Chapter-2

Conceptual Database Design: Entity Relationship Model

Prepared by: Marta G. (MSc.)

Contents

- Basic Concepts: Data Models
- DB development Lifecycle
- ER diagram
- Entity types, entity Sets and Weak Entity Types, Attributes and Keys
- Relationships, Roles and Structural Constraints
- E/R Diagram naming conventions
- Constraints in ER models and design issues
- Enhanced ER

Data models

- Data models define how the logical structure of a database is modeled.
- It provides the **conceptual tools** for describing the design of a database at each level of data abstraction.
- It categorized in to **High-level Conceptual Data Models** and **Record-based Logical Data Models**
- These models follow three approaches: conceptual, logical, and physical.

Data models Cont…

I. High-level Conceptual Data Models

- It provide concepts for presenting data in ways that are **close to the way people perceive data.**
- A typical **example** is the **entity relationship model**, which uses main concepts like **entities**, **attributes** and **relationships**.

Data models

Cont…

II. Record-based Logical Data Models

- It provide concepts users can understand but are not too far from the way data is stored in the computer. Three well-known data models are:
  - **The relational model** represents data as *relations*, or tables.
  - **The network model** represents data as record types. This model also represents a limited type of one to many relationship called a *set type*
  - **The hierarchical model** represents data as a hierarchical tree structure. Each branch of the hierarchy represents a number of related records.

Database development life cycle

1. Planning phase
2. Requirements collection and analysis phase
3. Design phase
   - Conceptual design
   - Logical design
   - Physical design
4. Implementation phase
5. Testing phase
6. Maintenance phase

Planning phase

- This phase starts when a customer request to develop a system
- The **resources** and the **time** required to develop the system will be **identified**
- Problem is seen on broader basis

Requirements collection and analysis

- Database designers **understand** and **document** the data **requirements** of the database users.
- The **result** of this step is the **data requirement of the users** written concisely

Along with users data requirement functional requirement also have to be specified

- Consists of user defined operations that has to be applied to the database like **retrieval** and **updating** of data

Design phase

- The design phase is **where the requirements identified in the previous phase are used as the basis to develop the new system**.
- In this phase the **business understanding** of the data structures is **converted** to a **technical understanding**.
- The *what* questions ("What data are required? What are the problems to be solved?") are replaced by the *how* questions ("How the data can be structured? How is the data to be accessed?")

Cont….

I. Conceptual schema/model/design:

- Creating conceptual schema.
- Conceptual schema is a **concise description** of the data **requirements** and **detailed description** of the **entity types**, **relationships** and **constraints**.
- A commonly-used conceptual model is called an ***entity-relationship* model.**

Cont…

II. Logical design:

- Actual implementation of the database, using commercial DBMS like oracle, Microsoft sql server, etc.

III. Physical design:

- The internal storage structures, record formats, file structures, indexes, access paths are specified.
- Along with this activities application programs are designed and implemented as transactions corresponding to high level transaction specification.
- Every details of the database components are fully stated.

Implementation phase

- Final design is implemented
- **DBMS** is **installed** on the required hardware, the **databases are created** and the **data** are **loaded** or imported.

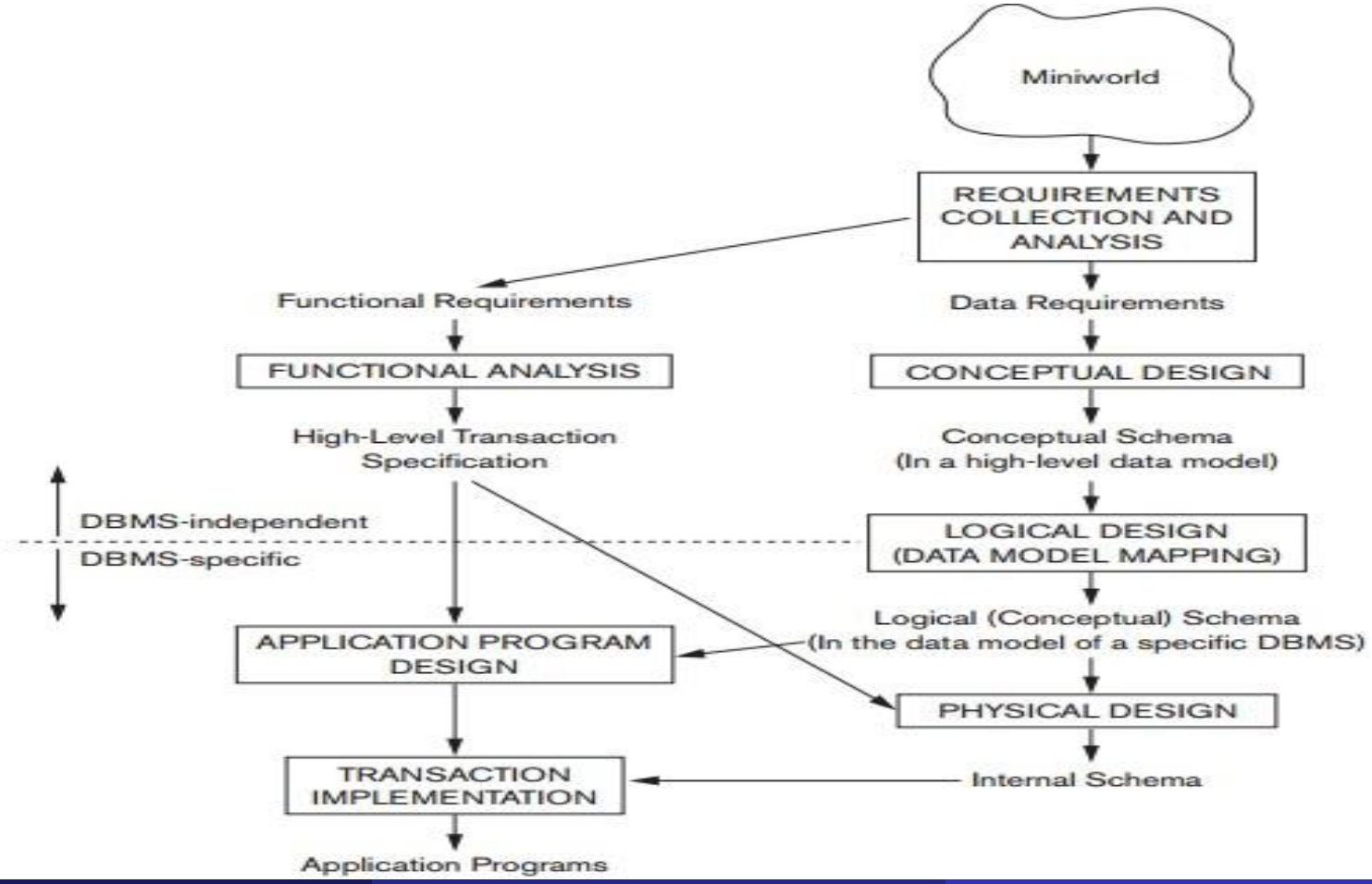
Testing phase

- The testing phase is **where the performance, security, and integrity of the data are tested**.
- Usually this will occur in conjunctions with the applications that have been developed.

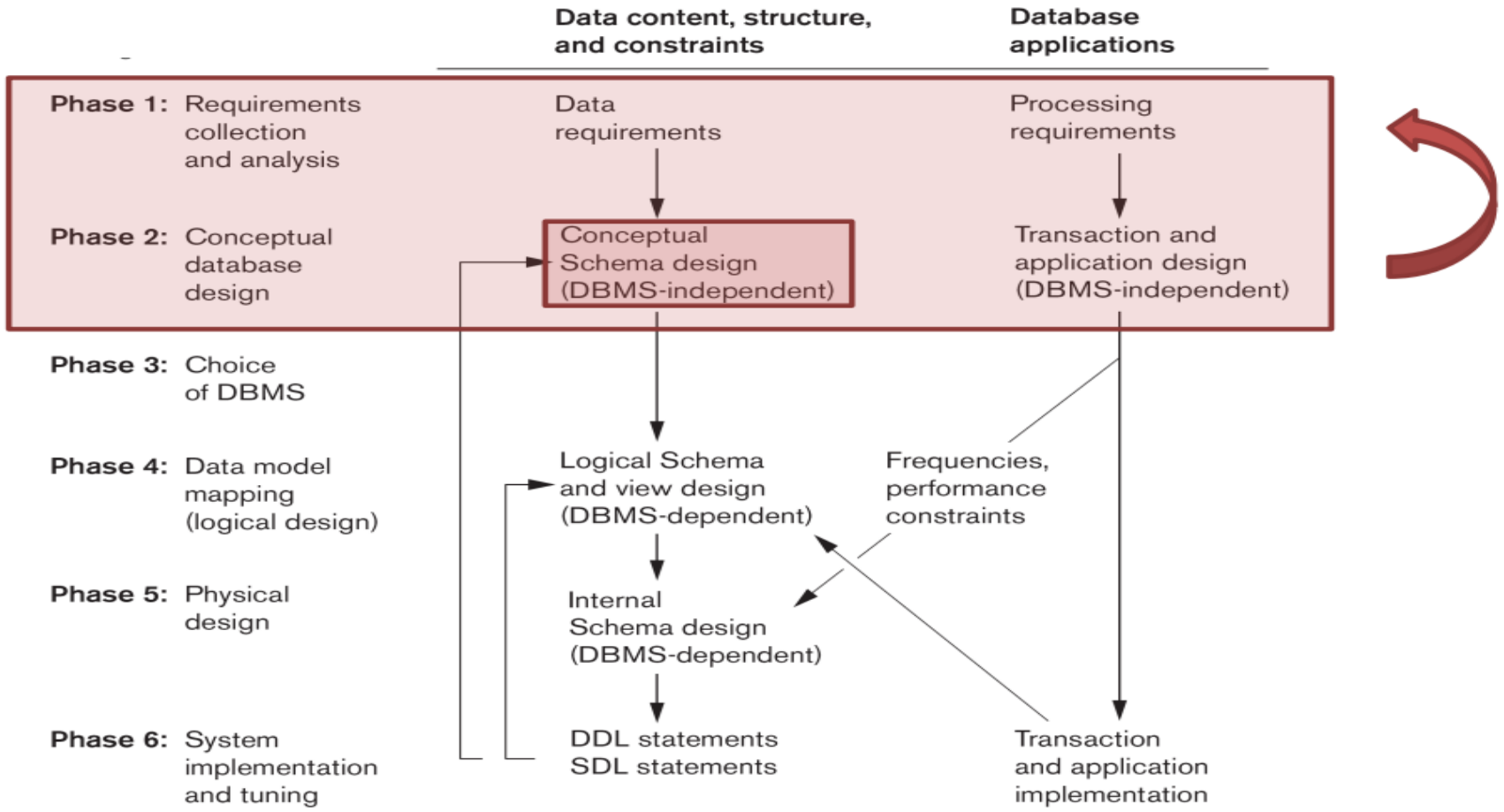
Maintenance

- The database maintenance phase incorporates general maintenance, such as **maintaining the indexes, optimizing the tables, adding and removing users, and changing passwords, as well as backups and restoration of backups in case of a failure**.
- **New requirements** also start to be requested, and this may result in new fields, or new tables, being created.

General phases of database design



Database Design and Implementation Process



High level Conceptual Data Models for Database Design

Requirements of a Conceptual Data Model

- **Expressiveness:**
  - Should be expressive enough to allow modeling of different types of r/ships, objects & constraints of the mini world.
- **Simplicity:**

  Non-specialists should be able to understand.
- **Diagrammatic Representation:**
  - To make it easy for interpretation.
- **Formality:**
  - There should be no ambiguity in the specification.

N.B.: A typical **example** of high level conceptual data model is **Entity relationship model**

ERD (Entity Relationship Diagram)

- Also known as an entity relationship model
- It is a **graphical representation** that depicts **relationships** among **people**, **objects**, places, concepts or **events** within an information technology (IT) system.
- ERD is a graphical technique for representing a database schema before it is actually implemented.
- It **shows the various entities** being modeled and the important **relationships** among them.
- ERD has three basic components, namely,
  - **Entity,**
  - **Attribute and**
  - **Relationship**

ERD (Entity Relationship Diagram)

- **Entity:**
  - An **entity** represents a **real-world object with an independent existence**
  - **An entity may be an object with**
    - A physical existence/ **Tangible** (e,g; a particular person, car, house, or employee) or
    - A conceptual existence/ **intangible** (e,g; course, project, job...)
  - **Entities have attributes which are the particular properties that describe them.**
- Example: A student with a particular roll number is an entity.
  - A company with a particular registration number is an entity.

NOTE: In a particular relation in RDBMS, a particular record is called an entity.

Cont…

Entity type

- It refers to the **category** that a **particular entity belongs to**
- **The category of a particular entity in the relation in RDBMS is called the entity type.**
- **Eg.** A table named **STUDENT** in a university database and a table named EMPLOYEE in a company database.

Entity set

- All rows of a relation (table)
- A **group** of **similar entities** and these entities can have **attributes**.
- Collection of entities of a particular entity type at a point in time.

STUDENT

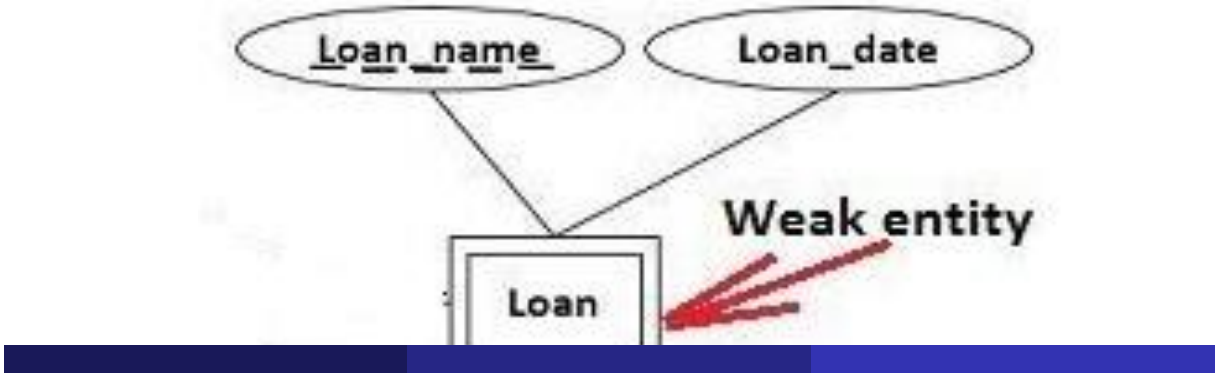
ID	NAME	AGE
01	YEABSIRA	18
02	SAMUEL	21
03	HALETA	16

Entity: Each row is an entity 01 yeabsira 18
Entity type: each entity belongs to the student type hence entity type is STUDENT
Entity set: {(01, YEABSIRA,18), (03, HALETA, 16)}

Weak Entities

- A weak entity is an entity that **depends on the existence of another entity**.
- In more technical terms it can be defined as an entity **that cannot be identified by its own attributes.**
- It **uses a foreign key** combined with its attributed to form the primary key.
- An entity like **order item** is a good example for this. The order item will be **meaningless without an order** so it depends on the existence of the order.

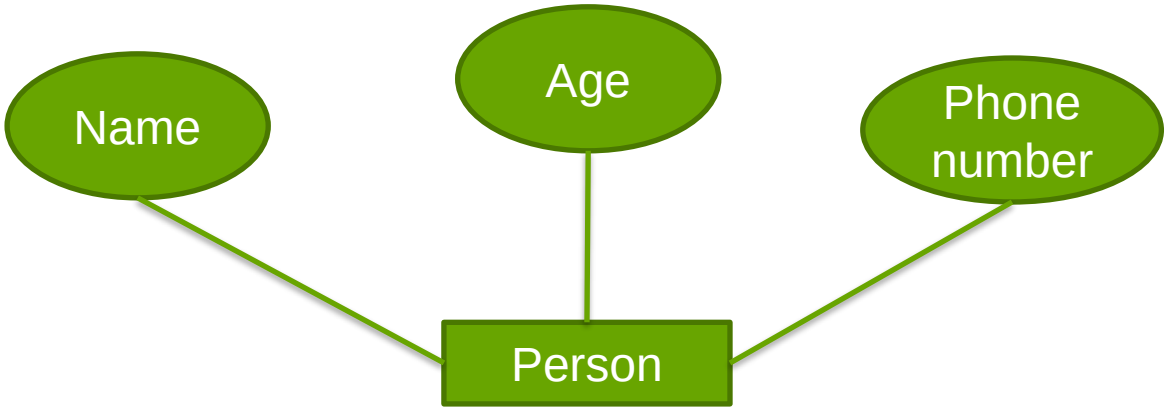
Read more examples for week entity types



ERD

Cont…

- **Attribute:**
  - The entity has **attributes**
  - Attributes are **Properties that describe the entities** such as an employee's name, phone number and **age**.



Cont…

There are several type of attributes in the ER model:

Simple attributes

- Simple attributes are atomic values that can not be divided for further
- Eg. Age, weight, salary, etc.

Composite attributes

- Are made of more than one simple attribute and can be divided into further parts
- Eg. Name→ First name, Middle Name, Last Name

Single valued attributes

- Have a single value for a particular entity
- Eg. Age → single valued attribute of a person

Multivalued attributes

- Can have more than one value or set of values at one time.
- Eg. College degree, languages known, etc.

Cont…

Derived attributes

- Can be derived from other attributes
- Eg. Age → can be derived from date of birth

Stored attribute

- From which the value of other attributes are derived
- Eg. Birth date of a person

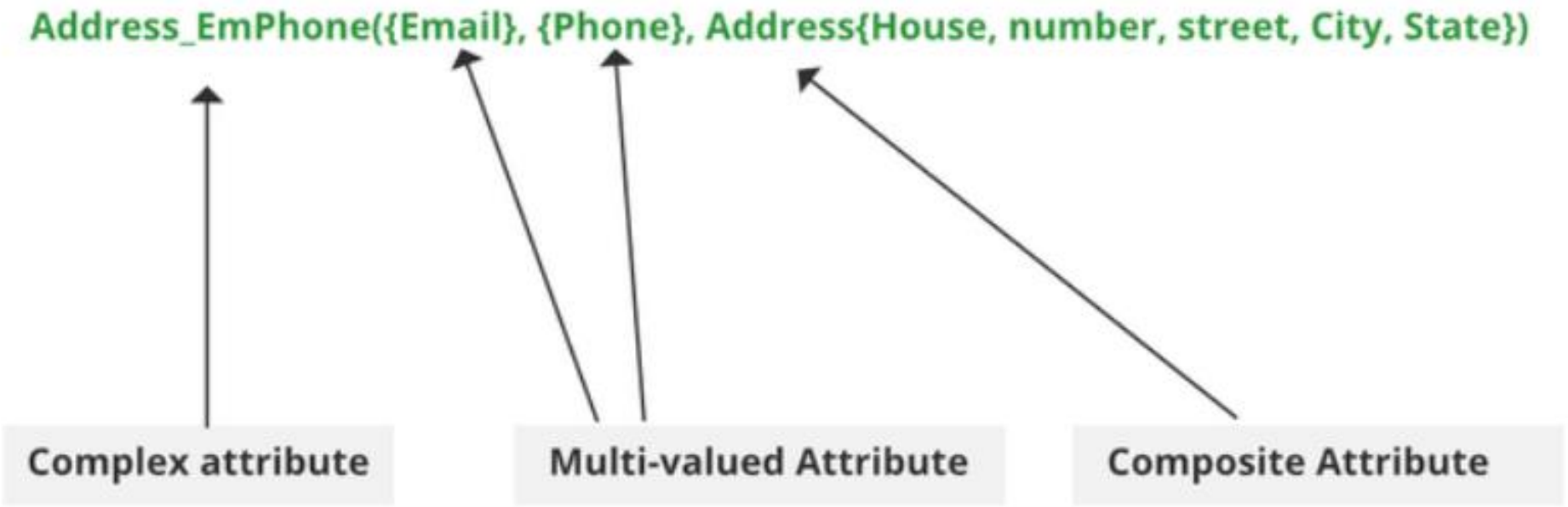
Key attribute

- It is capable of identifying each entity uniquely

Complex attributes

- Are the nesting of two or more composite and multi-valued attributes.
- Therefore, these multi-valued and composite attributes are called ‘Components’ of complex attributes.
  - Multi-valued attributes represented within ‘{ }’
  - Composite attributes represented by ‘( )’
  - Components are separated by commas ‘, ‘
- For example: let us consider a person having multiple phone numbers, emails, and an address.

Cont…

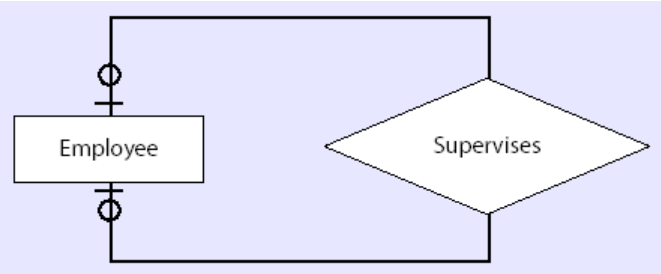

Address_EmPhone({Email}, {Phone}, Address{House, number, street, City, State})
Complex attribute
Multi-valued Attribute
Composite Attribute

ERD Cont…

- **Relationship :**
  - **Shows the communication among the entities in a database or we can say** it **represents** an **association** among two or more **entities**;
  - **F**or example, an **employee works on projects**. A relationship exists between the employee and each project.

Cont…

- **Degree of relationship:** denotes the **number of entity types** that participate in a relationship.
- It tells **how many entities are** associated or **linked together**
- Relationship type of **degree one** is called **unary relationship**
  - **unary relationship:** exists when there is an association with only one entity and its degree of relationship is one



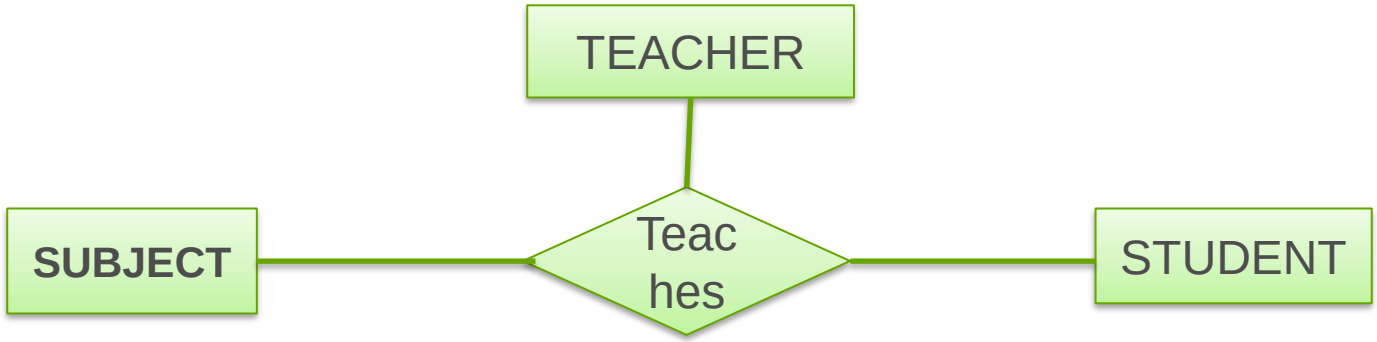
Cont…

Binary relationship

- Exists when there is an association among **two entities**.

Ternary relationship

- Exists when there is ana association among **three entities**



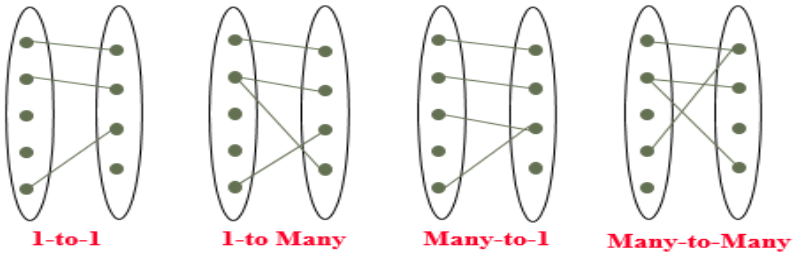
Reading assignment

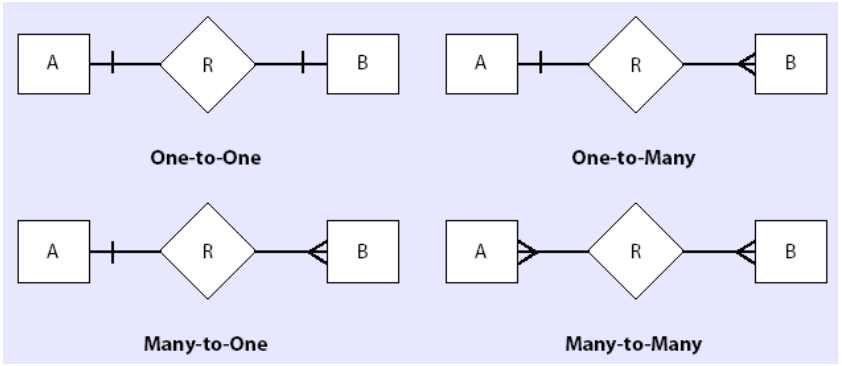
- There are also quaternary, quinary, senary, septenary, octal relationships

Relationship constraints

I. Cardinality Ratio (maximum cardinality)

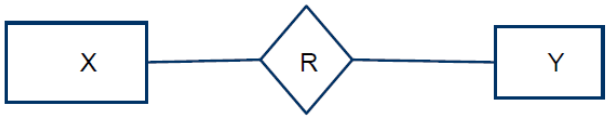
- **Maximum number** of **relationship** instances that an **entity can participate in**
- **For example:** when An employee can work for a departments, A department can have many employees, Each department has at most one manager….
- Therefore, relationship types by maximum cardinality are:
  - **One-to-one (1:1)**
  - **One-to-many (1:N),**
  - **Many-to-one (N:1) and**
  - **Many-to-many (M:N)**





Cont…

- How do you **know** the maximum cardinality between two entity types?
- Let's say we have **2 entity types** called X and Y



- Two important questions:
  - **Q1. Can an instance of X be associated with more than one instance of Y?**
  - **Q2. Can an instance of Y be associated with more than one instance of X?**
- The 2 questions above have 4 possible answers. Those answers determine the type of maximum cardinalities

QUESTION 1	QUESTION 2	TYPE OF MAXIMUM CARDINALITY
YES	YES	Many-2-Many
YES	NO	One-2-Many
No	YES	Many-2-One
NO	NO	One-2-One

Cont…

- **One-to-One Relationship**
  - An entity in X is associated with at most one entity in Y , and vice-versa.
  - E.g. Each passenger has his own Ticket

- **One -to-Many Relationship**
  - An entity in X is associated with any number(zero or more) of entities in Y. But an entity in Y can associate with at most one entity in X
  - E.g. A department contains many employees who work under it

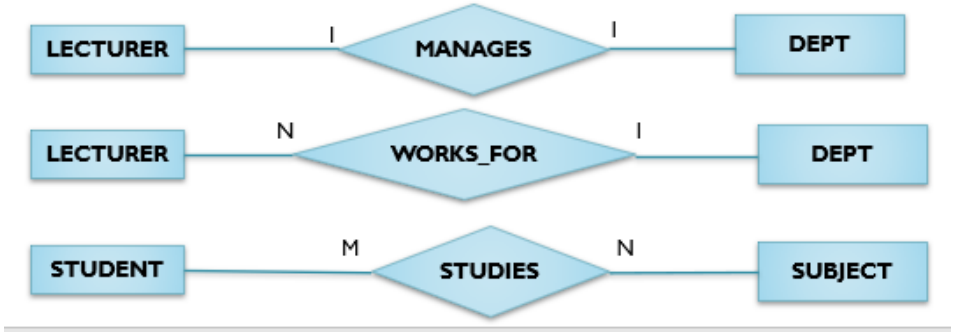
Cont…

- **Many-to-One Relationship**
  - An entity in X is associated with at most one entity in Y. But an entity in Y can be associated with any number (zero or more) of entities in X
  - E.g. Many needles can be injected into a single patient

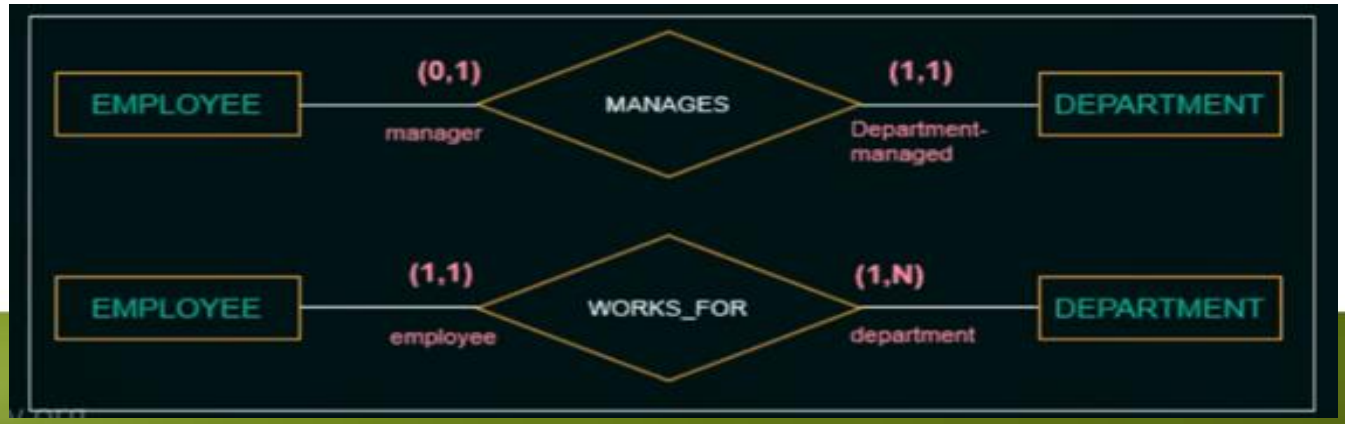
- **Many-to-Many Relationship**
  - An entity in X is associated with any number(zero or more) of entities in Y, and an entity in Y is associated with any number (zero or more) of entities in X

Notations to represent cardinality in ERD

- There are no. of notations to represent cardinality in ERD: Chen, UML, Crow's foot, Bachman etc.
- Cardinality with **chen notation**

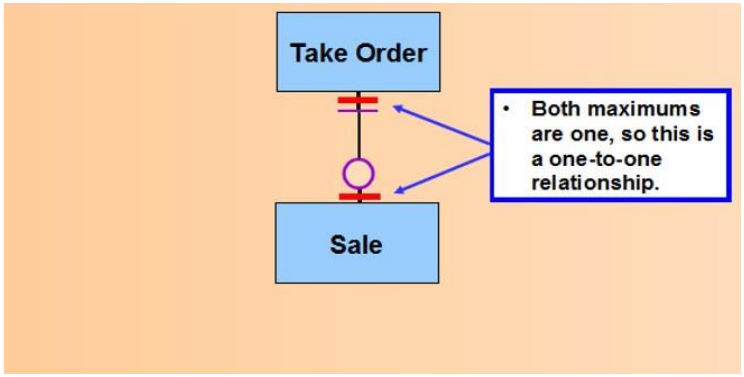


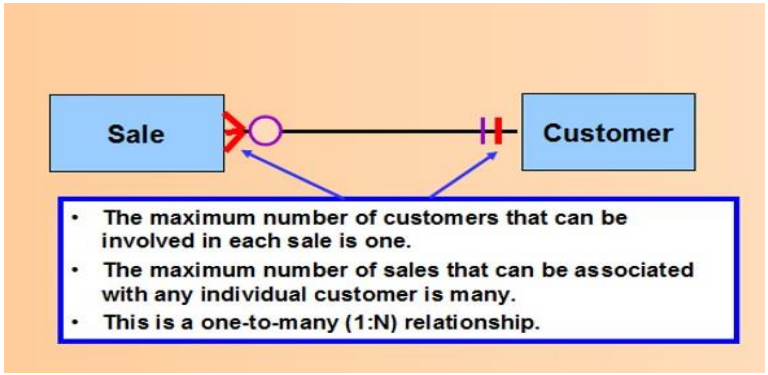
- Cardinalities of relationships with **UML(min max notation)**
  - Associates a pair of integer numbers **(min, max)** with each participation of an entity type in a relationship type, where 0<=min <=max and max >= 1.

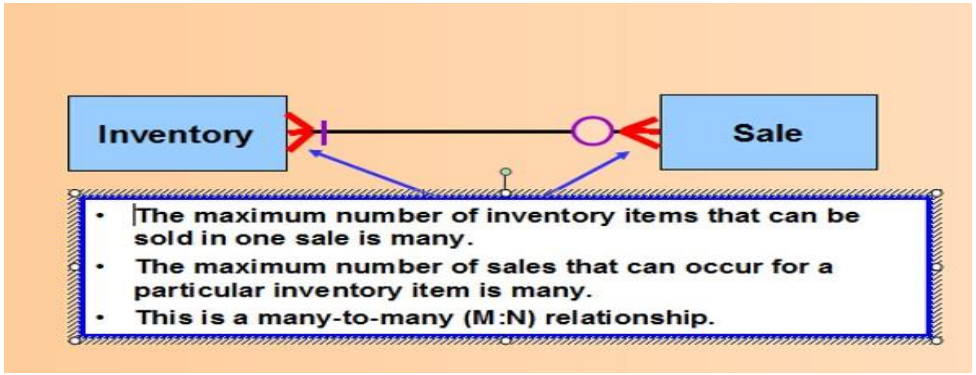


Cont…

- Cardinalities of relationships with Crow's foot



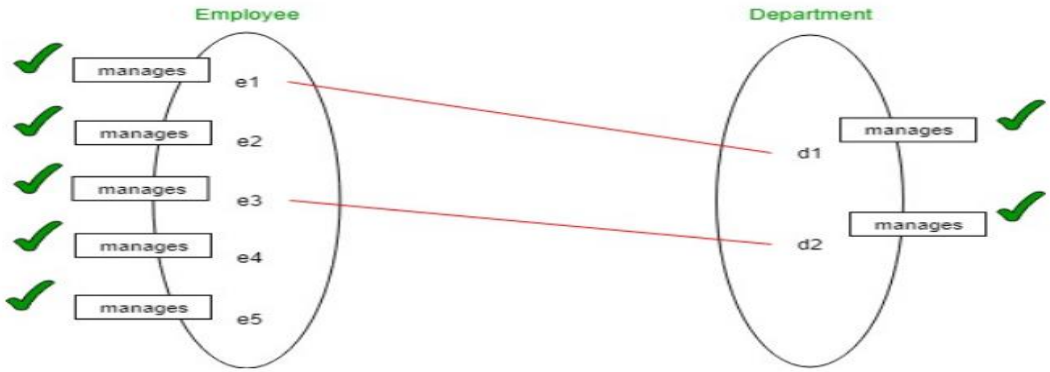




Cont…

II. Participation constraints (Minimum cardinality)

- The **minimum cardinality** of a relationship is defined as the **minimum number of instances** of **one entity type** that **must be associated with each instance of another entity type**
- Specifies whether **existence of an entity depends** on its **being related** to **another entity.**
- It also called **existence dependency constraints.** There are 2 types of participation constraints: **Total participation** and **Partial participation**



Cont…

- In **total participation**, each entity in the entity set occurs in at least one relationship in the relationship set
- The above example shows that the participation of the entity set Department in the relationship set Manages is total. This is because **every department must have a manager .**
- In **partial participation**, some entities (not all entities) in the entity set occurs in the relationship set
- The above example shows that the participation of the entity set employee in the relationship set manages is partial because **it is not possible for every employee entity to be a manager/ department head**

Relationships cont…

Attributes of r/n types

- In 1:1 relationship type, attributes **can be migrated** to **either** of the entity types (look at figure A)
- In 1:N or N:1 relationship type, attributes are **migrated** only **to the entity type on the N-side** of the relationship (look at figure B)
- In M:N relationship type, some attributes can be determined by a **combination of participating entities**. (look at figure C)

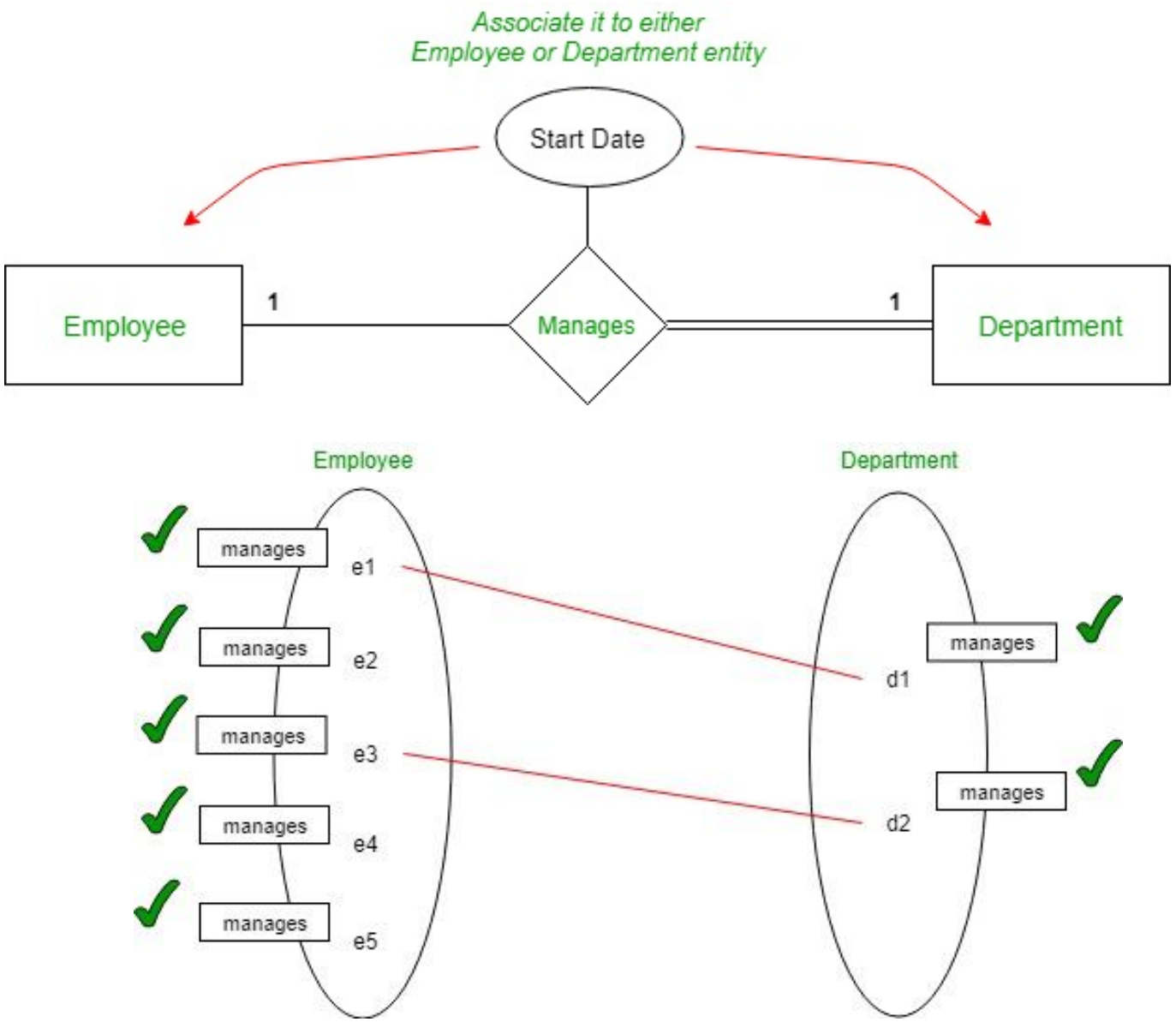

Associate it to either
Employee or Department entity
Start Date
Employee
1
Manages
1
Department
Employee
Department
manages
e1
manages
e2
manages
e3
manages
e4
manages
e5
d1
d2
manages
manages

Figure A

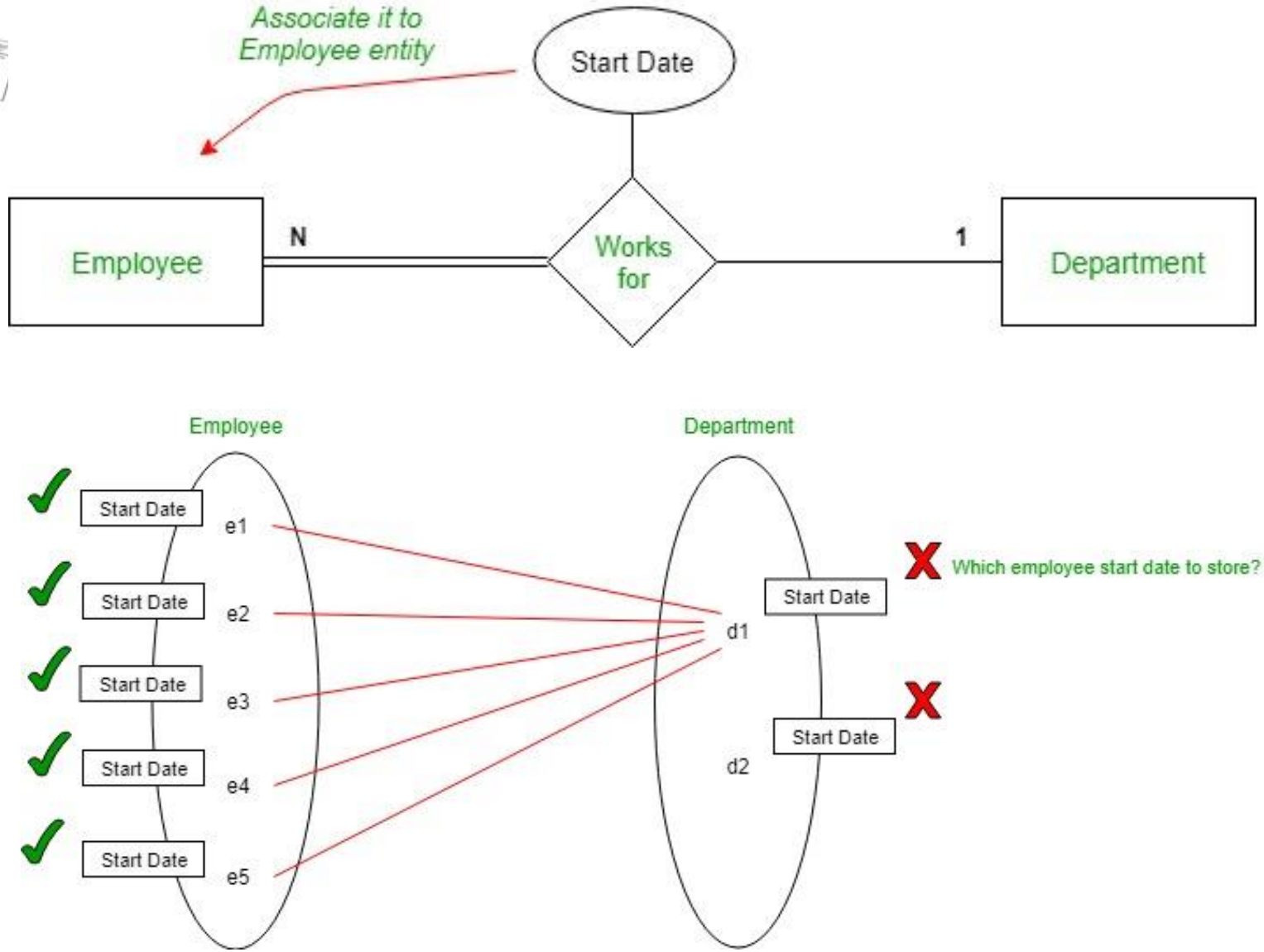
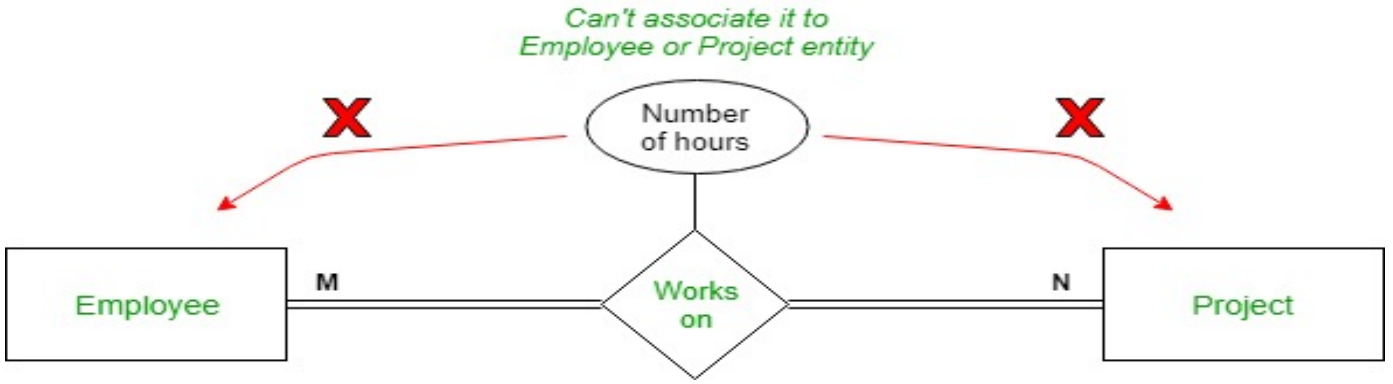

Associate it to Employee entity
Start Date
Employee
N
Works for
1
Department
Employee
Department
Start Date
e1
Start Date
e2
Start Date
e3
Start Date
e4
Start Date
e5
d1
d2
Start Date
Start Date
Which employee start date to store?

Figure B


Can't associate it to
Employee or Project entity
Number
of hours
Employee
M
Works
on
N
Project

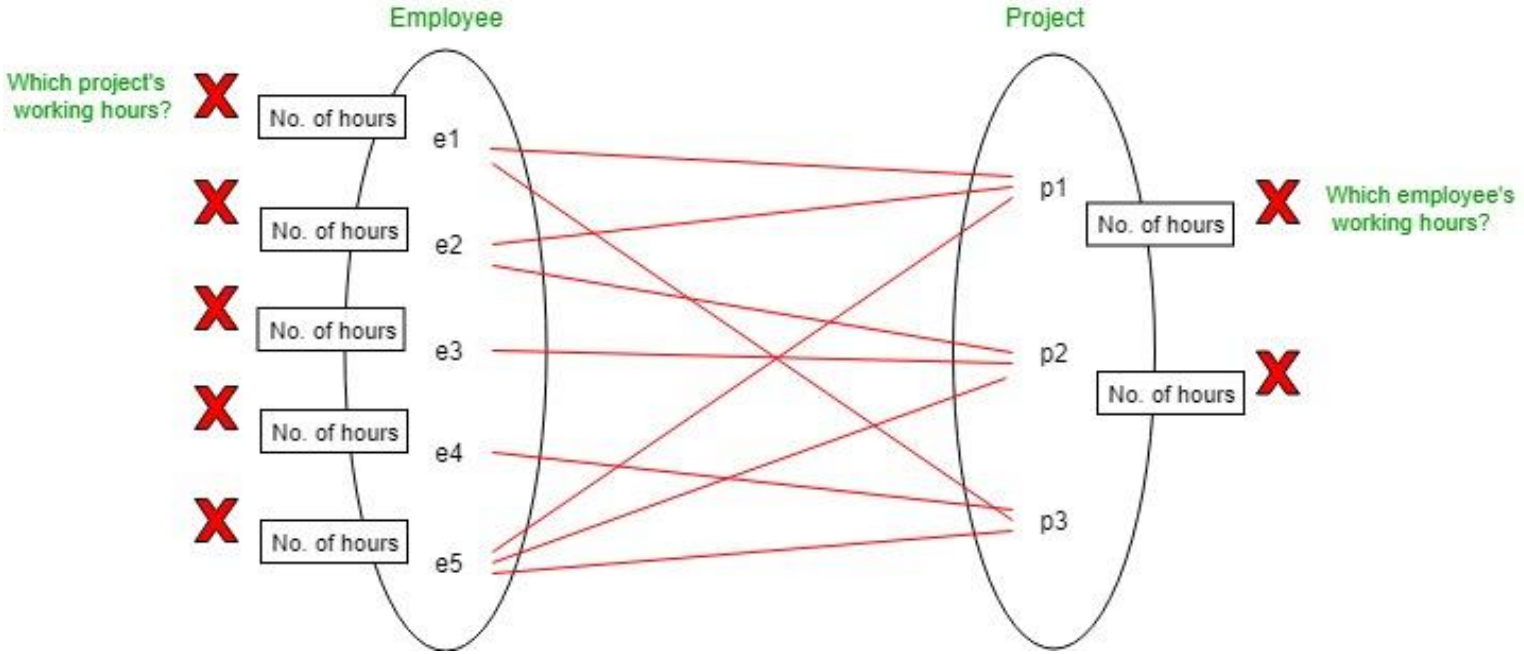

Employee
Project
Which project's
working hours?
No. of hours
e1
e2
e3
e4
e5
p1
p2
p3
Which employee's
working hours?

Figure C

Relationships cont…

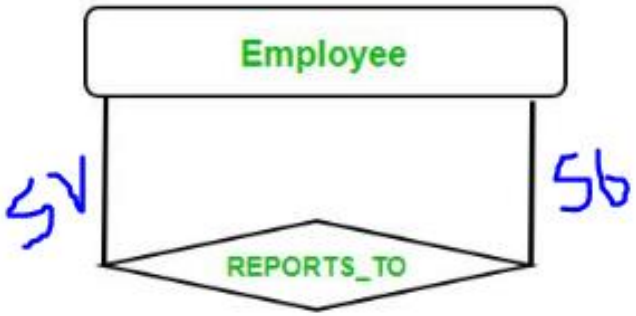
Role Names

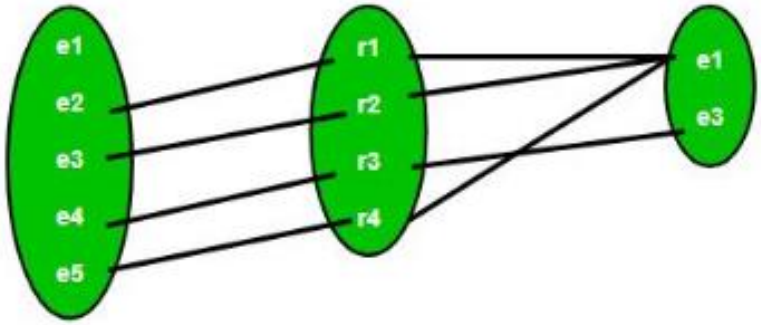
- Signifies **the role** that a **participating entity plays** in each **relationship instance**.
- It also explains what the relationship means
- It is necessary only in **recursive relationships**

Recursive relationships

- Same **entity type participates more than once** in a relationship type in **different roles**

Cont….





- Supervisors and subordinates are called **“Role Names”**.
- The degree of the REPORTS_TO relationship is 1 i.e. a **unary relationship.**

Cont…

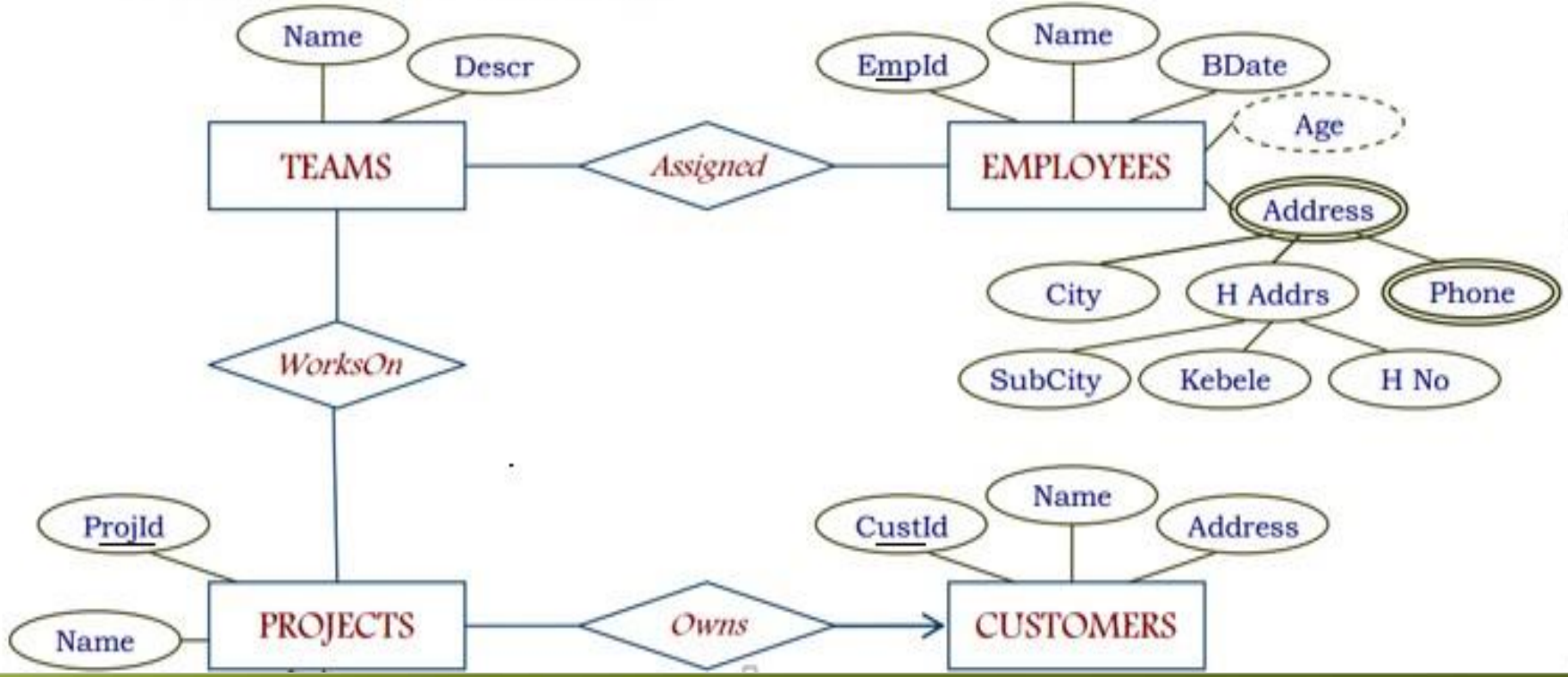
- The **minimum cardinality** of the **Supervisor** entity is **ZERO** since the lowest level employee may not be a manager for anyone.
- The **maximum cardinality** of the **Supervisor** entity is **N** since an employee can manage many employees.
- Similarly, the **Subordinate entity** has a **minimum cardinality** of **ZERO** to account for the case where CEO can never be a subordinate.
- Its **maximum cardinality** is **ONE** since a subordinate employee can have at most one supervisor.

Cont…

- **Sample ERD**

Example

- "EMPLOYEES" are *Assigned* to "TEAMS"
- "CUSTOMERS" *Owns* "PROJECTS"
- "TEAMS" *works on* "PROJECTS"



ERD Representation Naming convention

The Entity Relationship (E/R) data model is a diagrammatical data model. The elements of the E/R model are represented by:

- *Rectangles* - for the Entity sets,
- *Ellipses* - for the Attributes,
- *Diamonds* - for the Relationships, and
- *Lines* - for the links between the attributes and the entity sets and between the entity sets and the relationships.
- *Double border Rectangles* - for the weak entity sets.
- *Double border Ellipses* - for the multi-valued attributes.
- *Dashed border Ellipses* - for the derived attributes.
- *Arrow Head Line* - for the link between an entity set and a one-to-one or many-to-one relationship. The arrow is headed to the one side entity set.

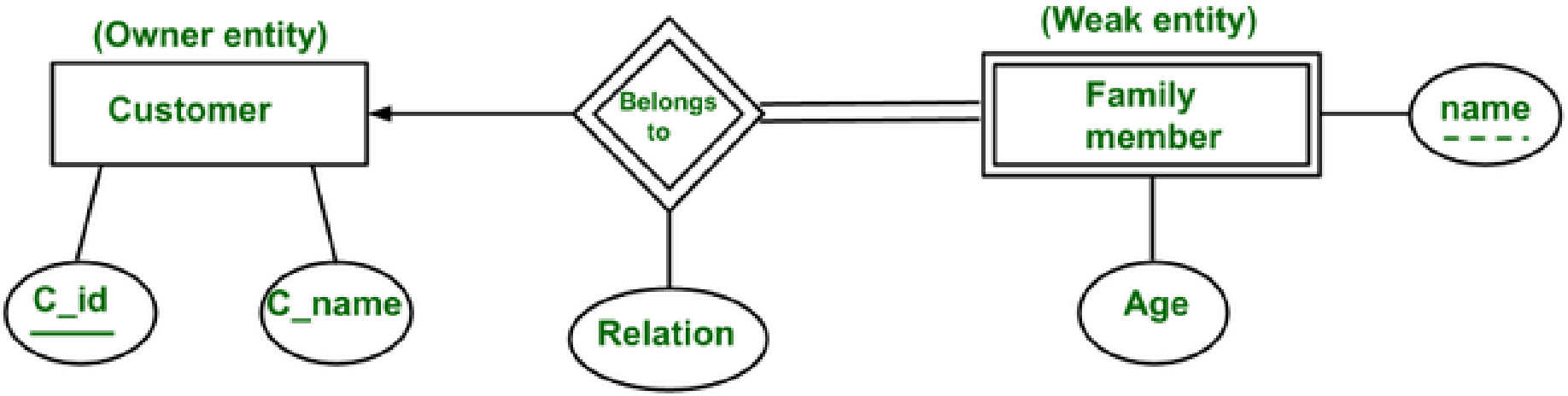
Alternative description

Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Indentifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute
	Total Participation of E_2 in R
	Cardinality Ratio 1: N for E_1:E_2 in R
	Structural Constraint (min, max) on Participation of E in R

Identifying relationship

- The relationship that **relates the weak entity type to an owner entity type** is known as identifying relationship.
- The **weak entity** type always has **total participation** (existence dependency) in a relationship because the **weak entity type can not be identified without an owner identity.**
- This doesn’t mean that every existence dependency(total participation) results in a weak entity type.

Cont…


(Owner entity)
Customer
Belongs to
(Weak entity)
Family member
name
C_id
C_name
Relation
Age

How to Draw ER Diagrams

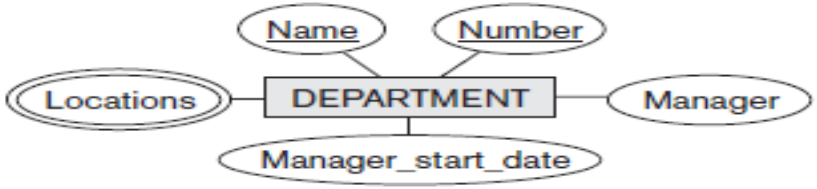
- **Identify all the entities** in the system. An entity should appear only once in a particular diagram. Create rectangles for all entities and name them properly.
- **Add attributes** for entities. Give meaningful attribute names so they can be understood easily.
- **Identify relationships** between entities. Connect them using a line and add a diamond in the middle describing the relationship.
- **Let us draw sample company database.**

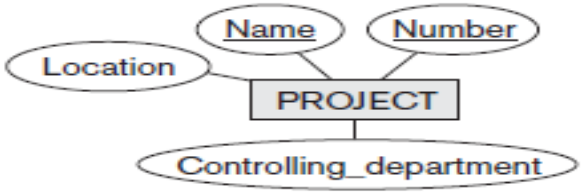
Initial Conceptual Design of the COMPANY Database

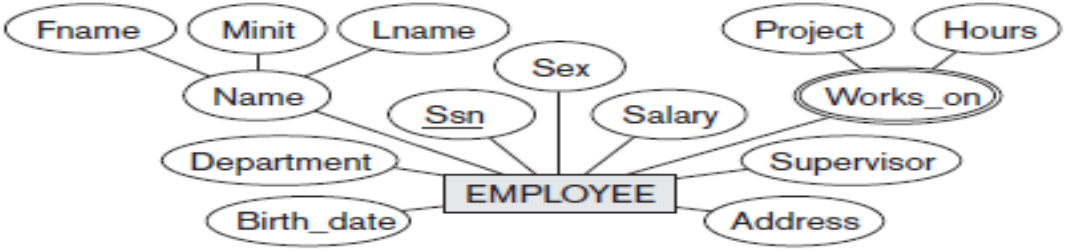
- An entity type DEPARTMENT
  - **with attributes Name, Number, Locations, Manager, and Manager_start_date. Locations is the only multivalued attribute.**
- An entity type PROJECT
  - **with attributes Name, Number, Location, and Controlling_department. Both Name and Number are (separate) key attributes.**
- An entity type EMPLOYEE
  - **with attributes Name, Ssn, Sex, Address, Salary, Birth_date, Department, and Supervisor. Both Name and Address may be composite attributes.**
- An entity type DEPENDENT
  - **with attributes Employee, Dependent_name, Sex,Birth_date, and Relationship (to the employee).**

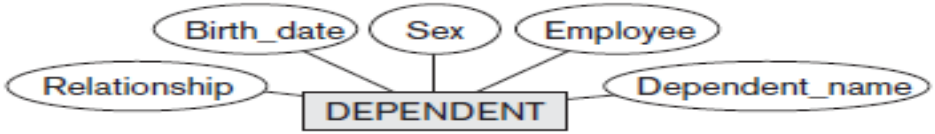
Cont…

Preliminary design of entity types for the COMPANY database.
Some of the shown attributes will be refined into r/ps.

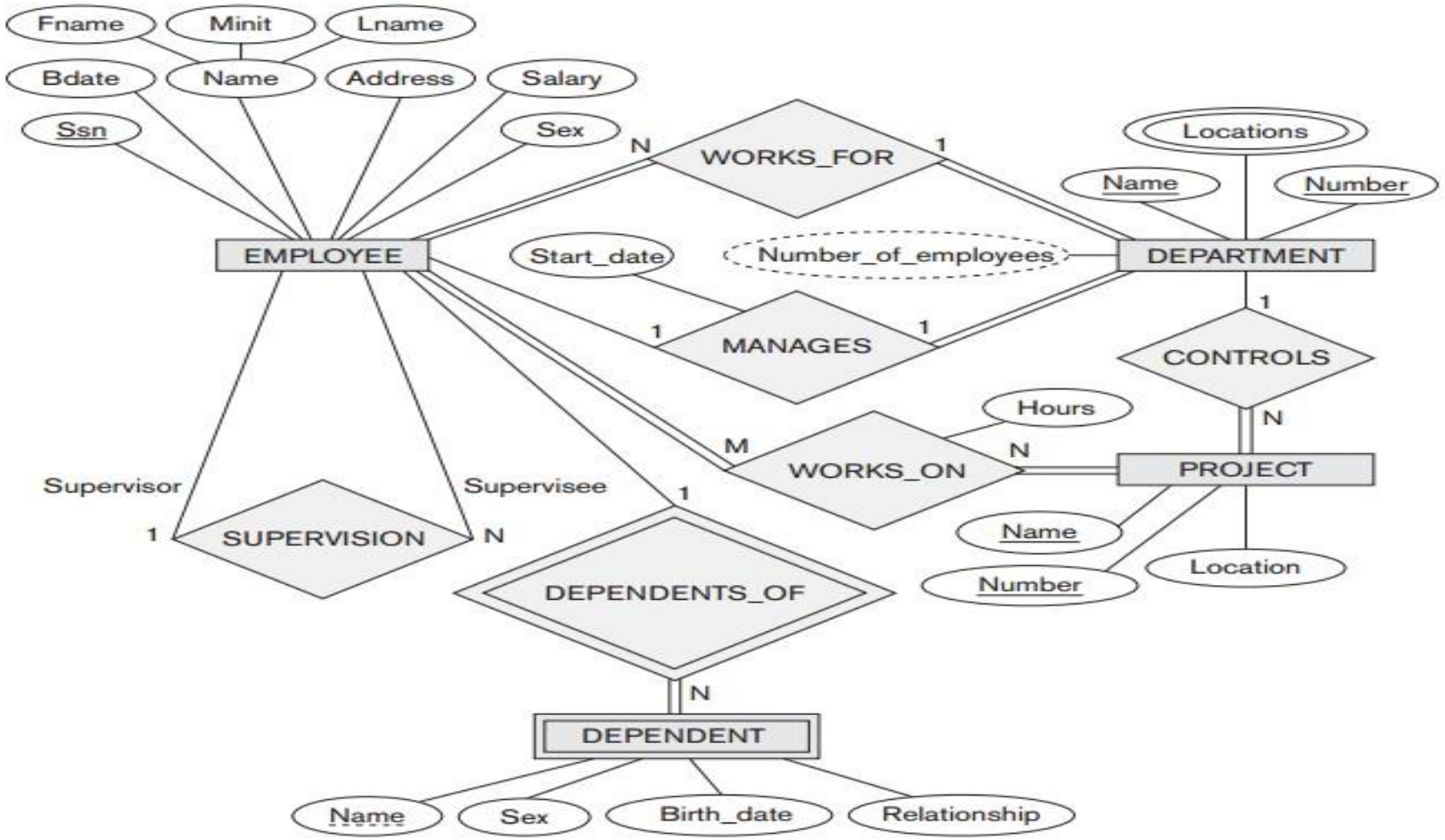








REFINING EXAMPLE ER DESIGN



Enhanced ER

- We use EER when basic concepts of the ER model are not sufficient to represent the requirements of the newer, more complex applications
- It is ER plus the concepts of (specialization & generalization, and aggregation).
- It also includes the concepts of Subclass, Superclass and Inheritance
- **Superclass:** is an entity type that **includes distinct subgroups with their own unique attributes** that should be represented in the model
- **Subclass:** is an entity type that **is a member of the superclass with its own distinct role**. Or it is any subgroup of the Superclass.

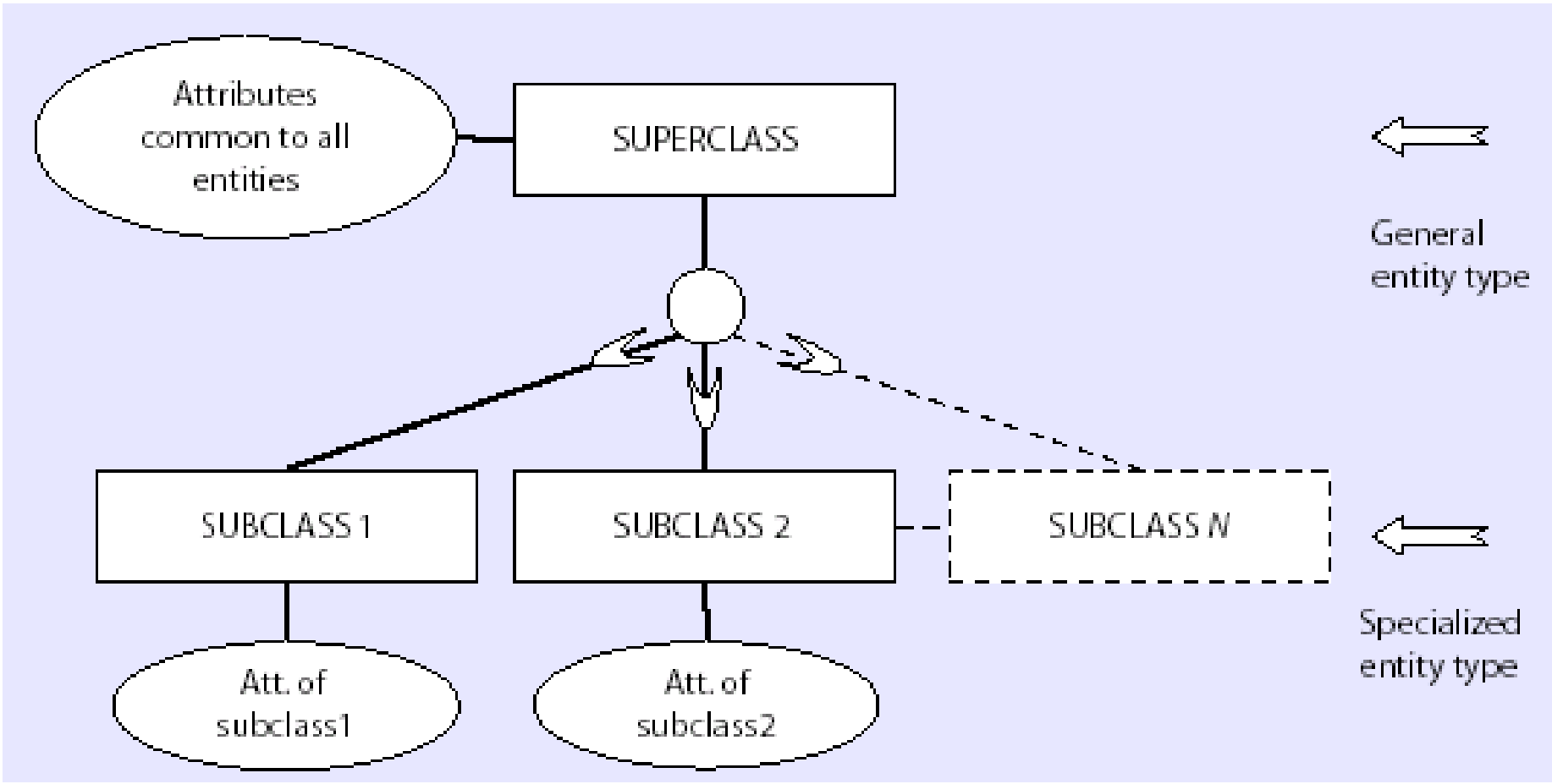
Cont…

- **Inheritance:** is an object oriented programming concept that allows one class of objects **to be defined** as a **special class** of a **more general class.**
- Subclass **inherits** all **properties** of its **super classes** and can define its **own unique properties**.
- Inheritance → superclass + subclass
- We can think of a **reverse process** of **abstraction** in which we suppress the differences among several entity types, **identify** their **common features**, and generalize them into a single superclass of which the original entity types are special subclasses.
- It is an **important feature of Generalization and Specialization**. It **allows lower-level entities to inherit the attributes of higher-level** entities.

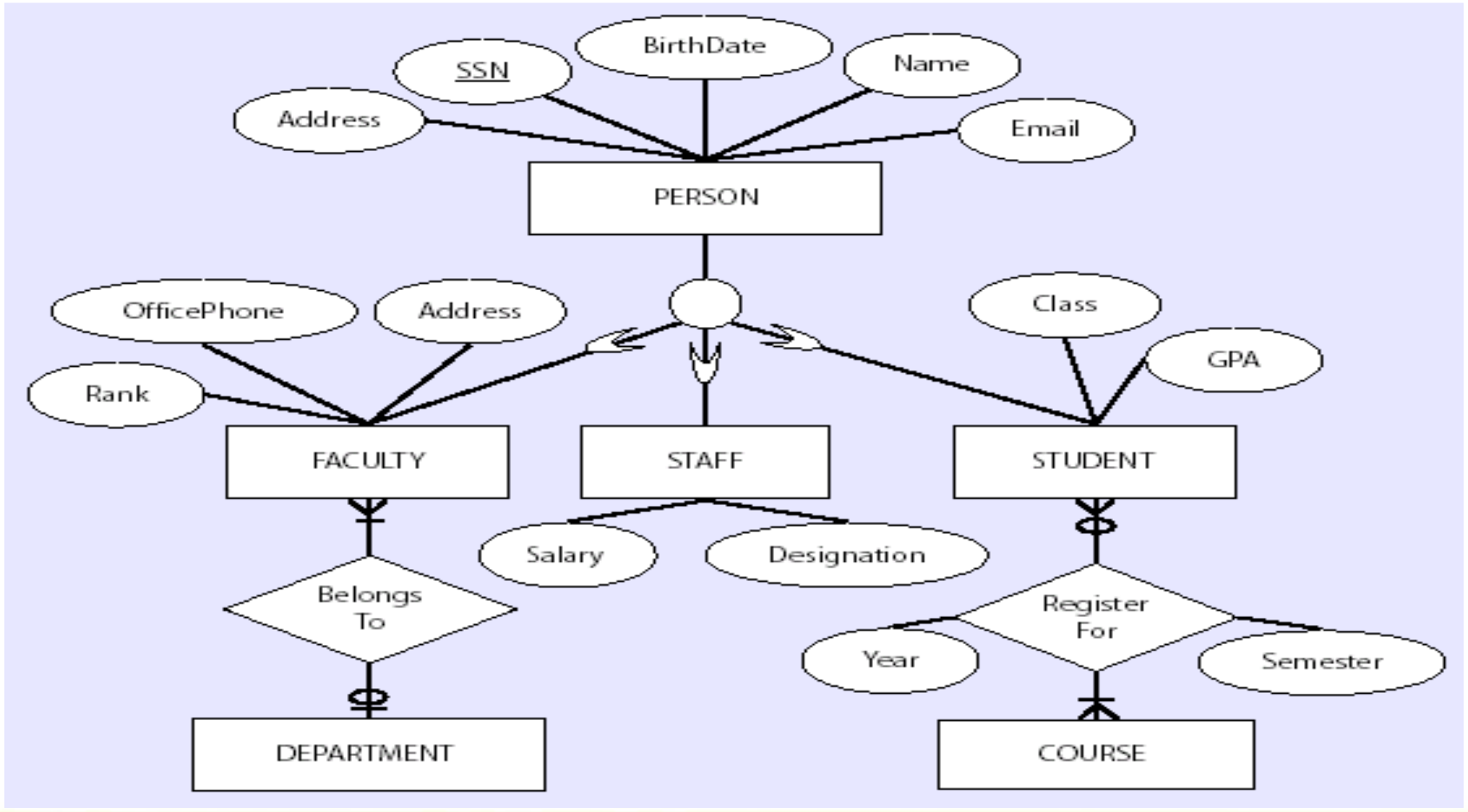
Cont…

- Inheritance is sometimes called generalization/specialization relationship.
  - **Subclass→ superclass - is Generalization**
  - **Superclass → subclass - is Specialization**
- **Generalization** is **taking** common **attributes** from the **subclasses** and **deriving a common superclass**
- **Specialization** is the process of **defining a set of subclasses of an entity type (superclass).**
  - The set of subclasses that form a specialization is defined on the basis of some distinguishing characteristic of the entities in the superclass.
  - The specialization of an entity type allows us to do the following:
    - Define a set of subclasses of an entity type
    - Establish **additional attributes** with subclasses
    - Establish **additional relationship types** between some subclasses and other entity types or other subclasses.

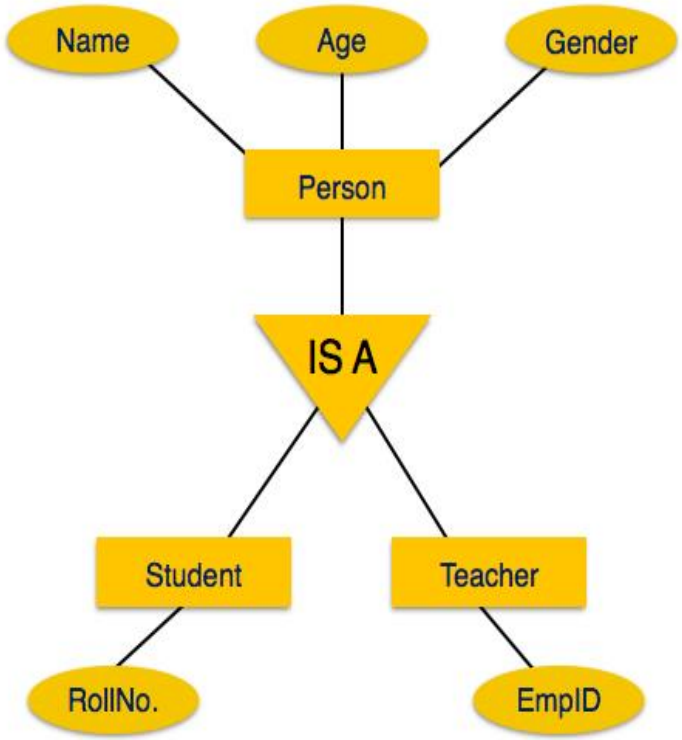
Example


Attributes common to all entities
SUPERCLASS
General entity type
SUBCLASS 1
SUBCLASS 2
SUBCLASS N
Specialized entity type
Att. of subclass1
Att. of subclass2

Superclass and Subclass Relationships Example

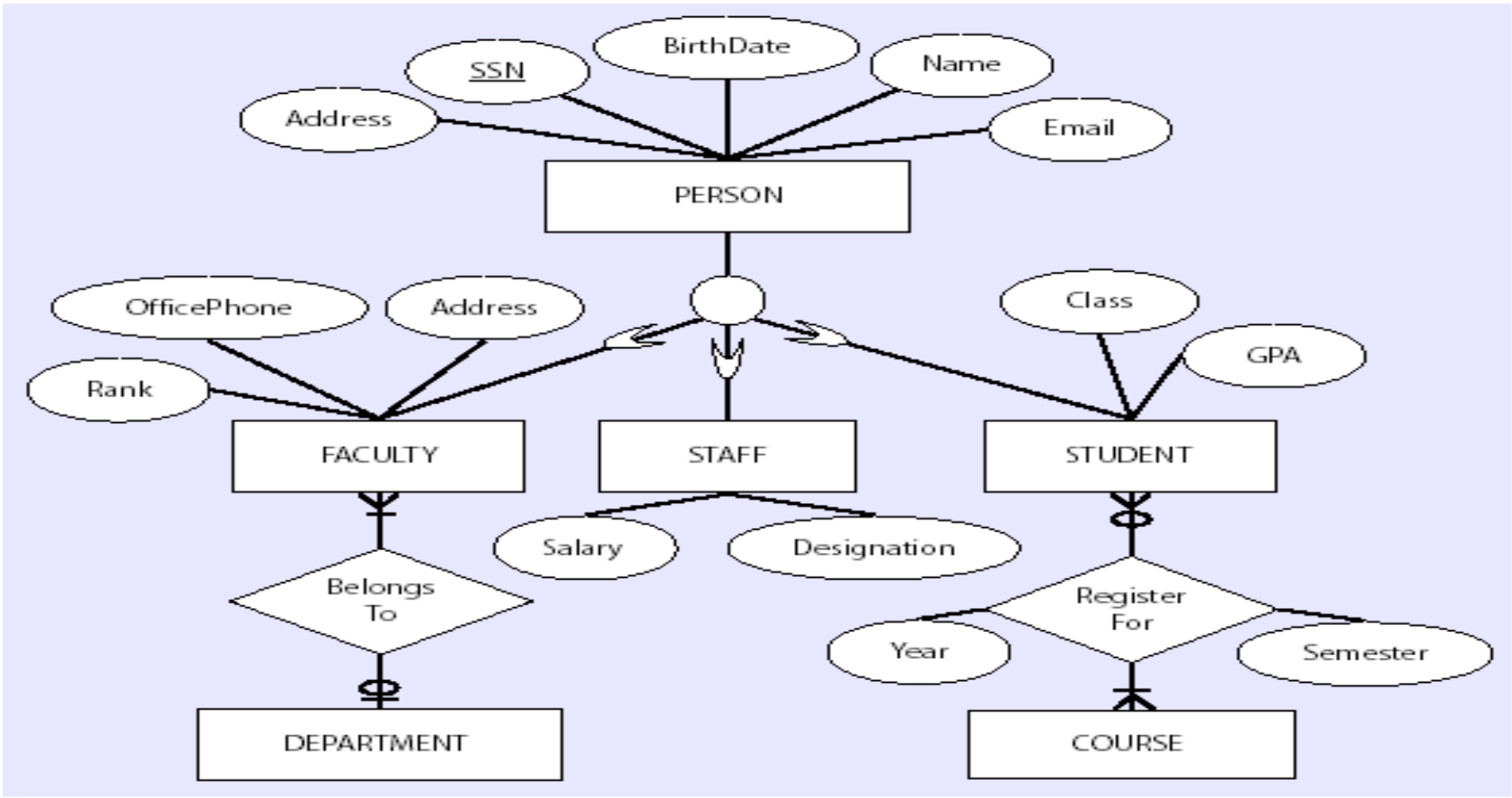


Inheritance example

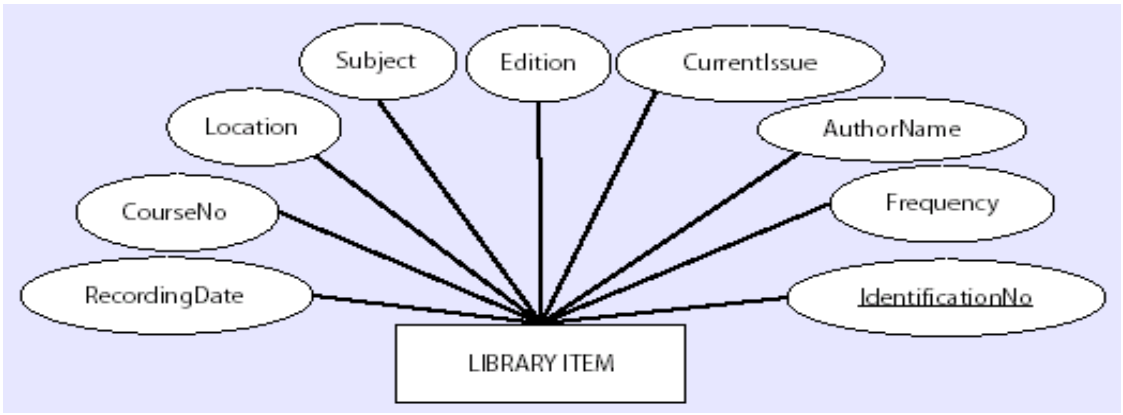


The attributes of a Person class such as name, age, and gender can be inherited by lower-level entities such as Student or Teacher.

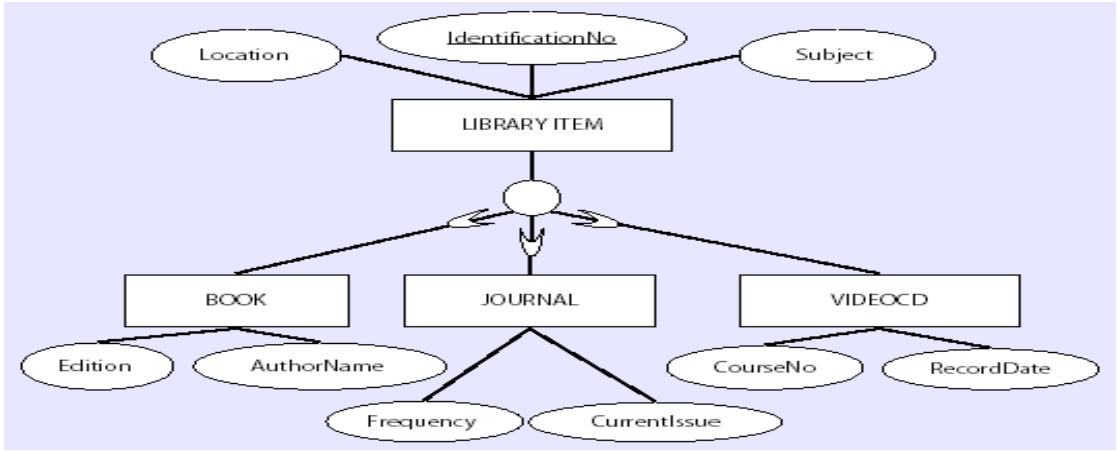
Specialization example



Specialization example



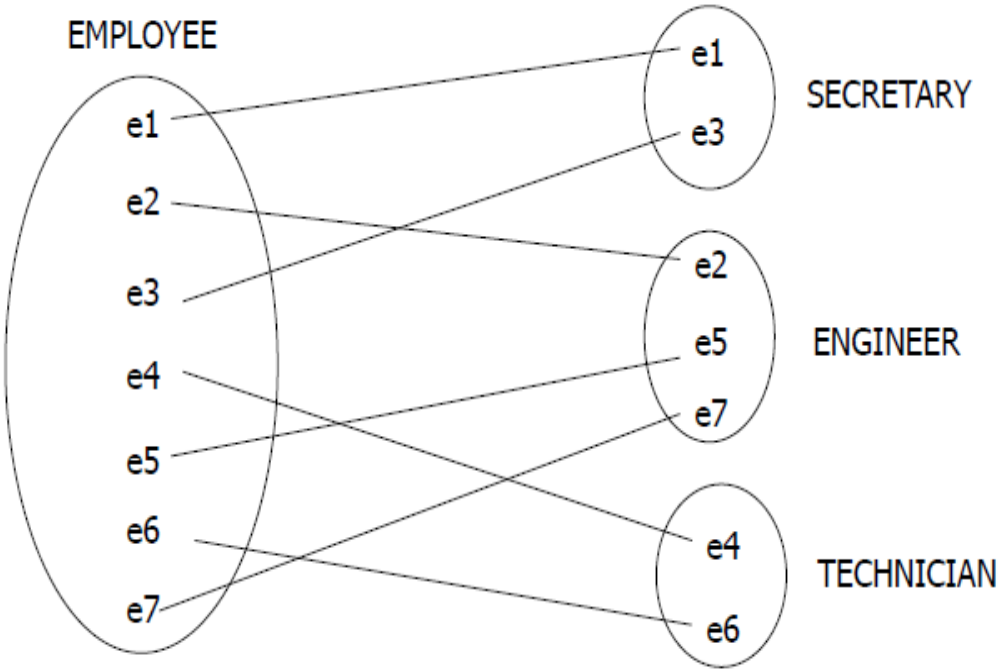
Entities Before Specialization



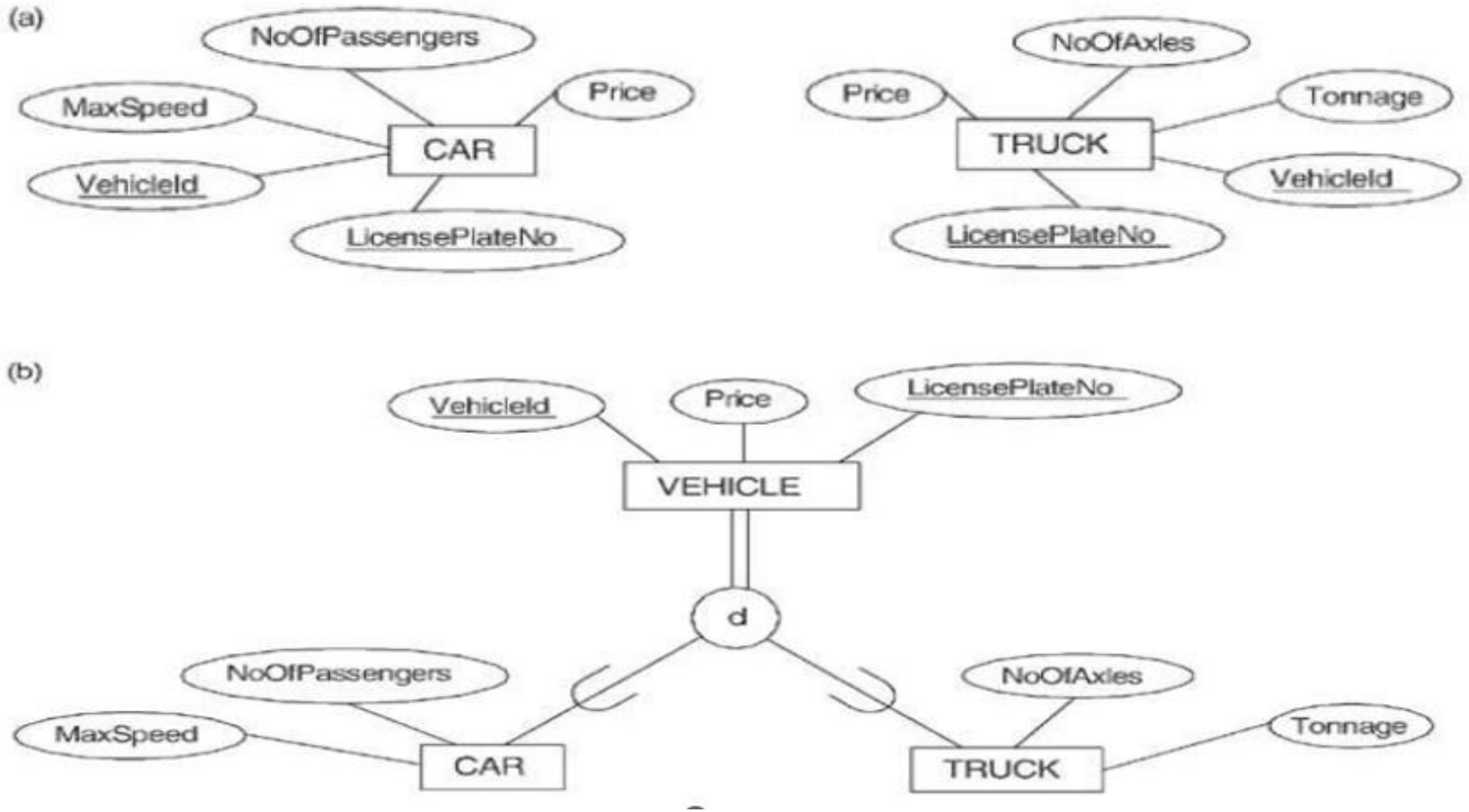
Superclass and Subclass Entities After Specialization

Specialization example

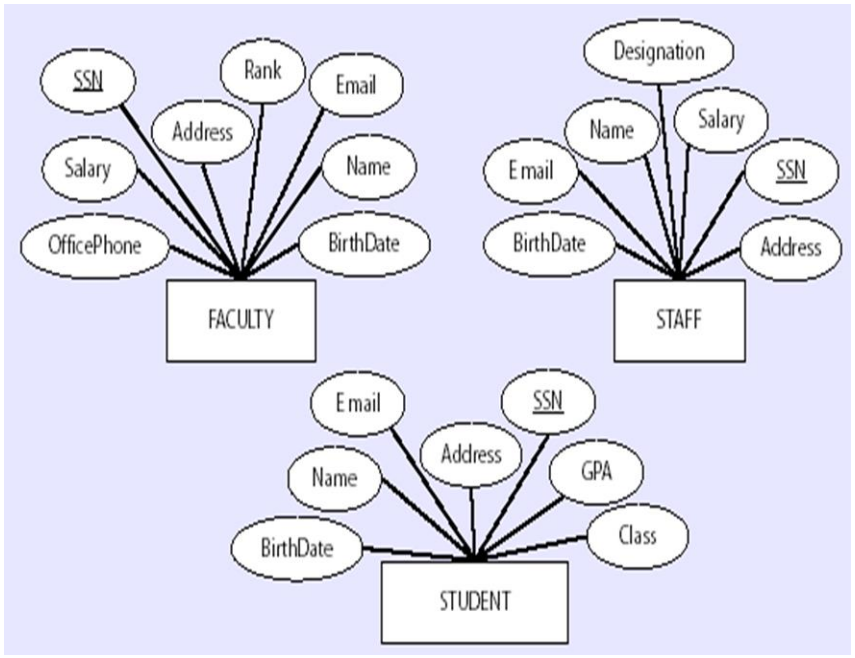
- The following Figure shows an example of some instances of the specialization of the superclass EMPLOYEE into the set of subclasses {SECRETARY, ENGINEER, TECHNICIAN}



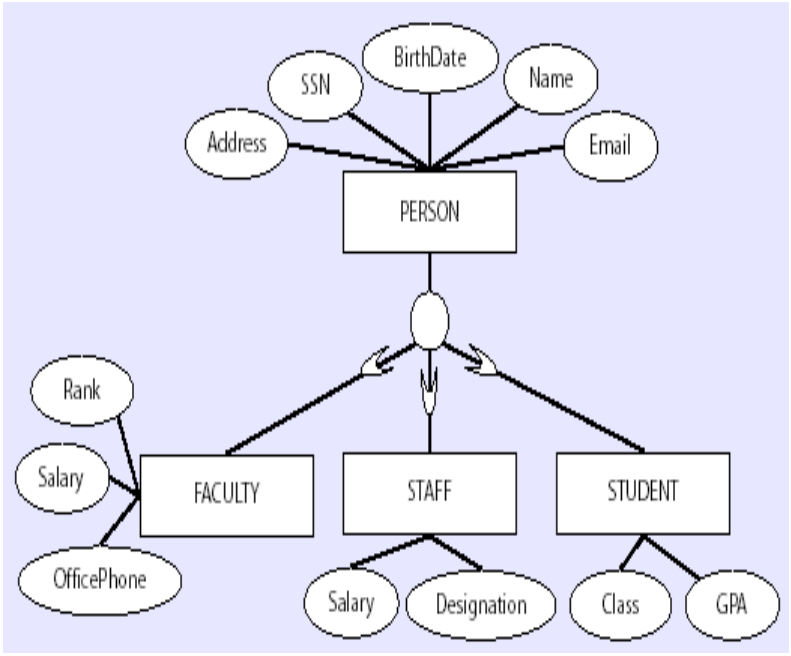
Generalization example



Generalization example



Entities Before Generalization

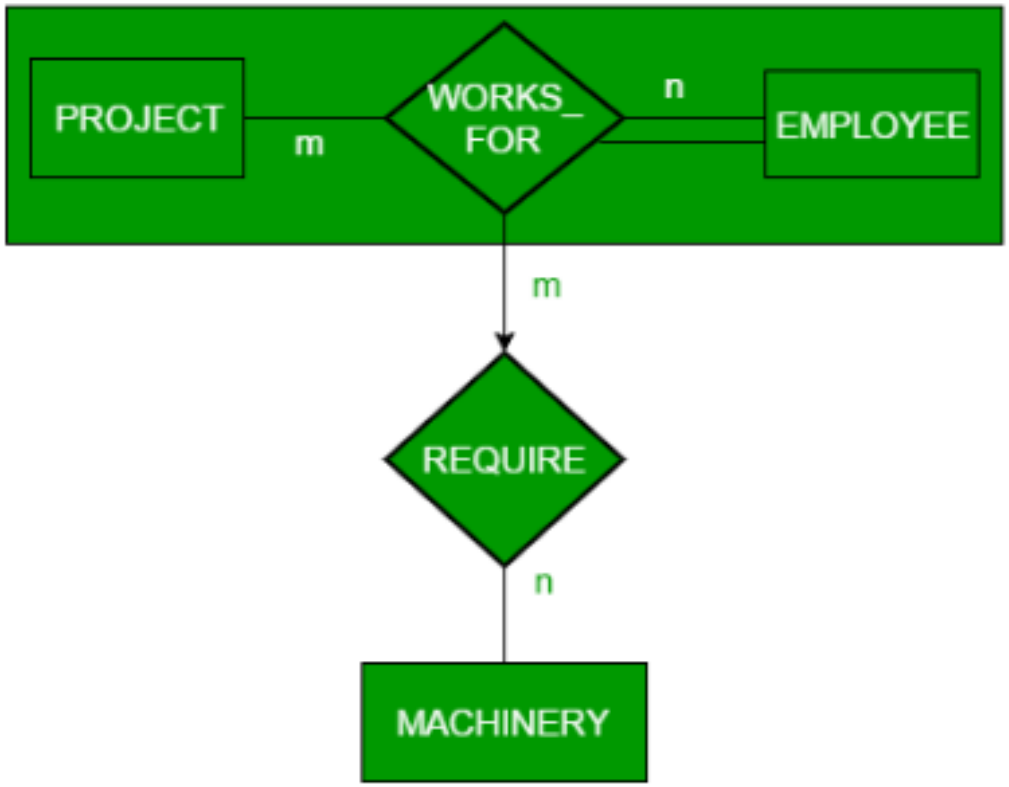


Superclass and Subclass Entities After Generalization

Aggregation

- **Relationships among relationships** are **not supported** by the **ER model. Groups of entities and relationships can be abstracted into higher level entities using aggregation.**
- Aggregation represents a **“HAS-A”** or **“IS-PART-OF”** relationship between entity types. One entity type is the whole, the other is the part. Aggregation **allows** us to **indicate** that a **relationship set participates in other relationship set.**
- For Example: **Employee working for a project may require some machinery.** So, REQUIRE relationship is needed between relationship WORKS_FOR and entity MACHINERY. Using aggregation, WORKS_FOR relationship with its entities EMPLOYEE and PROJECT is aggregated into single entity and relationship REQUIRE is created between aggregated entity and MACHINERY.
- It is represented by a line with a diamond at the end

Cont…



Aggregation

Constraints of Specialization & Generalization in DBMS

There are three constraints that may apply to specialization/generalization which are as follows:

1. Membership constraints
    - Condition-defined membership constraint
    - User-defined membership constraint
2. Disjoint constraints
    - Disjoint constraint
    - Overlapping constraint
3. Completeness constraints
    - Total completeness constraint
    - Partial completeness constraint

Cont…

1. Membership constraints: membership of a specialization generalization relationship.

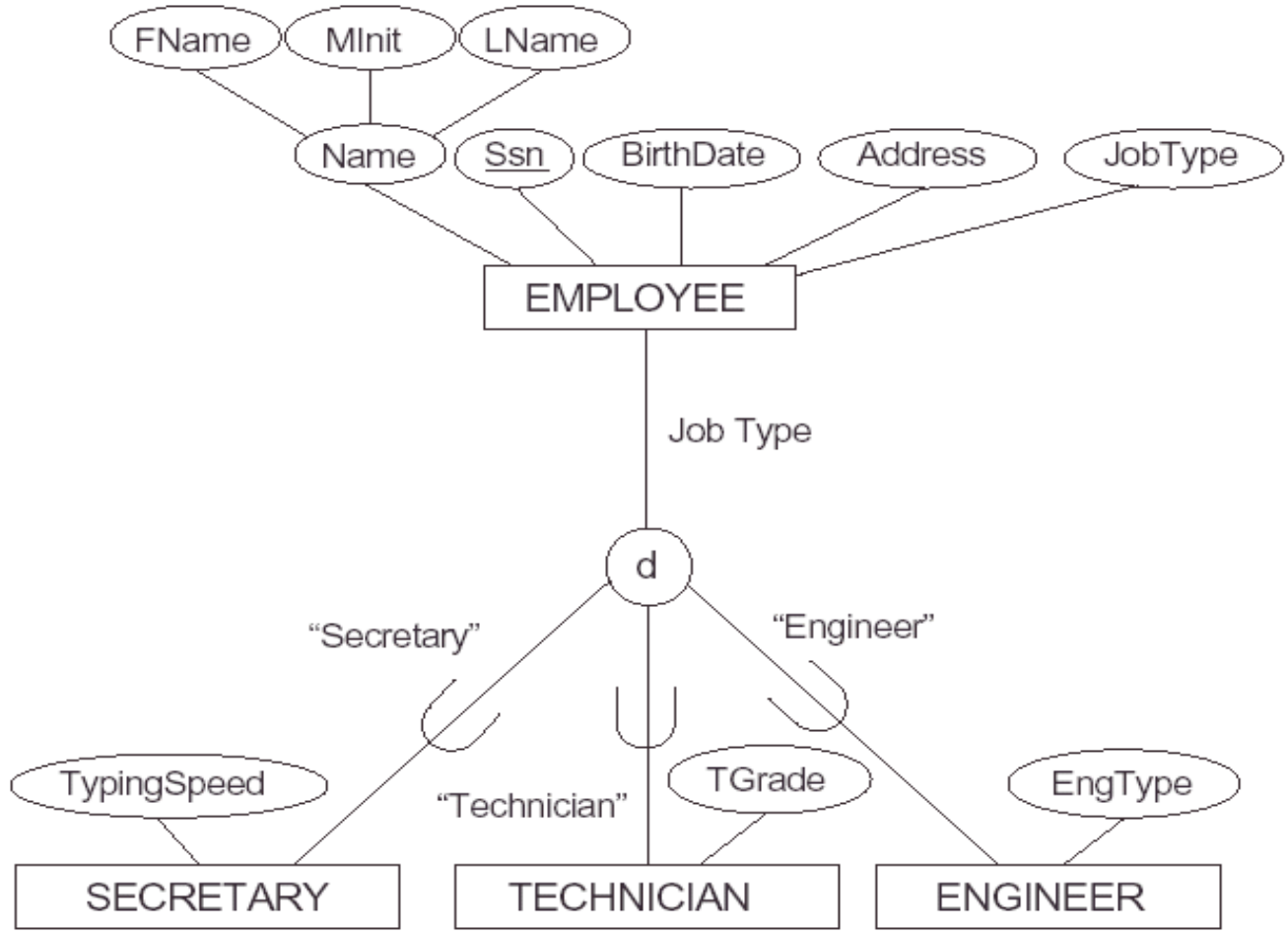
Condition/predicate defined

- In some specializations we can state a condition on the value of some attributes of the super classes.
- **The subclasses created with such specific condition are called predicate-defined (or condition-defined) subclasses.**
- Membership is **evaluated** on the basis of **whether or not an entity satisfies an explicit** condition or predicate.
- **Condition-defined constraints alone can be automatically handled by the system.**

Cont…

- Whenever any tuple is inserted into the database, its membership in the various lower level entity-sets can be automatically decided by evaluating the respective membership predicates.
  - **E.g. EMPLOYEE entity may have an attribute called JobType.**
- Then for a **SECRETARY subclass** we predicate (JobType = ‘Secretary’), which we call the defining predicate of the subclass.
- If all **subclasses in a specialization have a membership condition on the same attribute of the superclass**, the specialization itself is called an **attribute-defined specialization.**

Example of condition defined membership constraint



Cont….

User defined: sometimes the database user can define membership in subclass - superclass relationship.

- When we do not have a condition for determining membership in a subclass, the subclass is called user defined specialization.
- Hence, membership is specified individually for each entity by the user, not by any condition that may be evaluated automatically.

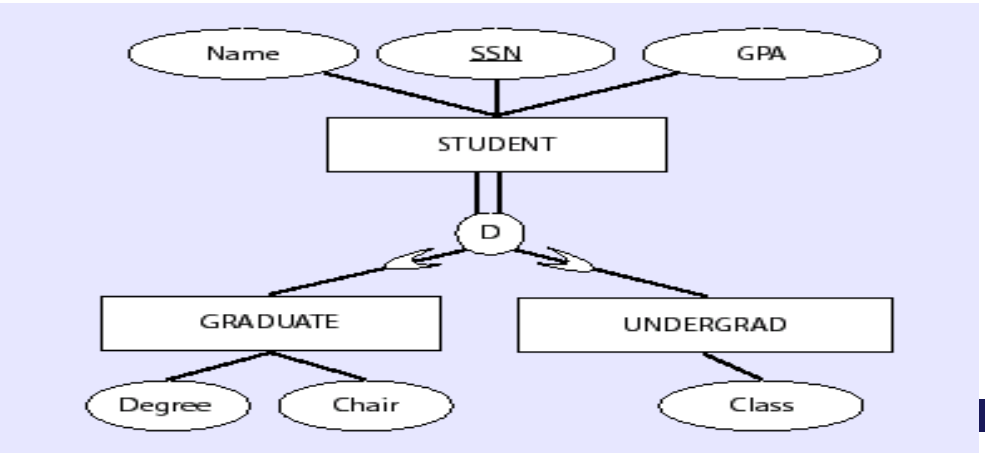
Condition defined vs user defined constraints

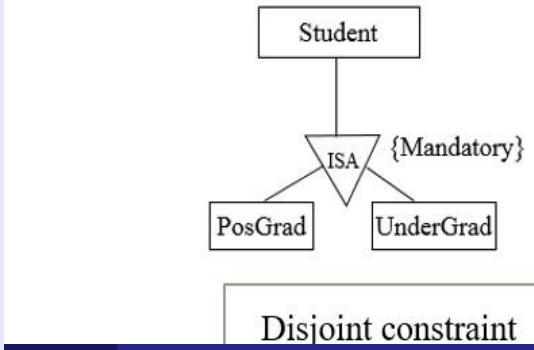
- Condition defined constraints are handled by the system while user defined constraints are handled by users.

Cont…

2. Disjoint constraints:

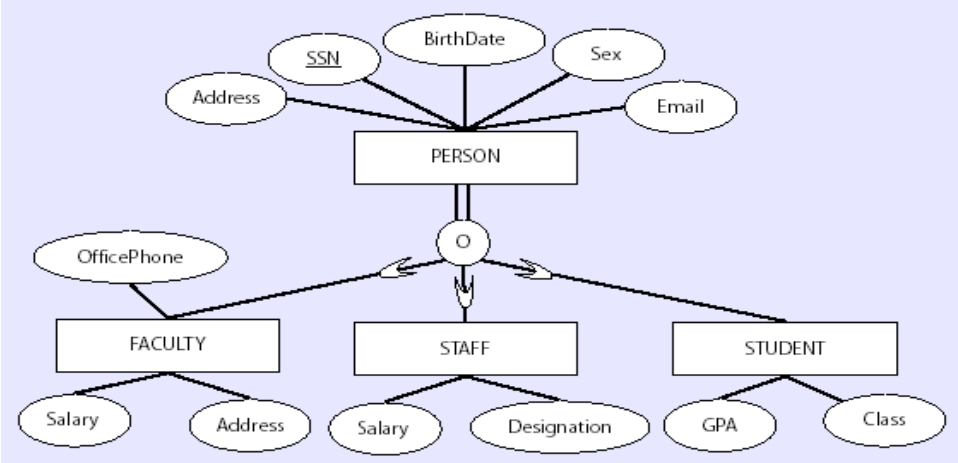
- Only **applies** when a superclass has **more than one subclass**.
- It specifies that the subclasses of the specialization must be disjoint → an **entity** can be a **member of at most one of the subclasses** of the specialization.
- **If** the **subclasses are disjoint**, then an **entity occurrence** can be a **member of only one of the subclasses.**
- E.g. postgrads or undergrads, you cannot be both at a time





Overlapping constraint

- In this category of generalizations, within a single generalization, the **same entity** may **belong** to **more than one lower-level entity set.**
- For example, in the employee work-team assume that certain **employees participate in more than one work team.** Thus, it offers a given **employee** that he **may appear** in **more than one of the team entity** sets that are lower-level entity sets of employee. Thus, generalization is overlapping.
- It is displayed by placing **O** in the circle.



Cont….

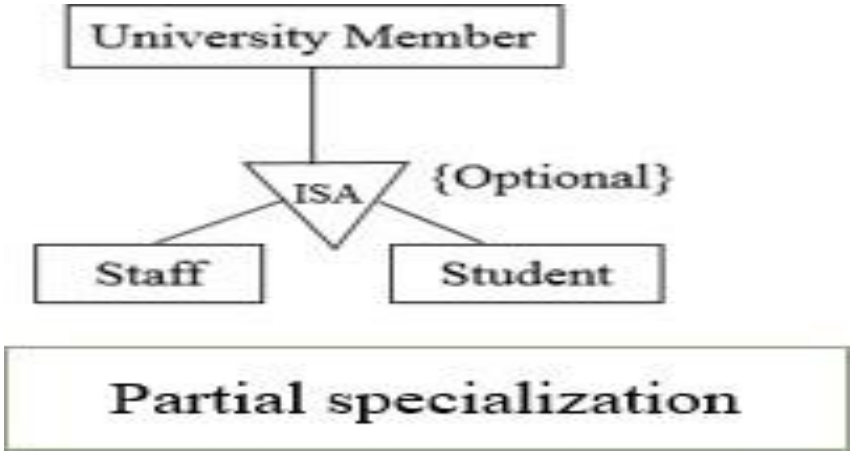
3. Completeness constraint:

- A **total specialization constraint** specifies that **every entity** in the **superclass** must be a **member** of **at least one subclass** in the specialization.
- **Each** higher-level entity must belong to a lower-level entity set.
- **E.g.** If every EMPLOYEE must be either an HOURLY-EMPLOYEE or a SALARIED-EMPLOYEE, then the specialization {HOURLY-EMPLOYEE, SALARIED-EMPLOYEE} is a total specialization of EMPLOYEE;
- This is shown in EER diagrams by **using a double line** to connect the superclass to the circle.

Cont…

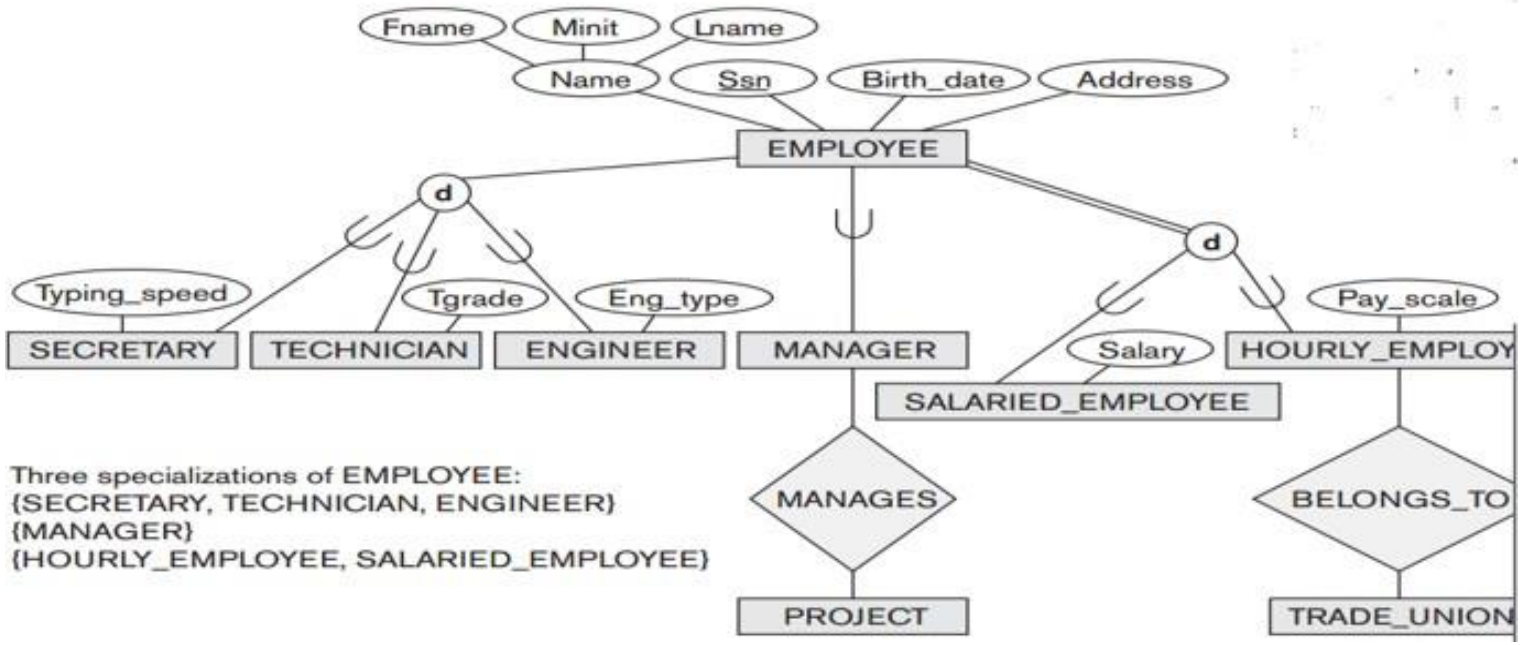
Partial specialization constraint

- **Some** higher-level entities **may not belong** to **any** lower-level entity set.
- A **single line** is used to display a partial specialization, which allows (may or may not[optional]) an entity not to belong to any of the subclasses.



Cont…

- E.g. If some EMPLOYEE entities belong to any one of the subclasses {SECRETARY, ENGINEER, TECHNICIAN }
- The figure below Shows both partial (LEFT) and total(RIGHT) specialization



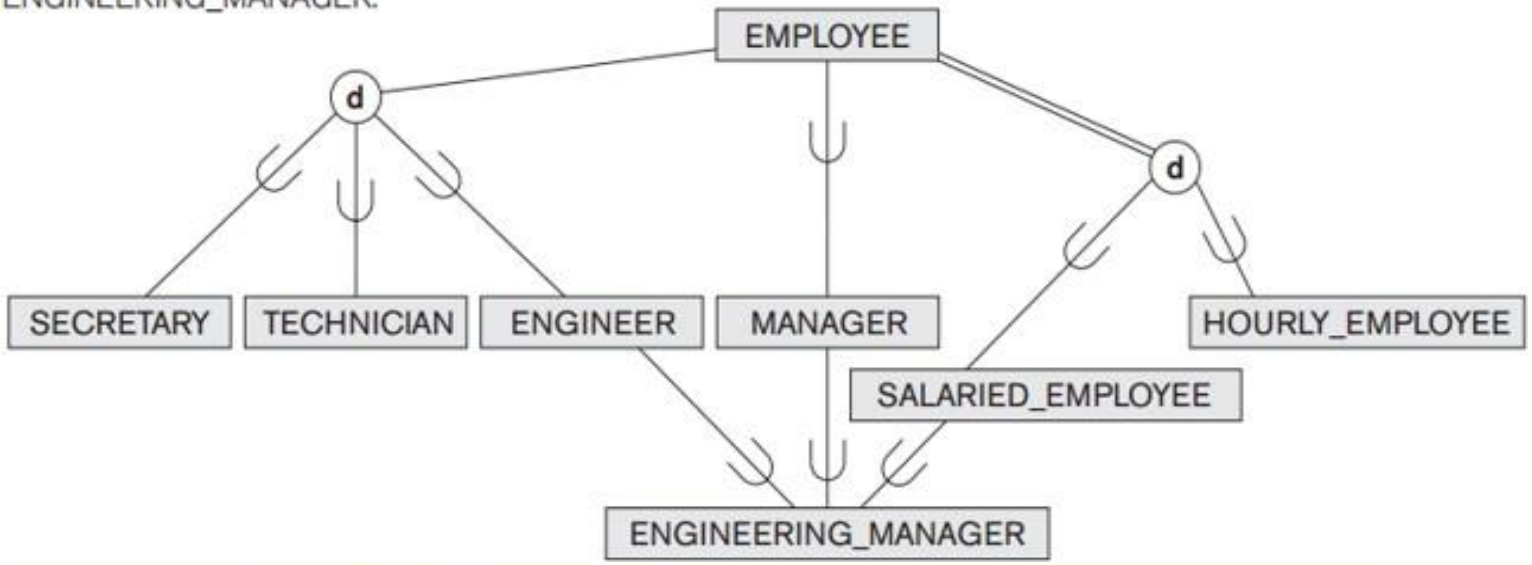
Specialization & Generalization Hierarchies, & Lattices

- A specialization hierarchy has the constraint that **every subclass participates as a subclass in only one class/subclass relationship;** i.e., **each subclass** has only **one parent,** which results in a tree structure or strict hierarchy.
- A subclass with **more than one** superclass is called a **shared subclass,** such as ENGINEERING-MANAGER
- This leads to the concept known as **multiple inheritance**, where the **shared subclass** ENGINEERING-MANAGER directly **inherits attributes** and **relationships** from **multiple super classes.**
- Notice that **the existence of at least one shared subclass** leads to **a lattice** (and **hence to multiple inheritance**).

Specialization & Generalization Hierarchies & Lattices

- If **no shared subclasses** existed, we would have a **hierarchy rather than a lattice** and only **single inheritance** would **exist**.

A specialization lattice with shared subclass ENGINEERING_MANAGER.



Modeling of UNION Types Using Categories

- It is sometimes **necessary** to **represent** a **collection** of **entities** from different entity types.
- In this case, a **subclass** will represent a **collection of entities** that is a **subset of the UNION of entities** from distinct entity types; we call such a subclass a **union type** or a **category**.
- E.g., suppose that we have three entity types: **PERSON**, **BANK**, and **COMPANY**.
- In a **database for motor vehicle registration**, an **owner** of a vehicle can be a **person**, a **bank** (holding a license on a vehicle), or a **company**.
- We **need to create a class** (collection of entities) that includes entities of all three types to play the role of vehicle **owner**.

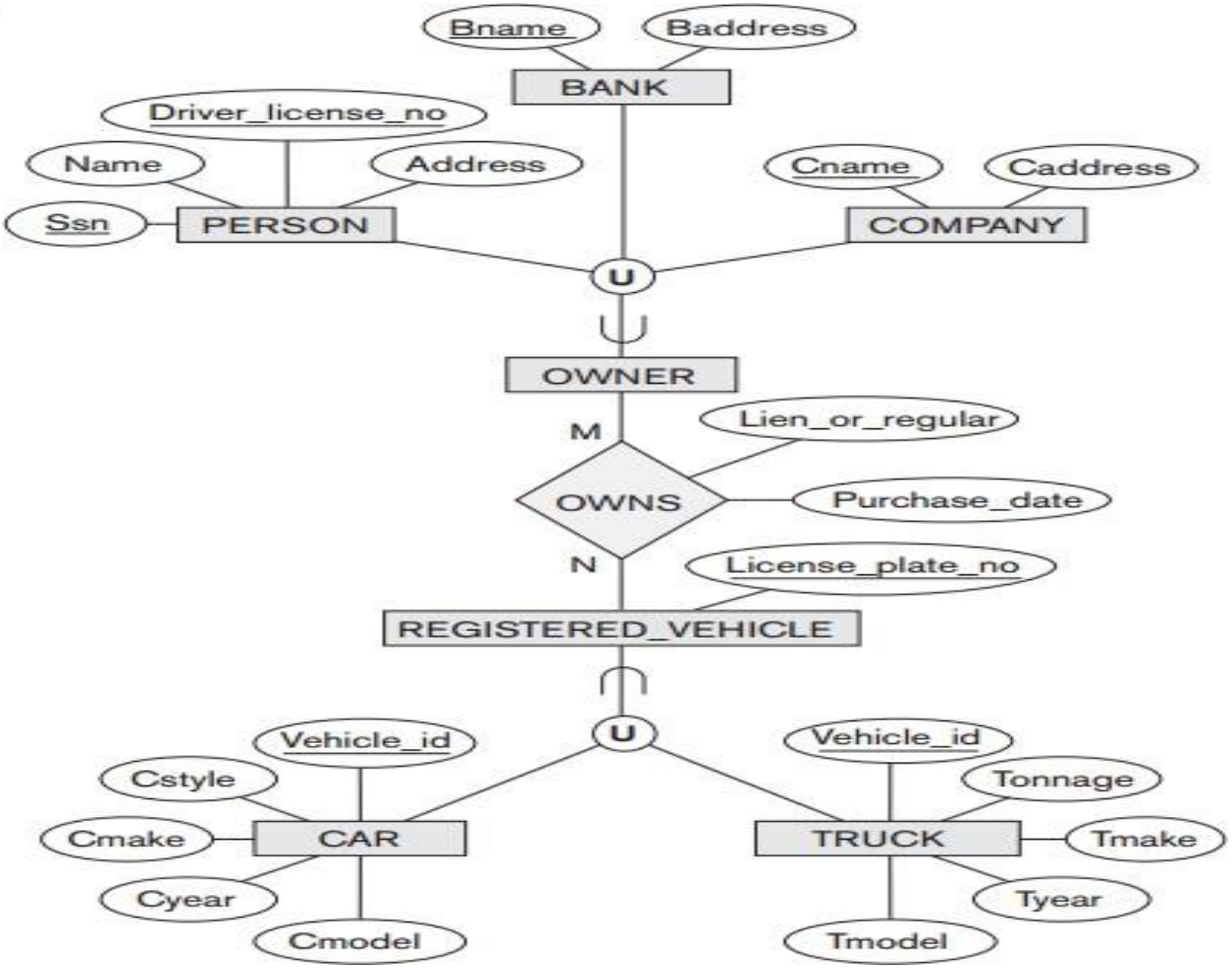
Modeling of UNION Types Using Categories

- A category (union type) **OWNER** that is a subclass of the UNION of the three entity sets of COMPANY, BANK, and PERSON can be created for this purpose.
- We display categories in an EER diagram as shown slide 76
- The super classes **COMPANY**, **BANK**, and **PERSON** are **connected** to the circle with the **U symbol**, which stands for the **set union operation**.
- An arc with the subset symbol connects the circle to the (subclass) OWNER category.

Modeling of UNION Types Using Categories

- In (slide 76) we have **two categories**:
  - OWNER, which is a union of PERSON, BANK, and COMPANY; and
  - REGISTERED-VEHICLE, which is a union of CAR and TRUCK.
- A **category has two or more super classes** that may represent collections of entities from distinct entity types **whereas other superclass/subclass relationships always have a single superclass.**

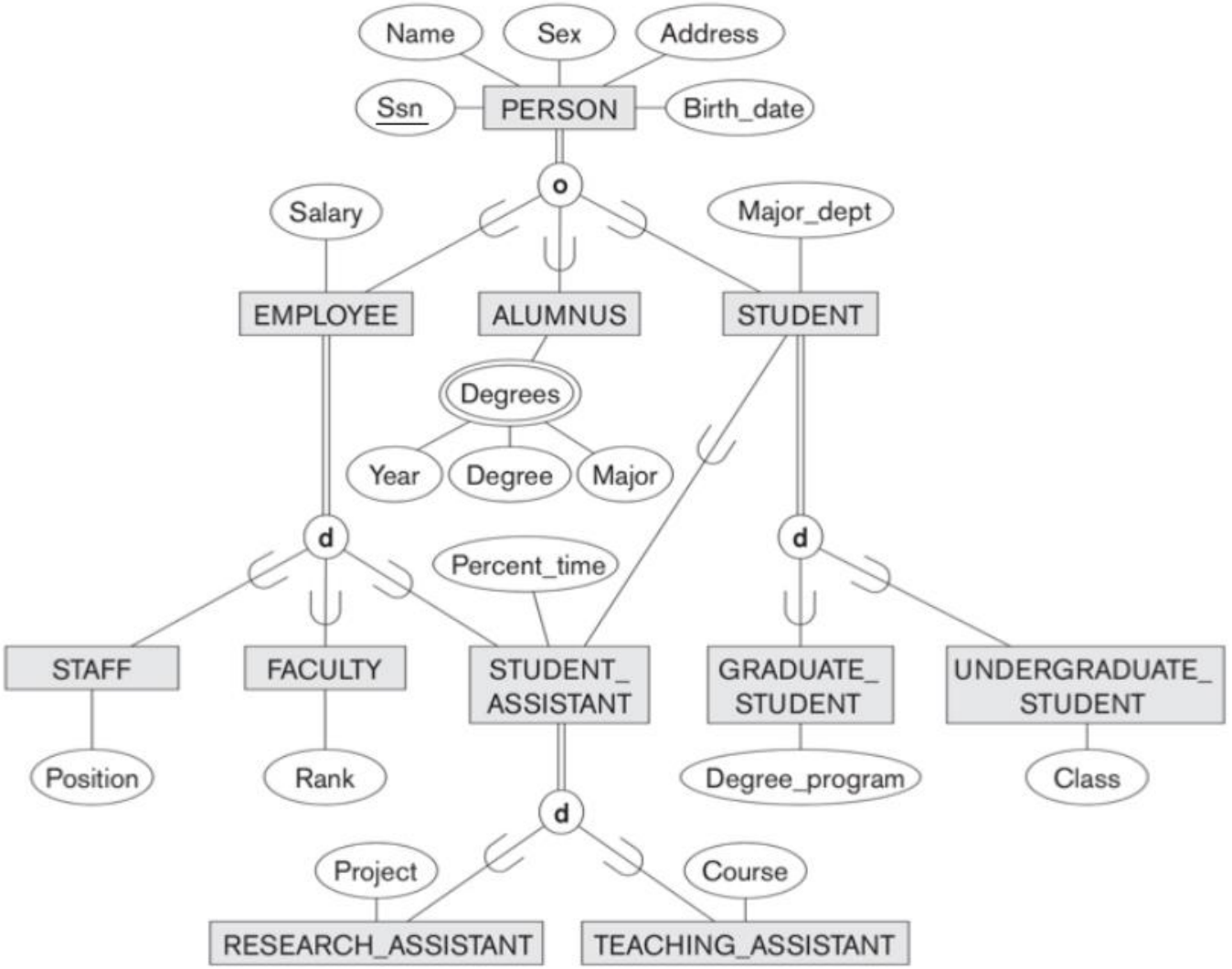
Modeling of UNION Types Using Categories



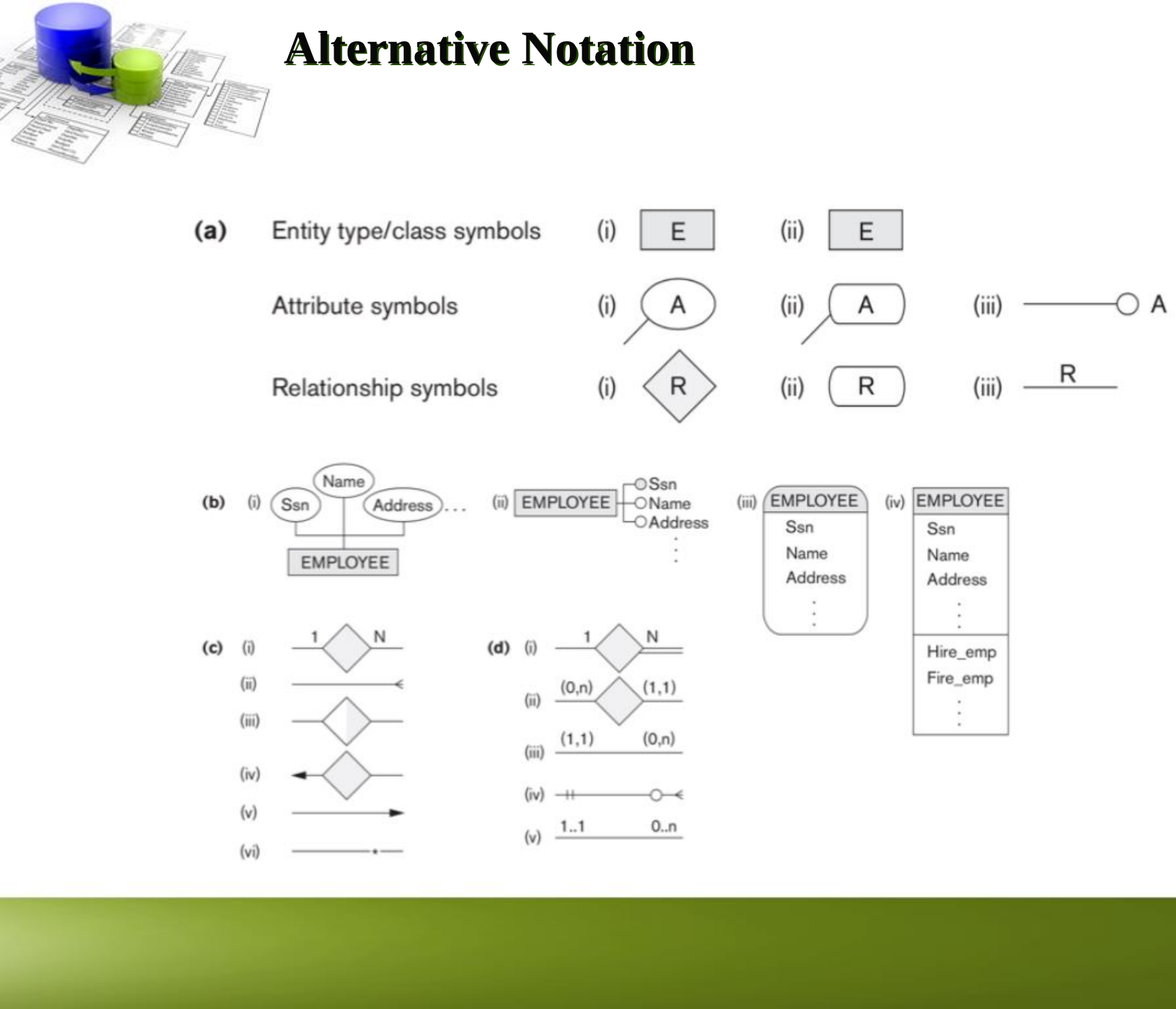
Exercise

- The database keeps track of three types of persons: employees, alumni, and students. A person can belong to one, two, or all three of these types. Each person has a name, SSN, sex, address, and birth date.
- Every employee has a salary, and there are three types of employees: faculty, staff, and student assistants. Each employee belongs to exactly one of these types. For each alumnus, a record of the degree or degrees that he or she earned at the university is kept, including the name of the degree, the year granted, and the major department. Each student has a major department.
- Each faculty has a rank, whereas each staff member has a staff position. Student assistants are classified further as either research assistants or teaching assistants, and the percent of time that they work is recorded in the database. Research assistants have their research project stored, whereas teaching assistants have the current course they work on.
- Students are further classified as either graduate or undergraduate, with the specific attributes degree program (M.S., Ph.D., M.B.A., and so on) for graduate students and class (freshman, sophomore, and so on) for under- graduates.

Answer

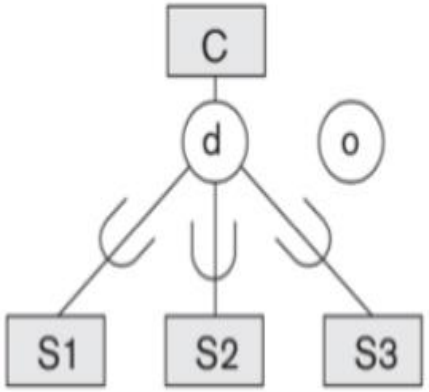

Name
Sex
Address
Ssn
PERSON
Birth_date
o
Salary
Major_dept
EMPLOYEE
ALUMNUS
STUDENT
Degrees
Year
Degree
Major
d
Percent_time
d
STAFF
FACULTY
STUDENT_ ASSISTANT
GRADUATE_ STUDENT
UNDERGRADUATE_ STUDENT
Position
Rank
Degree_program
Class
d
Project
Course
RESEARCH_ASSISTANT
TEACHING_ASSISTANT

Alternative Notation

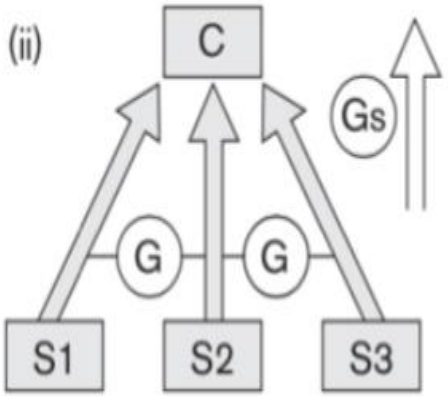


Cont…

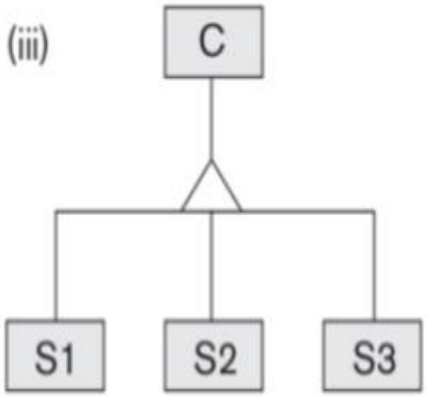
(e) (i)



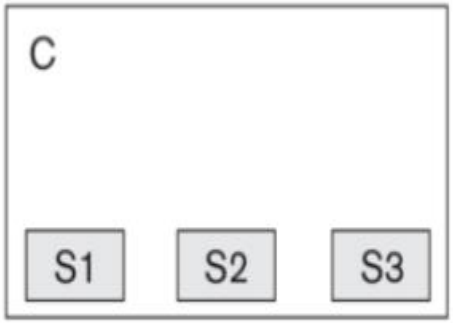
(ii)



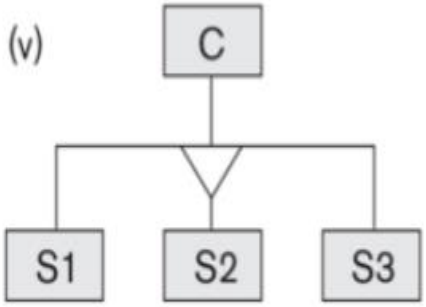
(iii)



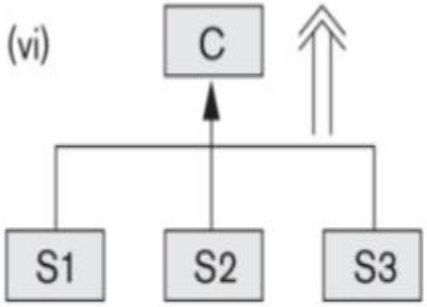
(iv)



(v)



(vi)



Thank you !!!