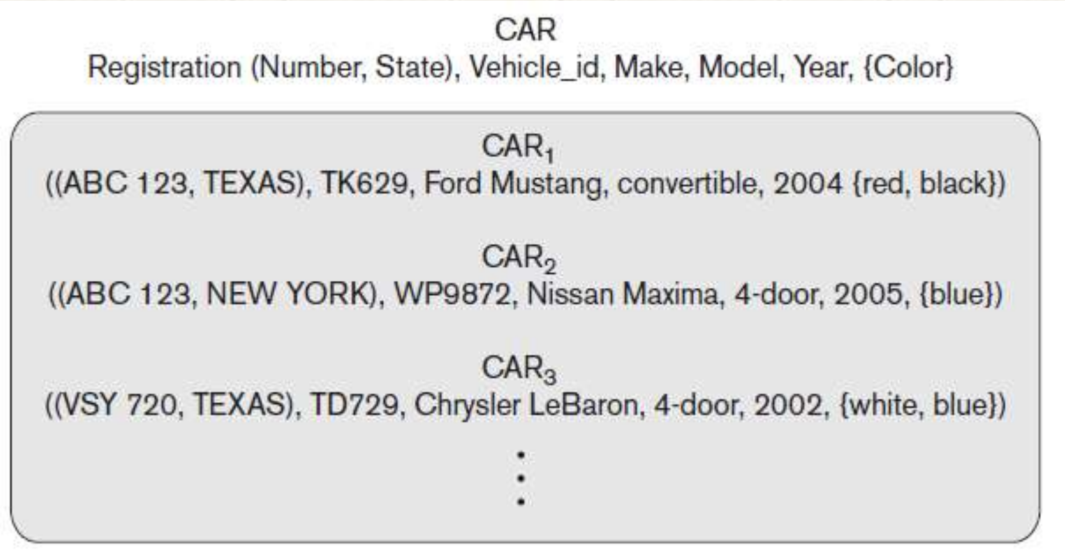


Keys Attributes of Entity Types

- In the entity Car, registration and vehichle_id are the primary key attributes. The Registration attribute is an example of a composite key formed from two simple component attributes, State and Number, neither of which is a key on its own.



The CAR entity type



CAR
Registration (Number, State), Vehicle_id, Make, Model, Year, {Color}

CAR_1
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR_2
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

CAR_3
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

⋮

CAR entity set with three entities.

Value Sets (Domains) of Attributes

- **Value Sets (Domains) of Attributes.** Each simple attribute of an entity type is associated with a **value set** (or **domain** of values), which specifies the set of values that may be assigned to that attribute for each individual entity. Value sets are not typically displayed in basic ER diagrams and are similar to the basic **data types** available in most programming languages, such as integer, string, Boolean, float, and so on.
- Mathematically, an attribute A of entity set E whose value set is V can be defined as a **function** from E to the power set $P(V)$ of V:

$$A : E \rightarrow P(V)$$

Value Sets (Domains) of Attributes

- In Figure 3.6, if the range of ages allowed for employees is between 16 and 70, we can specify the value set of the Age attribute of EMPLOYEE to be the set of integer numbers between 16 and 70. Similarly, we can specify the value set for the Name attribute to be the set of strings of alphabetic characters separated by blank characters, and so on.

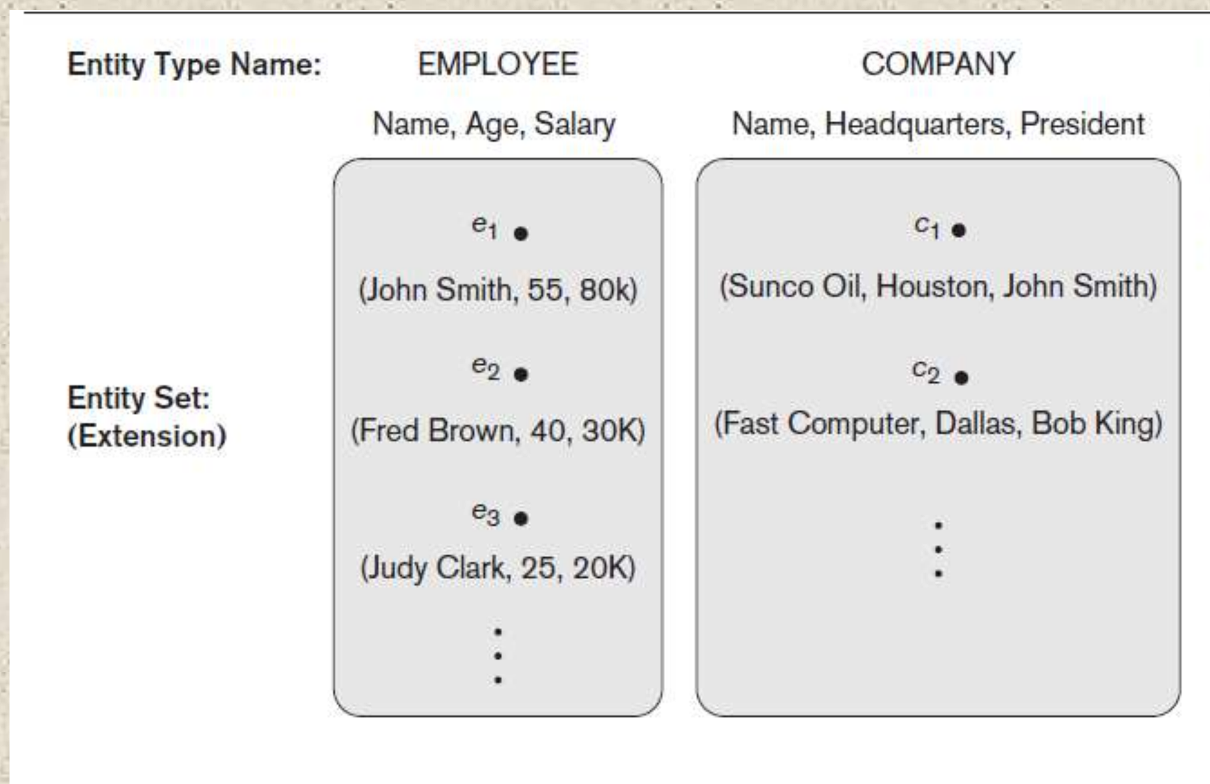


Figure 3.6
Two entity types, EMPLOYEE and COMPANY, and some member entities of each.

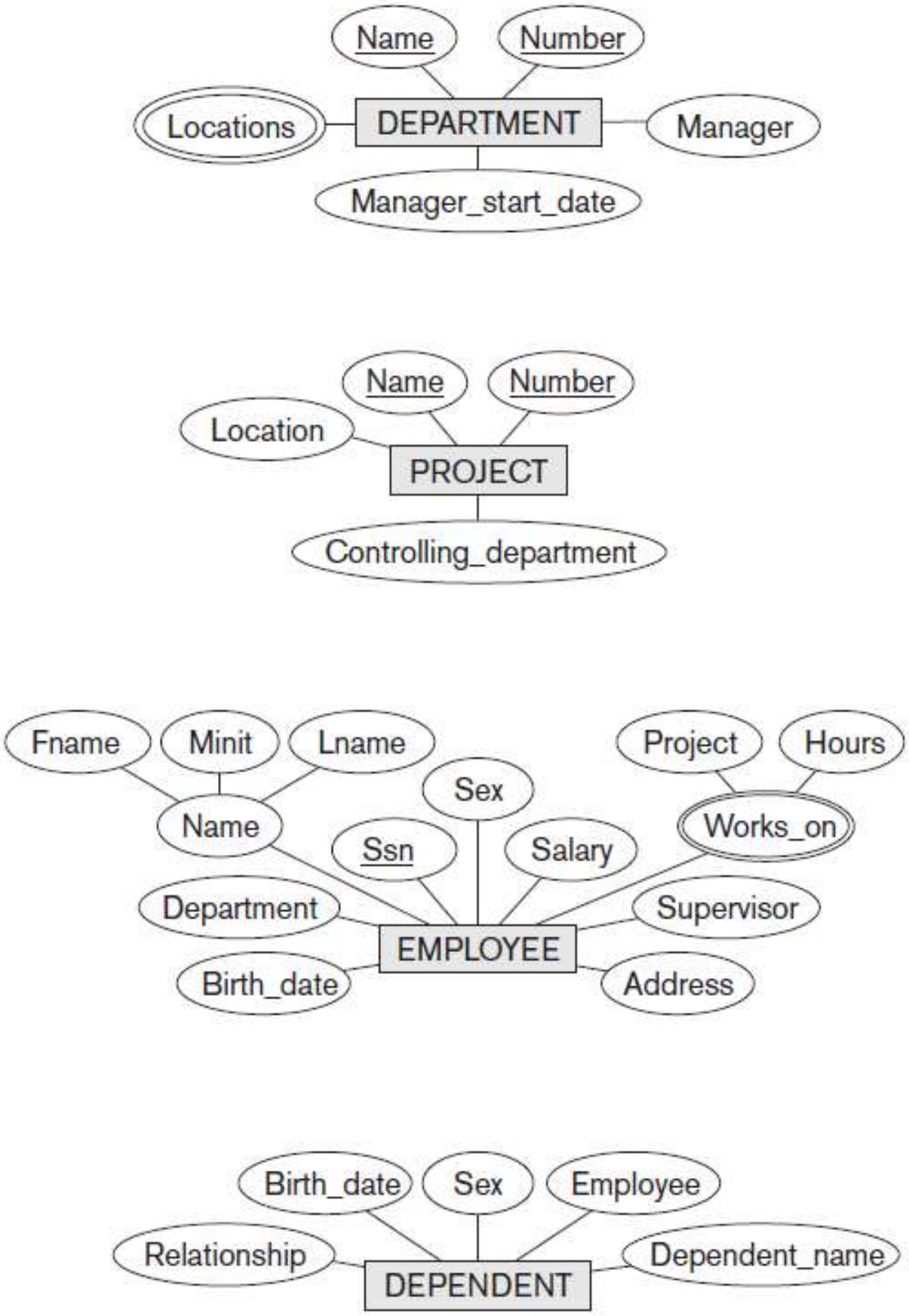
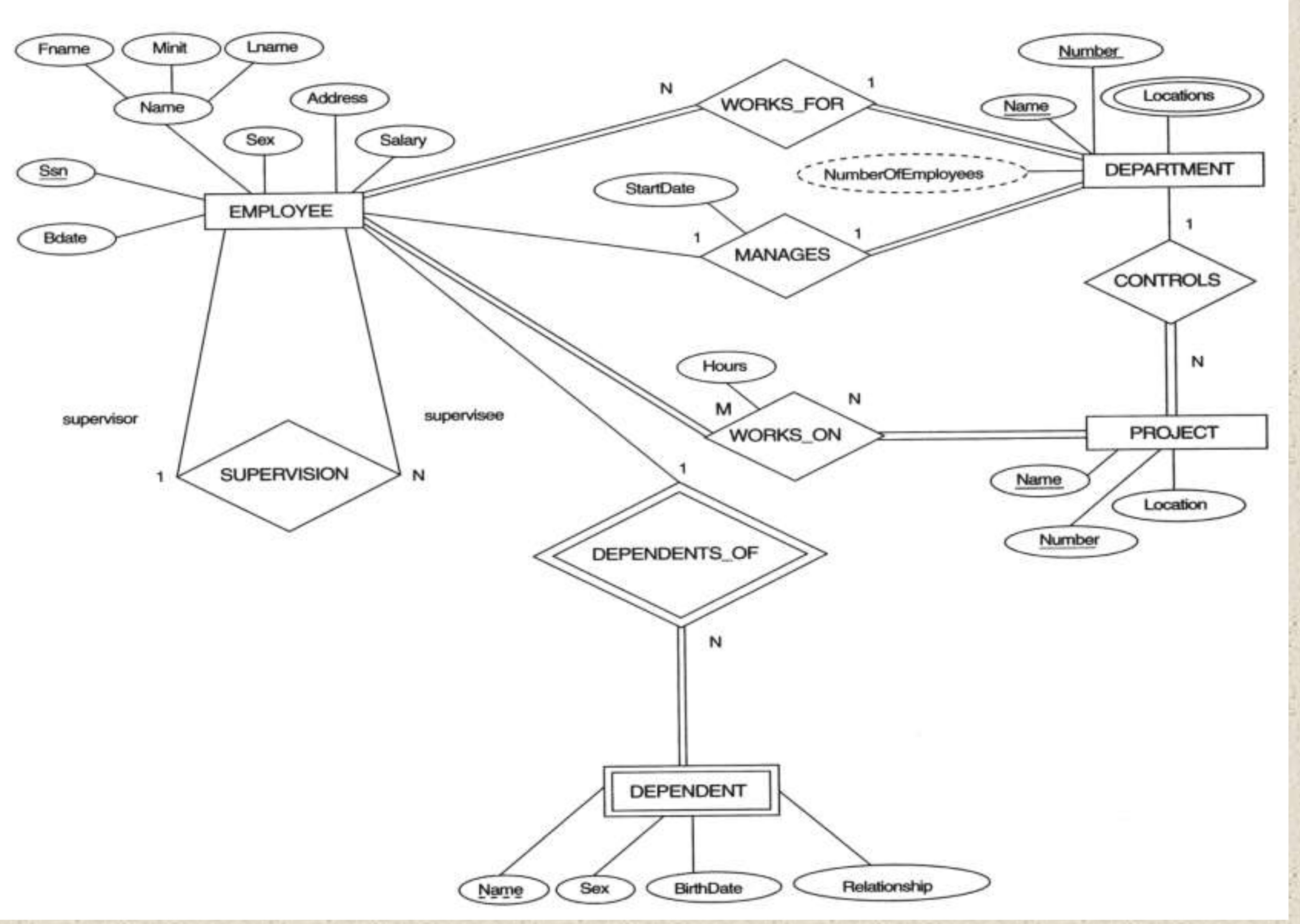


Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

ER Diagram: An Example



Relationship Types, Relationship Sets, Roles

- **Relationship** is an association among several entities.

  Relationships can be thought of as *verbs*, linking two or more nouns. Examples: a teaches relationship between a teacher and a student, a *supervises* relationship between an employee and a department etc.

Relationship Types, Relationship Sets, Roles

- A **relationship type** *R* among *n* entity types *E*1, *E*2, . . . , *En* defines a set of associations— or a **relationship set**—among entities from these entity types. Similar to the case of entity types and entity sets, a relationship type and its corresponding relationship set are customarily referred to by the *same name*, *R*. Mathematically, the relationship set *R* is a set of **relationship instances** *ri*, where each *ri* associates *n* individual entities (*e*1, *e*2, . . . , *en*), and each entity *ej* in *ri* is a member of entity set *Ej*, 1 ≤ *j* ≤ *n*. Hence, a relationship set is a mathematical relation on *E*1, *E*2, . . . , *En*; alternatively, it can be defined as a subset of the Cartesian product of the entity sets *E*1 × *E*2 × . . . × *En*. Each of the entity types *E*1, *E*2, . . . , *En* is said to **participate** in the relationship type *R*; similarly, each of the individual entities *e*1, *e*2, . . . , *en* is said to **participate** in the relationship instance *ri* = (*e*1, *e*2, . . . , *en*).

Relationship Types, Relationship Sets, Roles

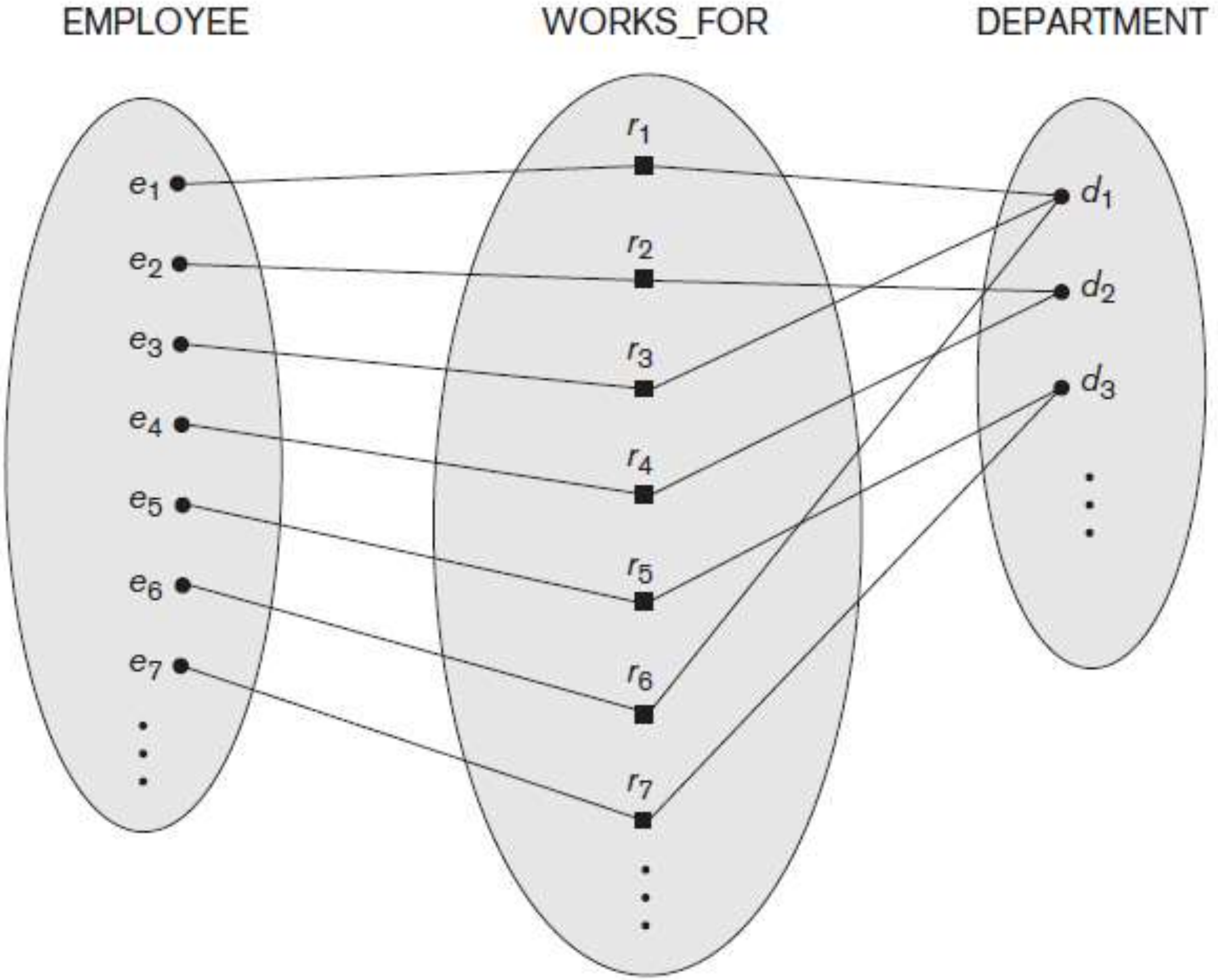


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Relationship Types, Relationship Sets, Roles

- **Degree of relationship type:-** The **degree of relationship type** is the number of participating entity types. Relationships can generally be of any degree, but the most common ones are binary relationships. i.e. having two participating entity types.

  having three participating entity types - Ternary relationship

  having n participating entity types – n-ary relationship
- The function that an entity plays in a relationship is called its **role**. **Roles** are normally explicit and not specified. They are useful when the meaning of a relationship set needs clarification. The **role name** signifies the role that a participating entity from the entity type plays in each relationship instance, and it helps to explain what the relationship means.
- When some entity type participates more than once in a relationship type in different roles, then we term it as a **recursive relationship or Self-Referencing**.

Relationship Types, Relationship Sets, Roles

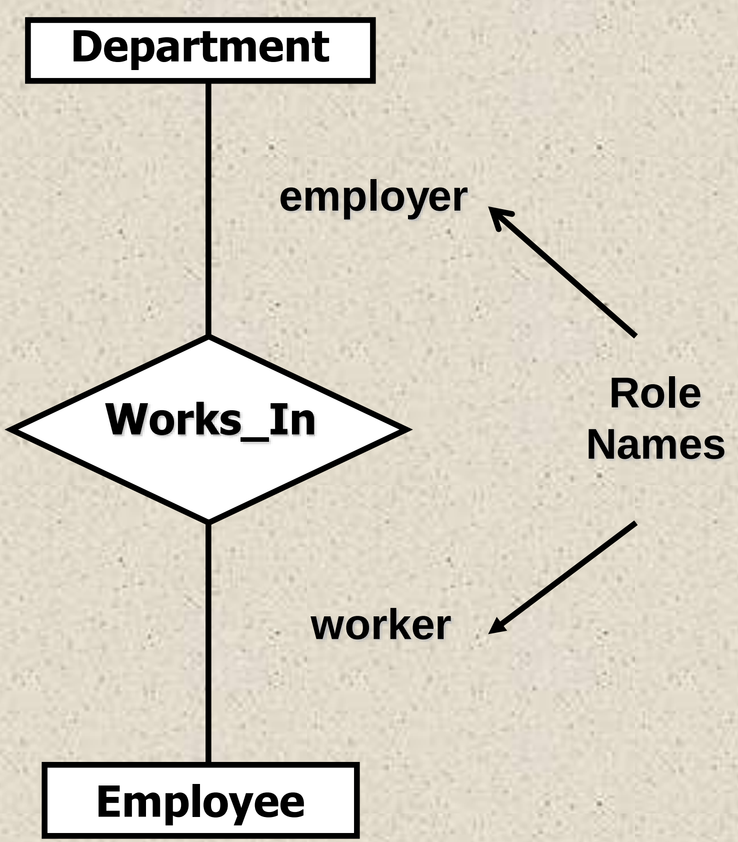


Fig 1: Role Names

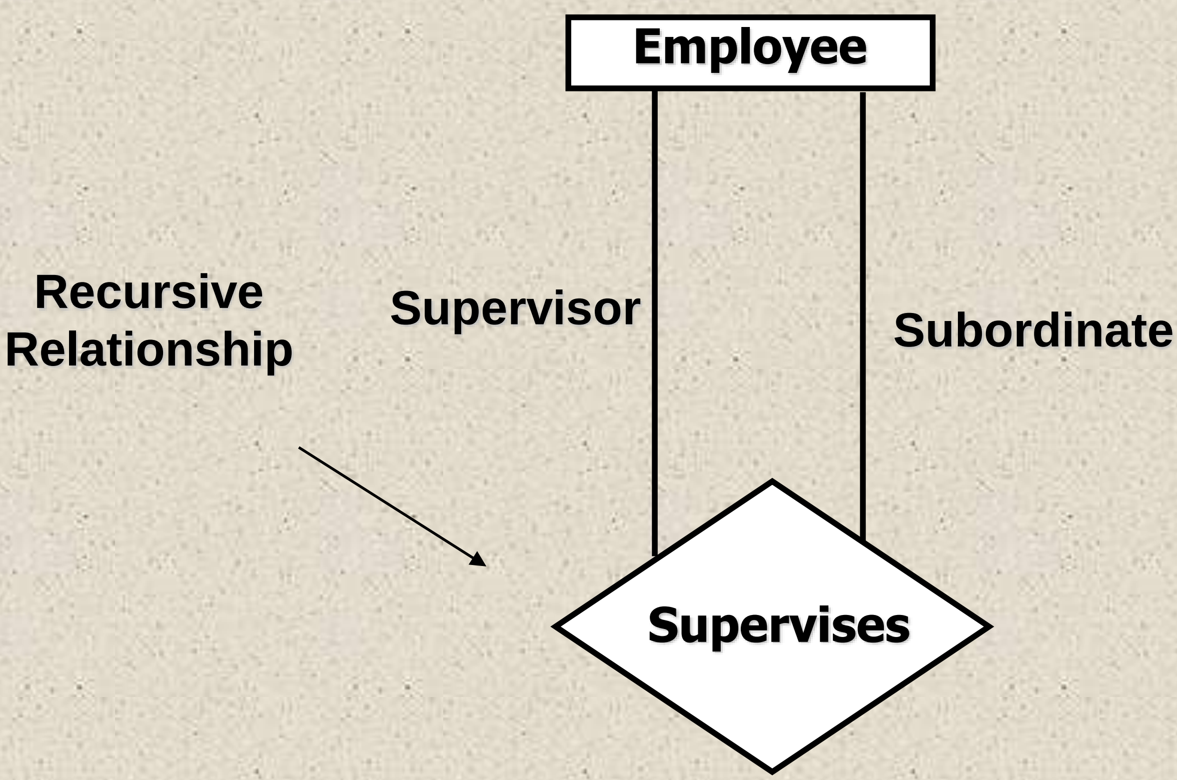


Fig 2: Recursive Relationship

The relationship in Fig. 1 has degree 2, so a binary relationship. While in Fig. 2 the degree is 1, so a unary relationship.

Constraints on Binary Relationship Types (Mapping / Structural Constraints)

- Relationship types have certain constraints that limit the set of possible combination of entities that may participate in the relationship type. There are mainly two types of constraints;
  - Cardinality Ratio
  - Participation Constraint
- **Cardinality Ratio**:- It specifies the **maximum number** of relationship instances that an entity can participate in.

  In other words, it defines how many entities of an entity set participate in a relationship type.

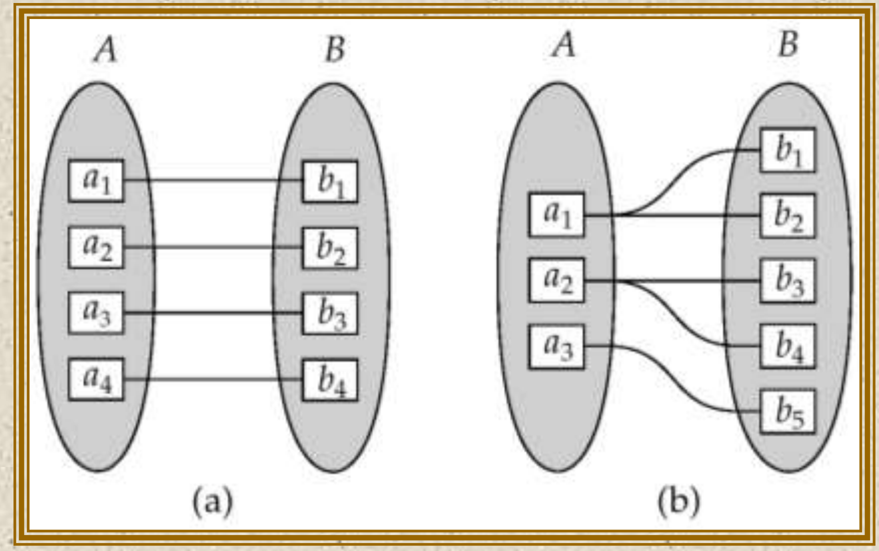
  Or, it express the number of entities to which another entity can be associated via a relationship set.
- Possible cardinality ratios for binary relationship types are:

  - One to one; One to many; Many to one; Many to many

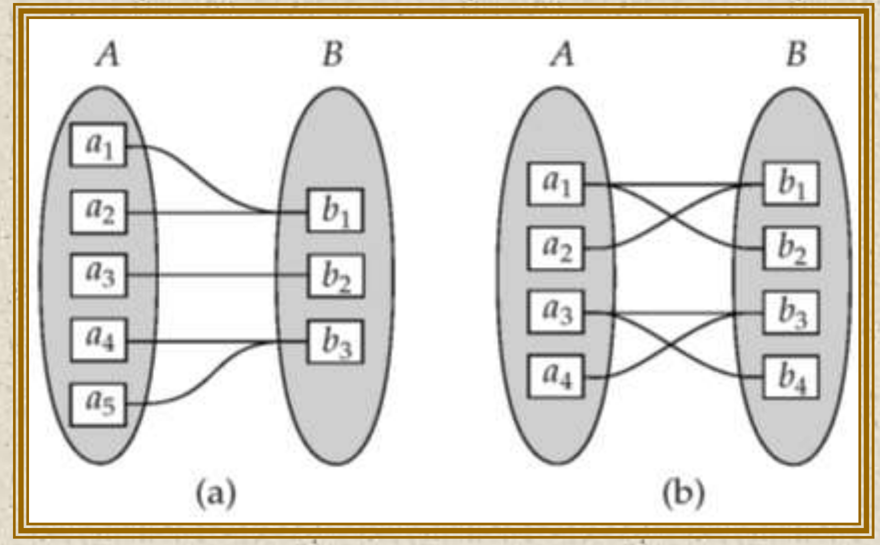
Mapping Constraints (Contd…..)

- **If R is a relationship set between two entity sets A and B, then cardinality ratio can be defined as;**
- **One to One**:- An entity in A is associated with at most one entity in B and an entity in B is associated with at most one entity in A.
- **One to Many**:- An entity in A is associated with any number of entities in B, while an entity in B can be associated with at most one entity in A.
- **Many to One**:- An entity in A is associated with at most one entity in B, while an entity in B can be associated with any number of entities in A.
- **Many to Many**:- An entity in A can be associated with any number of entities in B and an entity in B can be associated with any number of entities in A.

Mapping Constraints (Contd…..)



One to one One to many



Many to one Many to many

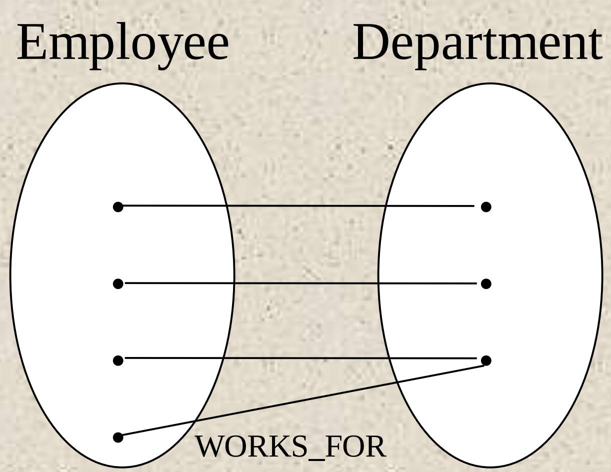
- EXAMPLES :
  - relationship set HAS from Department to Chairperson is one-to-one.
  - relationship set TEACHES from Teacher to Student is one-to-many & TAUGHT_BY from Student to Teacher is many-to-one.
  - relationship set STUDY from Student to Course is many-to-many .

Constraints on Binary Relationship Types (Mapping/Structural Constraints)

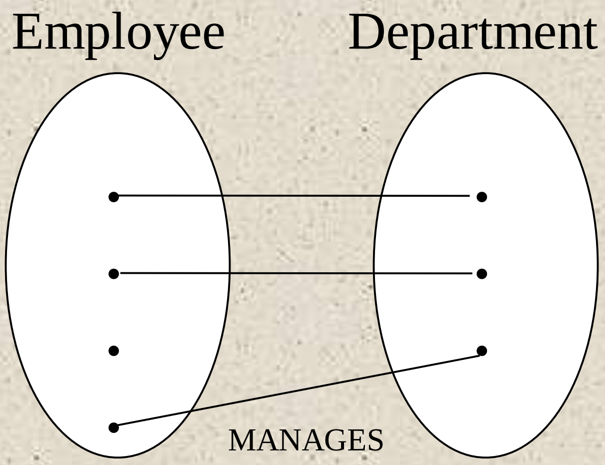
- **Participation Constraint:-** It describes the whether existence of an entity depends on another entity to which it is related through a relationship type
- It specifies the minimum number of relationship instances that each entity can participate in, and is sometimes called the **minimum cardinality constraint**. It may be;
  - Total Participation(Existence Dependency)
  - Partial participation
- **Total participation**:- The participation of an entity set E in a relationship set R is said to be total if each entity in E participates in at least one relationship in R i.e. each entity depends upon existence of the related entity.
- Consider Employee and Department entity sets, and a relationship WORKS_FOR between them indicating each employee must work for a department. Then there is total participation of entities from Employee entity set.

Constraints on Relationship Types (Mapping Constraints)

- **Partial participation**:- The participation of an entity set E in a relationship set R is said to be partial if only some entities in E participate in the relationships in R.
- Consider Employee and Department entity sets, and a relationship MANAGES between them indicating some employees manages department. Then there is partial participation of entities from Employee entity set.



Total Participation



Partial Participation

Attributes of Relationship Types

- Relationship types can also have attributes, similar to those of entity types and are often known as **descriptive attributes**. For example, to record the number of hours per week that a particular employee works on a particular project, we can include an attribute Hours for the WORKS_ON relationship type in Figure 3.13.

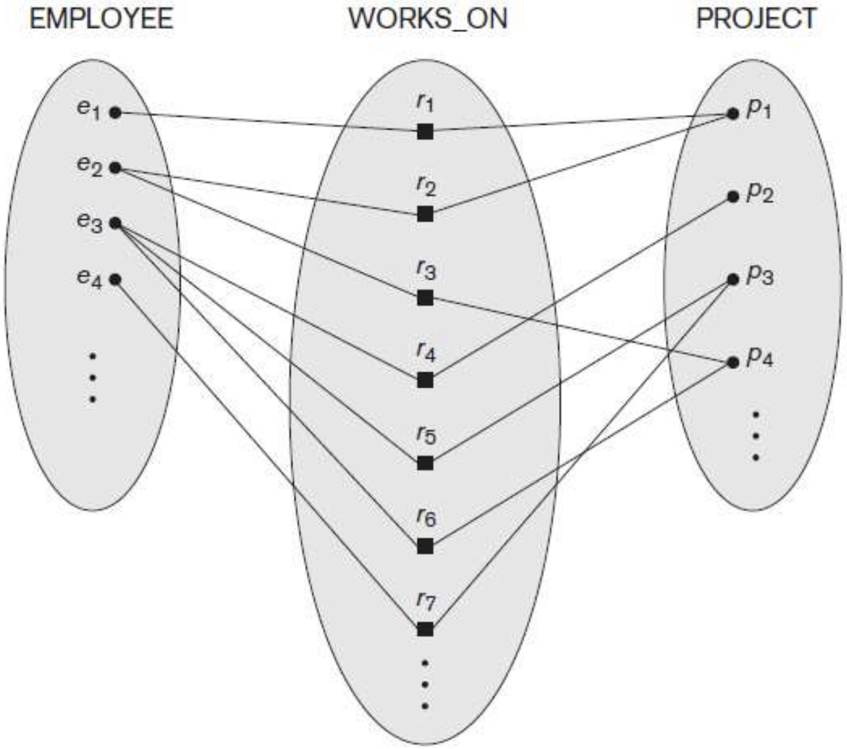


Figure 3.13
An M:N relationship, WORKS_ON.

Attributes of Relationship Types

- Attributes of 1:1 relationship types can be migrated to one of the participating entity types.
- For a 1:N relationship type, a relationship attribute can be migrated *only* to the entity type on the N-side of the relationship.
- For M:N (many-to-many) relationship types, some attributes may be determined by the *combination of participating entities* in a relationship instance, not by any single entity. Such attributes *must be specified as relationship attributes*.

Weak Entity Types

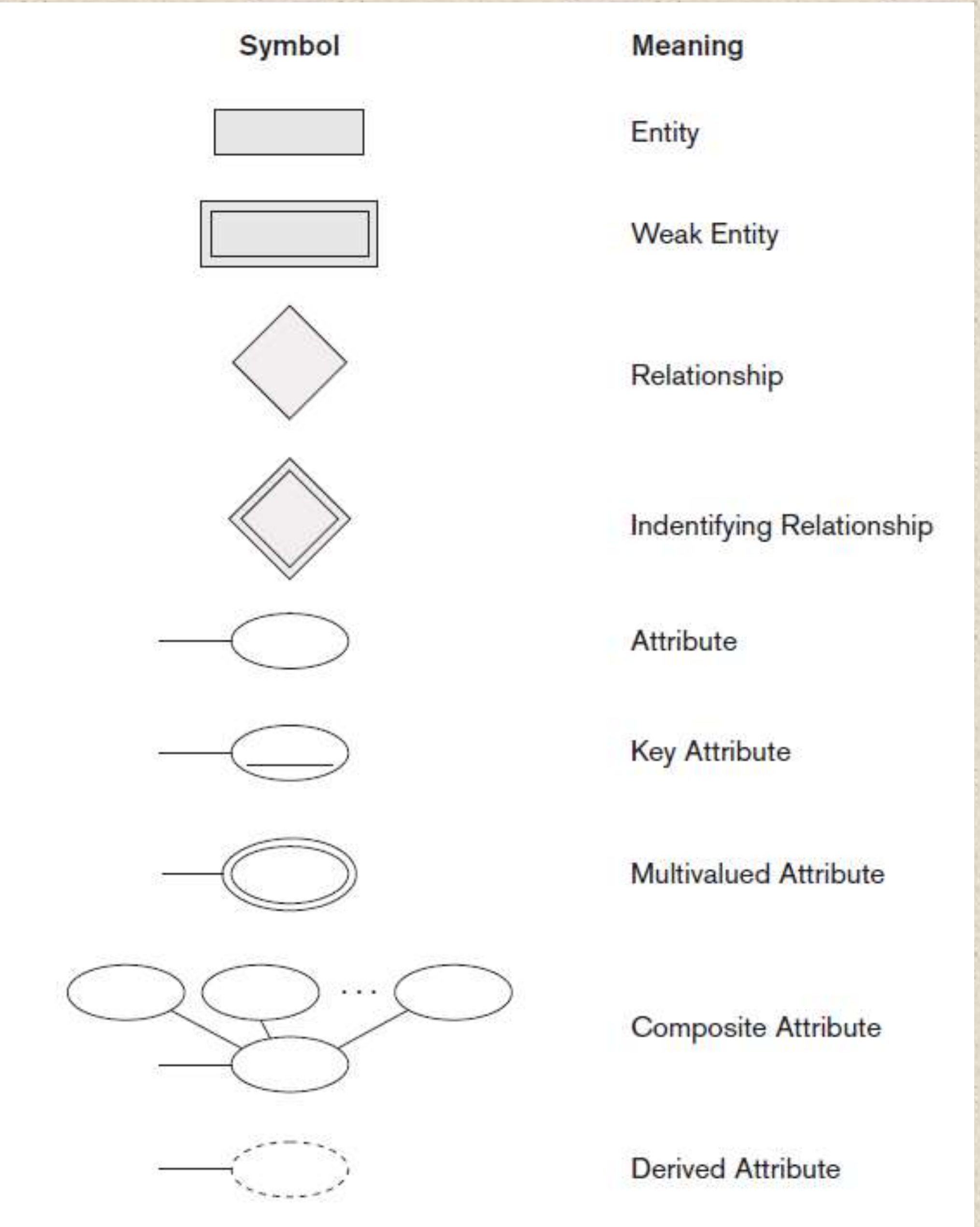
- Entity types that do not have key attributes of their own are called **weak entity types**. In contrast, **regular entity types** that do have a key attribute—which include all the examples discussed so far—are called **strong entity types**. Entities belonging to a weak entity type are identified by being related to specific entities from another entity type in combination with one of their attribute values. We call this other entity type the **identifying** or **owner entity type**, and we call the relationship type that relates a weak entity type to its owner the **identifying relationship** of the weak entity type.
- A weak entity type always has a ***total participation constraint*** (existence dependency) with respect to its identifying relationship because a weak entity cannot be identified without an owner entity. However, not every existence dependency results in a weak entity type. For example, a DRIVER_LICENSE entity cannot exist unless it is related to a PERSON entity, even though it has its own key (License_number) and hence is not a weak entity.

Weak Entity Types

- A weak entity type normally has a **partial key**, which is the attribute that can uniquely identify weak entities that are *related to the same owner entity.*
- In ER diagrams, both a weak entity type and its identifying relationship are distinguished by surrounding their boxes and diamonds with double lines. The partial key attribute is underlined with a dashed or dotted line.
- Weak entity types can sometimes be represented as complex (composite, multivalued) attributes. The choice of which representation to use is made by the database designer.
- A weak entity type may have more than one identifying entity type and an identifying relationship type of degree higher than two.
- The identifying relationship is many-to-one from the weak entity set to the identifying entity set

ER Diagram Notations

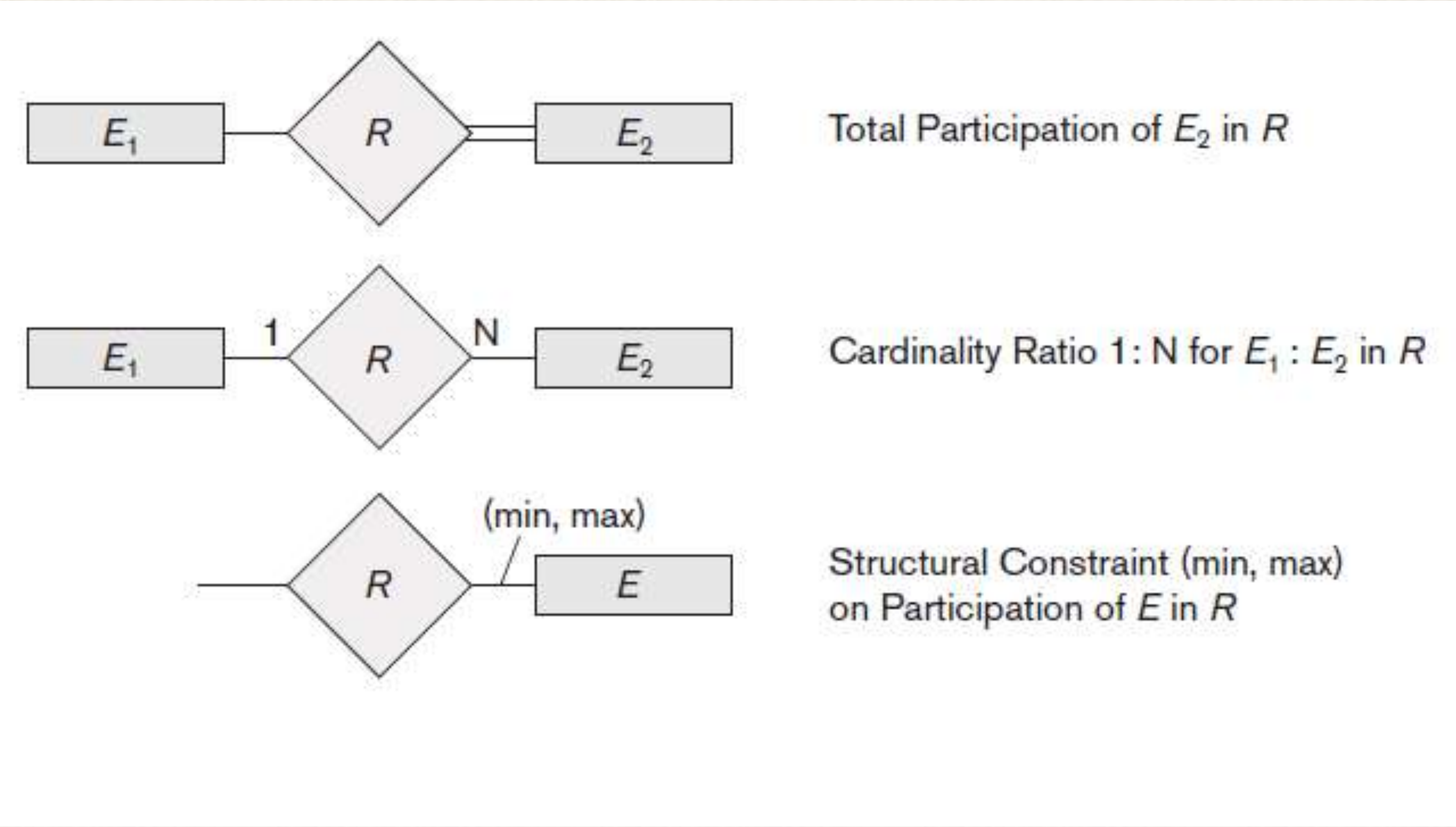
- **ER- Notations:**



Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Indentifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute

ER Diagram Notations

- **ER- Notations:**

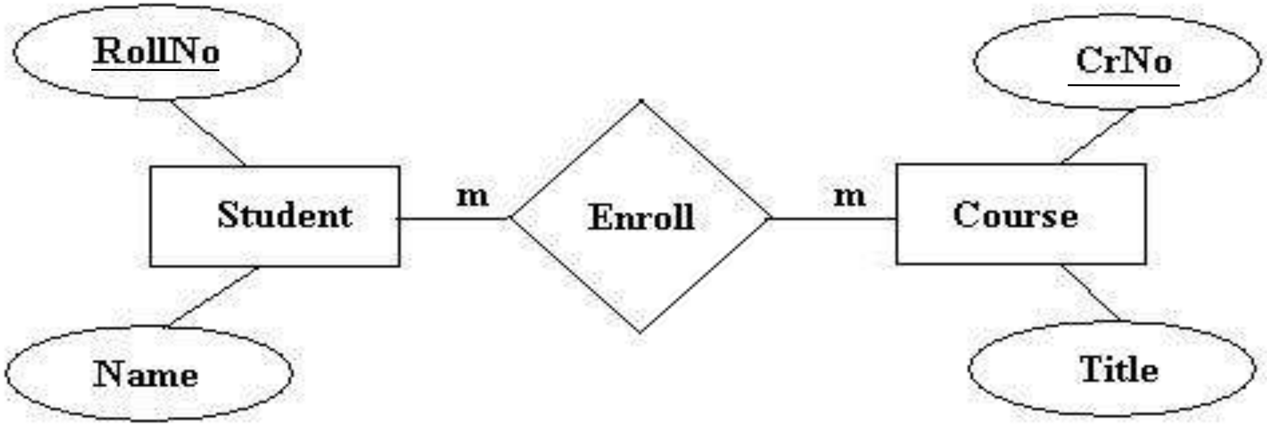


Total Participation of E_2 in R

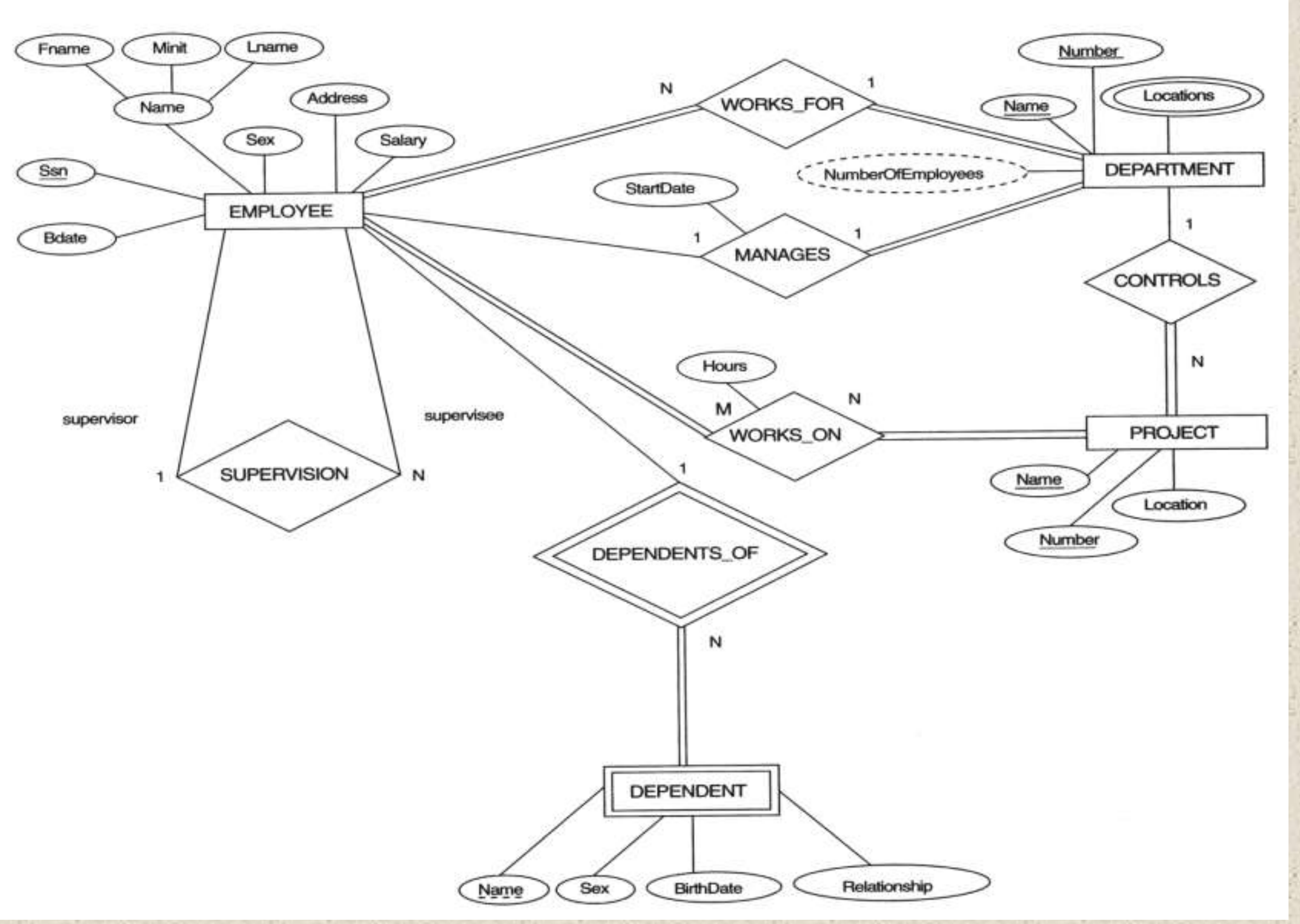
Cardinality Ratio 1: N for $E_1 : E_2$ in R

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

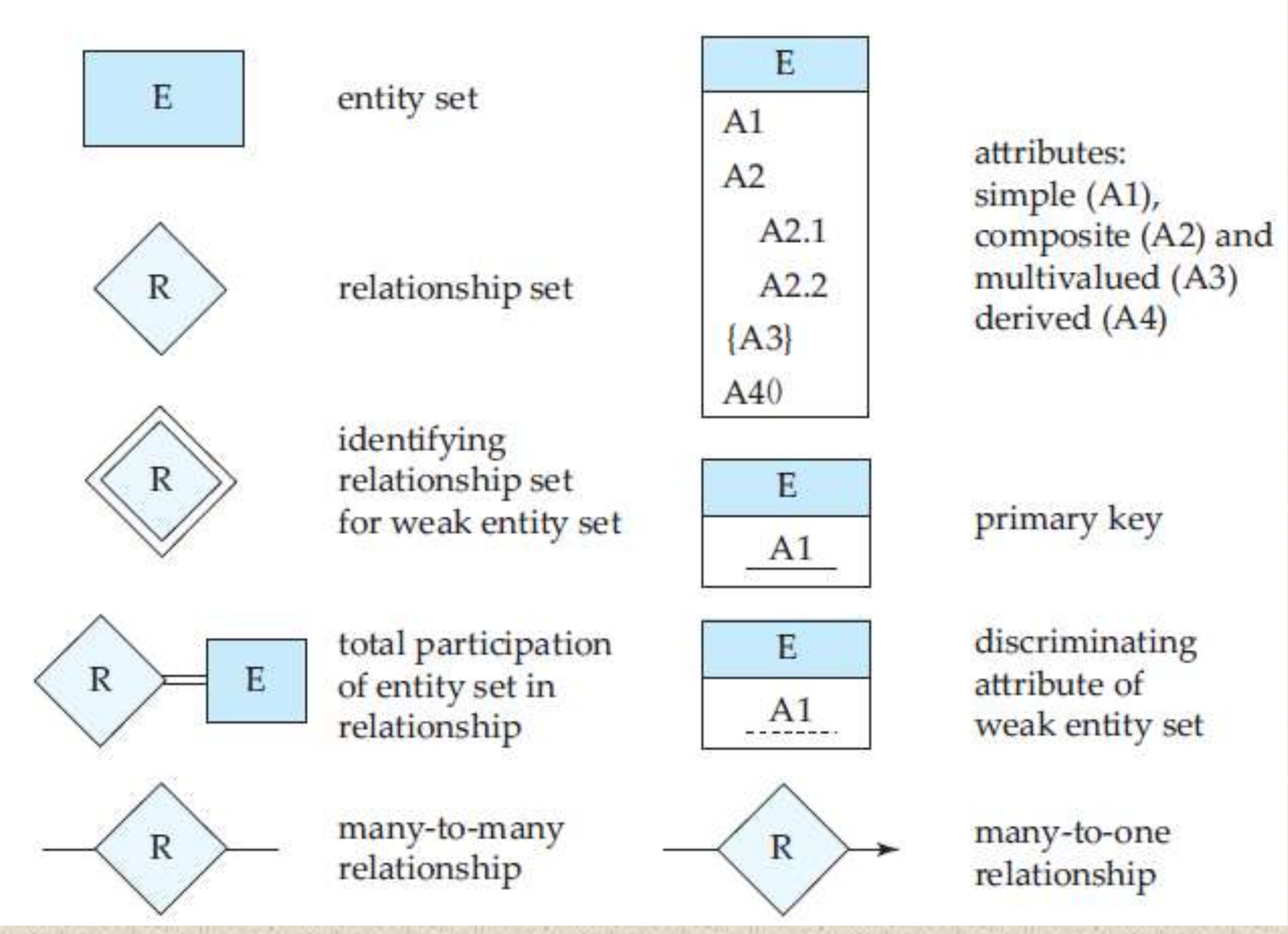
ER Diagram: An Example



ER Diagram: An Example



ER Diagram Notations: Based on Silberschatz Book



entity set

relationship set

identifying relationship set for weak entity set

total participation of entity set in relationship

many-to-many relationship

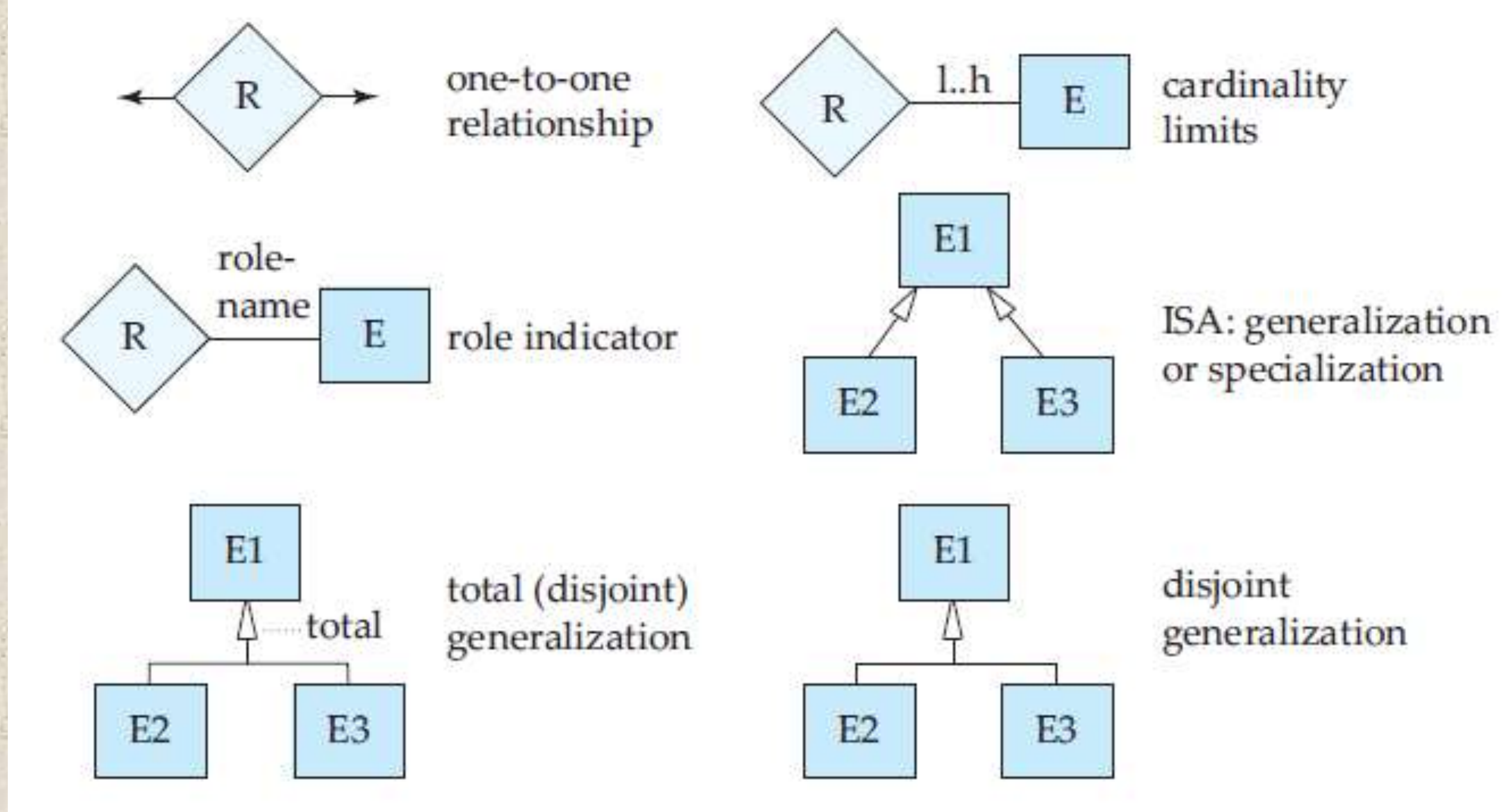
attributes: simple (A1), composite (A2) and multivalued (A3) derived (A4)

primary key

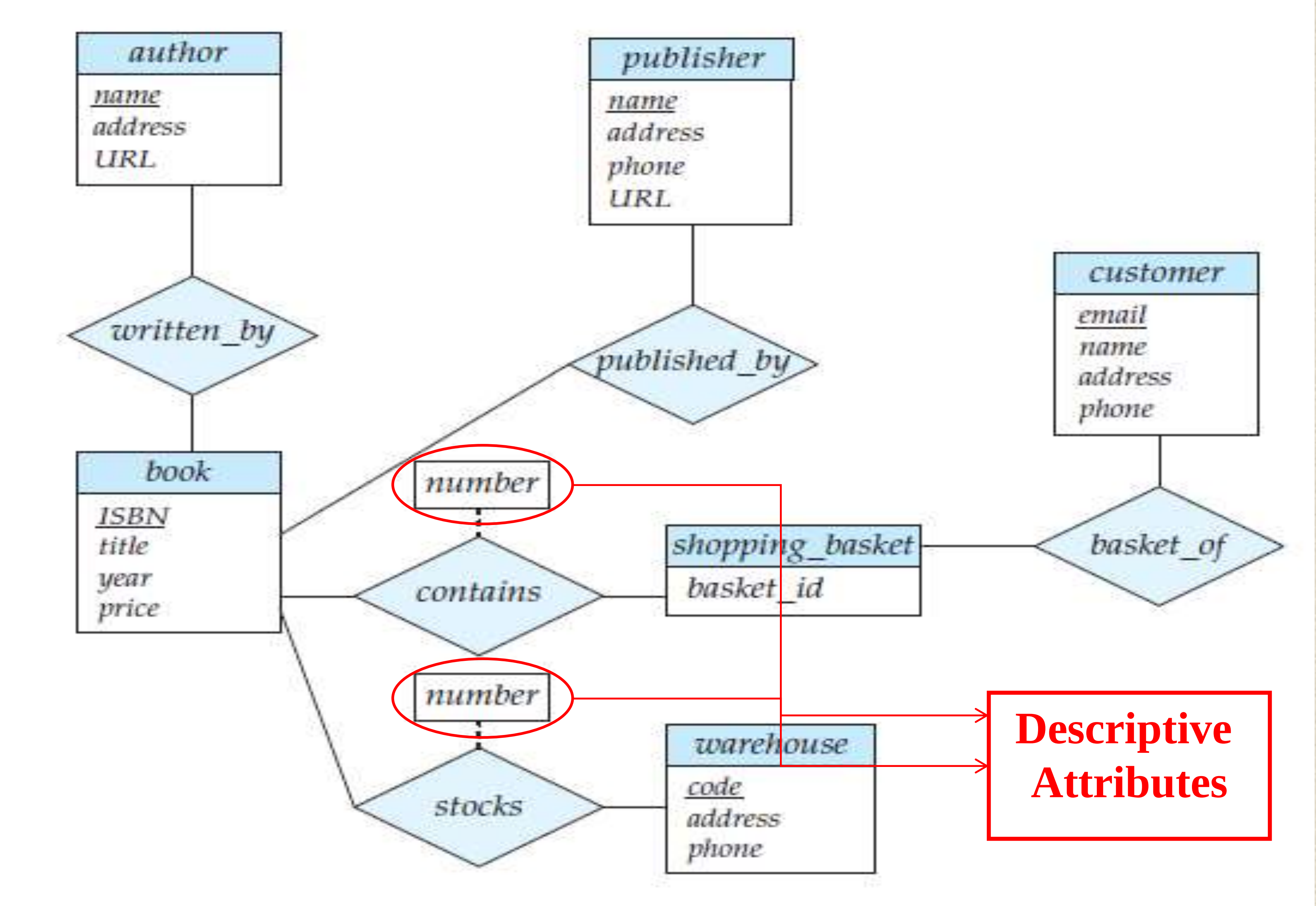
discriminating attribute of weak entity set

many-to-one relationship

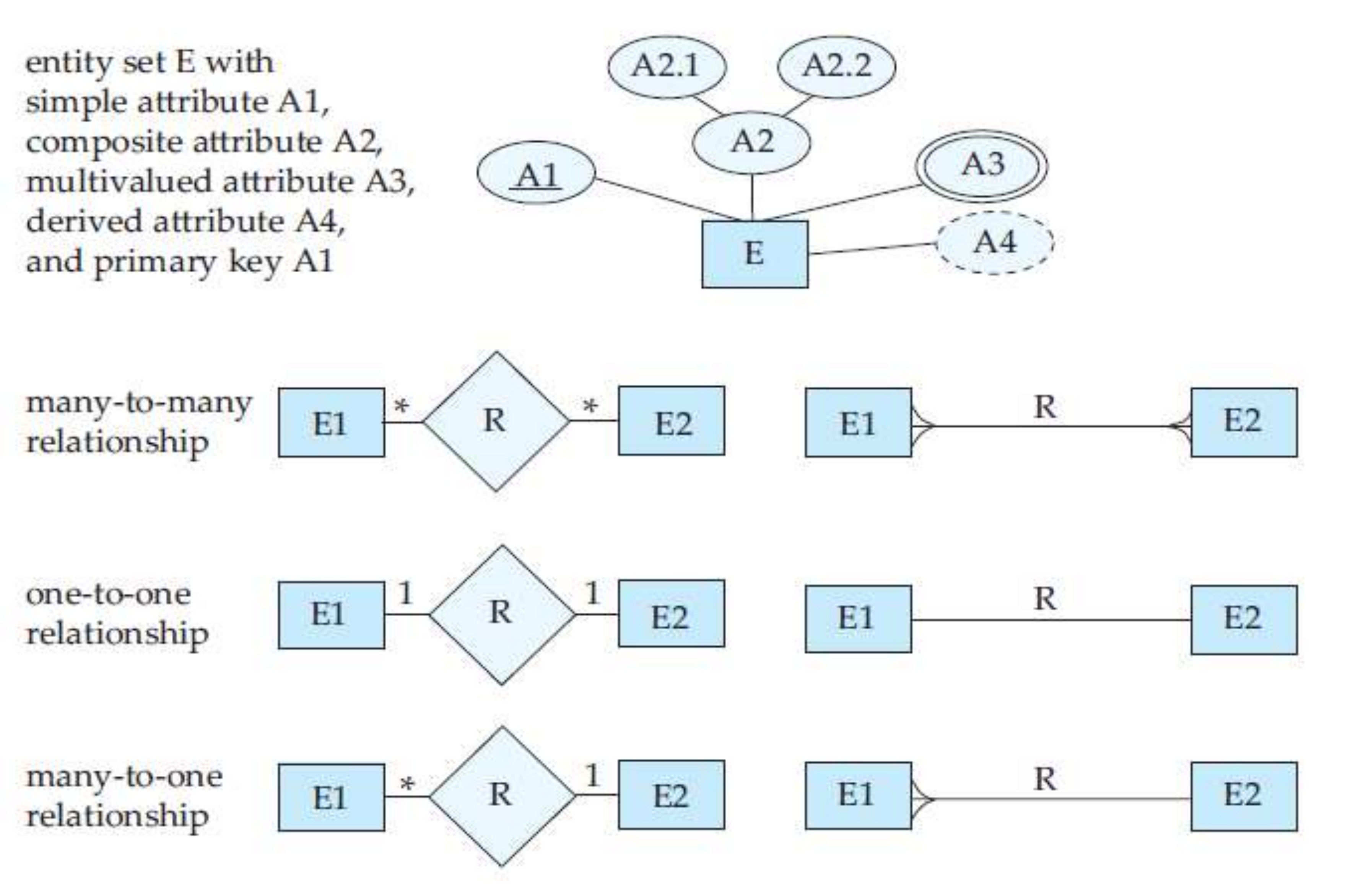
ER Diagram Notations: ….



ER Diagram Notations: ….



ER Diagram Notations: ….



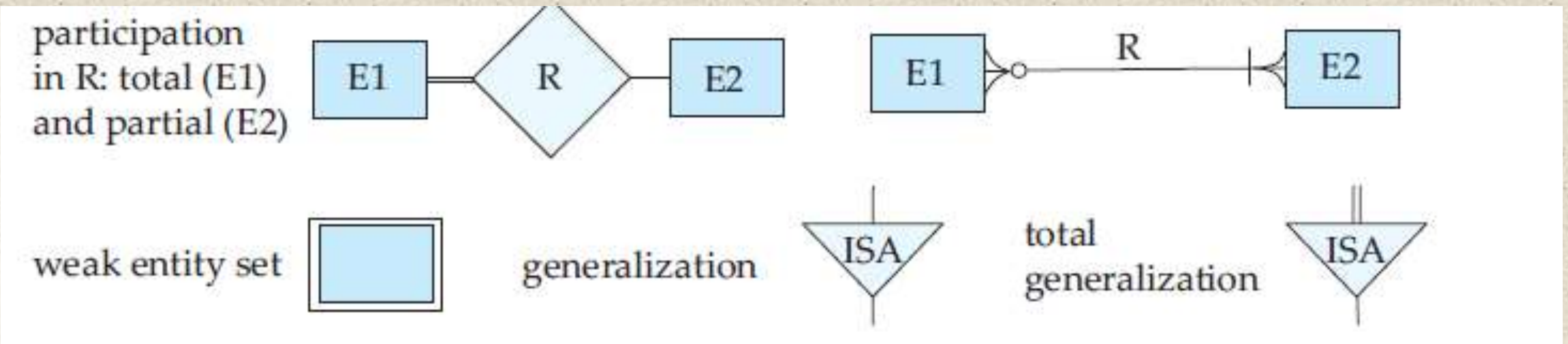
entity set E with
simple attribute A1,
composite attribute A2,
multivalued attribute A3,
derived attribute A4,
and primary key A1

many-to-many
relationship

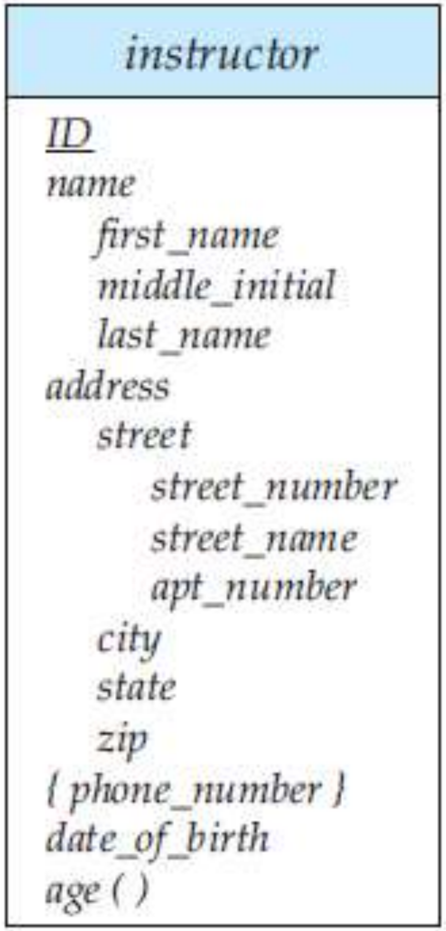
one-to-one
relationship

many-to-one
relationship

ER Diagram Notations: ….



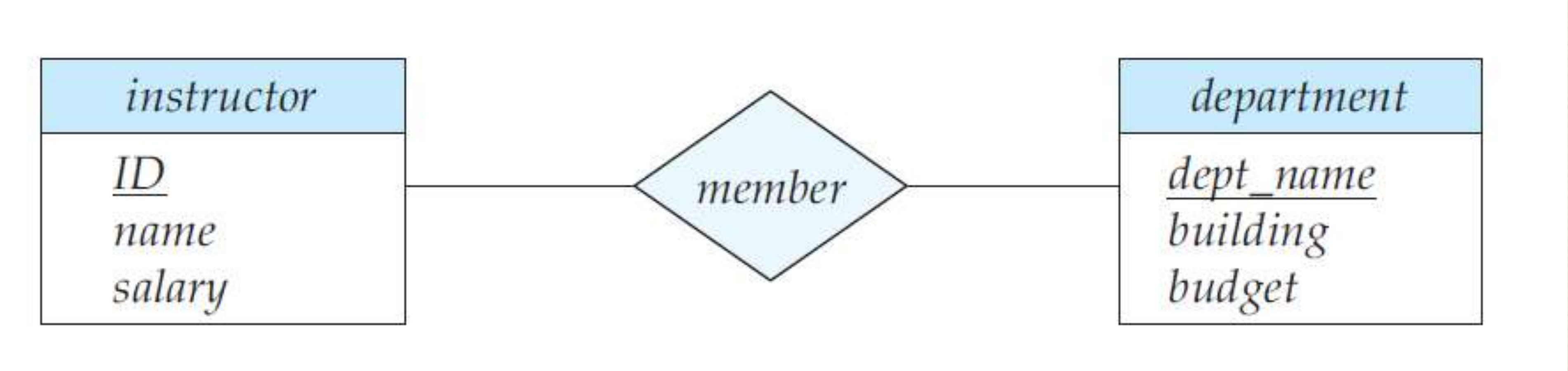
ER Diagram Notations: ….



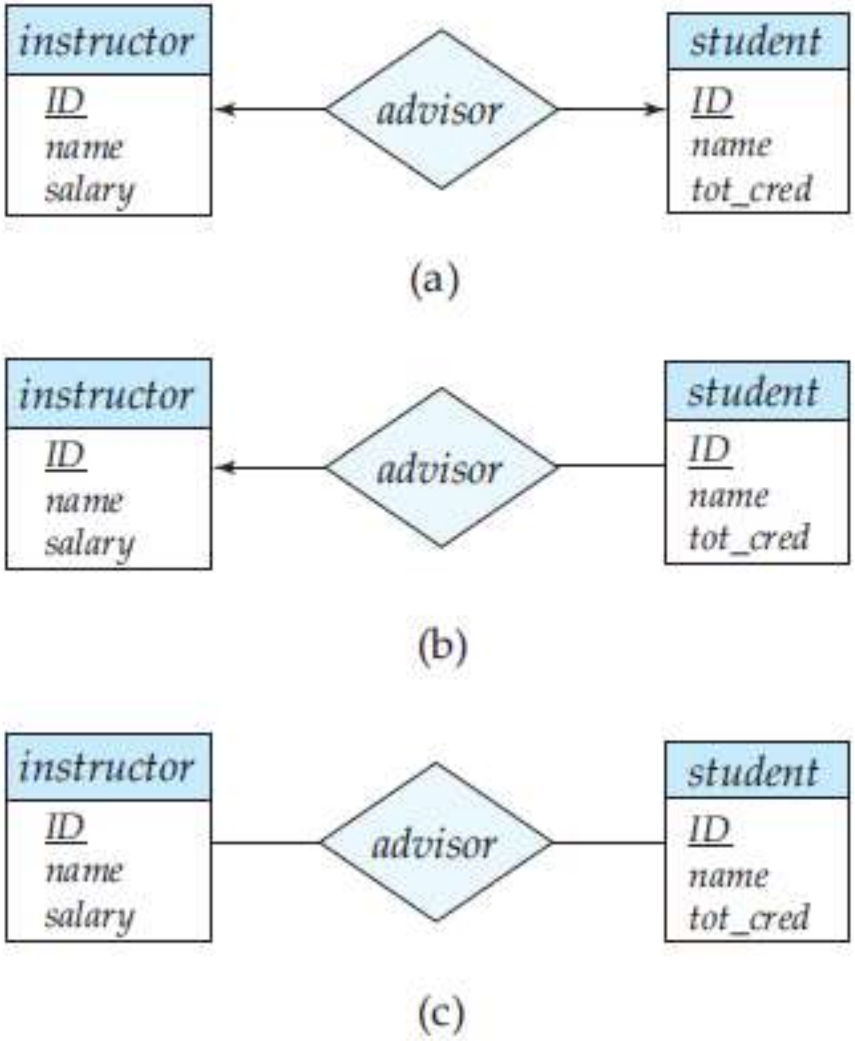
```
instructor
ID
name
    first_name
    middle_initial
    last_name
address
    street
        street_number
        street_name
        apt_number
    city
    state
    zip
{ phone_number }
date_of_birth
age ( )
```

Figure 7.11 E-R diagram with composite, multivalued, and derived attributes.

ER Diagram Notations: ….



ER Diagram Notations: ….



(a)

(b)

(c)

Figure 7.9 Relationships. (a) One-to-one. (b) One-to-many. (c) Many-to-many.

ER Diagram Notations: ….

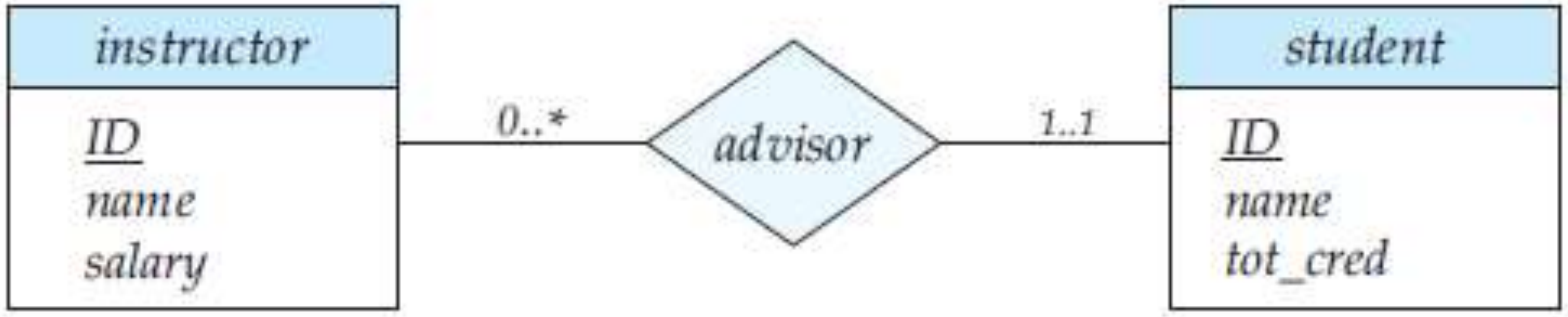


Figure 7.10 Cardinality limits on relationship sets.

ER Diagram Notations: ….

- **Roles**

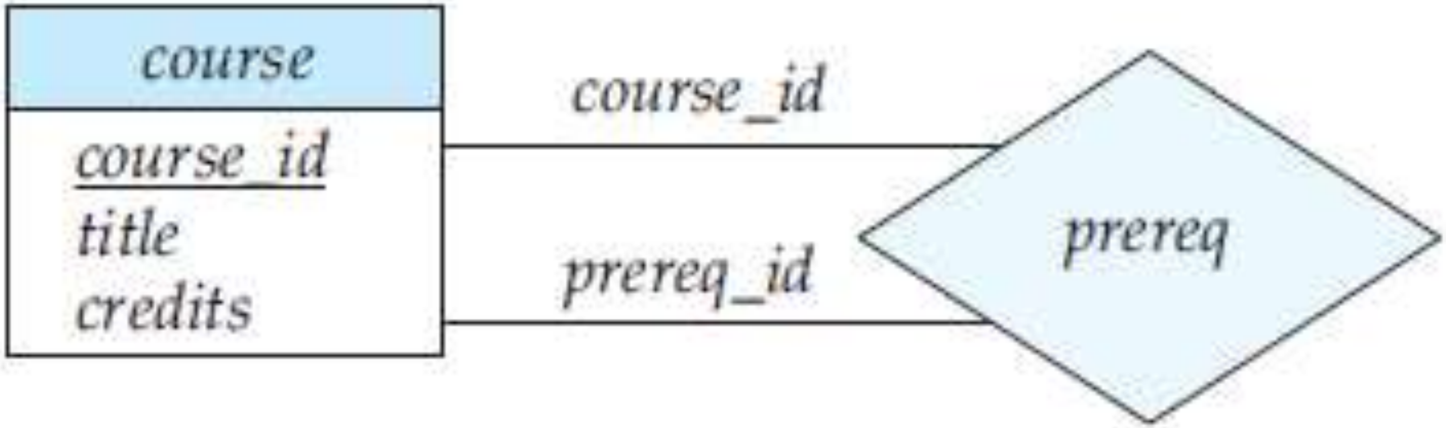


Figure 7.12 E-R diagram with role indicators.

ER Diagram Notations: ….

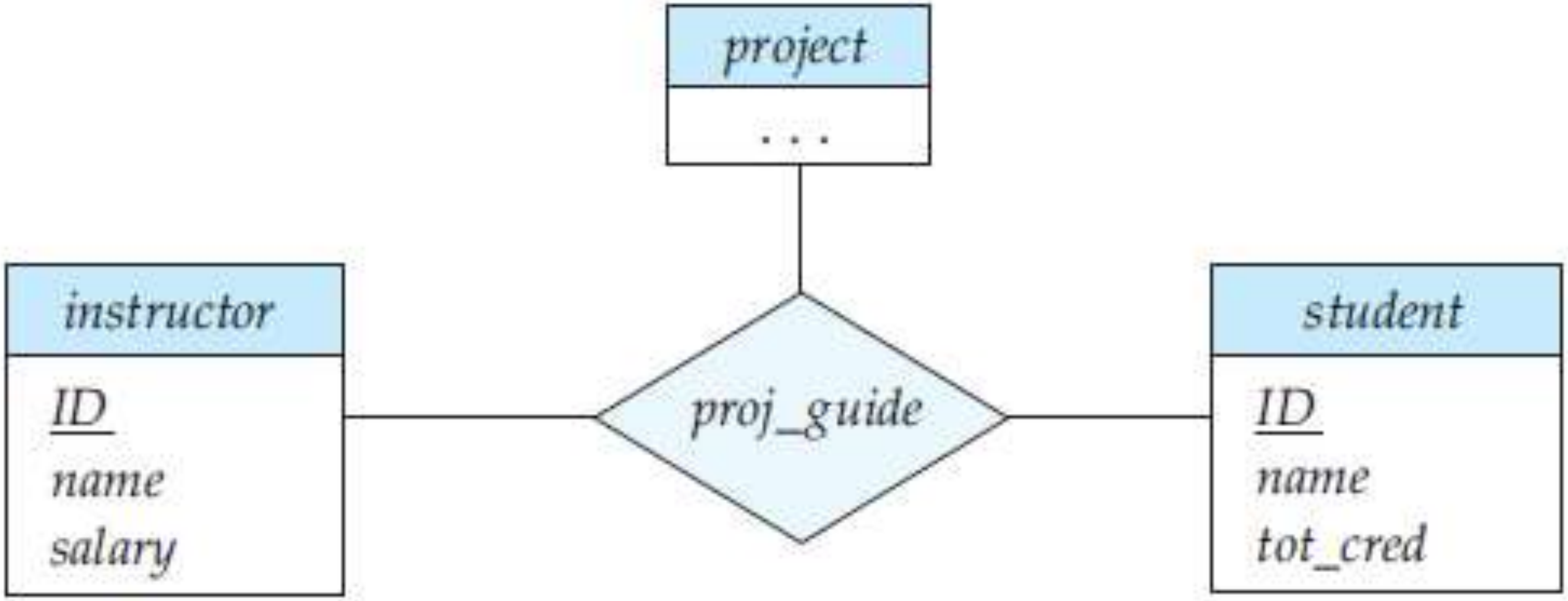
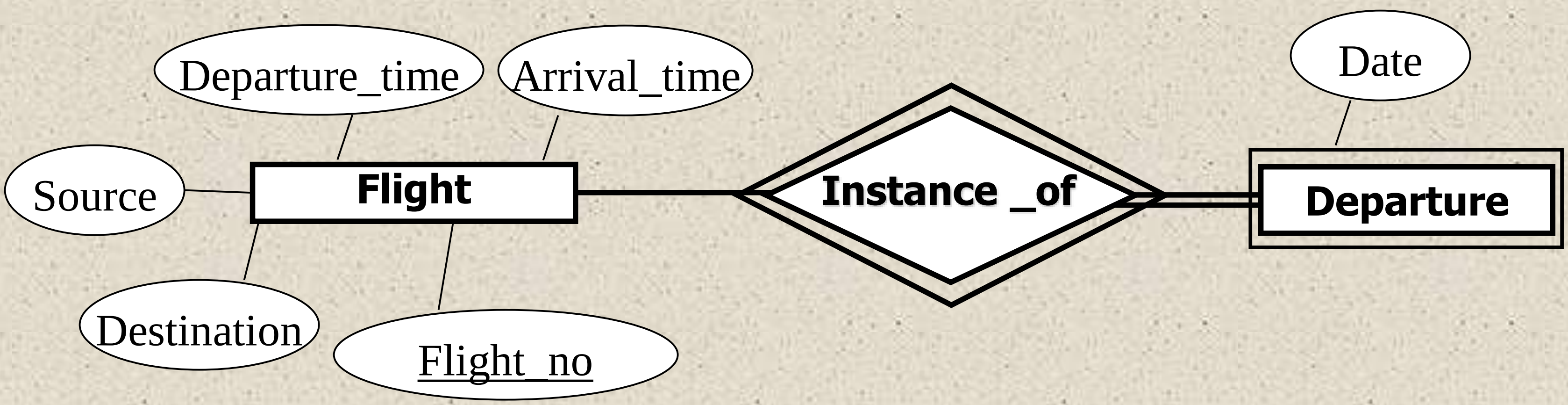


Figure 7.13 E-R diagram with a ternary relationship.

Weak Entity Example

- Here departure is a weak entity type. Even though each departure entity is distinct but different flights may have departure on same date. Thus it can be identified by relating with owner entity type “flight”, since for each flight *fight_no* is unique.



Weak Entity…

- **Partial Key**: A set of attributes that can uniquely identify weak entities that are related to the same owner entity. It is often known as **discriminator**.
- In the previous example;
  - {date} is a partial key.
  - {flight_no, date} becomes the primary key for the ***Departure*** entity.

- The entity types that do have key attributes of their own are called **strong or regular entity types**. They have their independent existence.

Weak Entity…

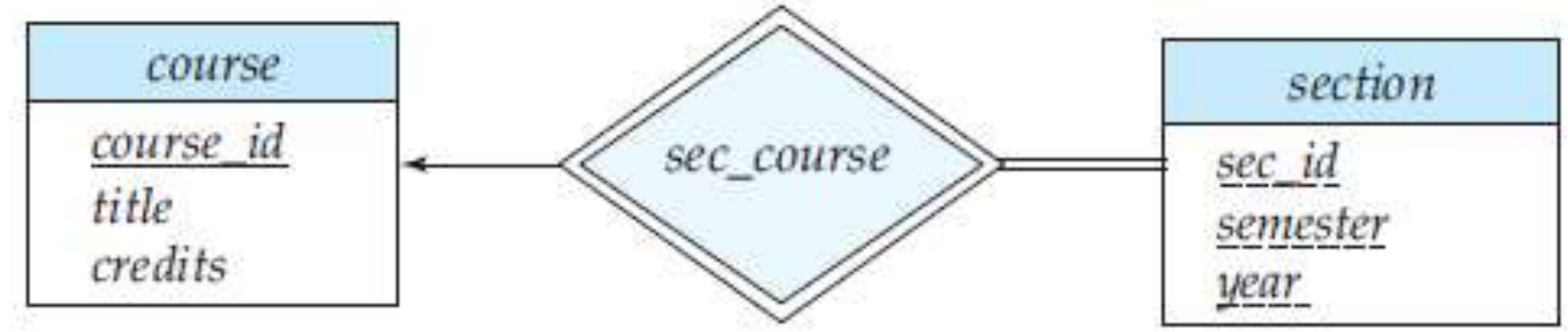
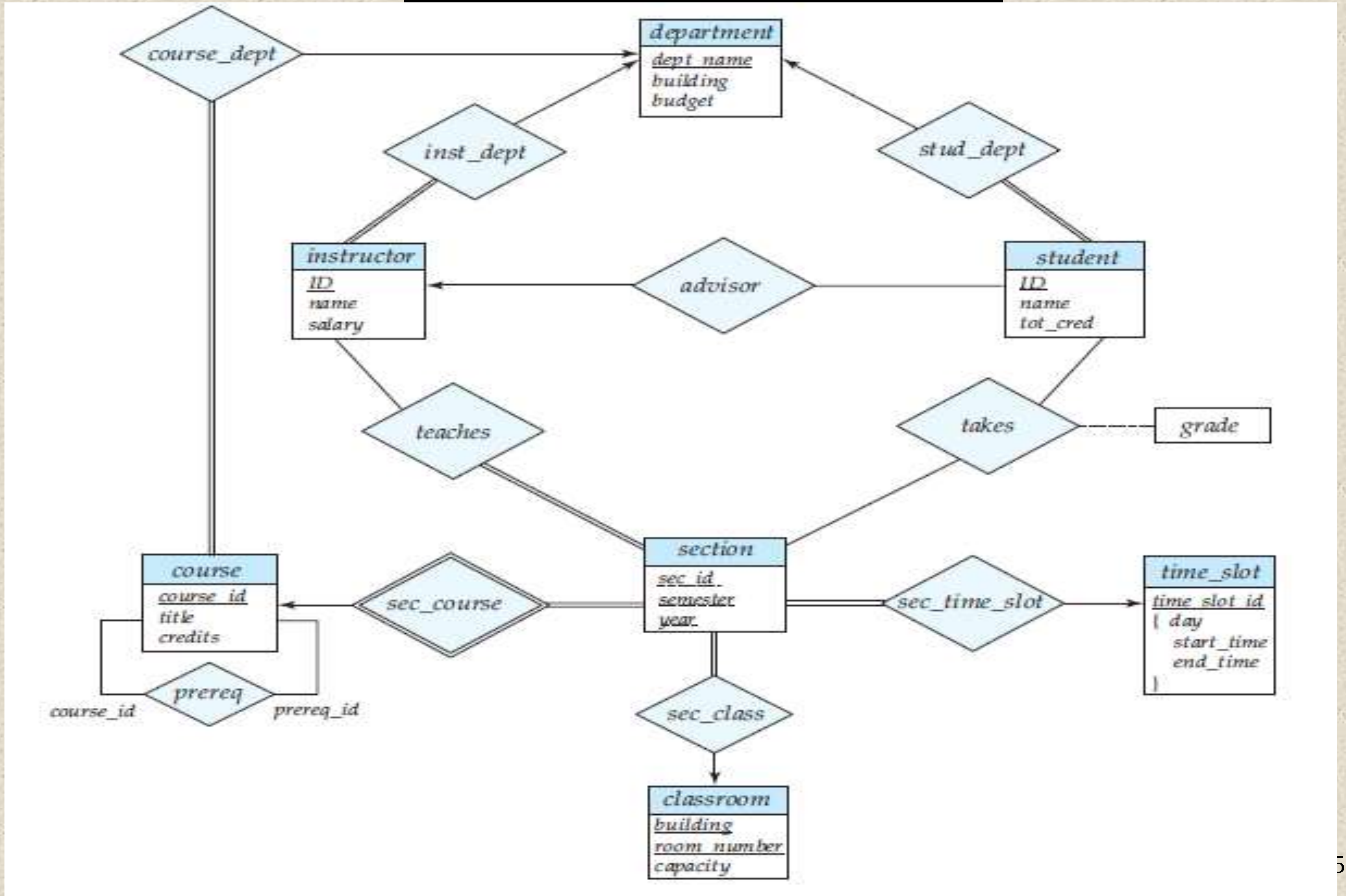


Figure 7.14 E-R diagram with a weak entity set.

ER Diagram for University



ER Diagram for Insurance Company

- Construct an E-R diagram for a car insurance company whose customers own one or more cars each. Each car has associated with it zero to any number of recorded accidents. Each insurance policy covers one or more cars, and has one or more premium payments associated with it. Each payment is for a particular period of time, and has an associated due date, and the date when the payment was received.

ER Diagram: Naming Convention

- When designing a database schema, the choice of names for entity types, attributes, relationship types, and (particularly) roles is not always straightforward. One should choose names that convey, as much as possible, the meanings attached to the different constructs in the schema.
- Choose ***singular names*** for entity types, rather than plural ones, because the entity type name applies to each individual entity belonging to that entity type.
- In ER diagrams, it is good to use the convention that entity type and relationship type names are in uppercase letters, attribute names have their initial letter capitalized, and role names are in lowercase letters.

ER Diagram: Naming Convention

- As a general practice, given a narrative description of the database requirements, the *nouns* appearing in the narrative tend to give rise to entity type names, and the *verbs* tend to indicate names of relationship types. Attribute names generally arise from additional nouns that describe the nouns corresponding to entity types.

- Another naming consideration involves choosing binary relationship names to make the ER diagram of the schema readable from left to right and from top to bottom

Design Choices for ER Conceptual Design

- It is occasionally difficult to decide whether a particular concept should be modeled as an entity type, an attribute, or a relationship type.
- In general, the schema design process should be considered an iterative refinement process, where an initial design is created and then iteratively refined until the most suitable design is reached.
- A concept may be first modeled as an attribute and then refined into a relationship because it is determined that the attribute is a reference to another entity type. It is often the case that a pair of such attributes that are inverses of one another are refined into a binary relationship. Once an attribute is replaced by a relationship, the attribute itself should be removed from the entity type to avoid duplication and redundancy.

Design Choices for ER Conceptual Design

- Similarly, an attribute that exists in several entity types may be elevated or promoted to an independent entity type. For example, suppose that each of several entity types in a UNIVERSITY database, such as STUDENT, INSTRUCTOR, and COURSE, has an attribute Department in the initial design; the designer may then choose to create an entity type DEPARTMENT with a single attribute Dept_name and relate it to the three entity types (STUDENT, INSTRUCTOR, and COURSE) via appropriate relationships. Other attributes/relationships of DEPARTMENT may be discovered later.
- An inverse refinement to the previous case may be applied—for example, if an entity type DEPARTMENT exists in the initial design with a single attribute Dept_name and is related to only one other entity type, STUDENT. In this case, DEPARTMENT may be reduced or demoted to an attribute of STUDENT

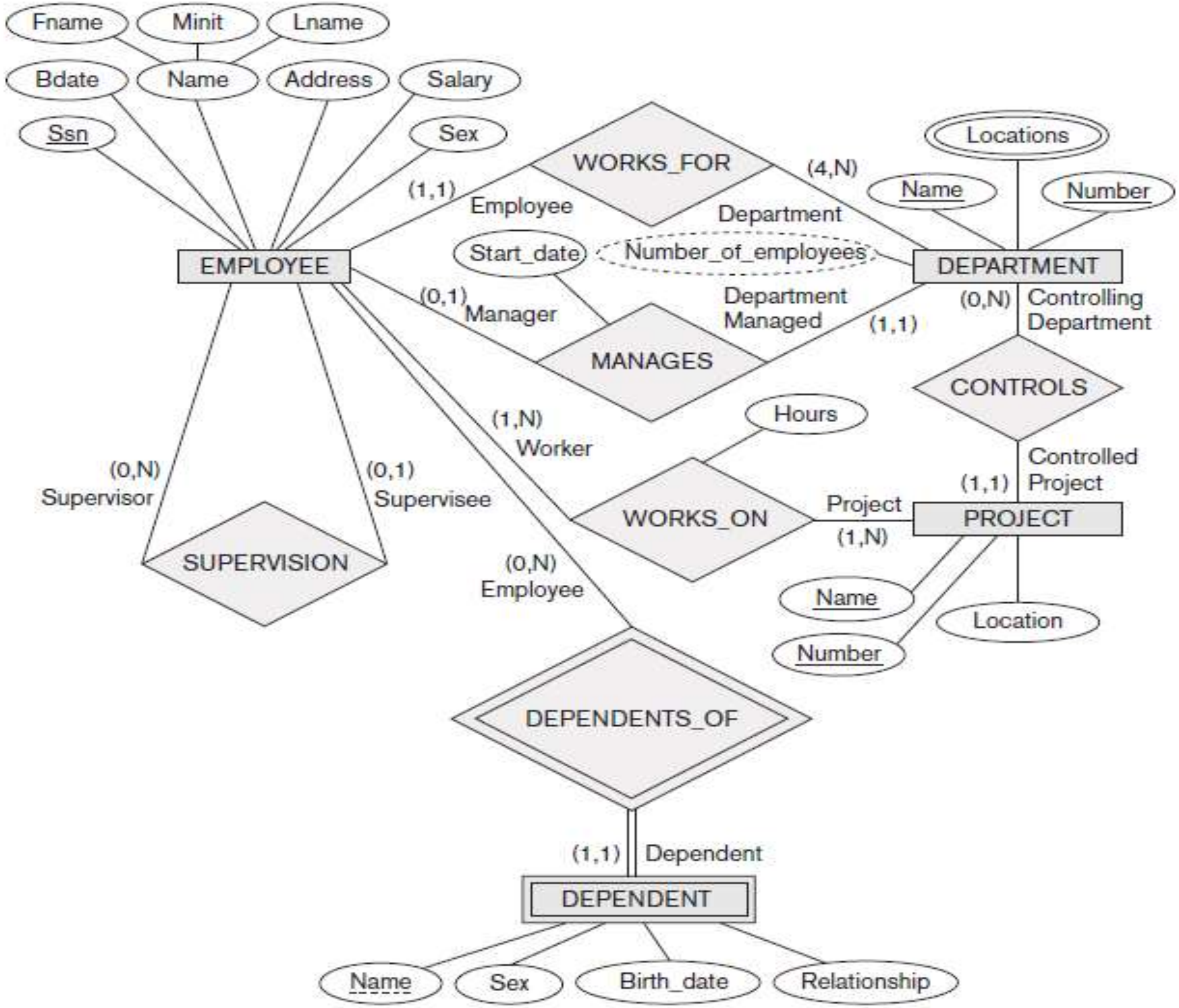


Figure 3.15
ER diagrams for the company schema, with structural constraints specified using (min, max) notation and role names.

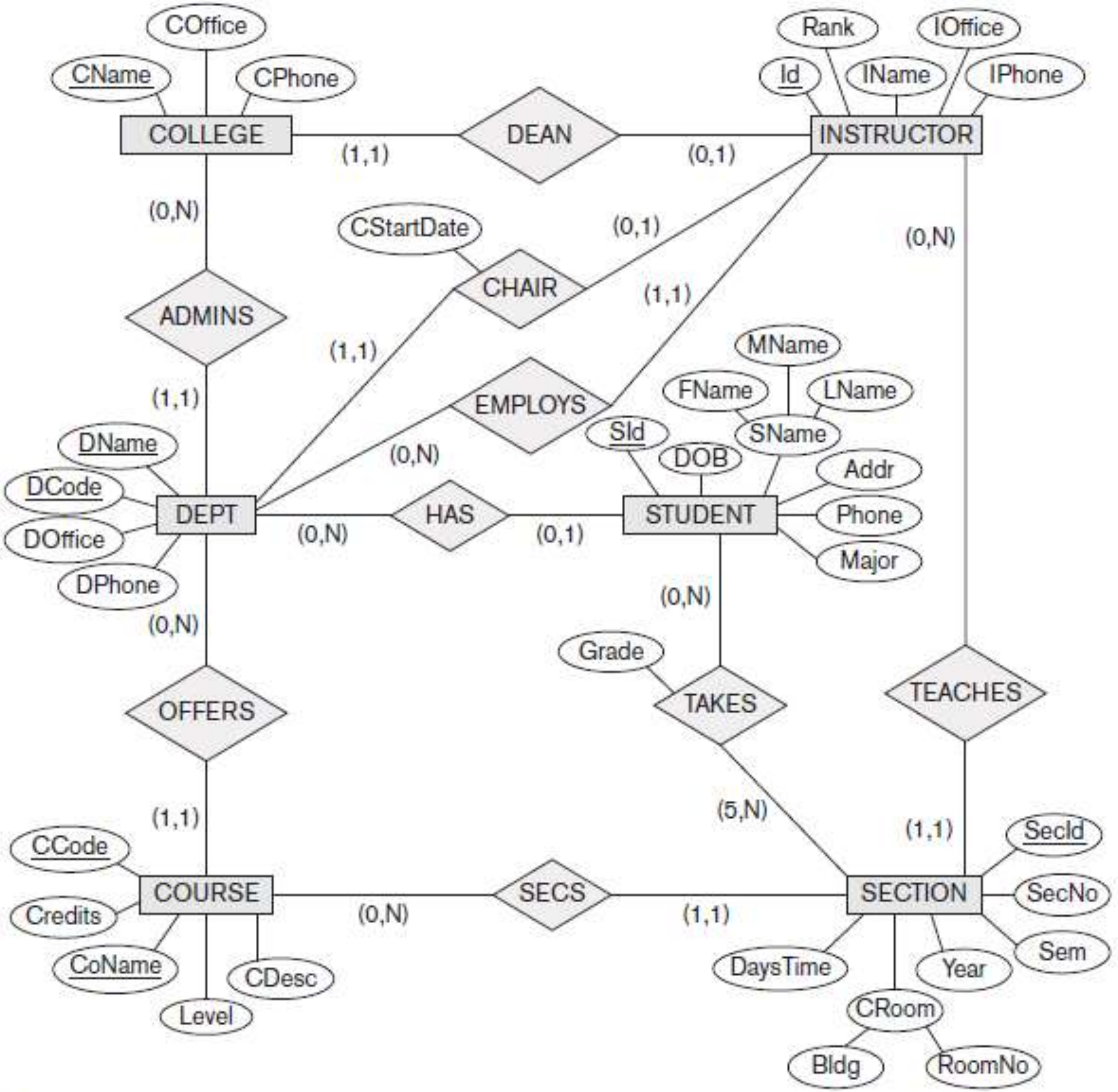


Figure 3.20
An ER diagram for a UNIVERSITY database schema.

Relationship Types of Degree Higher than Two

- We defined the **degree** of a relationship type as the number of participating entity types and called a relationship type of degree two *binary* and a relationship type of degree three *ternary*.

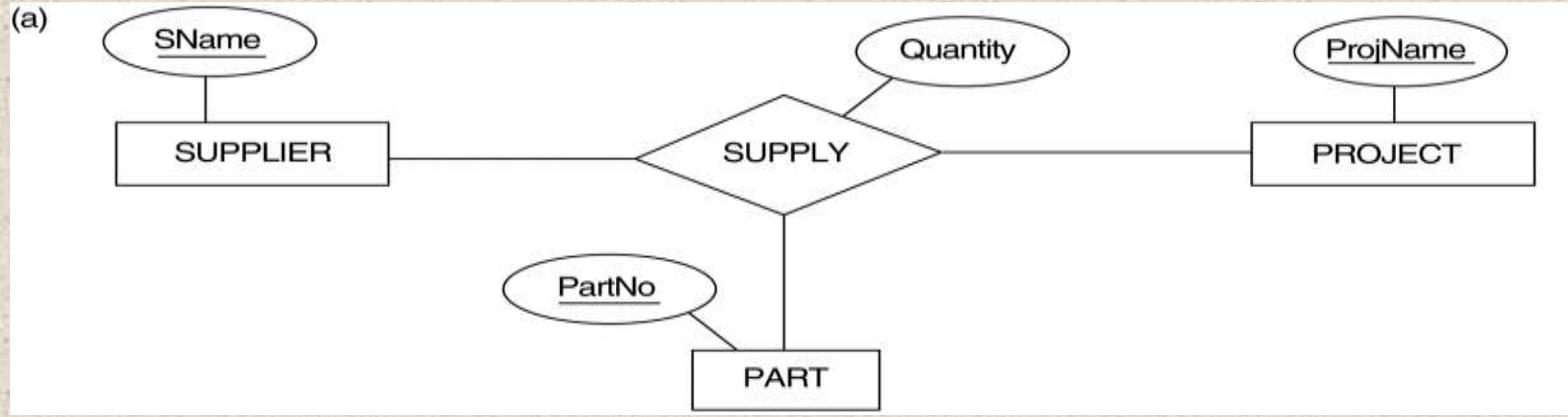
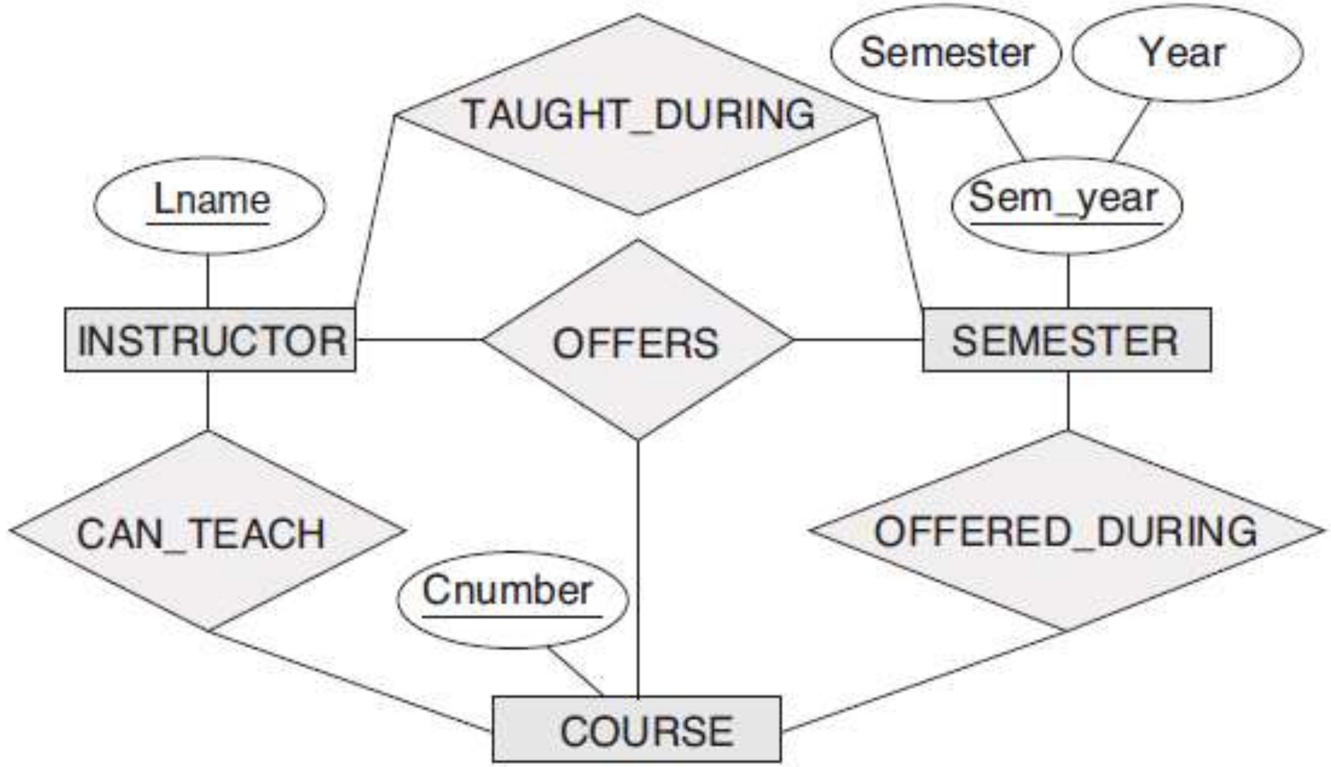


Fig: A Ternary relationship

Relationship Types of Degree Higher than Two

Figure 3.18
Another example of ternary versus binary relationship types.


TAUGHT_DURING
Semester
Year
Lname
Sem_year
INSTRUCTOR
OFFERS
SEMESTER
CAN_TEACH
OFFERED_DURING
Cnumber
COURSE