Database Management System (DBMS)

Avinash V. Gondal

email: avinash.gondal@gmail.com

Module 2

Entity–Relationship Model

2.Entity–Relationship Model

Contents :

Introduction

Introduction

- Building a database system is a complex process that normally requires analyzing the user's requirements followed by a design process and finally concluding with an implementation.
- During analysis and design, the database designer normally needs to build a model of the proposed database system.
- Database modeling is essential to adequately manage the database development process.
- A model is an abstract (and often incomplete) representation of the design or definition of a complex product, process or situation (e.g. information requirements of an enterprise)

Introduction

- The role of data modeling in building database systems is similar to the role of an architect in building a substantial house or a building.
- Data modeling is similar to the work of an architect whose job is to design a model that transforms the client requirements into specifications which are then given to the database builder.
- Just like the architect does not disappear when the builder arrives, a database modeler also does not disappear when the database builder takes over.
- Both the architect and the database modeler have a role in supervising the building activity.

Techniques for Database Modeling

- Entity Relationship Model
- Relational Model
- Hierarchical Model
- Network Model
- Object Oriented Model
- Object Relational Model
- Semi-structured Data Model (XML)

Entity-Relationship (ER) Model

Introduction

- The Entity-Relationship (ER) model was originally proposed by Peter Chen in 1976 as a way to unify the network and relational database views.
- Simply stated the ER model is a conceptual data model that views the real world as entities and relationships.
- A basic component of the model is the Entity-Relationship diagram which is used to visually represents data objects.

Database Design Techniques

- Top down Approach
  - E R Modeling
- Bottom Up approach
  - Normalization

The E-R Model

- **Basic constructs of the E-R model:**
  - Entities
  - Attributes
  - Relationships

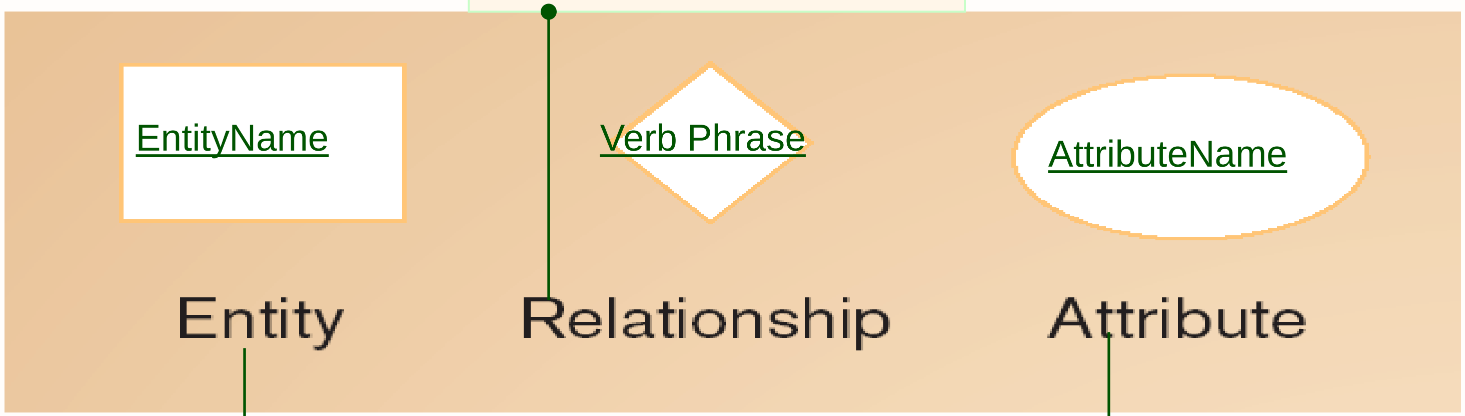
E-R Modeling

- **ER modeling:** A graphical technique for *understanding* and organizing the data independent of the actual database implementation
- **Entity:** Any thing that may have an independent existence and about which we intend to collect data.
  - Also known as **Entity type**.
- **Entity instance:** a particular member of the entity type
  - Example : a particular student
- **Attributes:** Properties/characteristics that describe entities
- **Relationships:** Associations between entities

Entity-Relationship (ER) Modeling

- **ER Modeling** is a *top-down* approach to database design.
- Entity Relationship (ER) Diagram
  - A detailed, logical representation of the entities, associations and data elements for an organization or business
- Notation uses three main constructs
  - Data entities
  - Relationships
  - Attributes

Association between the instances of one or more entity types



Person, place, object, event or concept about which data is to be maintained

named property or characteristic of an entity

Represents a set or collection of objects in the real world that share the same properties

Entities

- Examples of entities:
  - Person: EMPLOYEE, STUDENT, PATIENT
  - Place: STORE, WAREHOUSE
  - Object: MACHINE, PRODUCT, CAR
  - Event: SALE,REGISTRATION, RENEWAL
  - Concept: ACCOUNT, COURSE
- Guidelines for naming and defining entity types:
  - An entity type name is a singular noun
  - An entity type should be descriptive and specific
  - An entity name should be concise
  - Event entity types should be named for the result of the event, not the activity or process of the event.

Regular Vs. Weak Entity Type

- **Regular Entity**: Entity that has its own key attribute.
  - Example : Employee, student ,customer, policy holder etc.
- **Weak entity**: Entity that depends on other entity for its existence and doesn’t have key attribute of its own
  - Example : spouse of employee
    - The spouse data is identified with the help of the employee id to which it is related

Attributes

- The set of possible values for an attribute is called the **domain** of the attribute

  Example:

  - The domain of attribute marital status is just the four values: single, married, divorced, widowed
  - The domain of the attribute month is the twelve values ranging from January to December
- **Key attribute**: The attribute (or combination of attributes) that is unique for every entity instance.
  - Example : the account number of an account, the employee id of an employee etc.
- If the key consists of two or more attributes in combination, it is called a **composite key.**

Simple Vs. Composite Attributes

- **Simple Attribute**: cannot be divided into simpler components
  - Example : age of an employee
- **Composite Attribute**: can be split into components
  - Example : Date of joining of the employee.
    - Can be split into day, month and year

Single Vs. Multi-valued Attributes

- **Single valued** : can take on only a single value for each entity instance
  - Example : age of employee.
    - There can be only one value for this.
- **Multi-valued**: can take many values
  - Example : skill set of employee

Stored Vs. Derived Attribute

- **Stored Attribute**: Attribute that need to be stored permanently.
  - Example : name of an employee
- **Derived Attribute**: Attribute that can be calculated based on other attributes
  - Example : years of service of employee can be calculated from date of joining and current date

Relationships

- A **relationship type** between two entity types defines the set of all associations between these entity types
- Each instance of the relationship between members of these entity types is called a **relationship instance**
  - Example : if Works-for is the relationship between the Employee entity and the department entity, then Ram works for Comp.sc department, sham works –for electrical department ..etc. are relationship instances of the relationship, works-for

Degree of a Relationship

- **Degree**: the number of entity types involved

One	*Unary*
Two	*Binary*
Three	*Ternary*

  – Example:
    - employee manager-of employee is unary relationship
    - employee works-for department is binary relationship
    - customer purchase item, shop keeper is a ternary relationship

Constraints

Constraints

- An E-R enterprise schema may define certain constraints to which the contents of a database system must conform.
- Two main important types of constraints are :
  - Mapping Cardinalities
  - Participation Constraints.

Mapping Cardinality

- Mapping cardinalities express the number of entities to which another entity can be associated via a relationship set.
- Relationships can have different *connectivity*
  - **one-to-one** (1:1)
  - **one-to-many** (1:N)
  - **many-to- One** (M:1)
  - **many-to-many** (M:N)
- Example :

  Employee **head-of** department (1:1)

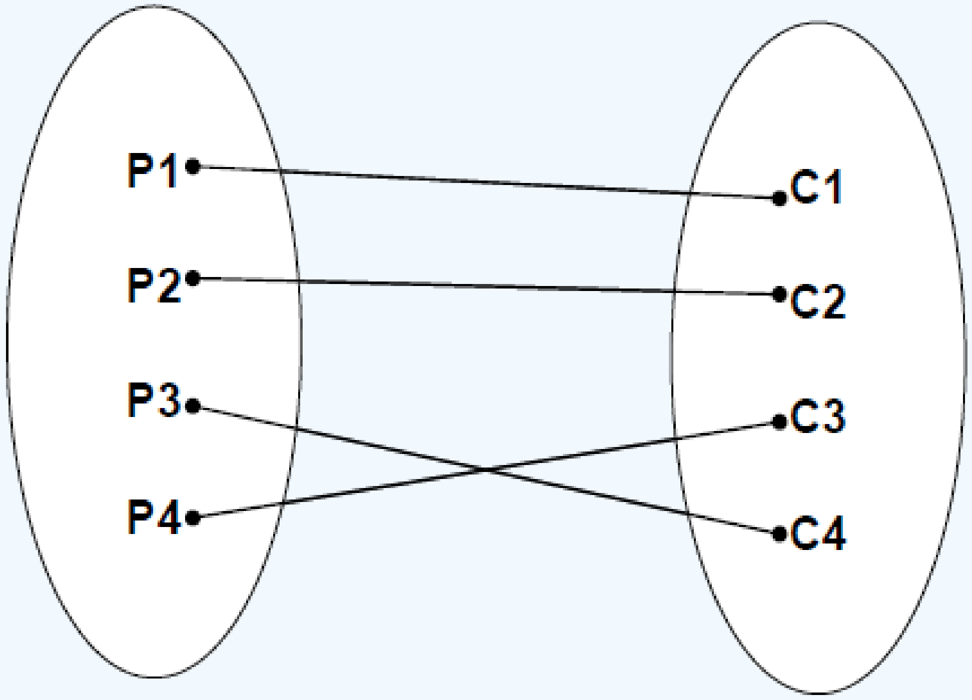
  Lecturer **offers** course (1:n) assuming a course is taught by a single lecturer

  Student **enrolls** course (m:n)

Mapping Cardinality

- The minimum and maximum values of this connectivity is called the **cardinality of the relationship**

Cardinality – One - To - One

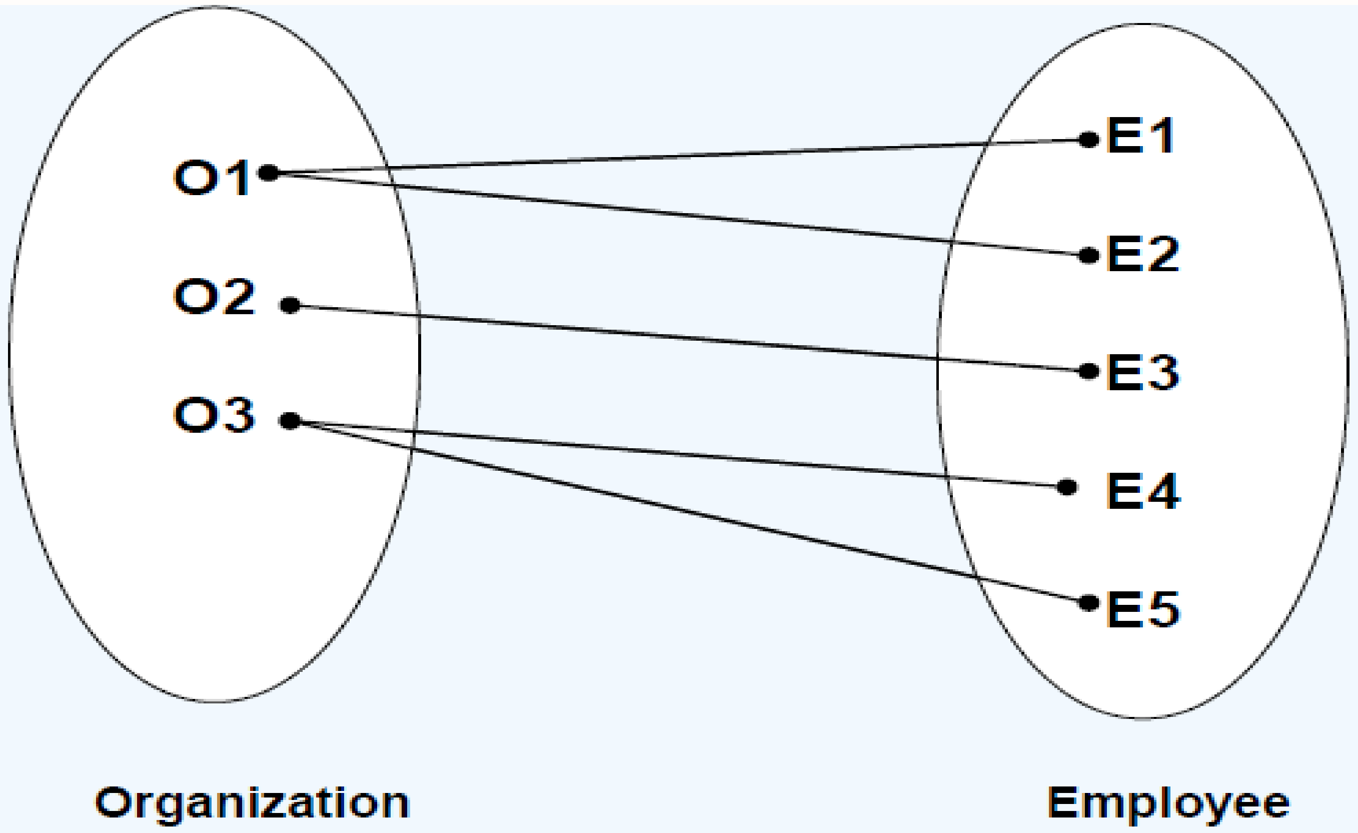


Person

Chair

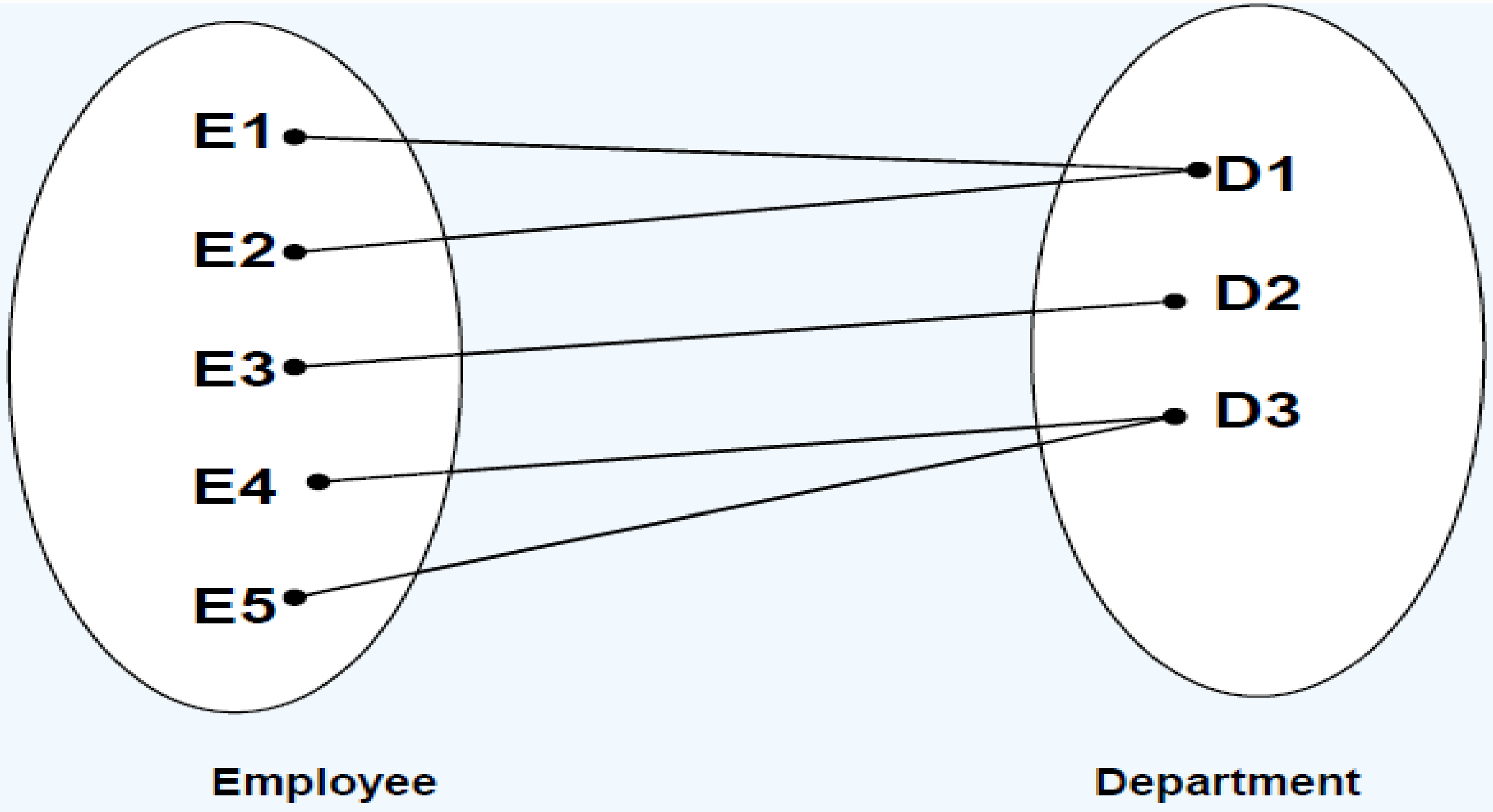
One instance of entity type Person
is related
to one instance of the entity type Chair.

Cardinality – One - To - Many



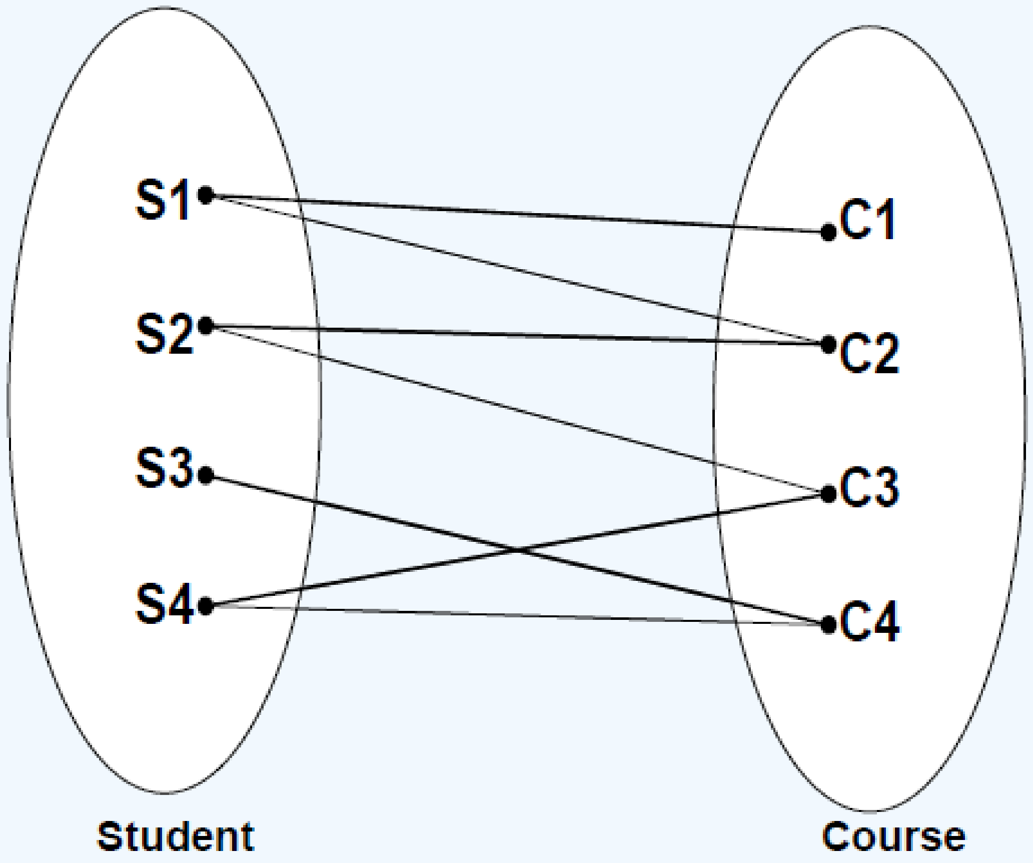
One instance of entity type Organization
is related
to multiple instances of entity type Employee

Cardinality – Many - To - One



Reverse of the One to Many relationship.

Cardinality – Many - To - Many



Multiple instances of one Entity are related to multiple instances of another Entity.

Participation Constraints

- There are two types of participation Constraints.
  - Total Participation
  - Partial Participation
- **Total Participation** : Every entity instance must be connected through the relationship to another instance of the other participating entity types
- **Partial Participation :** All instances need not participate

Example : Employee **Head-of** Department

Employee: partial

Department: total

Participation Constraints

All employees will not be head-of some department. So only few instances of employee entity participate in the above relationship. But each department will be headed by some employee. So department entity's participation is total and employee entity's participation is partial in the above relationship

Keys

Keys

- A *key* allows us to identify a set of attributes that suffice to distinguish entities from each other.
- Keys also help uniquely identify relationships, and thus distinguish relationships from each other.
- Types of Keys:
  - Super Key
  - Candidate Key
  - Primary Key
  - Alternate Keys
  - Foreign Key

Super Key

Definition:

An attribute, or group of attributes, that is sufficient to distinguish every tuple in the relation from every other one.

Example:

The combination of "SSN" and "Name" is a super key of the entity set ***customer***.

A super key may contain extraneous attributes and we are often interest in the smallest super key.

Therefore, we introduce **Candidate Key** next.

Candidate Key

Definition:

Candidate key is a set of one or more attributes whose value can uniquely identify an entity in the entity set, and any attribute in the candidate key cannot be omitted without destroying the uniqueness property of the candidate key. (It is minimal super key).

Example:

- (*SSN*, *Name*) is NOT a candidate key, because taking out "*name*" still leaves "*SSN*" which can uniquely identify an entity. "*SSN*" is a candidate key of ***customer***.
- Candidate key could have more than one attributes

Primary Key

Definition:

Primary key is the candidate key that is chosen by the database designer as the unique identifier of an entity. The database designer chooses only one candidate key as the primary key in building the system.

Example:

"SSN" and "License #" are both candidate keys of ***Driver*** entity set. The database designer can choose either one as the primary key.

In an E-R diagram, we usually underline the primary key attribute(s).

Overall,

- Super key is the broadest unique identifier.
- Candidate key is a subset of super key.
- Primary key is a subset of candidate key.

In practice, we would first look for super keys. Then we look for candidate keys based on experience and common sense. If there is only one candidate key, it naturally will be designated as the primary key. If we find more than one candidate key, then we can designate any one of them as primary key.

Alternate Keys:

The candidate keys that are not selected as primary keys, are called as alternate keys.

Foreign Keys

Definition:

A Foreign Key is a set of attribute (s) whose values are required to match values of a Candidate key in the same or another table.

Example

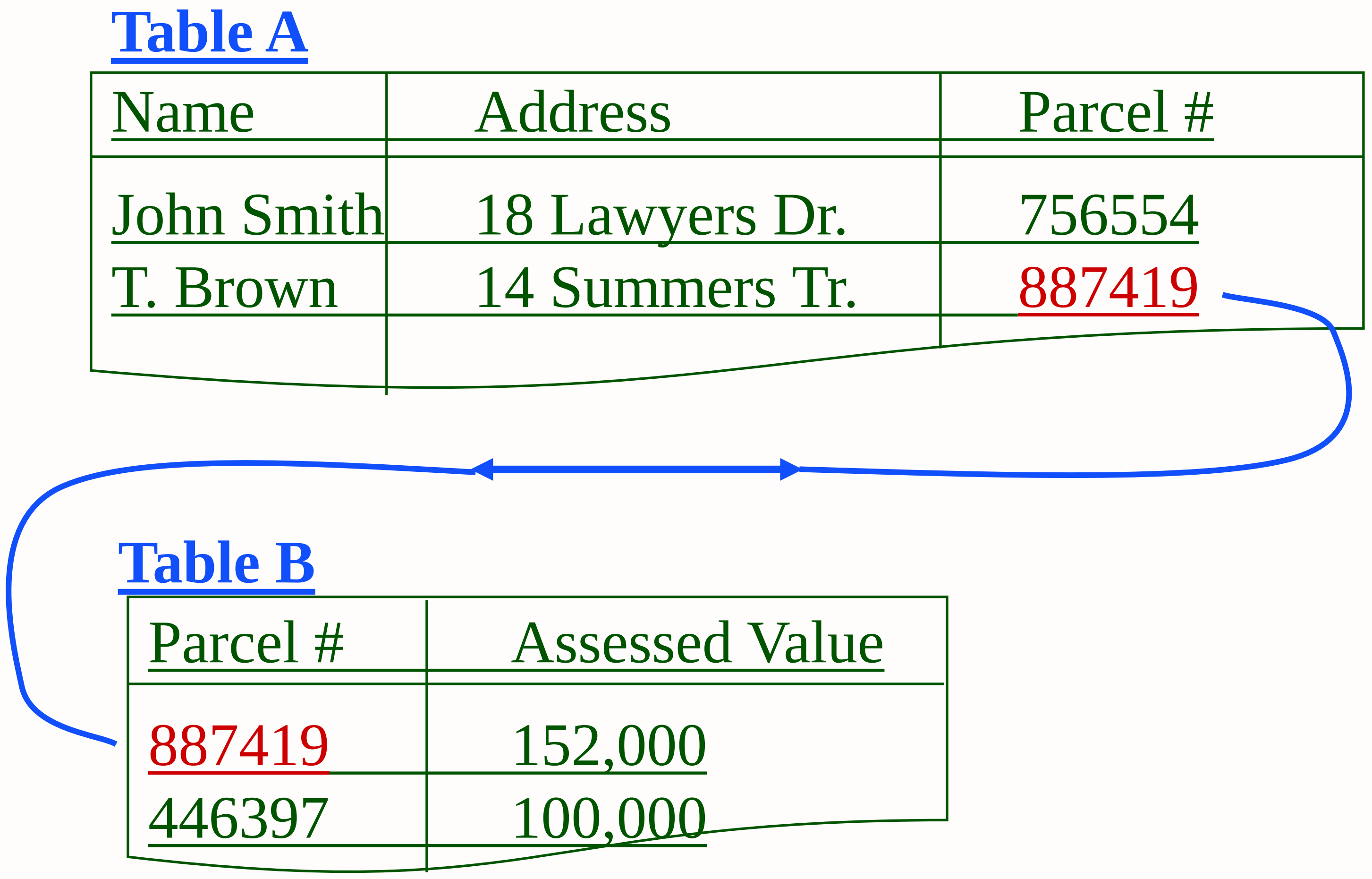


Table A

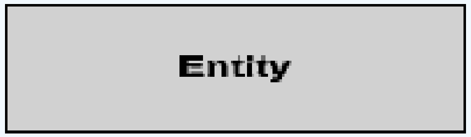
Name	Address	Parcel #
John Smith	18 Lawyers Dr.	756554
T. Brown	14 Summers Tr.	887419

Table B

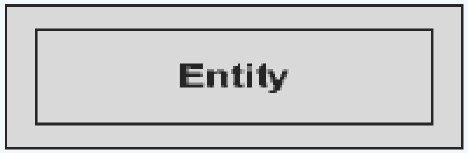
Parcel #	Assessed Value
887419	152,000
446397	100,000

ER Modeling -Notations

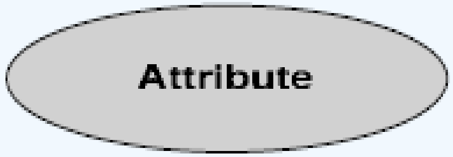
ER Modeling -Notations



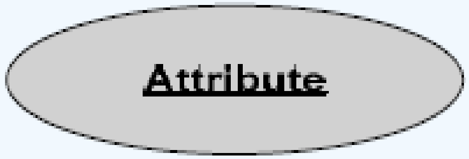
An Entity is an object or concept about which business user wants to store information.



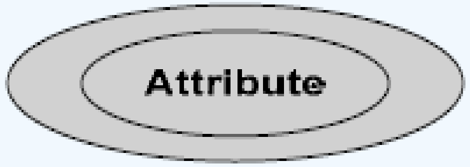
A weak Entity is dependent on another Entity to exist. Example Order Item depends upon Order Number for its existence. Without Order Number it is impossible to identify Order Item uniquely.



Attributes are the properties or characteristics of an Entity

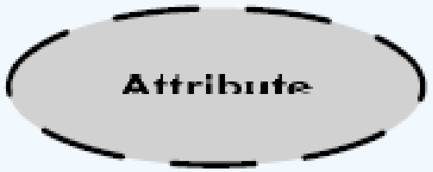


A key attribute is the unique, distinguishing characteristic of the Entity

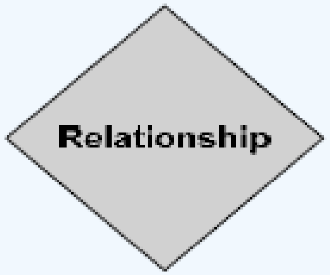


A multivalued attribute can have more than one value. For example, an employee Entity can have multiple skill values.

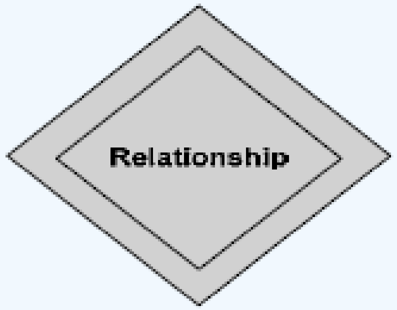
ER Modeling -Notations



A derived attribute is based on another attribute. For example, an employee's monthly salary is based on the employee's basic salary and House rent allowance.

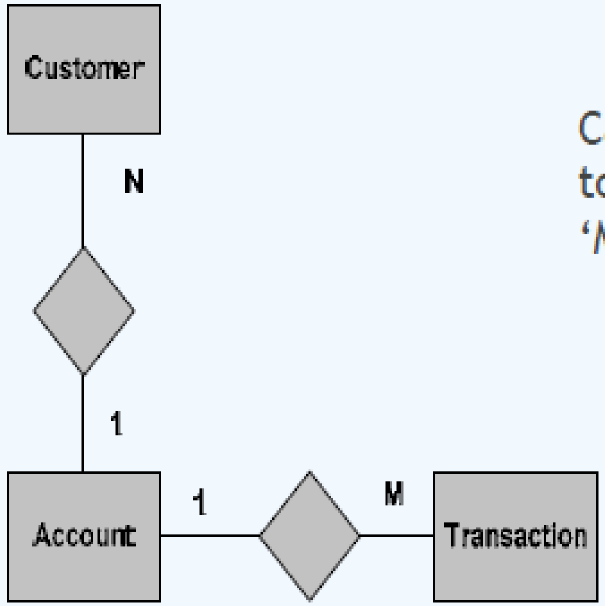


Relationships illustrate how two entities share information in the database structure.

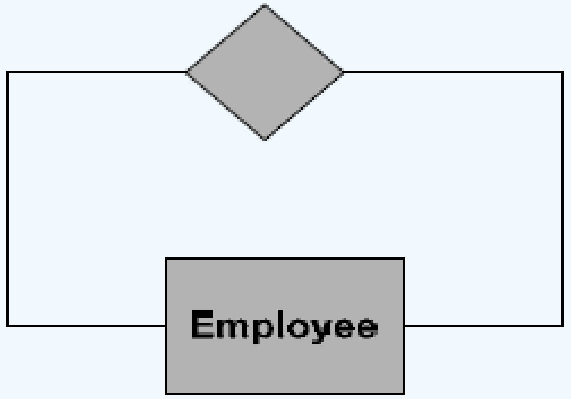


To connect a weak Entity with others, you should use a weak relationship notation.

ER Modeling -Notations

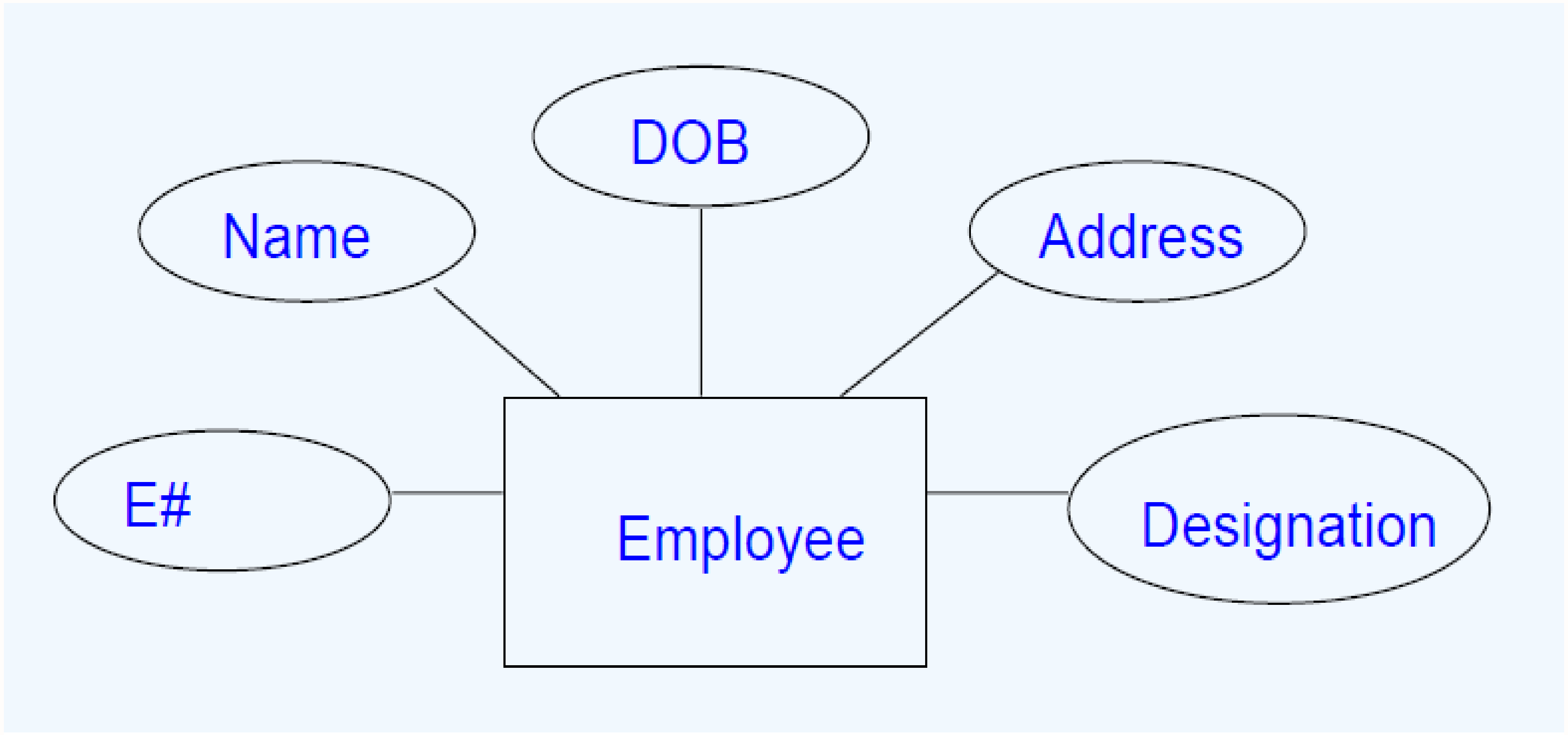


Cardinality specifies how many instances of an Entity relate to one instance of another Entity. M,N both represent 'MANY' and 1 represents 'ONE' Cardinality



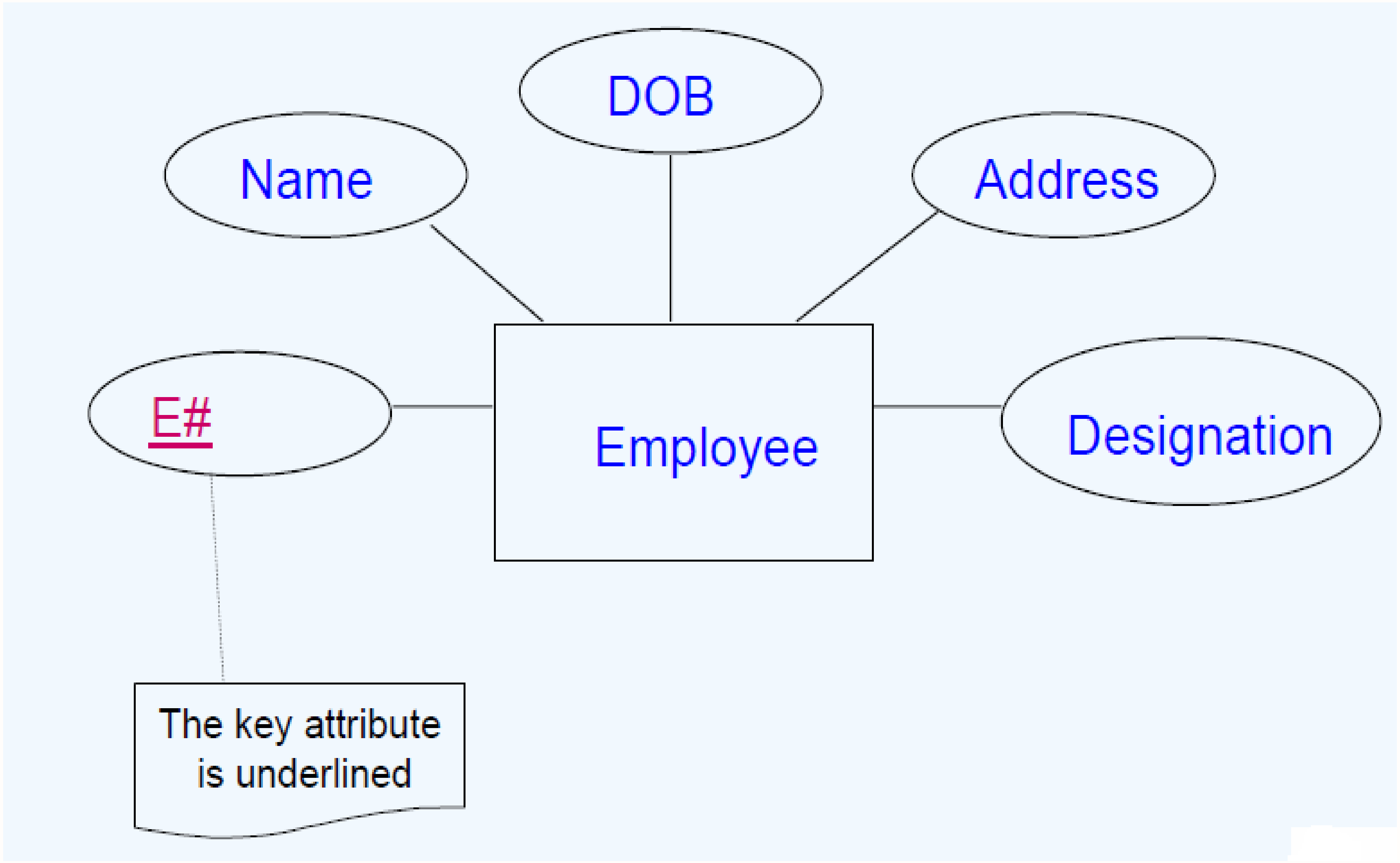
In some cases, entities can be self-linked. For example, employees can supervise other employees

Attributes

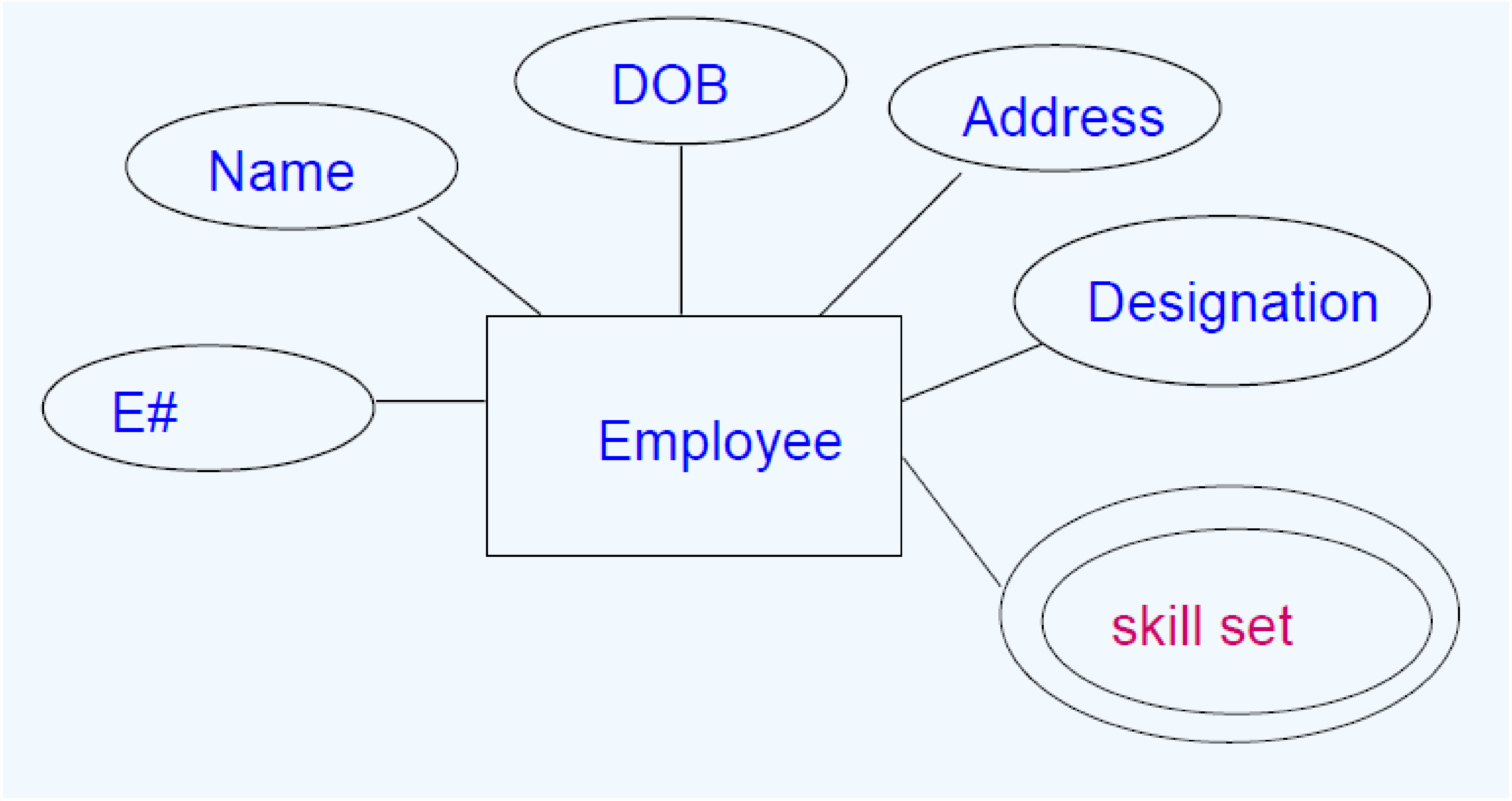


Represented by ellipses connected to theentity type by straight lines

Key attribute

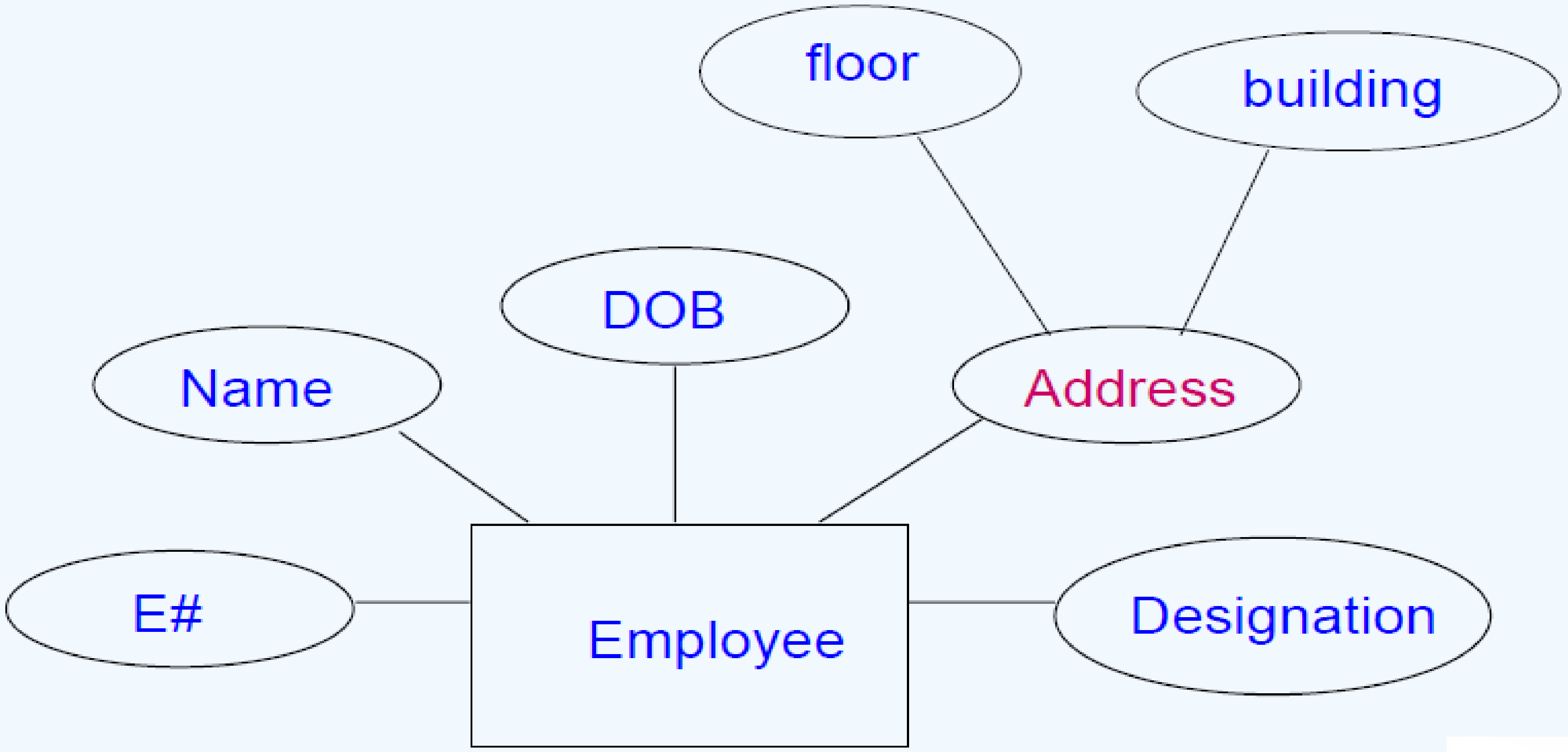


Multivalued Attribute



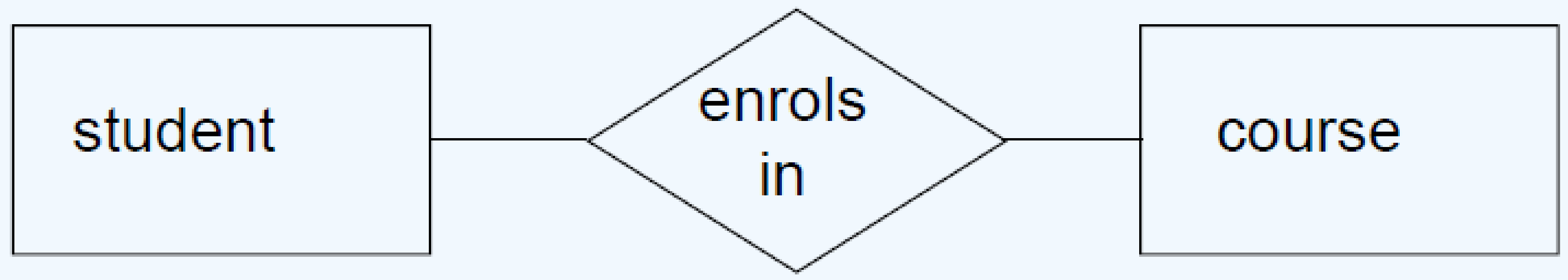
Indicated by a double lined ellipse as shown in figure

Composite attribute



Represented by an ellipse from which other ellipses emanate and represent the component attributes E.g Address

Relationship



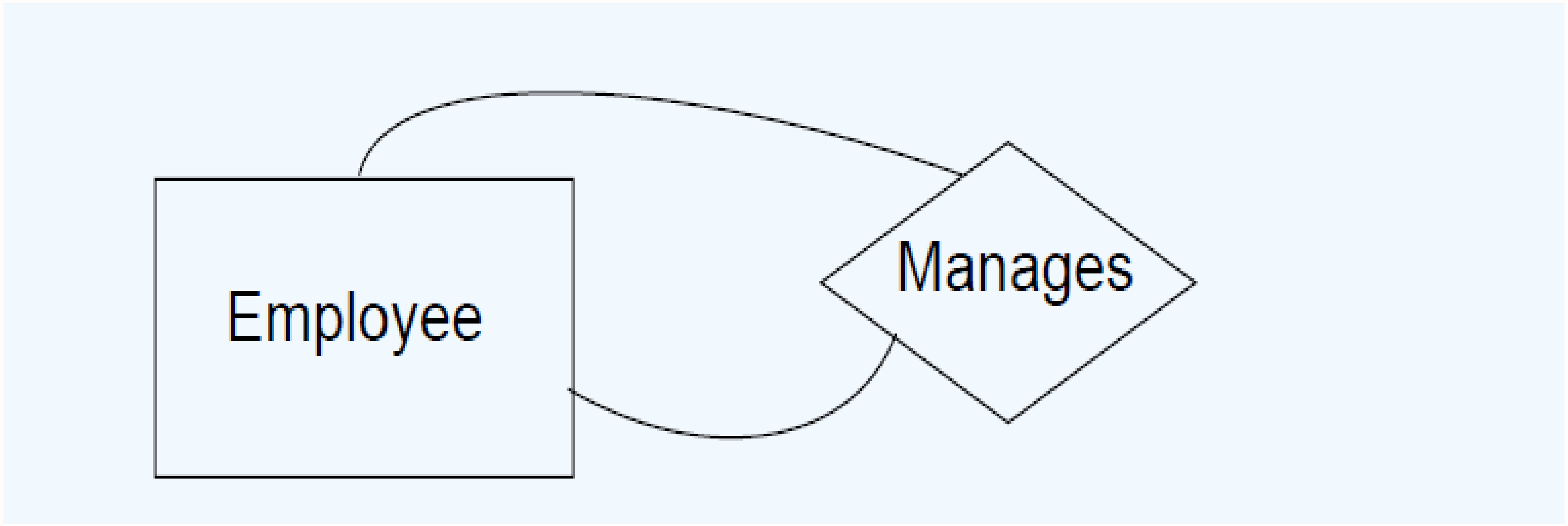
- A relationship is represented as a diamond between two entity types.

- It has a label that explains the relationship. Usually the convention to read the ER diagram from top to bottom and from right to left.

- So the relationship name is so chosen as to make sense when read from left to right.

- The relationship above is read as student enrolls - in course

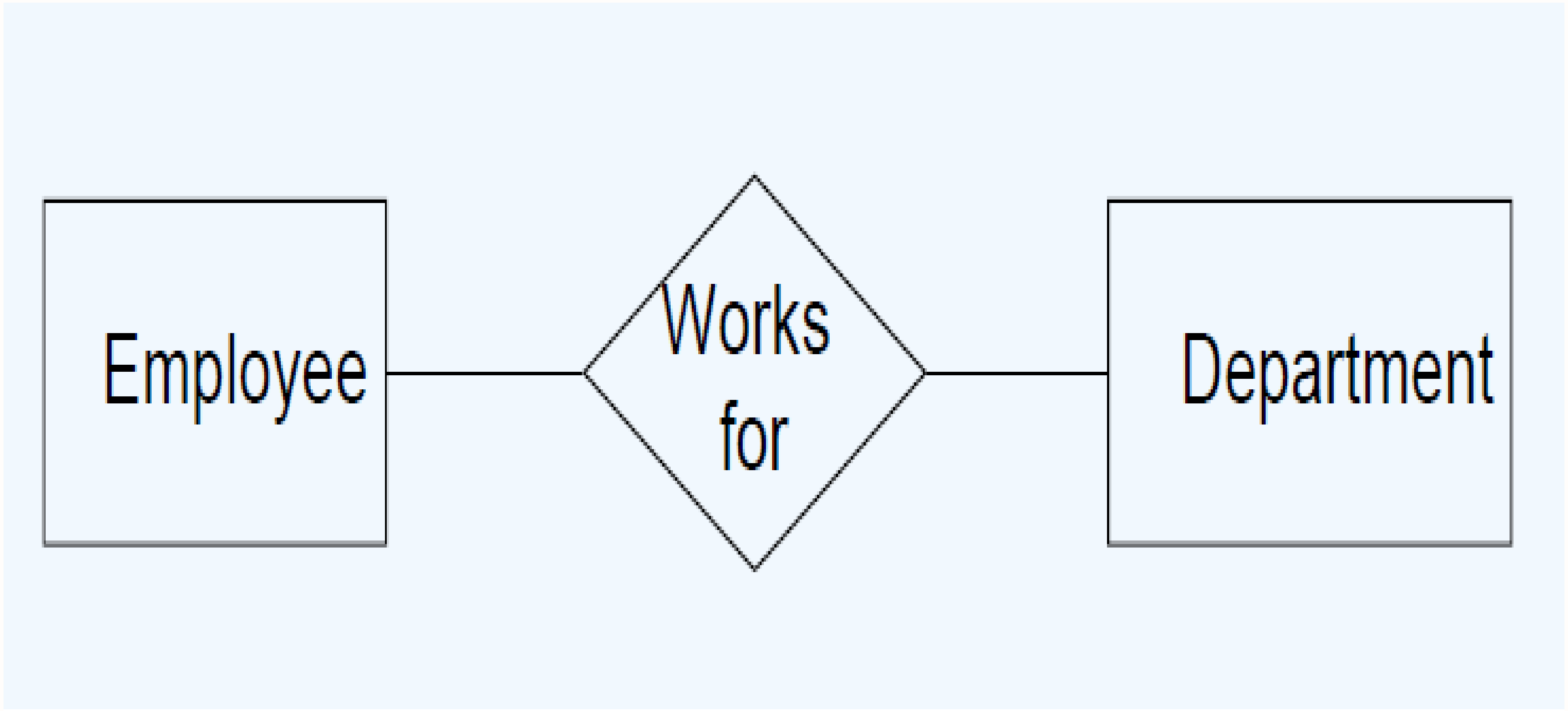
Unary Relationship



- A unary relationship is represented as a diamond which connects one entity to itself as a loop.

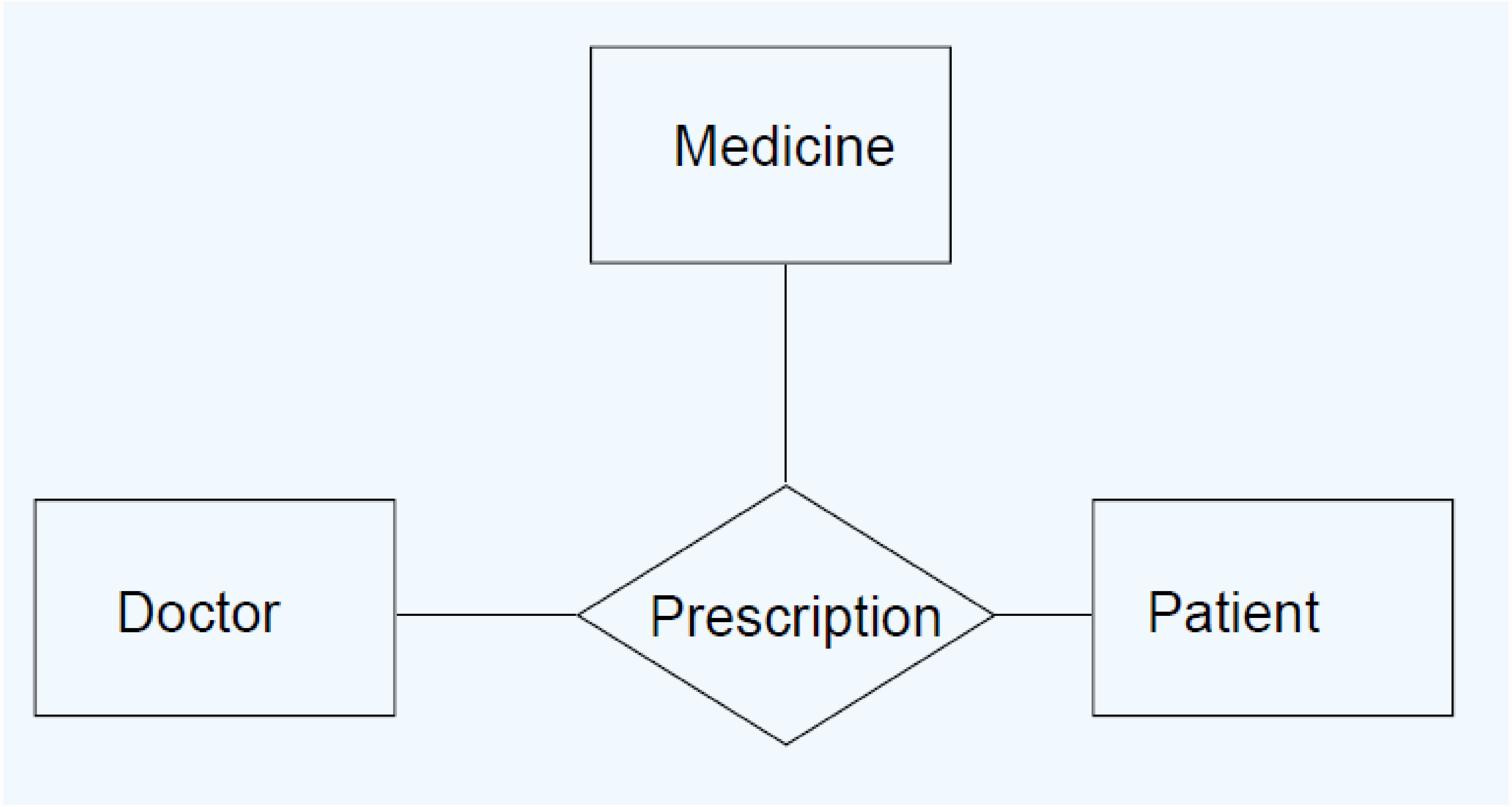
- the relationship above means, some instances of employee manage other instances of Employee.

Binary Relationship



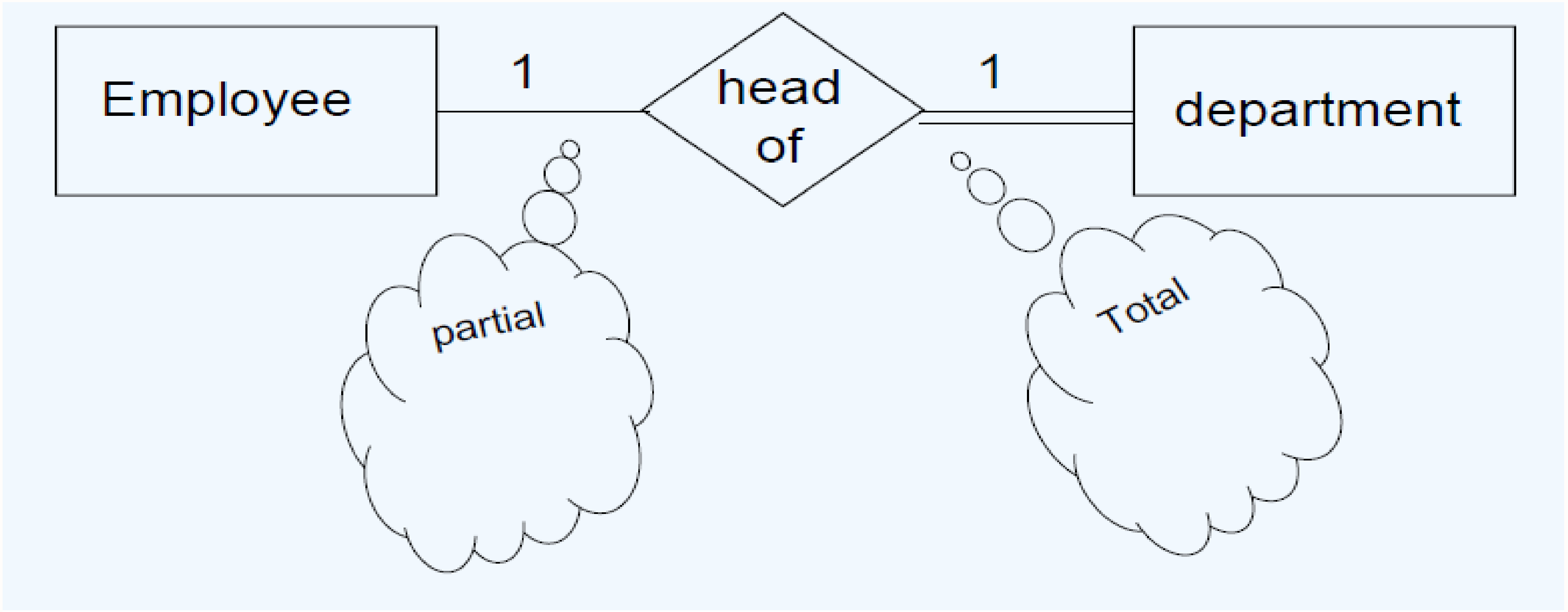
A relationship between two entity types

Ternary Relationship



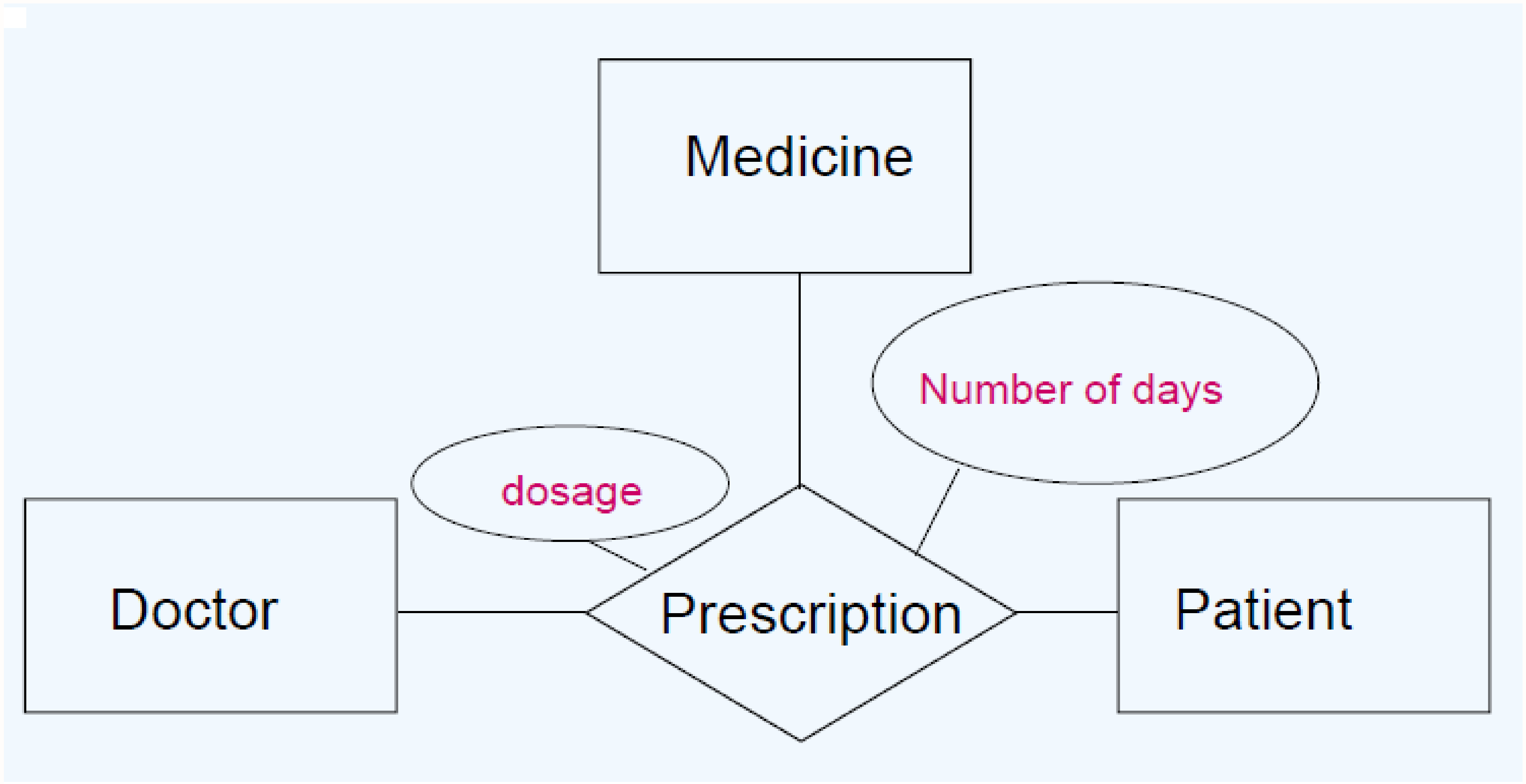
A relationship connecting three entity types.

Relationship Participation



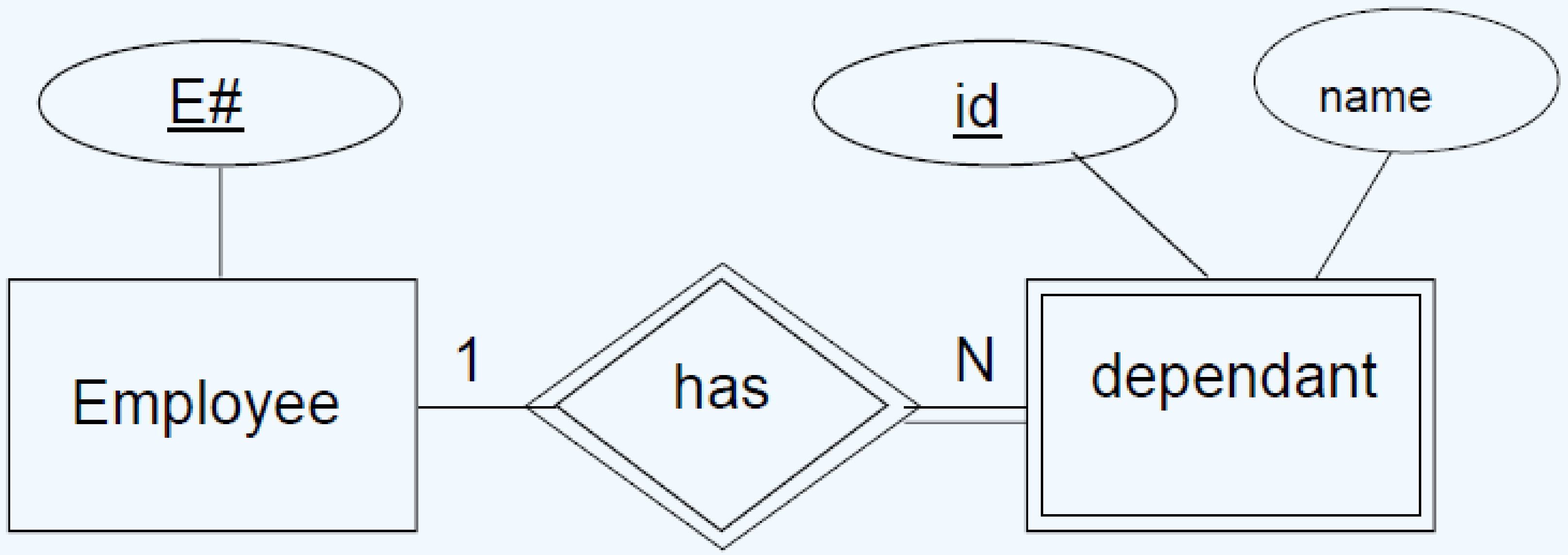
- All instances of the entity type Employee don't participate in the relationship, Head-of .
- Every employee doesn't head a department. So, employee entity type is said to be partially participate in the relationship.
- But, every department, would be headed by some employee.
- So, all instances of the entity type Department participate in this relationship. So, we say that it is total participation from the depatment side.

Attributes of a Relationship



These attributes best describe the relationship rather than any individual entity.

Weak Entity



The dependant entity is represented by a double lined rectangle and the identifying relationship by a double lined diamond

The identifying relationship is the one which relates the weak entity with the strong entity on which it depends.

ER Modeling –Case Study

Case study- ER model for a College DB

Assumptions :

- A college contains many departments
- Each department can offer any number of courses
- Many instructors can work in a department
- An instructor can work only in one department
- For each department there is a Head
- An instructor can be head of only one department
- Each instructor can take any number of courses
- A course can be taken by only one instructor
- A student can enroll for any number of courses
- Each course can have any number of students

Steps in ER Modeling

- Identify the Entities
- Find relationships
- Identify the key attributes for every Entity
- Identify other relevant attributes
- Draw complete E-R diagram with all attributes including Primary Key
- Review your results with your Business users

Step 1: Identify the Entities

- DEPARTMENT
- STUDENT
- COURSES
- INSTRUCTOR

Step 2: Find the relationships

- One course is enrolled by multiple students and one student enrolls for multiple courses, hence the cardinality between course and student is Many to Many.
- The department offers many courses and each course belongs to only one department, hence the cardinality between department and course is One to Many.
- One department has multiple instructors and one instructor belongs to one and only one department , hence the cardinality between department and instructor is one to Many.
- Each department there is a "Head of department" and one instructor is "Head of department ",hence the cardinality is one to one .
- One course is taught by only one instructor, but the instructor teaches many courses, hence the cardinality between course and instructor is many to one.

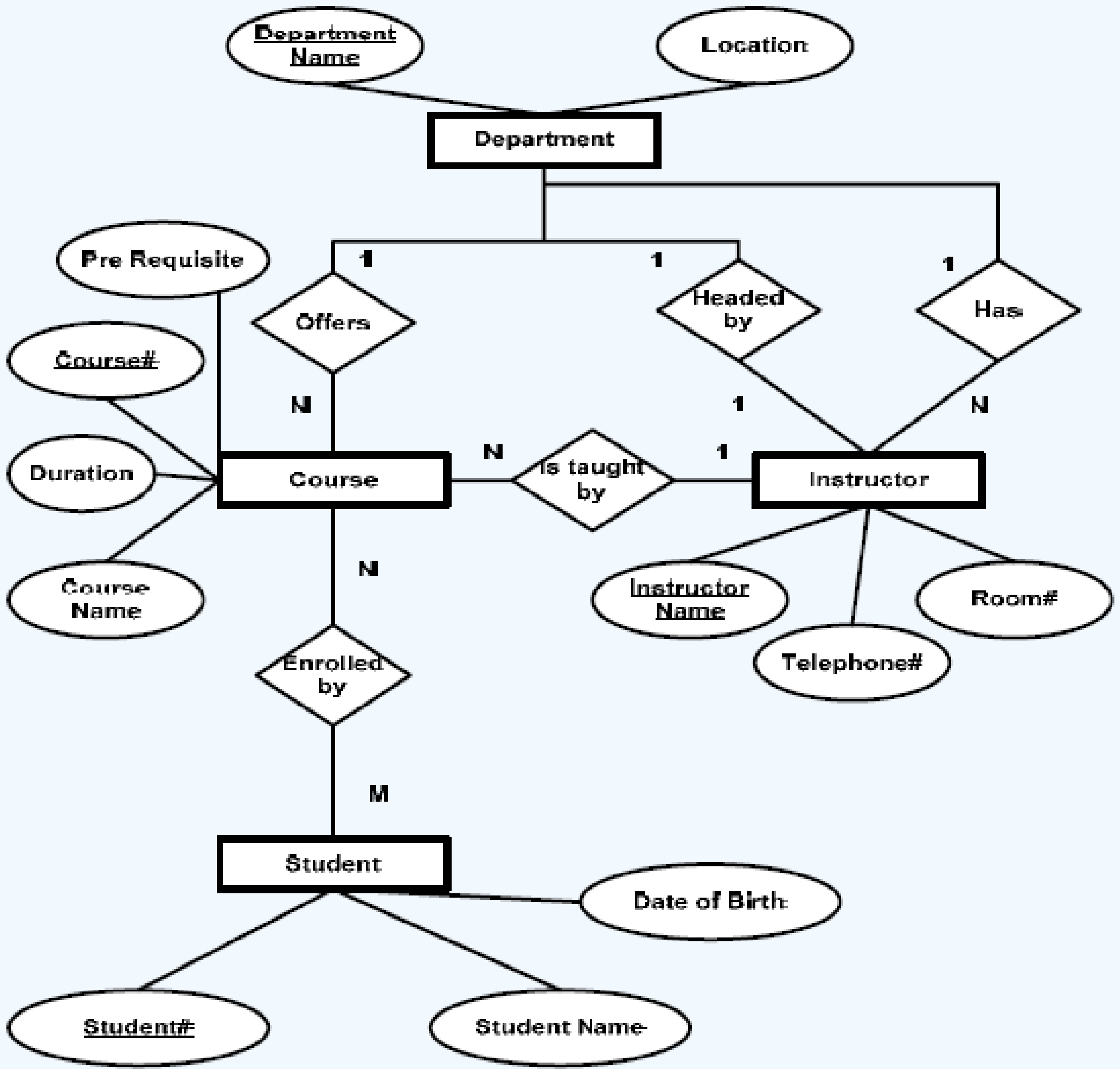
❑ Step 3: Identify the key attributes

- Deptname is the key attribute for the Entity “Department”, as it identifies the Department uniquely.
- Course# (CourseId) is the key attribute for “Course” Entity.
- Student# (Student Number) is the key attribute for “Student” Entity.
- Instructor Name is the key attribute for “Instructor” Entity.

❑ Step 4: Identify other relevant attributes

- For the department entity, the relevant attribute is location
- For course entity, course name, duration, prerequisite
- For instructor entity, room#, telephone#
- For student entity, student name, date of birth

Step 5: Draw complete E-R diagram with all attributes including Primary Key



Case Study – Banking Business Scenario

Assumptions :

- There are multiple banks and each bank has many branches.
- Each branch has multiple customers
- Customers have various types of accounts
- Some Customers also had taken different types of loans from these bank branches
- One customer can have multiple accounts and Loans

Steps in ER Modeling

- Identify the Entities
- Find relationships
- Identify the key attributes for every Entity
- Identify other relevant attributes
- Draw complete E-R diagram with all attributes including Primary Key
- Review your results with your Business users

Step 1: Identify the Entities

- BANK
- BRANCH
- LOAN
- ACCOUNT
- CUSTOMER

Step 2: Find the relationships

- One Bank has many branches and each branch belongs to only one bank, hence the cardinality between Bank and Branch is One to Many.
- One Branch offers many loans and each loan is associated with one branch, hence the cardinality between Branch and Loan is One to Many.
- One Branch maintains multiple accounts and each account is associated to one and only one Branch, hence the cardinality between Branch and Account is One to Many
- One Loan can be availed by multiple customers, and each Customer can avail multiple loans, hence the cardinality between Loan and Customer is Many to Many.
- One Customer can hold multiple accounts, and each Account can be held by multiple Customers, hence the cardinality between Customer and Account is Many to Many

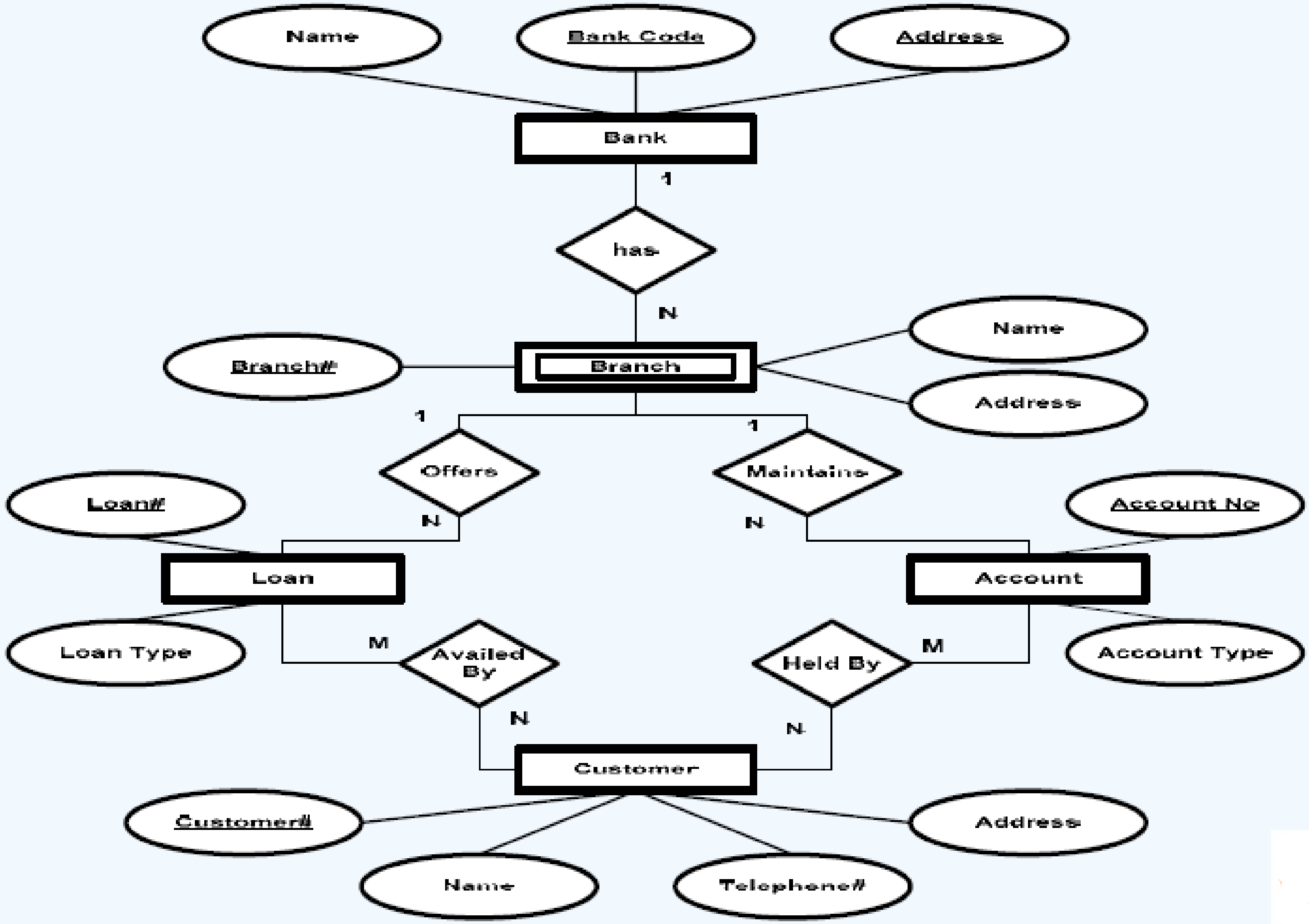
Step 3: Identify the key attributes

- BankCode (Bank Code) is the key attribute for the Entity “Bank”, as it identifies the bank uniquely.
- Branch# (Branch Number) is the key attribute for “Branch” Entity.
- Customer# (Customer Number) is the key attribute for “Customer” Entity.
- Loan# (Loan Number) is the key attribute for “Loan” Entity.
- Account No (Account Number) is the key attribute for “Account” Entity.

Step 4: Identify other relevant attributes

- For the “Bank” Entity, the relevant attributes other than “BankCode” would be “Name” and “Address”.
- For the “Branch” Entity, the relevant attributes other than “Branch#” would be “Name” and “Address”.
- For the “Loan” Entity, the relevant attribute other than “Loan#” would be “Loan Type”.
- For the “Account” Entity, the relevant attribute other than “Account No” would be “Account Type”.
- For the “Customer” Entity, the relevant attributes other than “Customer#” would be “Name”, “Telephone#” and “Address”.

Step 5: Draw complete E-R diagram with all attributes including Primary Key



Merits and Demerits of ER Modeling

- **Merits :**
  - Easy to understand. Represented in Business Users Language. Can be understood by non-technical specialist.
  - Intuitive and helps in Physical Database creation.
  - Can be generalized and specialized based on needs.
  - Can help in database design.
  - Gives a higher level description of the system.
- **Demerits :**
  - Physical design derived from E-R Model may have some amount of ambiguities or inconsistency.
  - Sometime diagrams may lead to misinterpretations

Extended ER Features

Motivation

- Since 1980s there has been an increase in emergence of new database applications with more demanding requirements.
- Basic concepts of ER modeling are not sufficient to represent requirements of newer, more complex applications, for example:
  - Design and manufacturing (CAD/CAM)
  - Geographic Information Systems (GIS)
  - Telecommunications
- These applications had more complex requirements
- Response is development of additional ‘semantic’ modeling concepts.
- Result$\Rightarrow$ Enhanced (or Extended) ER data model

Outline

- Subclasses, Superclasses and Inheritance
- Specialization and Generalization
- Aggregation

Enhanced-ER (EER) Model Concepts

- Includes all modeling concepts of basic ER
- Additional concepts: subclasses/superclasses, specialization/generalization, aggregation, categories, attribute inheritance
- The resulting model is called the enhanced-ER or Extended ER (E2R or EER) model
- It is used to model applications more completely and accurately if needed
- It includes some object-oriented concepts, such as inheritance

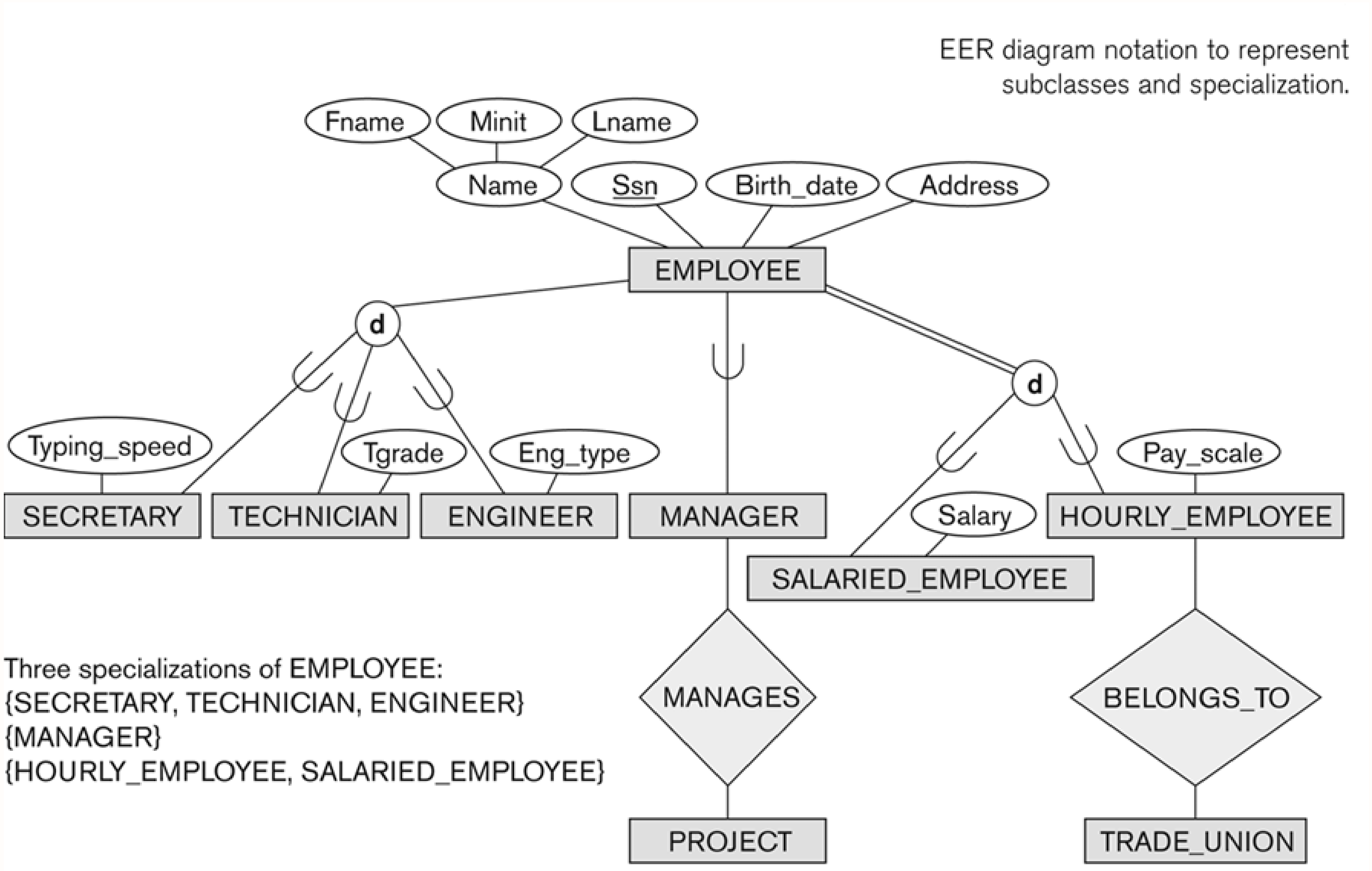
Subclass and Superclass

- An entity type may have additional meaningful subgroupings of its entities
- Example: EMPLOYEE may be further grouped into SECRETARY, ENGINEER, MANAGER, TECHNICIAN, SALARIED_EMPLOYEE, HOURLY_EMPLOYEE,…
  - Each of these groupings is a subset of EMPLOYEE entities
  - Each is called a subclass of EMPLOYEE
  - EMPLOYEE is the superclass for each of these subclasses
- These are called superclass/subclass relationships.
- Example: EMPLOYEE/SECRETARY, EMPLOYEE/TECHNICIAN

Subclass and Superclass

- These are also called IS-A relationships (SECRETARY IS-A EMPLOYEE, TECHNICIAN IS-A EMPLOYEE, …).
- Note: An entity that is member of a subclass represents the same real-world entity as some member of the superclass
  - The Subclass member is the same entity in a distinct specific role
  - An entity cannot exist in the database merely by being a member of a subclass; it must also be a member of the superclass
  - A member of the superclass can be optionally included as a member of any number of its subclasses
- Example: A salaried employee who is also an engineer belongs to the two subclasses ENGINEER and SALARIED_EMPLOYEE
  - It is not necessary that every entity in a superclass be a member of some subclass

Subclass and Superclass



EER diagram notation to represent subclasses and specialization.

Three specializations of EMPLOYEE:
{SECRETARY, TECHNICIAN, ENGINEER}
{MANAGER}
{HOURLY_EMPLOYEE, SALARIED_EMPLOYEE}

Attribute Inheritance in Superclass / Subclass Relationships

- An entity that is member of a subclass *inherits* all attributes of the entity as a member of the superclass.
- It also inherits all relationships.
- Example:
  - In the previous slide, SECRETARY (as well as TECHNICIAN and ENGINEER) inherit the attributes Name, SSN, …, from EMPLOYEE
  - Every SECRETARY entity will have values for the inherited attributes
  - Every SECRETARY entity will also keep all relationships

Specialization

- Top-down design process; we designate sub-groupings within an entity set that are distinctive from other entities in the set.
- These sub-groupings become lower-level entity sets that have attributes or participate in relationships that do not apply to the higher-level entity set.
- Depicted by a triangle component labeled ISA (E.g., instructor "is a" person).
- Is the process of defining a set of subclasses of a superclass
- The set of subclasses is based upon some distinguishing characteristics of the entities in the superclass
- Example: {SECRETARY, ENGINEER, TECHNICIAN} is a specialization of EMPLOYEE based upon *job type*.
  - May have several specializations of the same superclass

Specialization

- Example: Another specialization of EMPLOYEE based in *method of pay* is {SALARIED_EMPLOYEE, HOURLY_EMPLOYEE}.
  - Superclass/subclass relationships and specialization can be diagrammatically represented in EER diagrams
  - Attributes of a subclass are called specific attributes. For example, TypingSpeed of SECRETARY
  - The subclass can participate in specific relationship types. For example, BELONGS_TO of HOURLY_EMPLOYEE

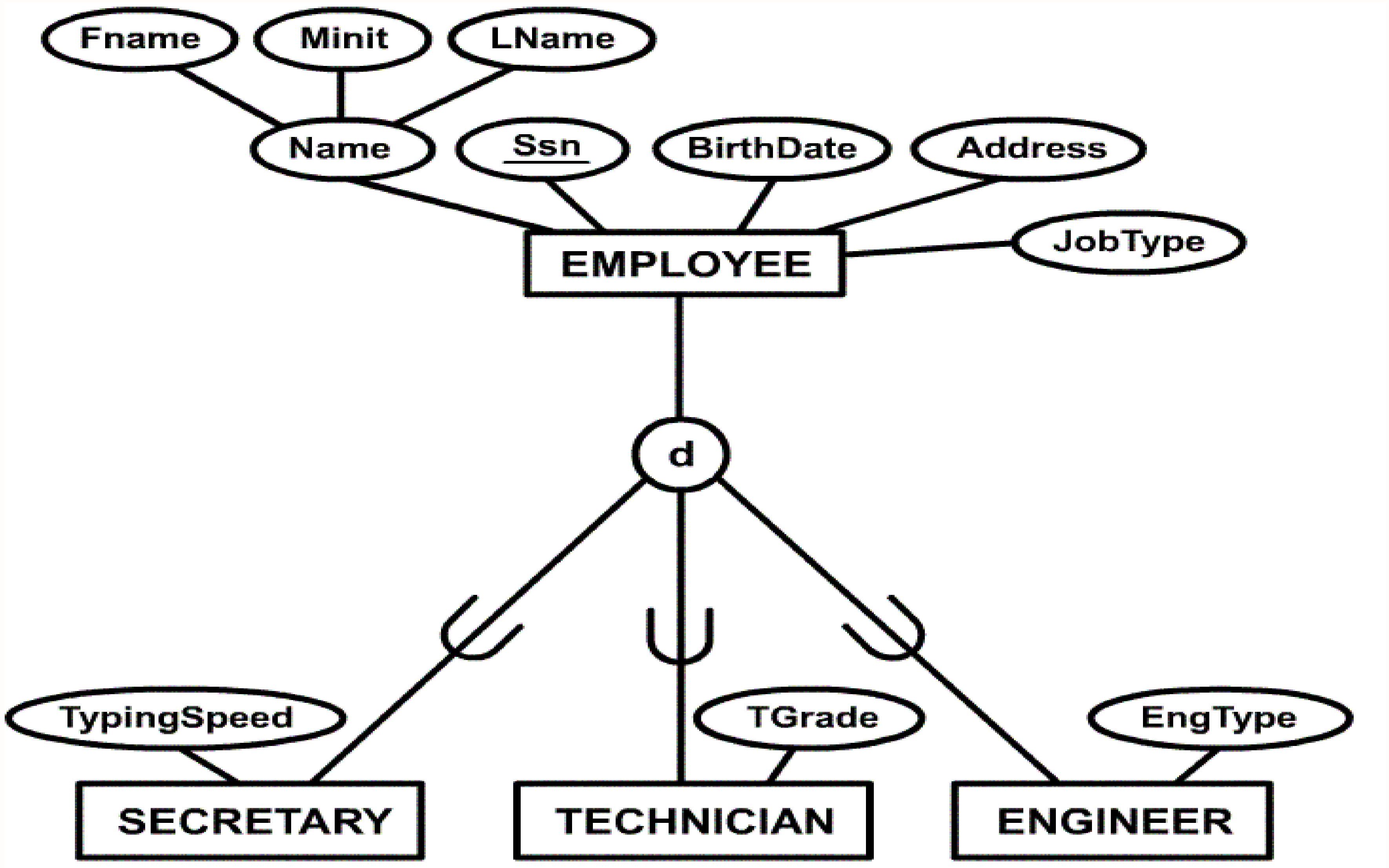
Specialization

- There are two major reasons for including class/subclass relationship and specialization in a data model:

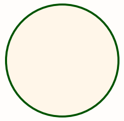
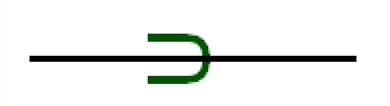
  1. Certain attributes may apply to some but not all entities of the superclass (secretary subclass has local attribute Typing speed where engineer has eng_type)

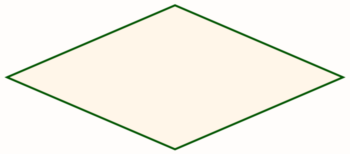
  2. Some relationship types may be participate in only by entities that are members of the subclass (Hourly_employees are related to Trade_union via Belongs_to)

Example of a Specialization



Model Shapes

- When you have more than one subclass based on the same defining attribute (*JobType*), use 
  - The circle can be empty or it can contain a symbol **d** (for disjointness) or **o** (for overlapping)
- To show class/subclass relationships, use 
  - Used for relationships between entity types
- To show relationship between two different entity types, use



Specialization

- In summary, the specialization process allows us to do the following:
  - Define a set of subclass of an entity type
  - Establish additional specific attributes with each subclass
  - Establish additional specific relationship types between each subclass and other entity types or other subclasses

Generalization

- The reverse of the specialization process
- A bottom-up design process – combine a number of entity sets that share the same features into a higher-level entity set.
- Several classes with common features are generalized into a superclass; original classes become its subclasses.
- Specialization and generalization are simple inversions of each other; they are represented in an E-R diagram in the same way.
- The terms specialization and generalization are used interchangeably.

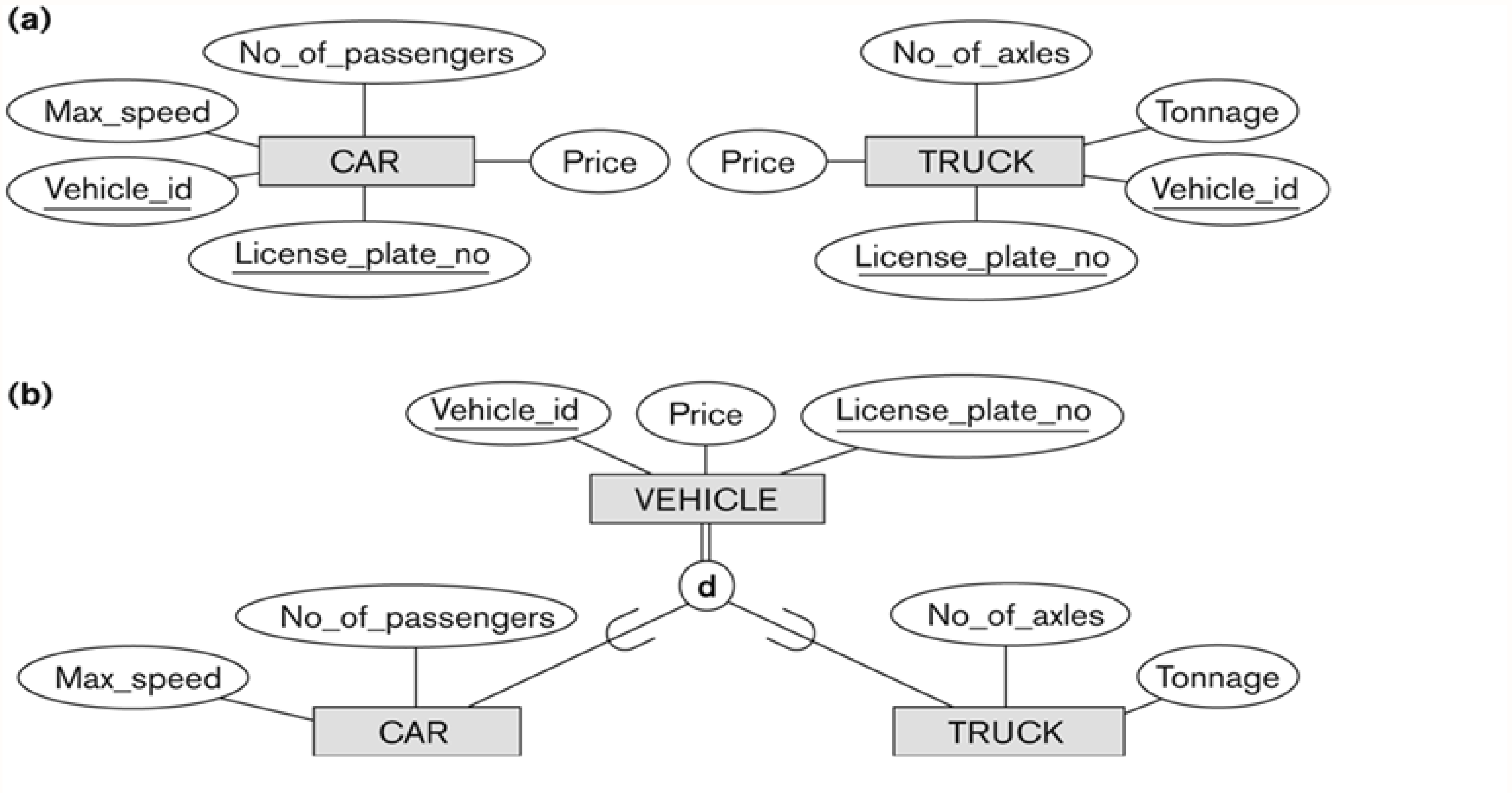
Generalization

- Example: CAR, TRUCK generalized into VEHICLE; both CAR, TRUCK become subclasses of the superclass VEHICLE.
  - We can view {CAR, TRUCK} as a specialization of VEHICLE
  - Alternatively, we can view VEHICLE as a generalization of CAR and TRUCK

Generalization and Specialization

- Diagrammatic notation sometimes used to distinguish between generalization and specialization
  - Arrow pointing to the generalized superclass represents a generalization
  - Arrows pointing to the specialized subclasses represent a specialization
  - We do not use this notation because it is often subjective as to which process is more appropriate for a particular situation
  - We advocate not drawing any arrows in these situations
- Data Modeling with Specialization and Generalization
  - A superclass or subclass represents a set of entities
  - Shown in rectangles in EER diagrams (as are entity types)
  - Sometimes, all entity sets are simply called classes, whether they are entity types, superclasses, or subclasses

Example of Generalization



Generalization. (a) Two entity types, CAR and TRUCK.
(b) Generalizing CAR and TRUCK into the superclass VEHICLE.

Example of Specialization and Generalization

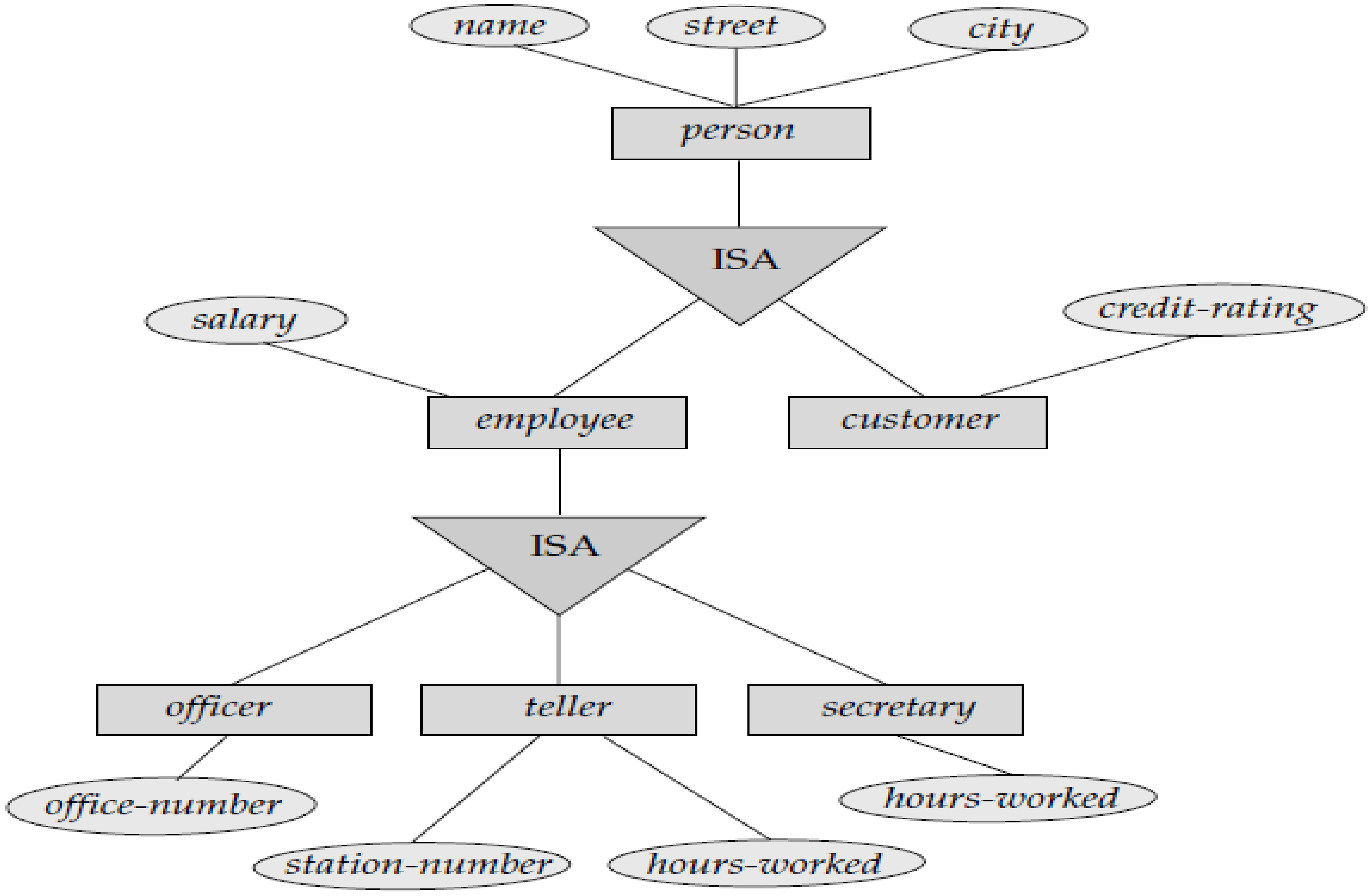


Figure Specialization and generalization.

Constraints on Specialization and Generalization

❑ **Constraints on Subclass Membership**

- If we can determine exactly those entities that will become members of each subclass by a condition, the subclasses are called ***predicate-defined*** **(or condition-defined)** subclasses
  - Condition is a constraint that determines subclass members
  - Display a predicate-defined subclass by writing the predicate condition next to the line attaching the subclass to its superclass
  - Example: all customers over 65 years are members of senior-citizen entity set; senior-citizen ISA person.

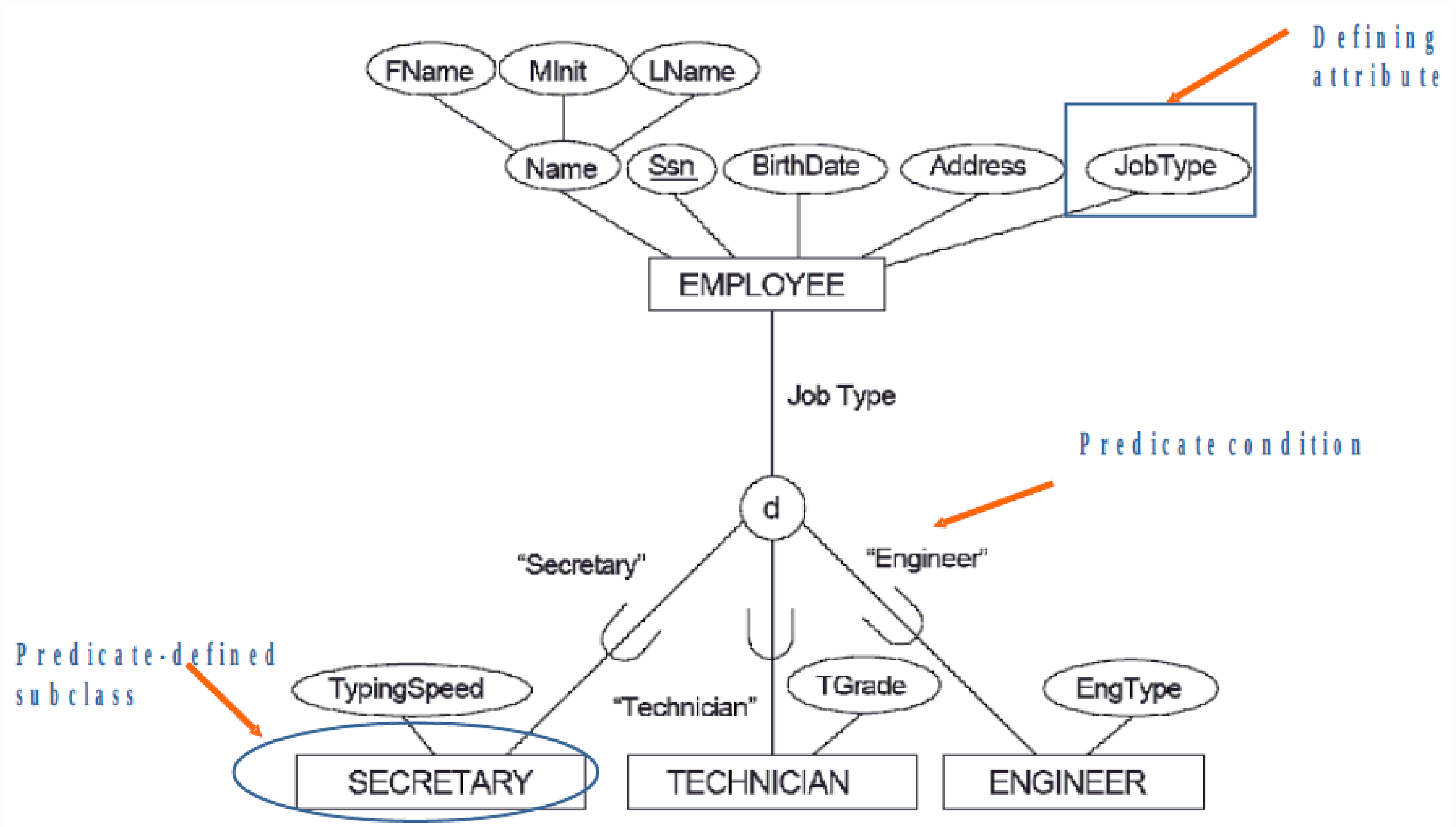
Constraints on Specialization and Generalization

- If all subclasses in a specialization have membership condition on same attribute of the superclass, specialization is called an ***attribute defined*-specialization**
  - Attribute is called the defining attribute of the specialization
  - Example: JobType is the defining attribute of the specialization {SECRETARY, TECHNICIAN, ENGINEER} of EMPLOYEE

Constraints on Specialization and Generalization

- If no condition determines membership, the subclass is called ***user-defined***
  - Membership in a subclass is determined by the database users by applying an operation to add an entity to the subclass
  - Membership in the subclass is specified individually for each entity in the superclass by the user
  - Example : Consider an entity CELEBRI TY. A celebrity could be a film star, a politician, a newsreader or a player. Thus the assignment of the entities of type CELEBRITY to a given subclass is specified explicitly by the database users when they apply the operation to add an entity to the subclass.

Displaying an attribute-defined specialization in EER diagrams



Constraints on Specialization and Generalization

- ❑Two other conditions apply to a specialization /generalization:
- **Disjointness Constraint :**
  - Specifies that the subclasses of the specialization must be disjointed (an entity can be a member of at most one of the subclasses of the specialization)
  - Specified by **d** in EER diagram
  - Example : An entity type BOOK can belong to either TEXTBOOK or NOVEL, but not both.
  - An attribute defined specialization in which the defining attribute is single valued implies a disjointness constraint.

Constraints on Specialization and Generalization

- **Overlapping Constraint :**
  - If not disjointed, **overlap**; that is the same entity may be a member of more than one subclass of the specialization
  - Specified by **o** in EER diagram
  - Example : The entity types PLAYER and POLITICIAN show an overlapping constraint, as the celebrity can be a player as well as a politician.
  - Similarly, an entity of type BOOK can belong to both TEXTBOOK and LANGUAGE_BOOK since a language book can also be a prescribed textbook in a university.

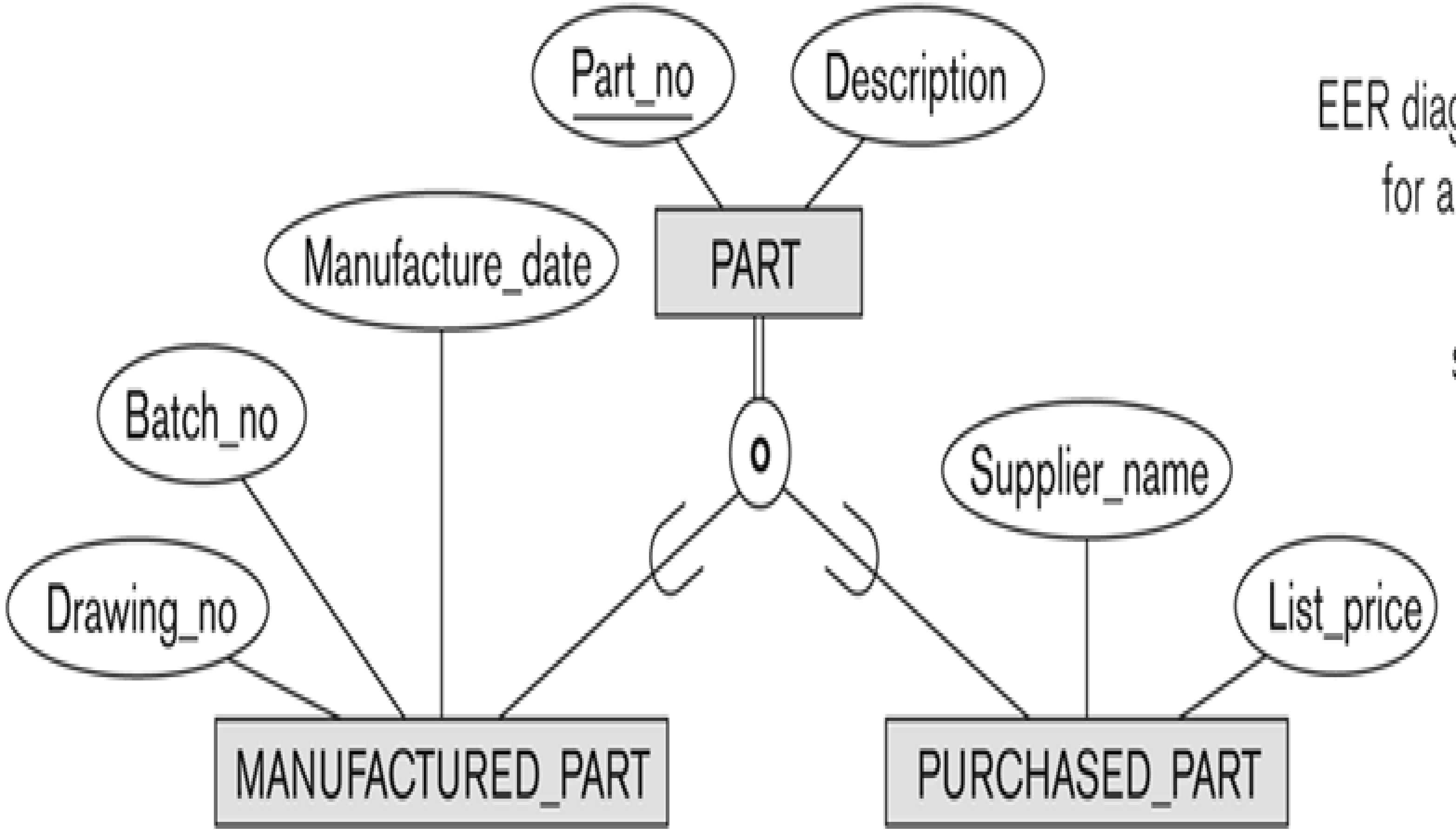
Constraints on Specialization and Generalization

- **Completeness Constraint**:
  - **Total** specifies that every entity in the superclass must be a member of some subclass in the specialization/ generalization
  - Example, the Salaried_Employee and Hourly_Employee
  - Shown in EER diagrams by a **double line**
  - **Partial** allows an entity not to belong to any of the subclasses
  - Example, Manager, Job type
  - Shown in EER diagrams by a **single line**

Constraints on Specialization and Generalization

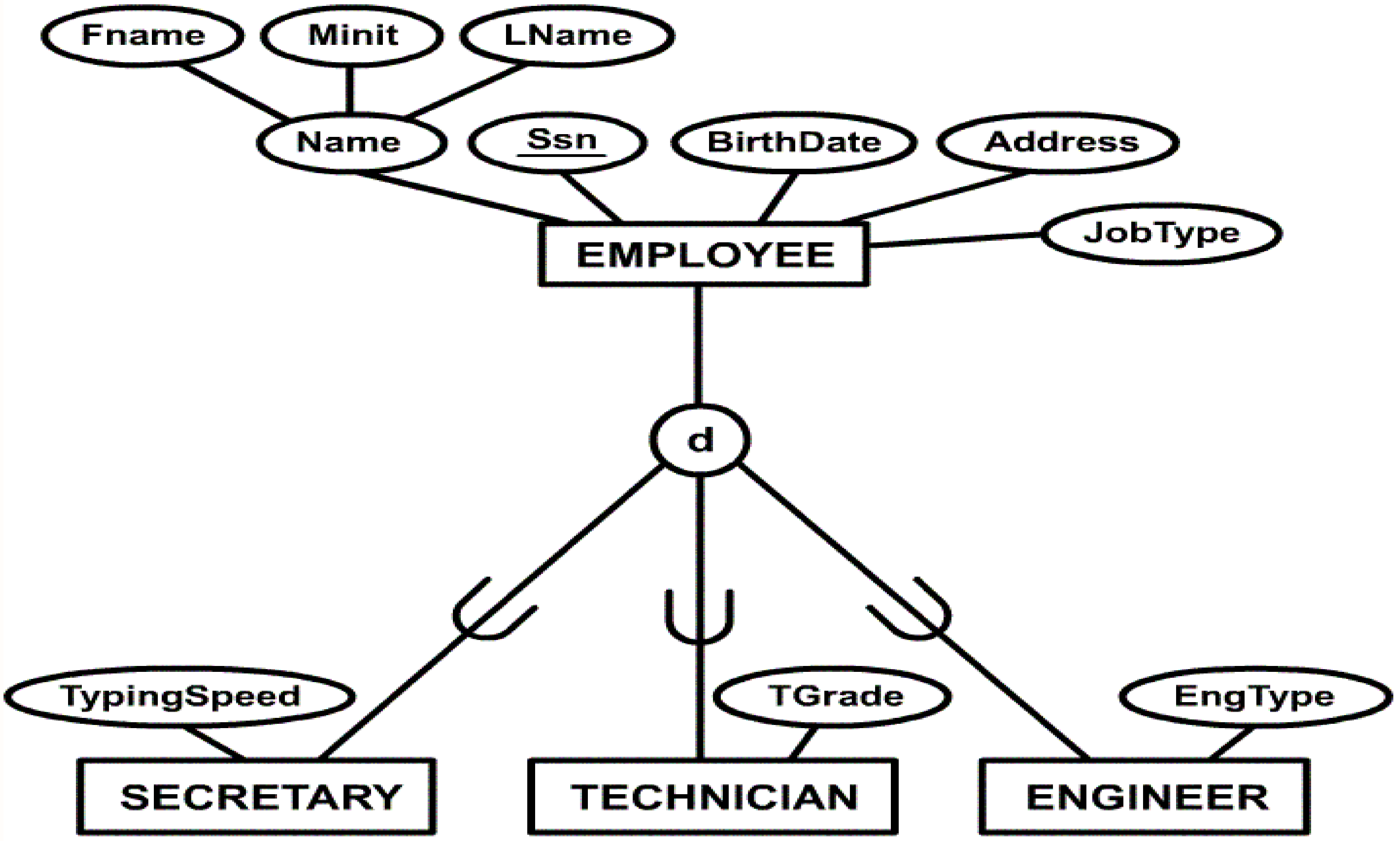
- Hence, we have four types of specialization/generalization:
  - Disjoint, total
  - Disjoint, partial
  - Overlapping, total
  - Overlapping, partial
- Note: Generalization usually is total because the superclass is derived from the subclasses.

Example of Overlapping total Specialization



EER diagram notation for an overlapping (nondisjoint) specialization.

Example of disjoint partial Specialization



Constraints on Specialization and Generalization

- Some general rules:
  - Deleting an entity from a superclass implies that it is automatically deleted from all the subclasses to which it belongs
  - Inserting an entity in a superclass of a total specialization implies that the entity is mandatorily inserted in at least one of the subclasses of the specialization

Aggregation

- One limitation of the E-R model is that it cannot express relationships among relationships.
- Consider the ternary relationship *works-on*, between an employee, branch, and job.

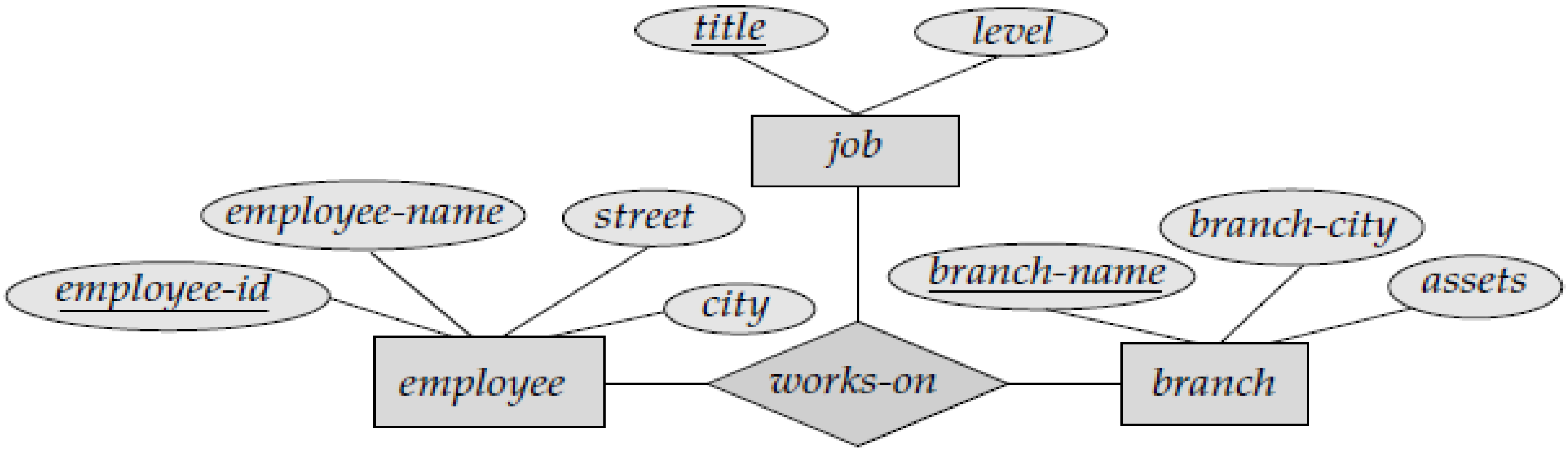


Figure E-R diagram with a ternary relationship.

Aggregation

- Now, suppose we want to record managers for tasks performed by an employee at a branch; that is, we want to record managers for (employee, branch, job) combinations.
- Let us assume that there is an entity set manager.

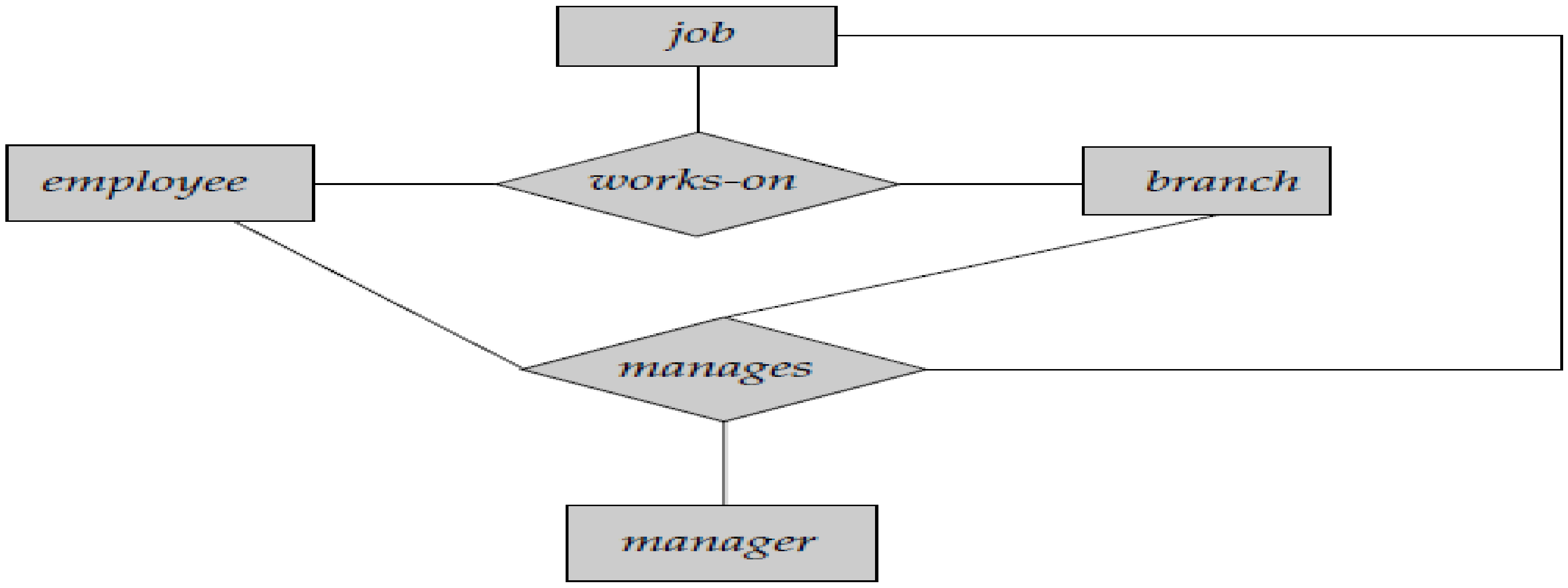


Figure E-R diagram with redundant relationships.

Aggregation

- It appears that the relationship sets works-on and manages can be combined into one single relationship set.
- Nevertheless, we should not combine them into a single relationship, since some employee, branch, job combinations many not have a manager.
- There is redundant information in the resultant figure, however, since every employee, branch, job combination in manages is also in works-on.
- If the manager were a value rather than an manager entity, we could instead make manager a multivalued attribute of the relationship works-on.

Aggregation

- But doing so makes it more difficult (logically as well as in execution cost) to find, for example, employee-branch-job triples for which a manager is responsible.
- Since the manager is a manager entity, this alternative is ruled out in any case.
- The best way to model a situation such as the one just described is to use aggregation.

Aggregation

- Aggregation is an abstraction through which relationships are treated as higher level entities.
- Thus, for our example, we regard the relationship set works-on (relating the entity sets employee, branch, and job) as a higher-level entity set called works-on.
- Such an entity set is treated in the same manner as is any other entity set.
- We can then create a binary relationship manages between works-on and manager to represent who manages what tasks.

Aggregation

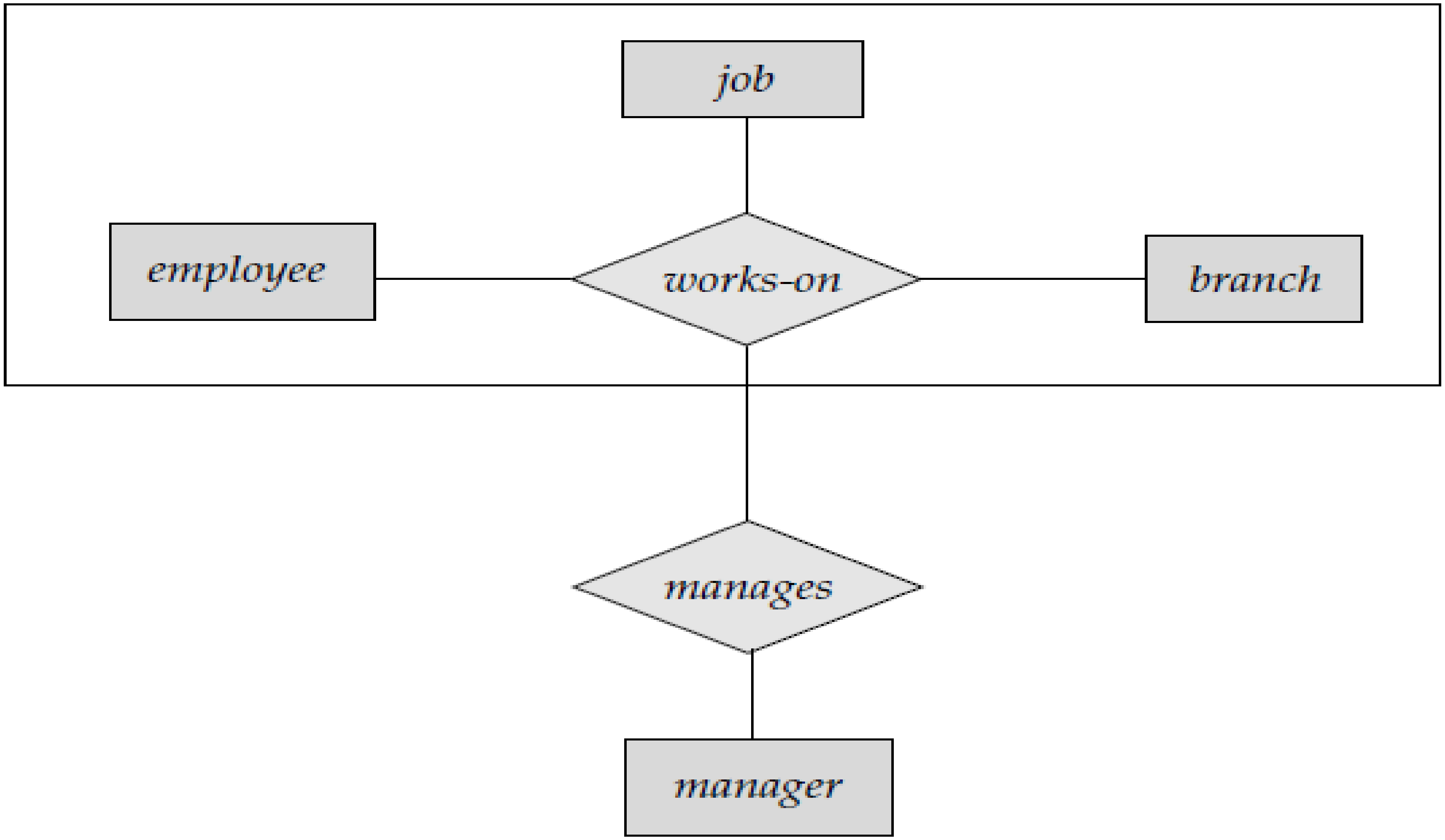
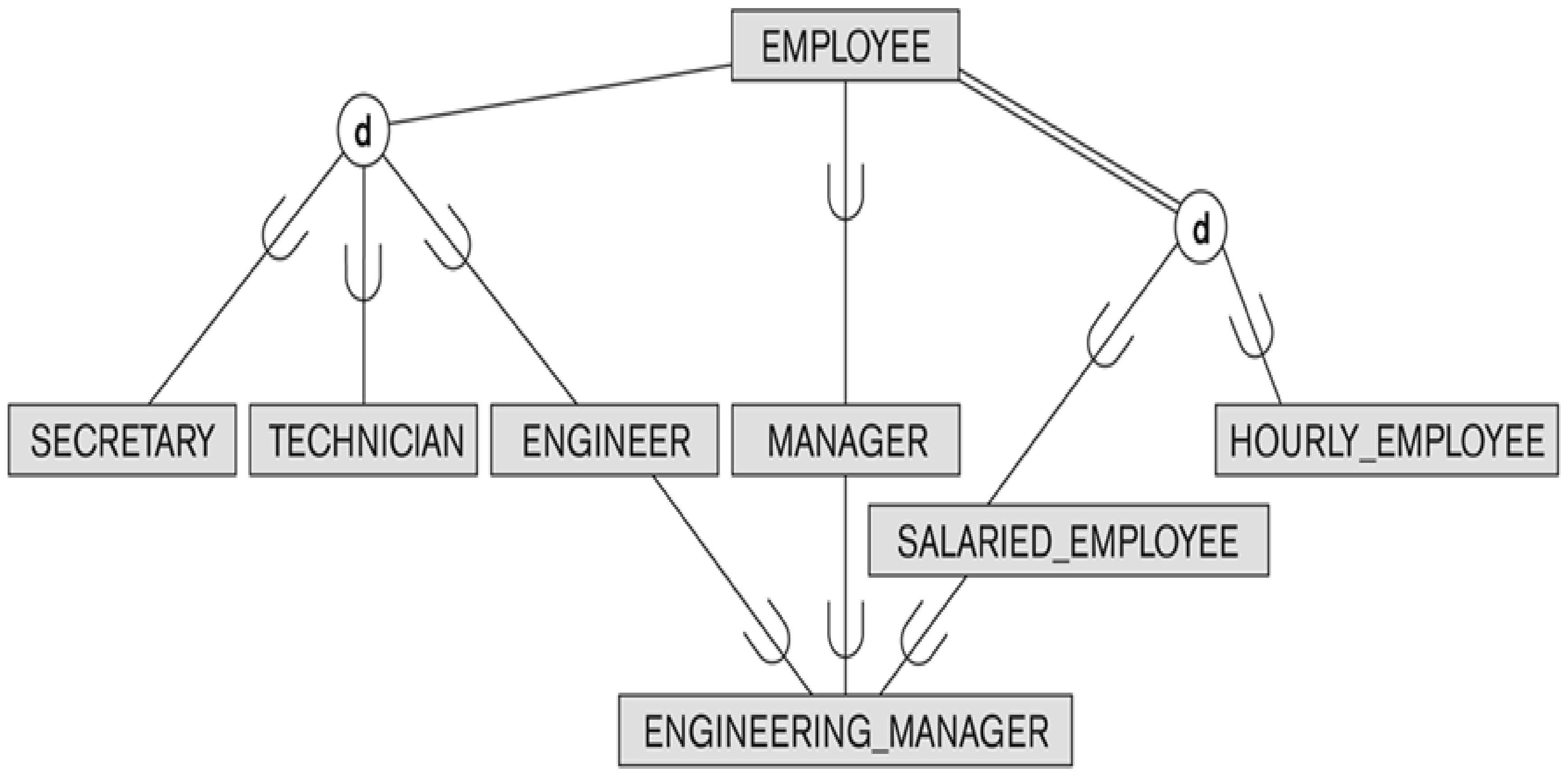


Figure E-R diagram with aggregation.

Specialization / Generalization Hierarchies, Lattices and Shared Subclasses (1)

- A subclass may itself have further subclasses specified on it
  - forms a hierarchy or a lattice
- ***Hierarchy*** has a constraint that every subclass has only one superclass (called ***single inheritance***); this is basically a ***tree structure***
- In a ***lattice***, a subclass can be subclass of more than one superclass (called ***multiple inheritance***)

Shared Subclass “Engineering_Manager”



A specialization lattice with shared subclass ENGINEERING_MANAGER.

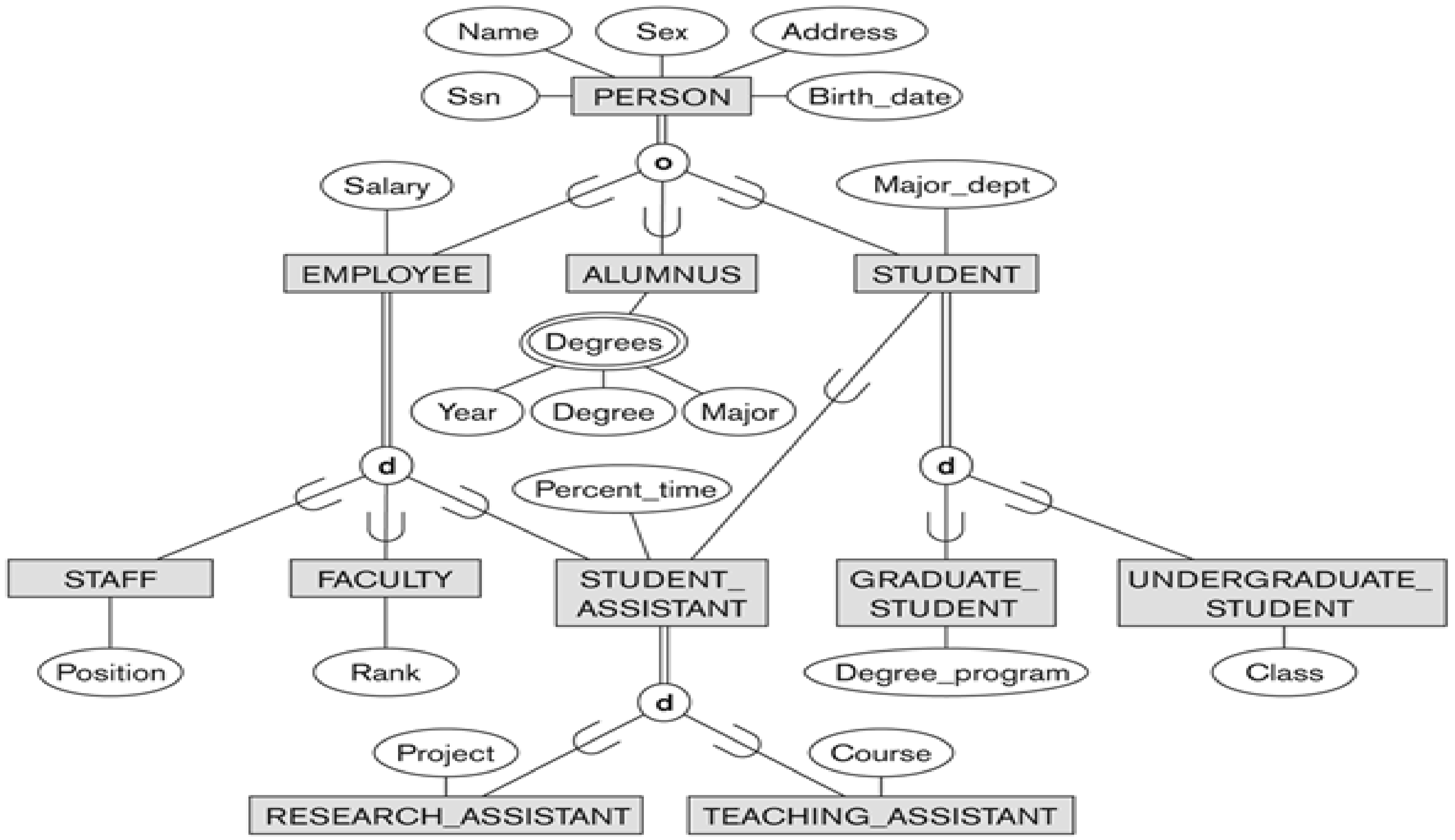
Specialization / Generalization Hierarchies, Lattices and Shared Subclasses (2)

- In a lattice or hierarchy, a subclass inherits attributes not only of its direct superclass, but also of all its predecessor superclasses
- A subclass with more than one superclass is called a shared subclass (multiple inheritance)
- Can have:
  - *specialization* hierarchies or lattices, or
  - *generalization* hierarchies or lattices,
  - depending on how they were *derived*
- We just use *specialization* (to stand for the end result of either specialization or generalization)

Specialization / Generalization Hierarchies, Lattices and Shared Subclasses (3)

- In *specialization*, start with an entity type and then define subclasses of the entity type by successive specialization
  - Called a *top down* conceptual refinement process
- In *generalization*, start with many entity types and generalize those that have common properties
  - Called a *bottom up* conceptual synthesis process
- In practice, a *combination of both processes* is usually employed

Specialization / Generalization Lattice Example (UNIVERSITY)



A specialization lattice with multiple inheritance for a UNIVERSITY database.

Categories (UNION TYPES) (1)

- All of the *superclass/subclass relationships* we have seen thus far have a single superclass
- A shared subclass is a subclass in:
  - *more than one* distinct superclass/subclass relationships
  - each relationships has a single superclass
  - shared subclass leads to multiple inheritance
- In some cases, we need to model a *single superclass/subclass relationship* with *more than one* superclass
- Superclasses can represent different entity types
- Such a subclass is called a category or UNION TYPE
- In this case, the subclass will represent a collection of objects that is a subset of the UNION of distinct entity types

Categories (UNION TYPES) (2)

- Engineering_Manager has 3 distinct relation, each relation has 1 superclass
- In our new case, a subclass has a single relationship with 3 distinct superclass.
- The subclass represent collection of objects, which we call **union type or category**

Categories (UNION TYPES) (3)

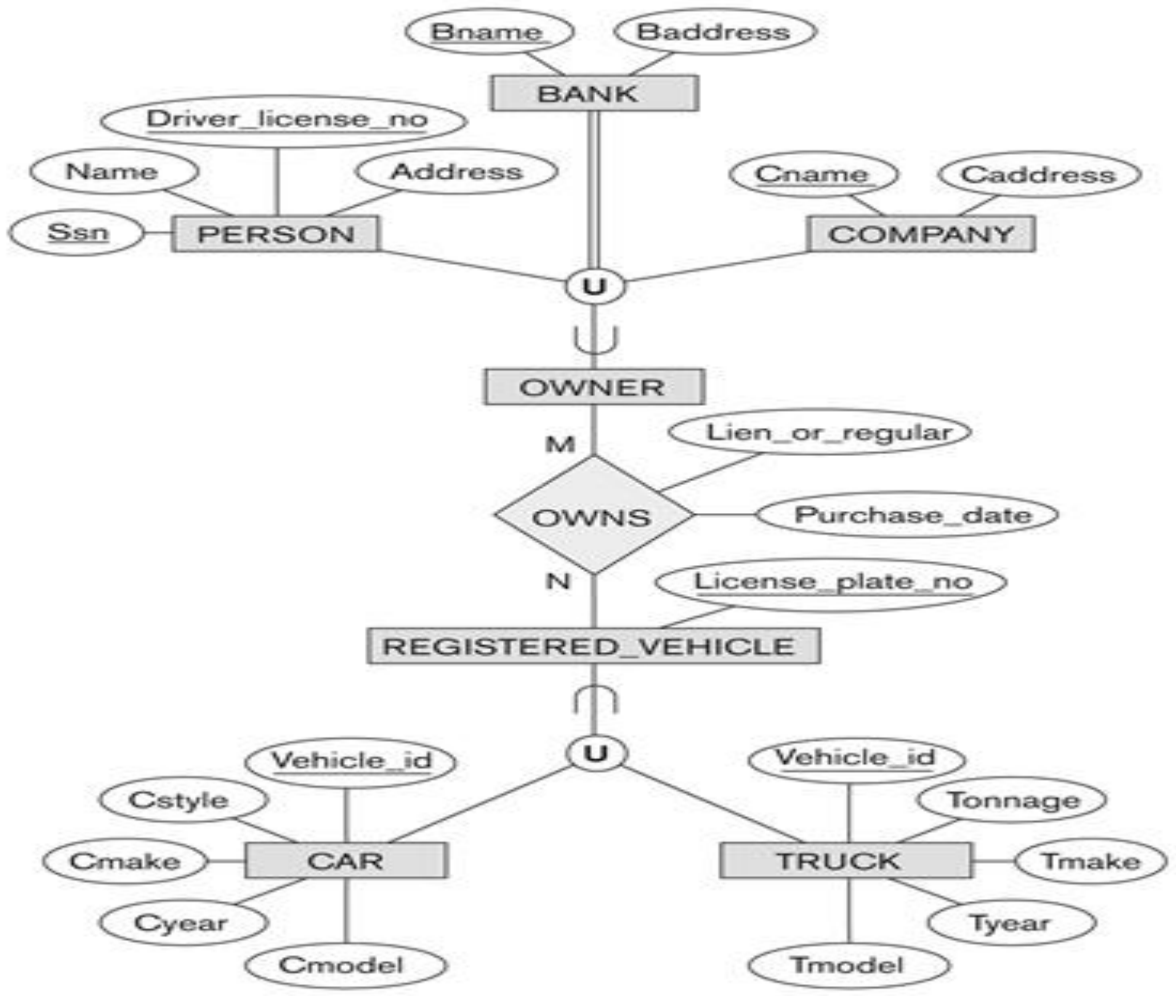
- Example: In a database for vehicle registration, a vehicle owner can be a PERSON, a BANK (holding a lien on a vehicle) or a COMPANY.
  - A *category* (UNION type) called OWNER is created to represent a subset of the *union* of the three superclasses COMPANY, BANK, and PERSON
  - A category member must exist in ***at least one*** of its superclasses
  - A category OWNER is a union subclass of COMPANY, BANK and PERSON
  - Use the (U) symbol (set union operation)
  - Registered_Vehicle is a union subclass of Car & Truck

The Difference….

Note: The difference from shared subclass, which is subset of the intersection of its superclasses (shared subclass member must exist in all of its superclasses).

- Engineering_Manager must exist in all three superclass: Manager, Engineer, Salaried_Employee
- Owner, must exist in only *one* superclasses
- Engineering_Manager: inherited all superclasses attributes
- Owner, selective attribute inheritance, depending on the superclass

Example of categories (UNION TYPES)



Two categories (union types): OWNER and REGISTERED_VEHICLE.

Categories (UNION TYPES) (4)

- A category can be **total** or **partial**.
- A total category holds the *union* of all entities in its superclasses, whereas a partial category can hold a *subset of the union*.
- A total category is represented diagrammatically by a double line connecting the category and the circle, whereas a partial category is indicated by a single line.
- The superclasses of a category may have different key attributes, as demonstrated by the OWNER category in Figure, or they may have the same key attribute, as demonstrated by the REGISTERED_VEHICLE category.

Categories (UNION TYPES) (5)

- Notice that if a category is total (not partial), it may be represented alternatively as a total specialization (or a total generalization).
- In this case, the choice of which representation to use is subjective.
- If the two classes represent the same type of entities and share numerous attributes, including the same key attributes, specialization/generalization is preferred;
- otherwise, categorization (union type) is more appropriate.

University Questions

1. Define the concept of aggregation. Give two examples, where this concept is useful. [2T– 5 M]
2. Write Short notes on [5 M each]
   i) Generalization and Aggregation [3T]
   ii) Generalization and Specialization [4T]
   iii) Constraints on Specialization and Generalization [2T]
3. What is an attribute ? What are the different types of attribute ? How they are represented in E-R diagram ? [1T– 4M]

University Questions

4. Define the following terms with example [5T-- 2 M each]

    i) Primary Key [3T]

    ii) Candidate Key [3T]

    iii) Foreign Key [4T]

    iv) Super Key [3T]

    v) Single and composite attributes [2T]

    vi) Single valued and multivalued attributes [2T]

    vii) Entity and Entity set [2T]

    viii) Relationship set [1T]

    ix) Aggregation [3T]

    x) Weak and strong entity set [3T]

    xi) Specialization and Generalization [3T]

    xii) Mapping Cardinalities [3T]

University Questions

5. Draw the E-R diagram for the following scenario and convert the E-R diagram into tables [10-12 M each]
   - i) Hospital [2T]
   - ii) University Database [3T]
   - iii) Car insurance company [1T]
   - iv) Railway ticket reservation system [1T]
   - v) Banking enterprise [3T]
   - vi) Software company [1T]
   - vii) Library Management System [2T]
   - vii) Art Gallery [2T]

<u>For problem statements refer question bank.</u>