Module 2:
Entity– Relationship Data Model

Outline

- The Entity-Relationship (ER) Model
- Entity types: Weak and strong entity sets
- Entity sets
- Types of Attributes
- Keys

Database Design

- In the creation of database system, the design of database is the most important and initial part.
- The **overall success of the system** is completely depends upon the design of database.
- The design is mainly focused on the security and access module of data by the application program.
- The requirements of user play an important role in database design.
- The process of doing database design generally consists of a number of steps which will be carried out by the database designer. Usually, the designer must :
  1. Determine the data to be stored in the database
  2. Determine the relationships between the different data elements
  3. Superimpose a logical structure upon the data on the basis of these relationships

1. Determine the data to be stored in the database

- Database designer is a person with expertise in the area of database design. It is **not necessary that he should** be expertise in the domain from which is the data is retrieved e.g. financial information, biological information etc.
- Hence it is **important that the data to be stored in the database should be determined in cooperation with the domain expertise** that has knowledge that which data must be stored within the system.
- **This process is a part of requirements analysis** and required that the database designer should get the required data from domain expertise.
- **Data to be stored can be determined by Requirement Specification**. In the life cycle of software it is known as requirement gathering phase.

2. Determining data relationships

- Once a database designer is aware of the data which is to be stored within the database, they must then **determine where dependency is within the data**.
- For example, in **a list of names and residential addresses**, assuming a situation where multiple people can have the same address, but one person cannot have more than one address; the address is dependent upon the name.
- When provided a name and the list the address can be uniquely determined; however, the inverse does not hold - when given an address and the list, a name cannot be uniquely determined because multiple people can reside at an address.
- Because an address is determined by a name, an address is considered dependent on a name.

3. Logically structuring data

- Once the relationships and dependencies amongst the various pieces of information have been determined, it is possible to **arrange the data into a logical structure** which can then be mapped into the storage objects supported by the database management system.
- Now the **important part in design is that how to represent things** like person, product, place etc.

These things can be represented in term of **entities**.

- The various entities are related with each other by one or other way.

Database Design Problems

Redundancy

- Sometimes as per requirement same data may be stored in multiple files.
- **Consider an employee having record in both Employee and Team Details files.**
- The name and address of employee is stored in both of these tables i.e, the data get duplicated.
- If such data increases, it leads to higher storage and access cost.
- This duplication of data in various files is termed as data redundancy. Such redundancy can also occur in relational schema also due to which information may be repeatedly shown.

Database Design Problems

Incompleteness

- An incomplete design is very difficult to model.
- Suppose in a IT enterprise database system, if a department like R&D is there, to which no employee is still appointed, then it becomes very difficult to represent information about this department.

Entity-Relationship (E-R) Model

- The **Entity Relationship (E-R) model** allows specifications of an enterprise schema and represents the overall logical structure of the database.
- Now a days the database related to real world applications becomes very vast and complex. Representing relations between the different elements of the database becomes difficult.

ER Model simplifies this task. It is nothing but the **design technique for database**.

- It is a graphical technique which helps to understand and organize the complex data which should not depend upon the actual database implementation.

Entity-Relationship (E-R) Model

- The real world objects can be easily mapped with entities of E-R model.
- In Entity Relationship Model, a graphical representation of a database system is generated.
- Diagrams are used in this model which are known as entity-relationship diagrams, ER diagrams or ERDs.

ER Model Components

- An ER Diagram has three main components:

1. **Entity**
2. **Attribute**
3. **Relationship**

Entity

- An entity is nothing but a thing having its own properties. These properties help to differentiate the object (entity) from other objects.
- ***An entity is a thing that exists either physically or logically.***
- An entity may be a physical object
    - such as a house or a car

OR

- an entity may be a logical concept
    - such as a house sale or a car service

OR

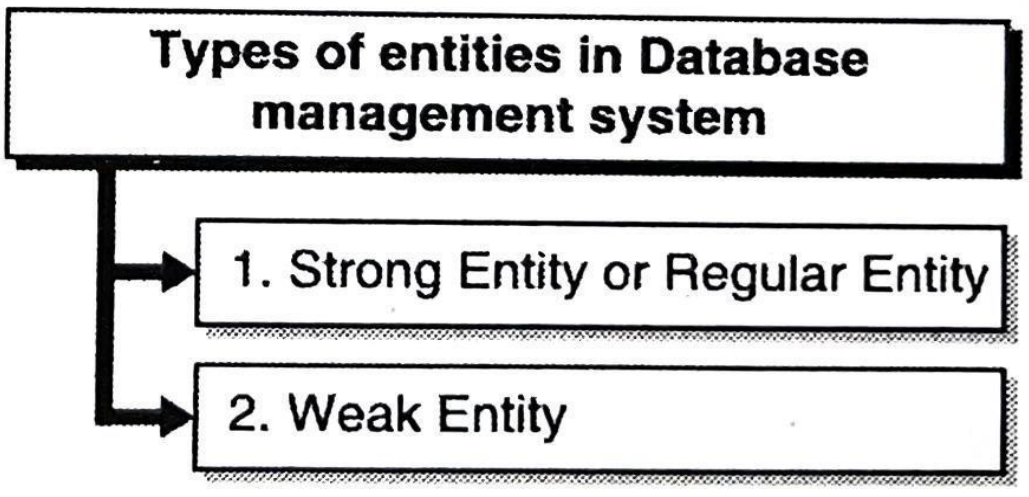
- an entity may be a concept
    - such as a customer transaction or order.

Entity

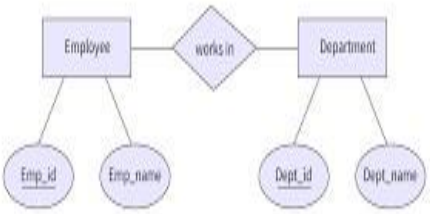
- An **entity set** is a set of entities which share the same properties.
- In a Company, employee is the **entity set** which has similar properties like Employee_ID, emp_name, salary etc.

Entity Types

- An **entity-type** is a category.
- An entity, strictly speaking, is an instance of a given entity-type.
- There are usually many instances of an entity-type.



Entity Types



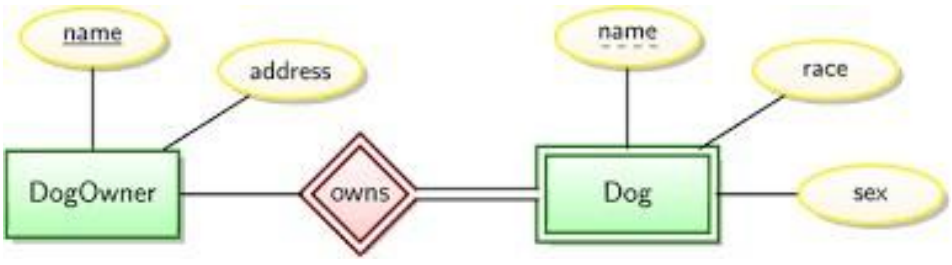
1. **Strong Entity or Regular Entity**

- If an entity having it's own key attribute specified then it is a strong entity.
- Key attribute is used to identify that entity uniquely among set of entities in entity-set.
- **Example**: In a parent/child relationship, a parent is considered as a strong entity.
- Strong entity is denoted by a single rectangle.
- The relation between two strong entities is denoted by a single diamond simply called relationship.

Entity Types

2. **Weak Entity**

- The entity which does not have any key attribute is known as weak entity.
- The weak entity has a partial discriminator key.
- Weak entity depends on the strong entity for its existence.
- Weak entity is denoted with the double rectangle.
- **Example**: In a parent/child relationship, a child is considered as a weak entity which is completely depends upon the strong entity 'parent'.



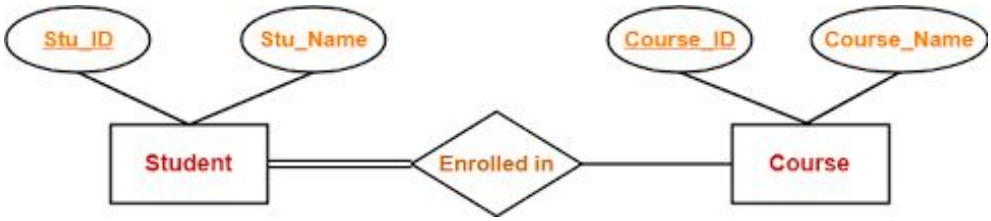
Difference: Strong and Weak Entity

Parameter	Strong Entity Set	Weak Entity Set
Basic	The Strong entity has a primary key.	The weak entity has a partial discriminator key.
Depends	The Strong entity is independent of any other entity in a schema	Weak entity depends on the strong entity for its existence.
Denoted	Strong entity is denoted by a single rectangle	Weak entity is denoted with the double rectangle
Relation	The relation between two strong entities is denoted by a single diamond simply called relationship	The relationship between a weak and a strong entity is denoted by double diamond
Example	In a parent/child relationship, a parent is considered as a strong entity.	In a parent/child relationship, a child is considered as a weak entity which is completely depends upon the strong entity 'parent'

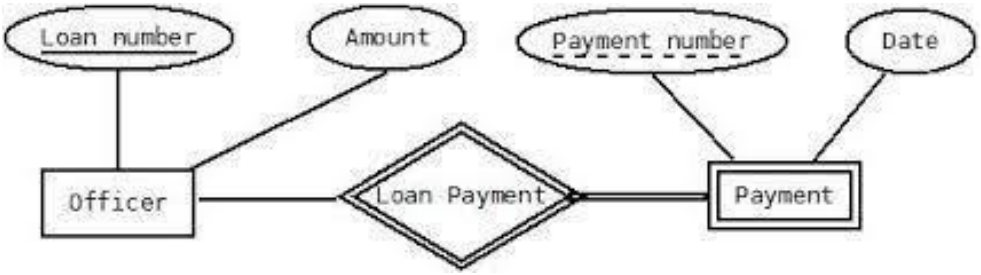
Exercise

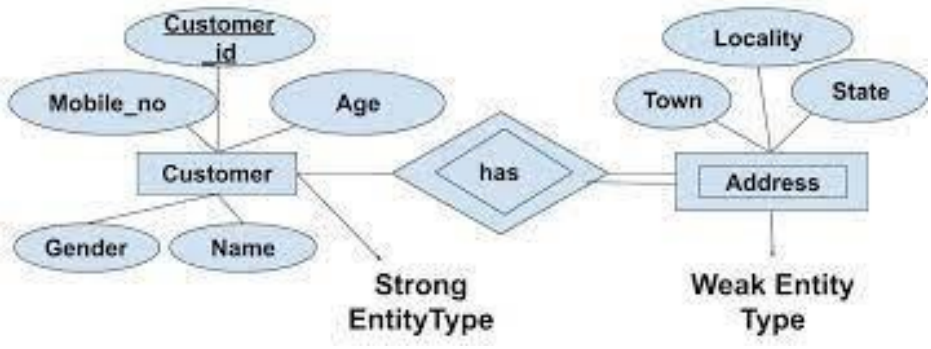
- Draw and explain one example on
  - Relationship between two strong entity
  - Relationship between one strong and one weak entity

Strong entity examples



Weak entity examples





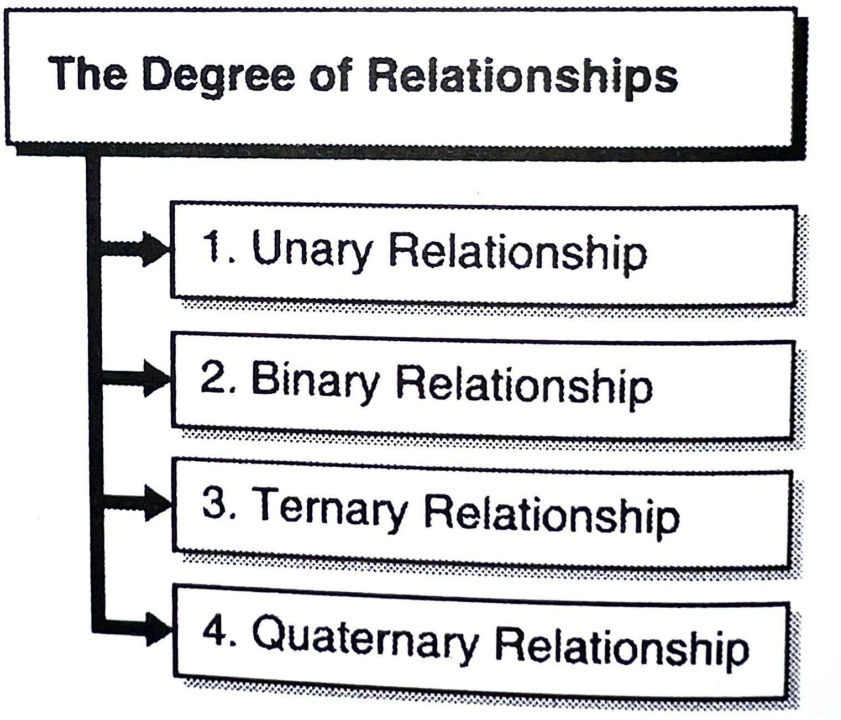
Relationships

- The association between two different entities is called as relationship.
- In the real world application, what does one entity do with the other, how do they connect to each other?
- **Example**: An employee works at a department, a student enrolls for a course. Here, works at a department, enrolls for a course.
- Here, **works** and **enrolls** are called relationships.

The Degree of Relationships

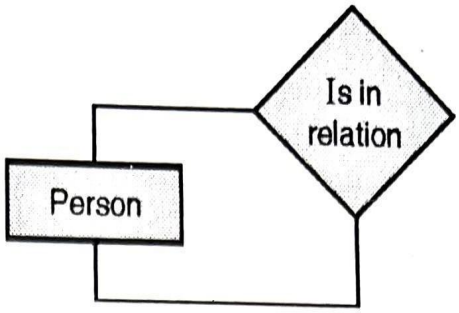
- The degree of relationship refers to number of entities participated in the relation.

Relationships

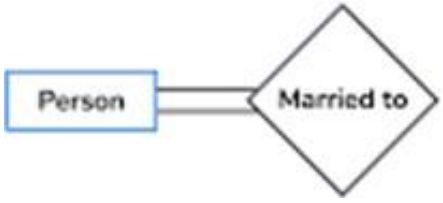


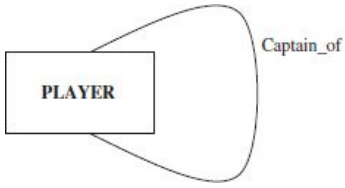
1. Unary Relationship

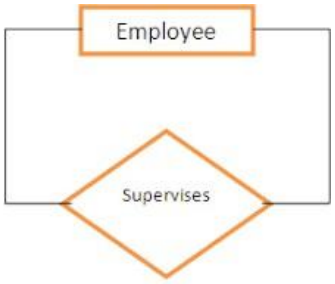
- A unary relationship exists when there is relation between single entity.
- A unary relationship is also known as recursive relationship in which an entity relates with itself.
- **Example**: A person can be in the relationship with another person, such as :
  - A woman who can be someone's mother
  - A person that is a someone's child

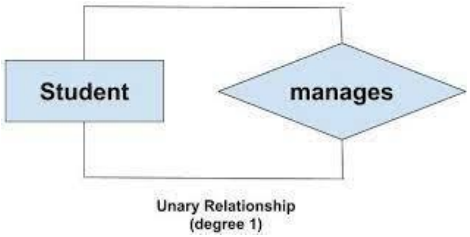


Unary Relationship Example





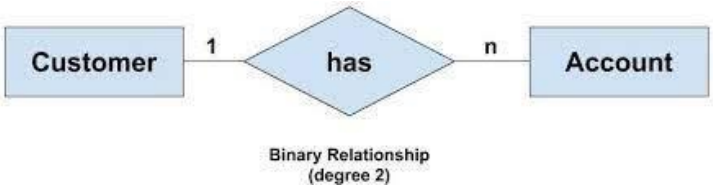




2. Binary Relationship

- A binary relationship exist only when there is relation between only two entities. In this case the degree of relation is two.
- **Example**: A teacher teaches student.
- In this teacher and student are two different entities which are connected with each other via relation Teaches.

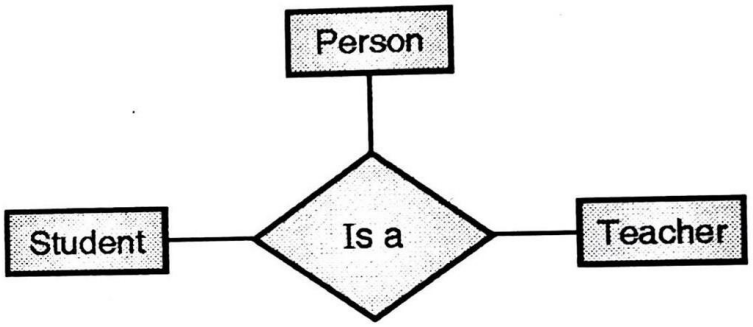
Binary Relationship Example



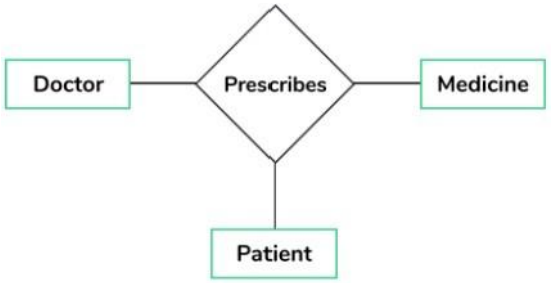
Binary Relationship
(degree 2)

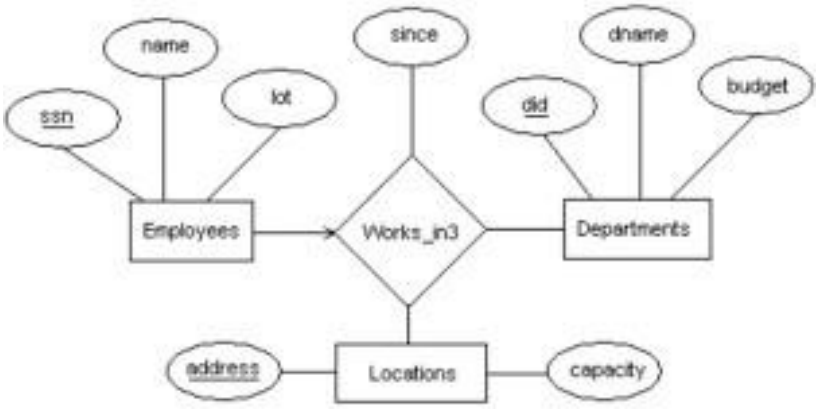
3. Ternary Relationship

- A ternary relationship exists when there are relations between three entities.
- **Example**: A person can be a student and a person also can be teacher.
- Here teacher, student and person are three entities which are related to each other.



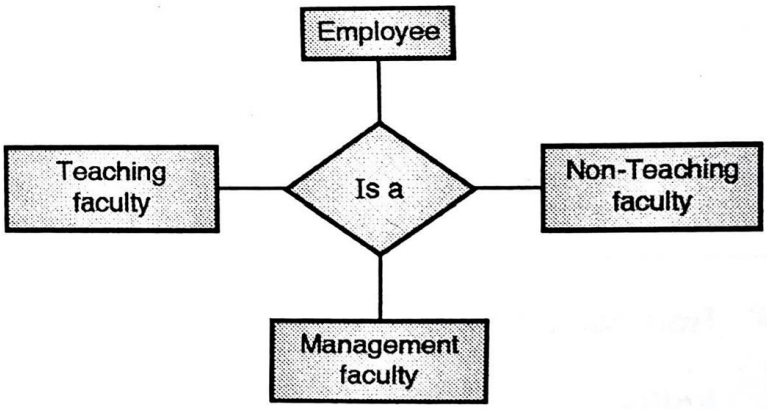
Ternary Relationship Example



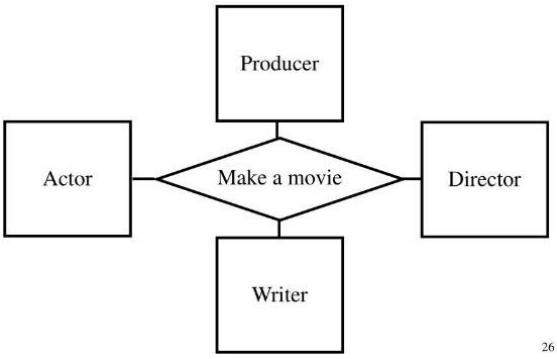


4. Quaternary Relationship

- A quaternary relationship exists when there are relations between four entities.
- **Example**: The four entities Employee, Management Faculty, Teaching Faculty, and Non-Teaching Faculty are connected with each other via is a relationship



Quaternary Relationship Example



26

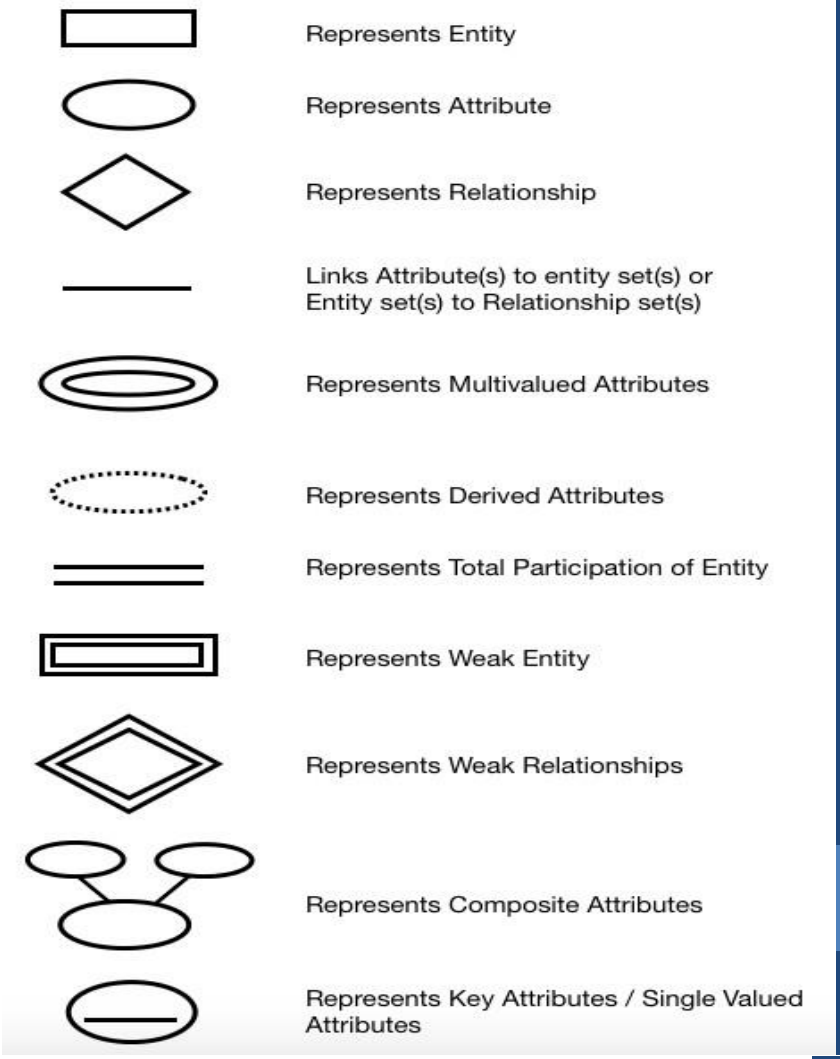
Exercise

- Draw ER Diagram to represent following relationships:
  - Unary
  - Binary
  - Ternary
  - Quaternary

ER Diagram: Symbols

- ER model is a graphical representation of entities.
- The pictorial representation of data using different conventions which state that how these data are related with each other is known as **Entity Relationship Diagram**.
- ER diagrams express the logical structure of database in graphical manner.
- Special symbols are used to draw an ER Diagram.
- Every symbol has its own meaning.

ER Diagram: Symbols



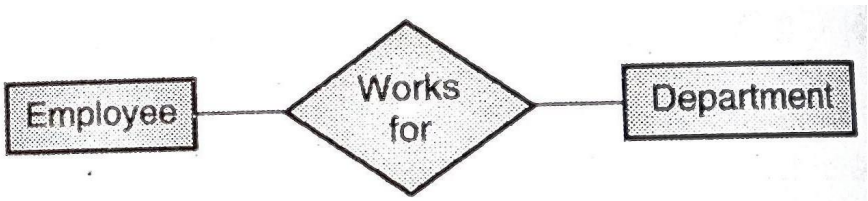
ER Diagram: Symbols

1. **Entity**

- An Entity is any object, place, person or class.
- In E-R Diagram, an entity is represented using rectangles.
- Consider an example of an Organization.

Employee, Manager, Department, Product etc. are considered as entities.

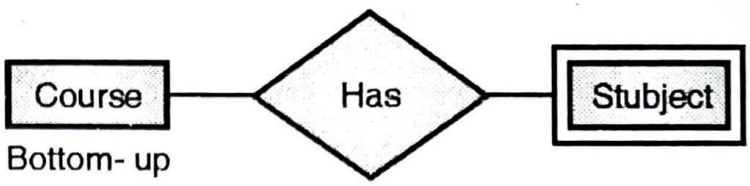
- Here **employee** and **department** are entities.



ER Diagram: Symbols

2. **Weak Entity**

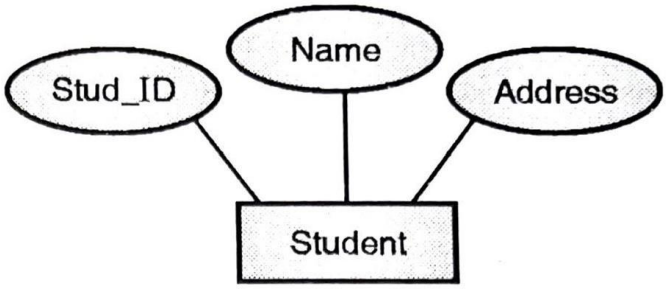
- Weak entity is an entity which depends upon another entity.
- Weak entity is represented by double rectangle.
- Subject is the weak entity Because **subject** is depends on course.



ER Diagram: Symbols

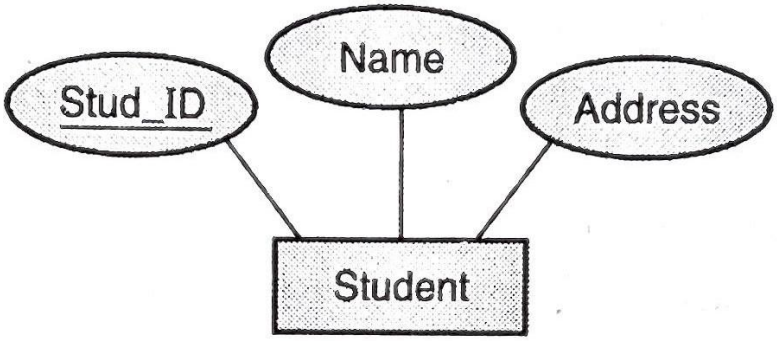
3. **Attribute (Single)**

- Attributes are nothing but the properties of entity. Here Stud_id, Name and address are attributes of entity Student.



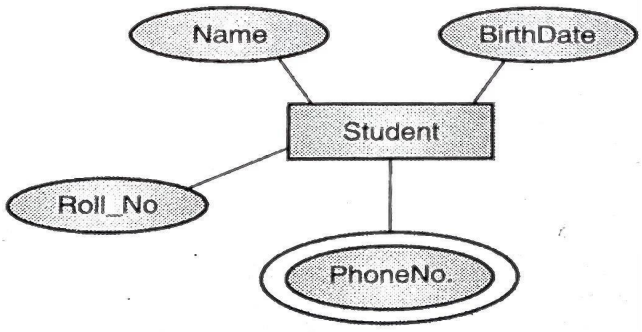
ER Diagram: Symbols

4. **Key Attribute (Primary key)**
   - To identify attribute uniquely we set the key to the attribute. It is denoted by underline.



ER Diagram: Symbols

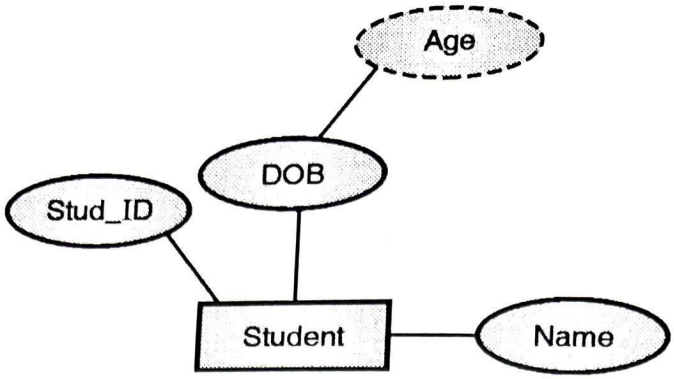
5. **Multi valued Attribute**
   - The attribute which have multiple values is known as multi valued attribute.
   - Here Phone_No is multi valued attribute as a person can have more than one phone numbers.



ER Diagram: Symbols

6. Derived Attribute

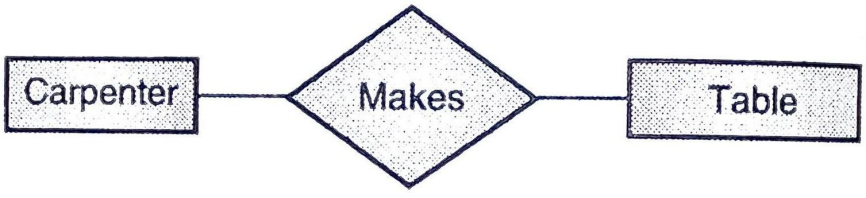
- Derived attributes are the attributes that do not exist physically in the database, but their values can be derived from other attributes present in the database.
- Example: Age can be derived from data_of_birth.



ER Diagram: Symbols

7. Relationship

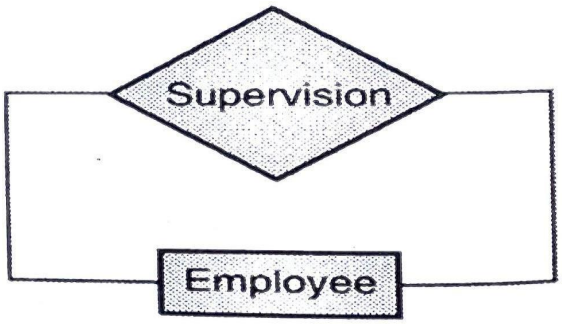
- A relationship describes how entities interact with each other.
- For example, the entity "carpenter" may be related to the entity "table" by the relationship "builds".
- Relationships are represented by diamond shapes and are labelled using verbs.



ER Diagram: Symbols

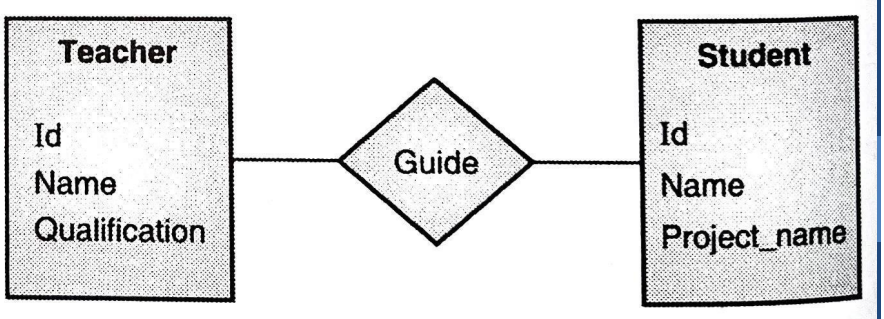
8. **Recursive Relationship**

- If the same entity participates more than once in a relationship it is known as a recursive relationship.
- Consider an example where an employee can be a supervisor and be supervised by manager, so there is a recursive relationship.



ER Diagram: Symbols

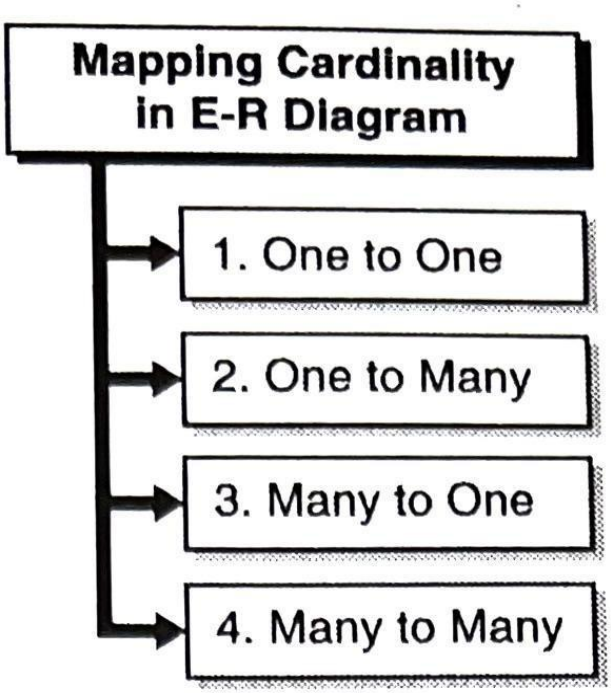
- Following E-R diagram represent the relationship between two entity sets teacher and student related through binary relationship guide.
- **The attributes of entity set teacher are:**
  - Id
  - Name
  - Qualification
- **The attributes of entity set student are :**
  - Id
  - Name
  - Project_name



Exercise: ER Diagram Symbols

- Represent following ER Diagram Symbols by considering any one sample relation
  - Entity
  - Weak Entity
  - Attributes
  - Key Attribute(primary key)
  - Multivalued attributes
  - Derived Attribute
  - Relationship

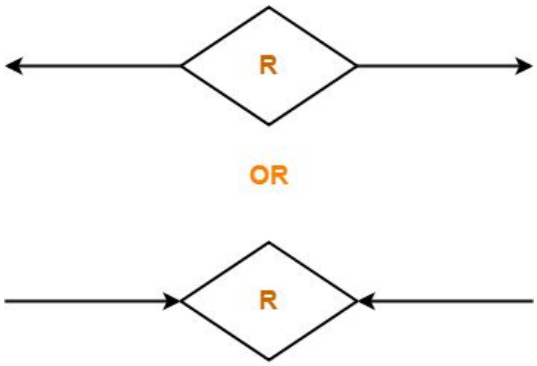
Relationship Constraints: Cardinality and Participation



Relationship Constraints: Cardinality

1. **One to One**

- An entity in set A can be associated with at most one entity in set B.
- An entity in set B can be associated with at most one entity in set A.
- **Symbol Used-**



Cardinality Ratio = 1 : 1

Relationship Constraints: Cardinality

- **Example-** Consider the following ER diagram-

One to One Relationship

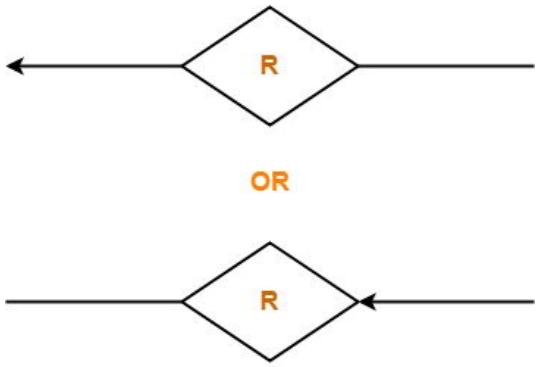
Here,

- One student can enroll in at most one course.
- One course can be enrolled by at most one student.

Relationship Constraints: Cardinality

2. **One to Many**

- An entity in set A can be associated with any number (zero or more) of entities in set B.
- An entity in set B can be associated with at most one entity in set A.
- **Symbol Used-**



Cardinality Ratio = 1 : n

Relationship Constraints: Cardinality

- **Example-** Consider the following ER diagram-

One to Many Relationship

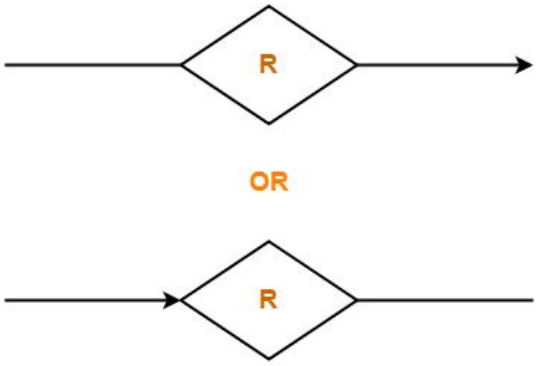
Here,

- One student can enroll in any number (zero or more) of courses.
- One course can be enrolled by at most one student.

Relationship Constraints: Cardinality

3. Many to One

- An entity in set A can be associated with at most one entity in set B.
- An entity in set B can be associated with any number (zero or more) of entities in set A.
- **Symbol Used-**



Cardinality Ratio = m : 1

Relationship Constraints: Cardinality

- **Example-** Consider the following ER diagram-

Many to One Relationship

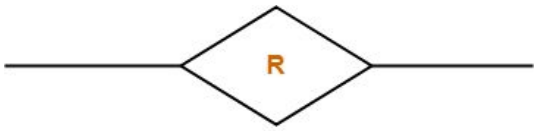
Here,

- One student can enroll in at most one course.
- One course can be enrolled by any number (zero or more) of students.

Relationship Constraints: Cardinality

4. Many to Many

- An entity in set A can be associated with any number (zero or more) of entities in set B.
- An entity in set B can be associated with any number (zero or more) of entities in set A.
- **Symbol Used-**



Cardinality Ratio = m : n

- **Example-** Consider the following ER diagram-

Many to Many Relationship

Here,

- One student can enroll in any number (zero or more) of courses.
- One course can be enrolled by any number (zero or more) of students.

Points to Remember: ER Diagram

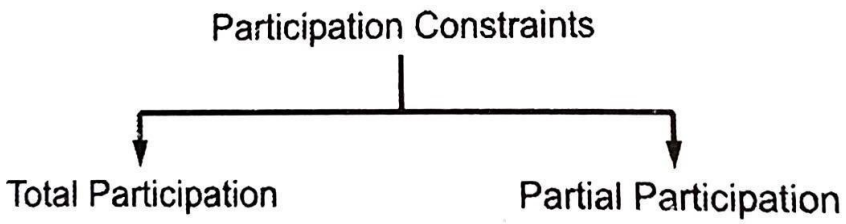
1. Initially identify all entities and their relationships with each other in the given database system.
2. No entity should be repeated in a particular diagram.
3. Provide a precise and appropriate name for each entity, attribute, and relationship in the diagram. Try to give user friendly words while naming. The name should also be meaningful, unique and easily understandable.
4. Do not set unclear, redundant or unnecessary relationships between entities.
5. Never connect a relationship to another relationship.
6. Using colors helps to make the diagram easily understandable. It helps in differentiation and classification

Participation Constraints

- Participation constraints define the least number of relationship instances in which an entity must compulsorily participate.

Types of Participation Constraints

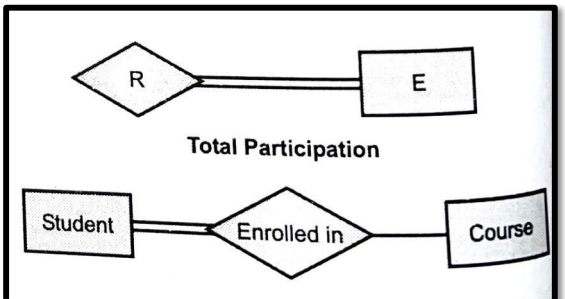
- There are two types of participation constraints.



Participation Constraints

1. **Total Participation**

- It specifies that each entity in the entity set must compulsorily participate in at least one relationship instance in that relationship set.
- That is why, it is also called as mandatory participation.
- Total participation is represented using a double line between the entity set and relationship set.
- **Example**:
  - Here, Double line between the entity set "Student" and relationship set "Enrolled in" signifies total participation.
  - It specifies that each student must be enrolled in at least one course.



Participation Constraints

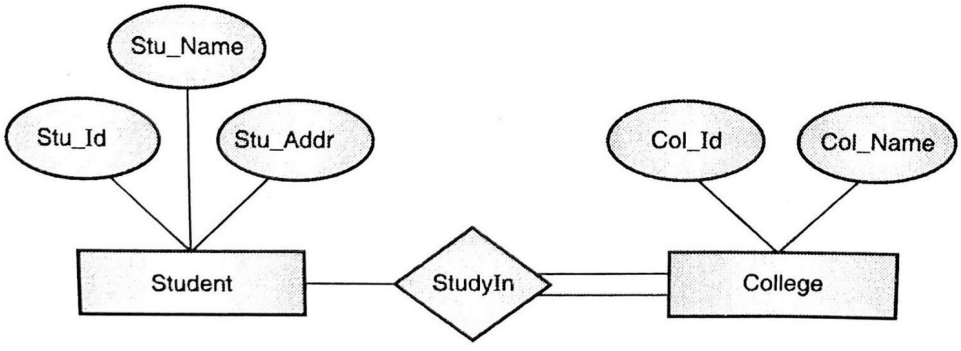
2. Partial Participation

- It specifies that each entity in the entity set may or may not participate in the relationship instance in that relationship set.
- That is why, it is also called as optional participation.
- Example
  - Partial participation is represented using a single line between the entity set and relationship set.

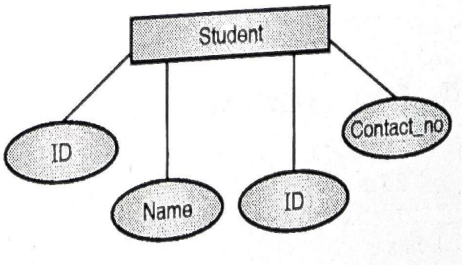
Partial Participation

Examples: ER Diagram

- Each college must have at least one associated student.
- A Total participation of an entity set represents that all the entities in the entity set should have minimum one relationship in a relationship set.
- For example, below each college must have at-least one associated Student.



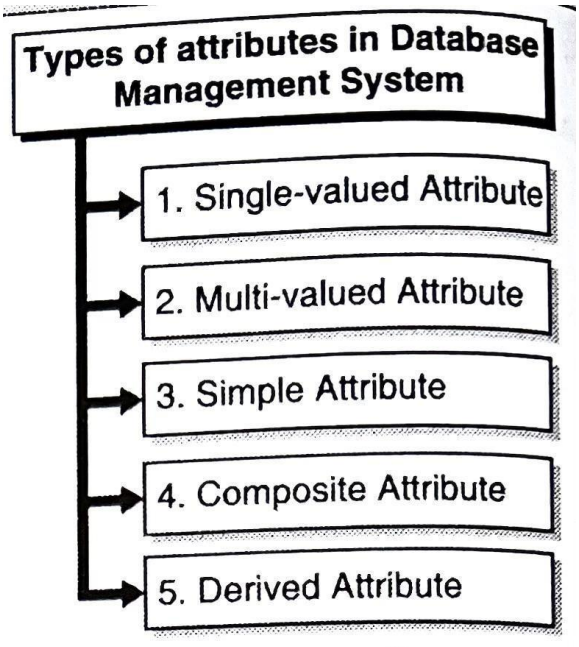
Attributes



- An attribute is a characteristic of an entity.
- Entities are represented by means of their attributes.
- All attributes have their own specific values.
- For example, an employee entity may have Employee _ID, emp_name, salary as attributes.
- In a database management system an attribute is a database component, such as field or column of a table.
- **Example**: The entity student has attributes like student_id, student_name.
  - In this every attribute has a value.
  - Here 101 is the value for the attribute student_id, Kunal is the value for attribute student_name.

Types of Attributes

- There are **five** different types of attributes in Database Management System



Types of Attributes

1. **Single-valued Attribute**

- A single-valued attribute is the attribute which can hold a single value for the single entity.
- ***Example:***In the entity student, **student_name** is the single-valued attribute since a student have a single value for name attribute.

2. **Multi-valued Attribute**

- A multi-valued attribute is the attribute which can hold multiple values for the single entity.
- ***Example:*** In the entity student, the attribute **student_ contactno** could be considered a multi-value attribute since a student could have multiple contact numbers.

Types of Attributes

3. **Simple Attribute**

- An attribute whose value cannot be further divided is known as simple attribute. That means it is atomic in nature.
- ***Example:*** In the entity student, the attribute **student_ age** cannot be divided. Therefore student_age is the simple attribute of student entity.

4. **Composite Attribute**

- The composite attributes are the attributes which can be further divided into sub parts. These sub parts represent the basic entities with their independent meaning.
- ***Example:*** In the entity student, **student_name** is the composite attribute, we can divide this attribute in three different sub parts: First_name, Middle_name and Last _name.

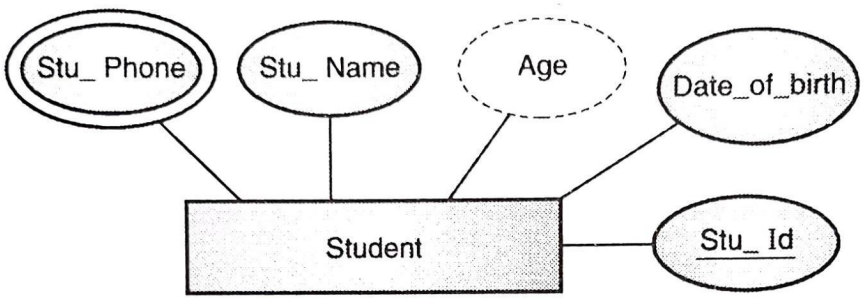
Types of Attributes

5. **Derived Attribute**

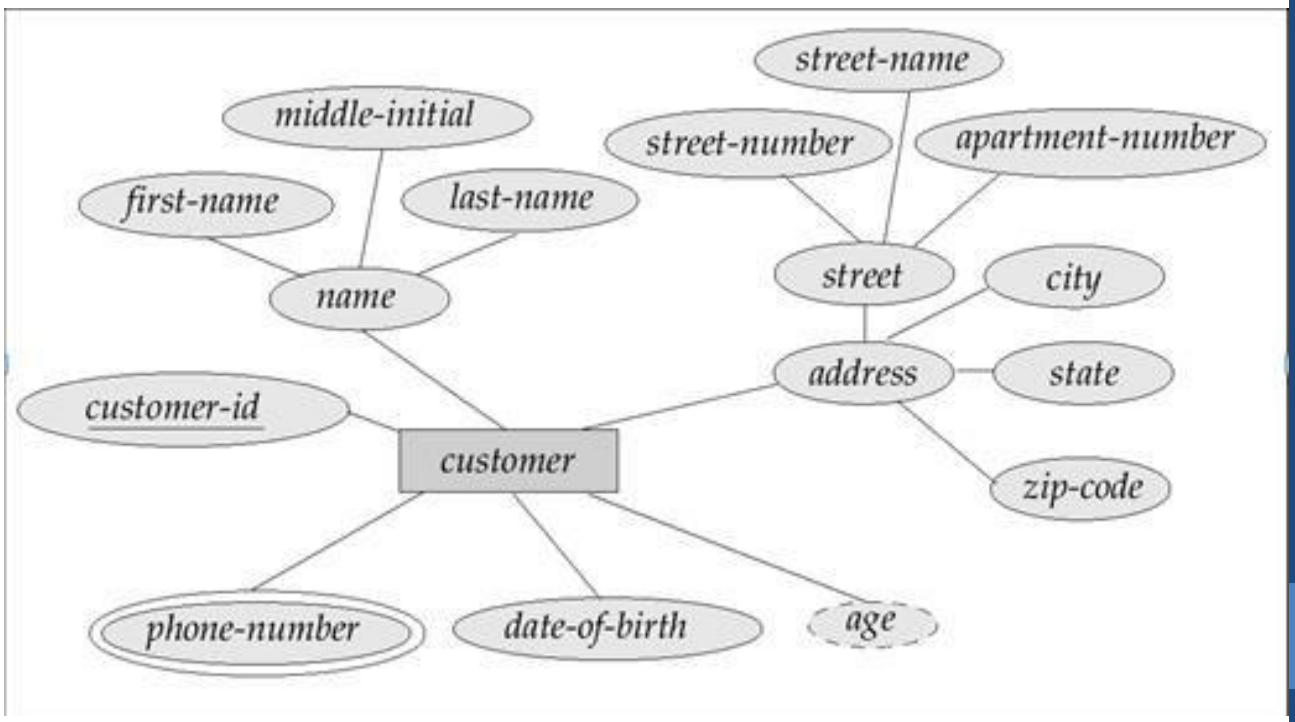
- The attribute which is not physically exist in database, but its value can be calculated from the other present attributes is known as derived attribute.
- ***Example***: In the entity student, we can calculate the average **age** of students. This average age is not physically present in the database but it can be derived from the attribute student_age.

Examples: ER Diagram

- **Give an example of E-R diagram with multi valued and derived attributes**

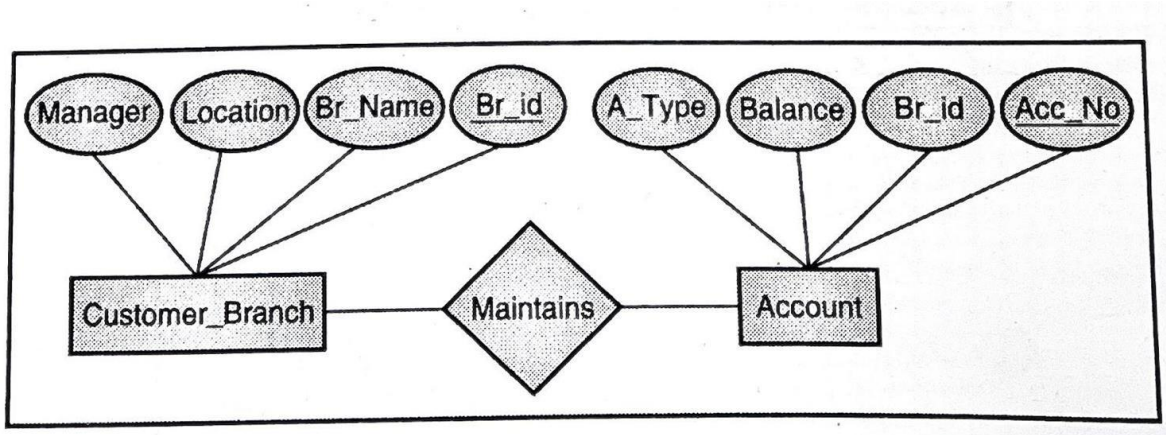


E-R Diagram With Composite, Multivalued, and Derived Attributes



Examples: ER Diagram

- **Draw an E-R diagram for customer branch and account relationship**
-



Relationship Attributes

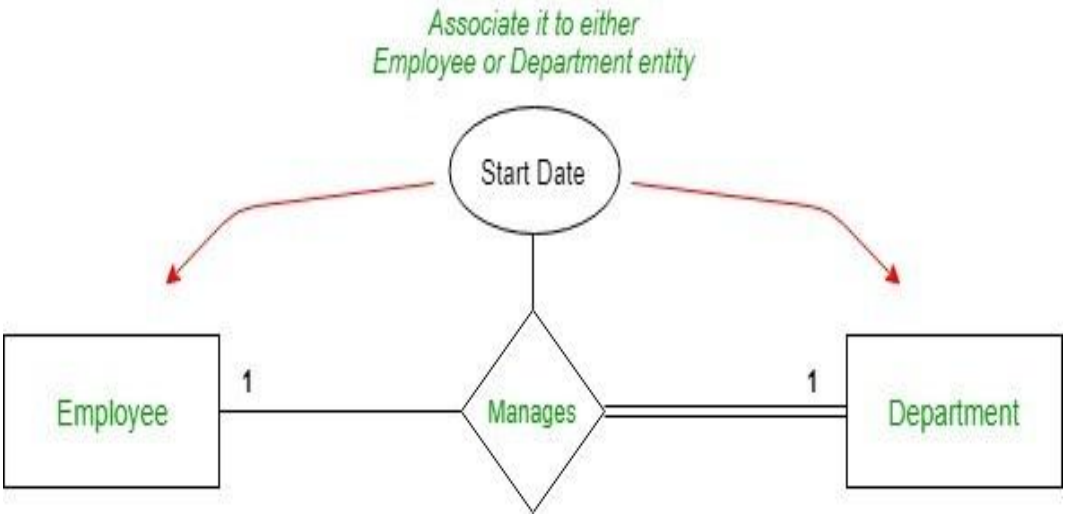
- Relationship set may have attributes which describe each relation with entities.
- The problem arises where to place this descriptive attribute; whether it is places with one of the entity or it will be placed with the separate relationship-set.
- Generally it is not recommended to give attributes to the relationships if not required because while converting the ER model into Relational model, things may get complex and we may require to create a separate table for representing the relationship.
- Let us see various cases and when we need to give attributes to the relationship with the help of examples:

Relationship Attributes

One to one relationship:

- In an organisation an employee manages a department and each department is managed by some employee.
- So, there is a total participation of employee and Department entity and there is *one to one* relationship between the given entities.
- Now, if we want to store the *Start_Date* from which the employee started managing the department then we may think that we can give the *Start_Date* attribute to the relationship *manages*.
- But, in this case we may avoid it by associating the *Start_Date* attribute to either *Employee* or *Department* entity.

Relationship Attributes

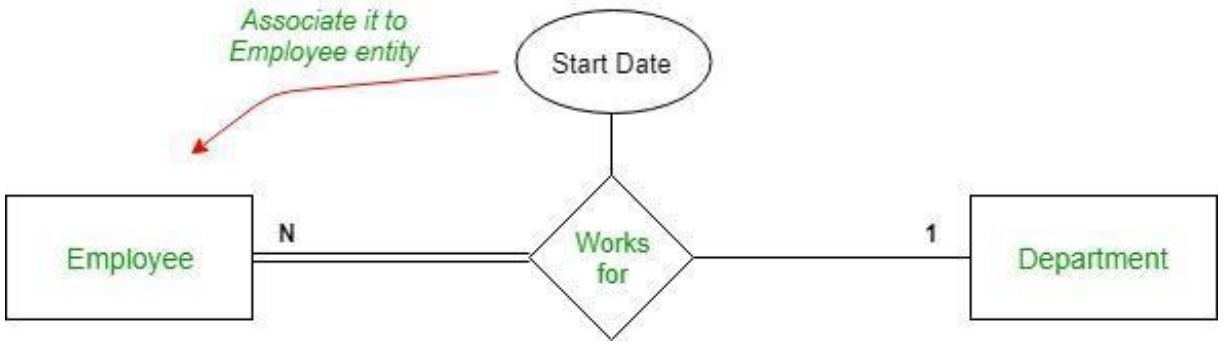


Relationship Attributes

One to many relationship:

- In an organisation many employees can work for a department but each employee can work for only a single department.
- So, there is a *one to many* relationship between the entities.
- Now if we want to store the *Start_Date* when employee started working for the department, then instead of assigning it to the relationship we should assign it to the *Employee* entity.
- Assigning it to the *employee* entity makes sense as each employee can work for only single department but on the other hand one department can have many employees working under it and hence, it wouldn't make sense if we assign Start_Date attribute to Department.

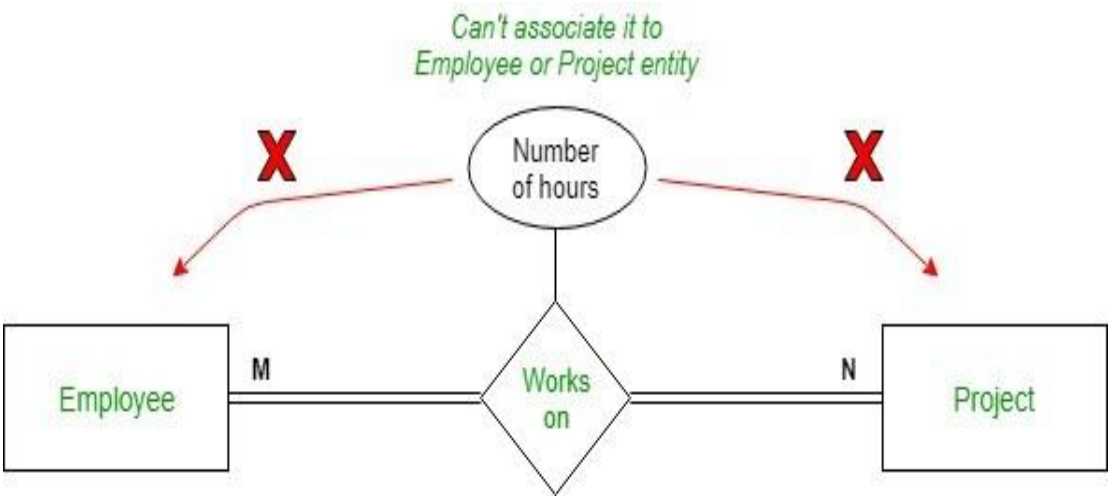
Relationship Attributes



Relationship Attributes

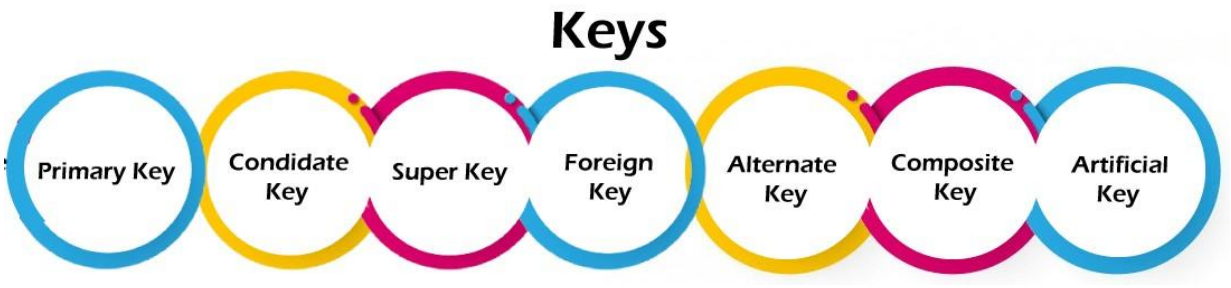
- **Many to many relationship:**
- In an organisation an employee can work on many projects simultaneously and each project can have many employees working on it.
- Hence, it's a *many to many* relationship.
- So here assigning the *Number_of_Working_hours* to the employee will not work as the question will be that it will store which project's working hours because a single employee can work on multiple projects.
- Similar the case with the *project* entity.
- Hence, we are forced to assign the *Number_of_Working_hours* attribute to the relationship.

Relationship Attributes

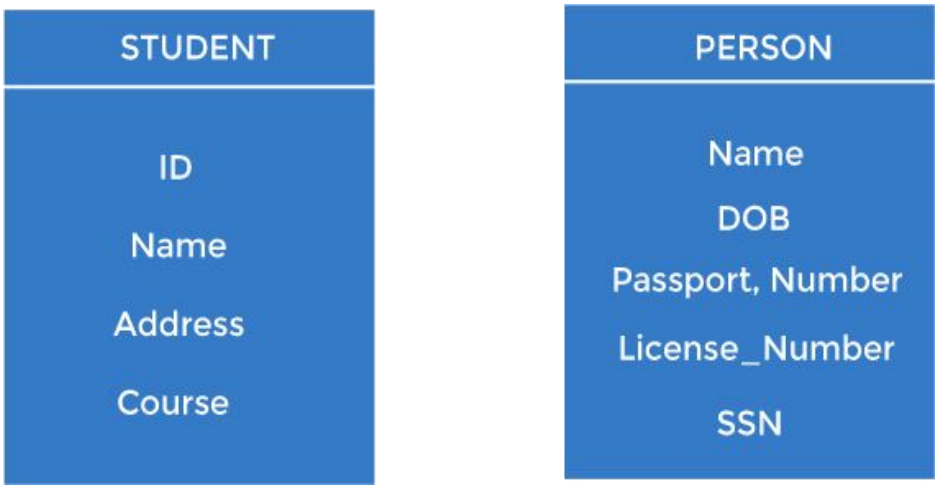


Keys

- A key refers to an attribute/a set of attributes that help us identify a row (or tuple) uniquely in a table (or relation).
- A key is also used when we want to establish relationships between the different columns and tables of a relational database.
- Keys are also used to identify relationships uniquely and differentiate these relationships from each other.
- There are **different types** of keys available in DBMS :



Keys

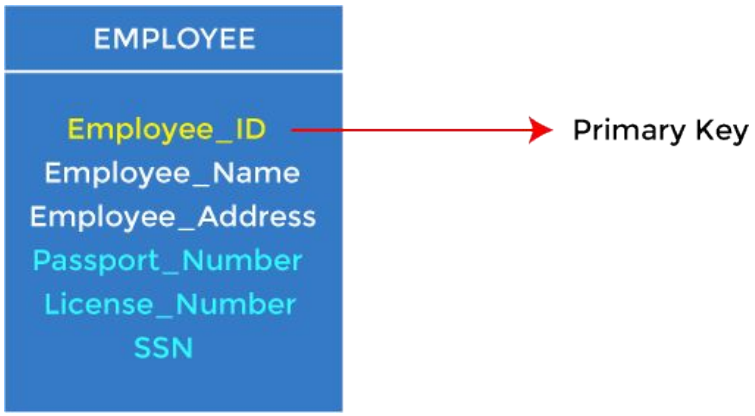


1. Primary Key

- **Definition** : Primary key *uniquely identify each entity in the entity set.*
- It must have **unique values and cannot hold null values.**
- Let R be a relationship set having entity sets E_1, E_2,..E_n.
- Consider primary key (E_i) denotes the set of attributes that forms the primary key for entity set E_i.
- The set of attributes associated with the relationship R is responsible for that relationship set
- ***Example***: In bank database, **account_number** should be primary key because this field cannot be null as well as no account_number can be repeated

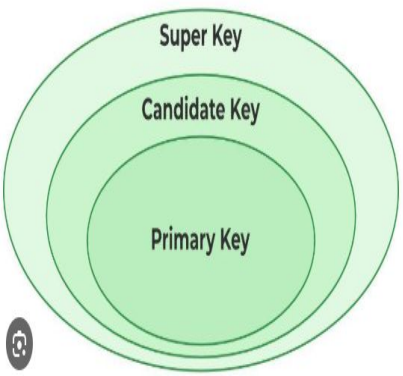
1. Primary Key

- In the EMPLOYEE table, **ID** can be the primary key since it is unique for each employee.
- In the EMPLOYEE table, we can even select **License_Number** and **Passport_Number** as primary keys since they are also unique.



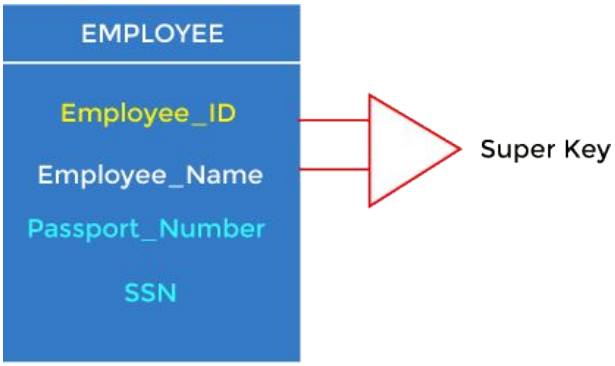
2. Super Key

- **Definition** : Super key is a *key which is set of one or more attributes for the purpose of uniquely identifying entities.*
- *It may include extra attributes that are not important for uniqueness but still uniquely identify the row*
- ***Example***
- In student database having attributes Student_reg_id, Student_roll_no, Student_name, Address, Contact_no.
- The Super keys are
  - {Student_reg_id}
  - (Student_roll_no}
  - {Student_reg_id, Student_roll_no}
  - Student_reg_id, Student_name}
  - {Student_reg_id, Student_roll_no, Student_name}
- A super key is a superset of a candidate key.



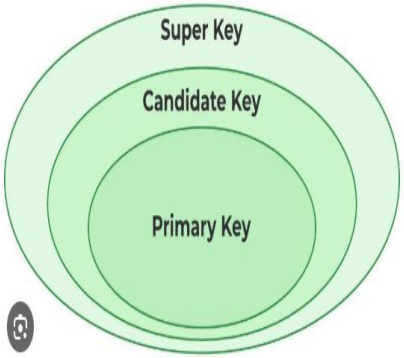
2. Super Key

The super key would be EMPLOYEE-ID (**EMPLOYEE_ID, EMPLOYEE-NAME**) etc.



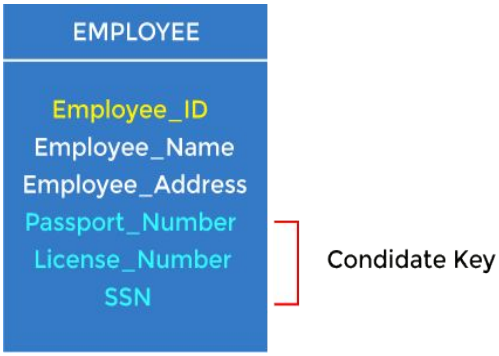
3.Candidate Key

- **Definition** : Candidate key *is formed by collection of attributes which hold unique values*.
- A super key without redundant values is known as **candidate key.**
- Candidate keys are selected from the set of super keys.
- Candidate key are also known as minimal super key having uniqueness property.
- The attribute which do not contain duplicate value, may be a candidate key.



3.Candidate Key

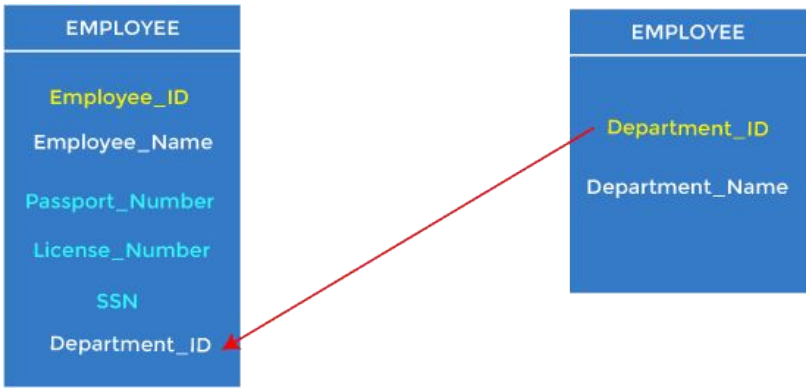
- ***Example***:
- In student database with attributes Student_reg_id, Student_roll_no, Student_name, Address, Contact_no
- The Candidate keys are:
  - **{Student_reg_id}**
  - **{Student_roll_no}**
  - **{Student_reg_id, Student_roll_no}**



4. Foreign Key

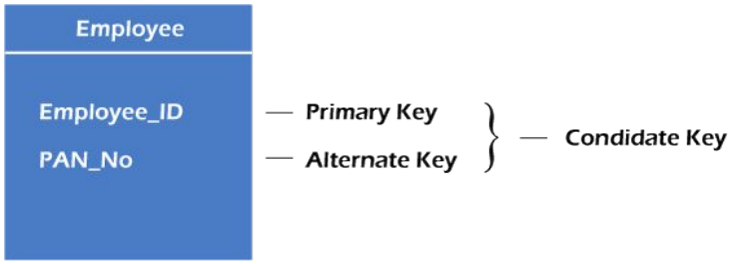
- Foreign keys are the *column of the table used to point to the primary key of another table.*
- Every employee works in a specific department in a company, and employee and department are two different entities.
- So we can't store the department's information in the employee table.
- That's why we link these two tables through the primary key of one table.
- We add the primary key of the DEPARTMENT table, **Department_Id,** as a new attribute in the EMPLOYEE table.
- In the EMPLOYEE table, Department_Id is the foreign key, and both the tables are related.

4. Foreign Key



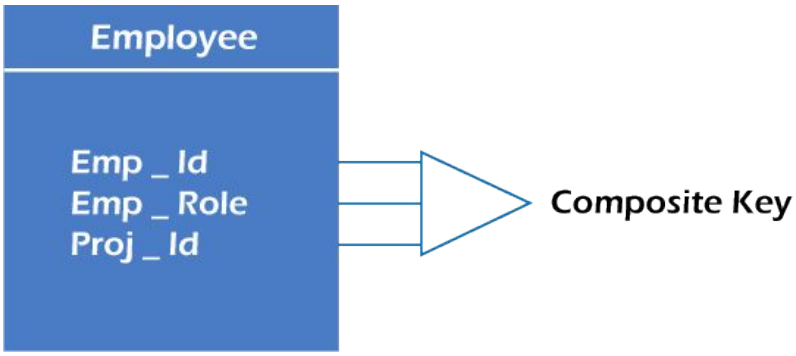
5. Alternate Keys or Secondary Keys

- From all candidate keys, only one key gets selected as primary key, remaining keys are known as **alternative or secondary keys**.
- The total number of the alternate keys is the total number of candidate keys minus the primary key.
- The alternate key may or may not exist. If there is only one candidate key in a relation, it does not have an alternate key.
- ***Example*** : In student table **Student_address, Contact_no, Date_Of_Birth** are the alternative keys.



6. Composite Key

- A key which consists of *more than one attributes to uniquely identify rows* in a table is called composite key.
- It i also known as compound key.
- ***Example***:



7. Artificial Keys

- The key created using *arbitrarily assigned data* are known as artificial keys.
- These keys are created when a primary key is large and complex and has no relationship with many other relations.
- The data values of the artificial keys are usually numbered in a serial order.
- **For example,** the primary key, which is composed of Emp_ID, Emp_role, and Proj_ID, is large in employee relations.

So it would be better to add a *new virtual attribute* to identify each tuple in the relation uniquely.

ER Diagram: Naming Conventions

- **Entities:**
  - Each entity should be a noun, singular or present tense(something about which we want to keep information)
  - First letter should be uppercase
  - Wherever necessary, entities name must be joined using underscore symbol
  - Entity names must be meaningful and must not conflict with other entity names
  - Ex: Account, Customer, Product, Employee, Department, Player etc.

ER Diagram: Naming Conventions

- **Attributes:**
  - Attribute name should be noun
  - Where necessary, attribute name can use underscore to join two words
  - Attribute name should be unique. Attributes of different entities can have same name
  - Ex: PID, EmpID, Course_No, Status, Reason, Date_of_Birth etc.

- **Relationships:**
  - Each relationship name should be a verb that fits sentence structure
  - Where necessary, relationship name can use underscore to join two words
  - Ex: Occupies, Married_to, Represents, Batting etc.

Displaying an Entity type

- In ER diagrams, an entity type is displayed in a rectangular box
- Attributes are displayed in ovals
  - Each attribute is connected to its entity type
  - Components of a composite attribute are connected to the oval representing the composite attribute
  - Each key attribute is **underlined**
  - Multivalued attributes displayed in **double ovals**
  - Derived attributes are represented with **dotted oval**

Ex: Entity Type CAR with two keys and a corresponding Entity Set

(a)

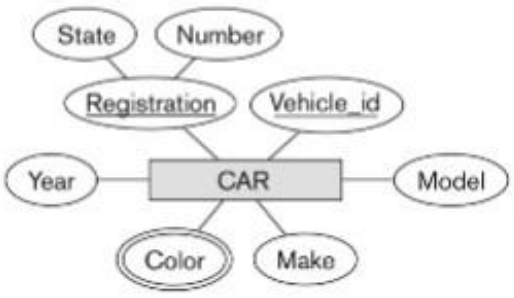


Figure 3.7
The CAR entity type with two key attributes, Registration and Vehicle_id. (a) ER diagram notation. (b) Entity set with three entities.

(b)

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

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

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

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

.
.
.

Example: Company Database

- We need to create a database schema design based on the following (simplified) **requirements** of the COMPANY Database:
  - The company is organized into DEPARTMENTs. Each department has a name, number and an employee who *manages* the department. We keep track of the start date of the department manager. A department may have several locations.
  - Each department *controls* a number of PROJECTs. Each project has a unique name, unique number and is located at a single location.

Example: Company Database

- We store each EMPLOYEE's social security number, address, salary, sex, and birthdate.
  - Each employee *works for* one department but may *work on* several projects.
  - We keep track of the number of hours per week that an employee currently works on each project.
  - We also keep track of the *direct supervisor* of each employee.

- Each employee may *have* a number of DEPENDENTs.
  - For each dependent, we keep track of their name, sex, birthdate, and relationship to the employee.

Initial Design of Entity Types for the COMPANY Database Schema

- Based on the requirements, we can identify four initial entity types in the COMPANY database:
  - **DEPARTMENT**
  - **PROJECT**
  - **EMPLOYEE**
  - **DEPENDENT**

- The initial attributes shown are derived from the requirements description

Initial Design of Entity Types: EMPLOYEE, DEPARTMENT, PROJECT, DEPENDENT

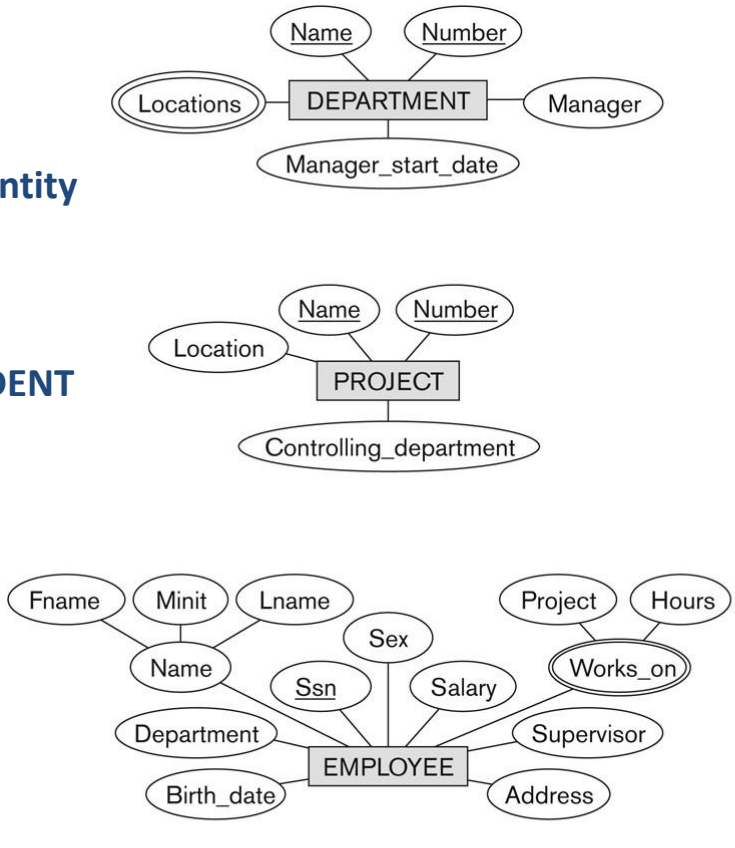


Figure 3.8
Preliminary design of entity types for the COMPANY database. Some of the shown attributes will be refined into relationships.

Refining the initial design by introducing relationships

- The initial design is typically not complete
- Some aspects in the requirements will be represented as **relationships**
- ER model has three main concepts:
  - **Entities** (and their entity types and entity sets)
  - **Attributes** (simple, composite, multivalued, derived)
  - **Relationships** (and their relationship types and relationship sets)

Relationships and Relationship Types

- A **relationship** relates two or more distinct entities with a specific meaning.
  - For example, **EMPLOYEE** John Smith ***works on*** the ProductX **PROJECT**, or **EMPLOYEE** Franklin Wong ***manages*** the Research **DEPARTMENT.**
- Relationships of the same type are grouped or typed into a **relationship type**.
  - For example, the **WORKS_ON** relationship type in which EMPLOYEEs and PROJECTs participate, or the **MANAGES** relationship type in which EMPLOYEEs and DEPARTMENTs participate.
- The degree of a relationship type is the number of participating entity types.
  - Both MANAGES and WORKS_ON are *binary* relationships.

Relationship instances of the WORKS_FOR N:1 relationship between EMPLOYEE and DEPARTMENT

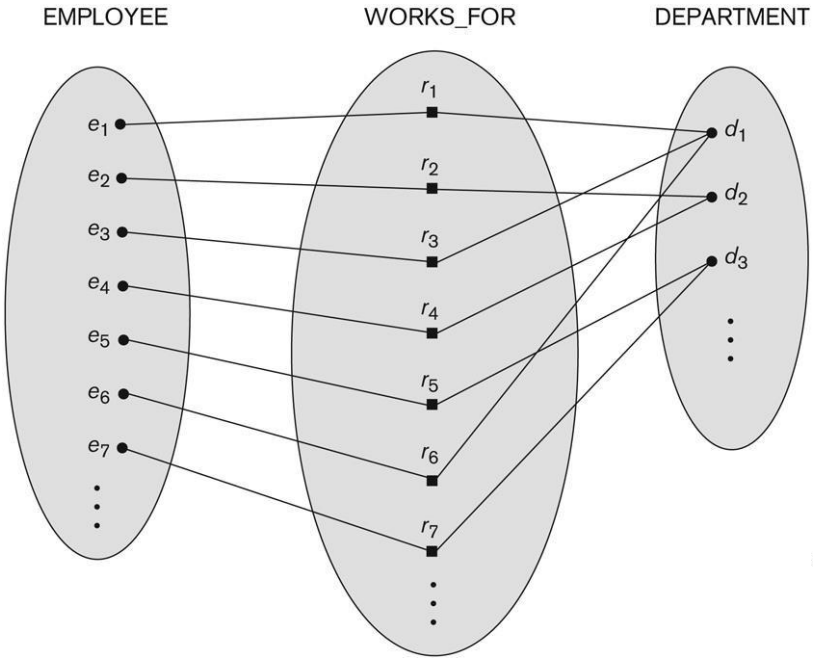


Figure 3.9
Some instances in the WORKS_FOR relationship set, which represents a relationship type WORKS_FOR between EMPLOYEE and DEPARTMENT.

Relationship instances of the M:N WORKS_ON relationship between EMPLOYEE and PROJECT

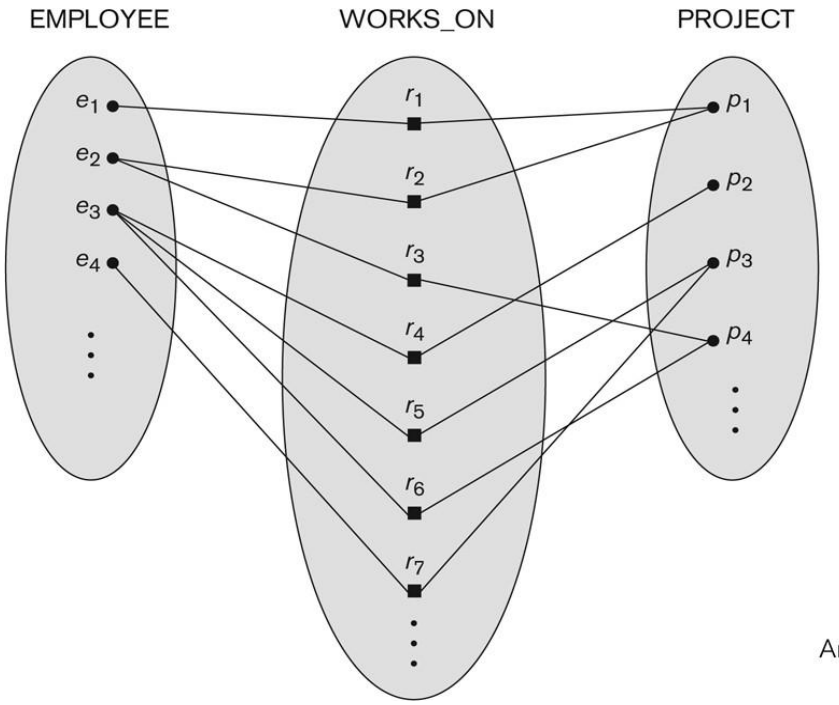


Figure 3.13
An M:N relationship, WORKS_ON.

Refining the COMPANY database schema by introducing relationships

- By examining the requirements, six relationship types are identified
- All are *binary* relationships(degree 2)
- Listed below with their participating entity types:
  - **WORKS_FOR** (between EMPLOYEE, DEPARTMENT)
  - **MANAGES** (also between EMPLOYEE, DEPARTMENT)
  - **CONTROLS** (between DEPARTMENT, PROJECT)
  - **WORKS_ON** (between EMPLOYEE, PROJECT)
  - **SUPERVISION** (between EMPLOYEE (as subordinate), EMPLOYEE (as supervisor))
  - **DEPENDENTS_OF** (between EMPLOYEE, DEPENDENT)

ER DIAGRAM – Relationship Types are: WORKS_FOR, MANAGES, WORKS_ON, CONTROLS, SUPERVISION, DEPENDENTS_OF

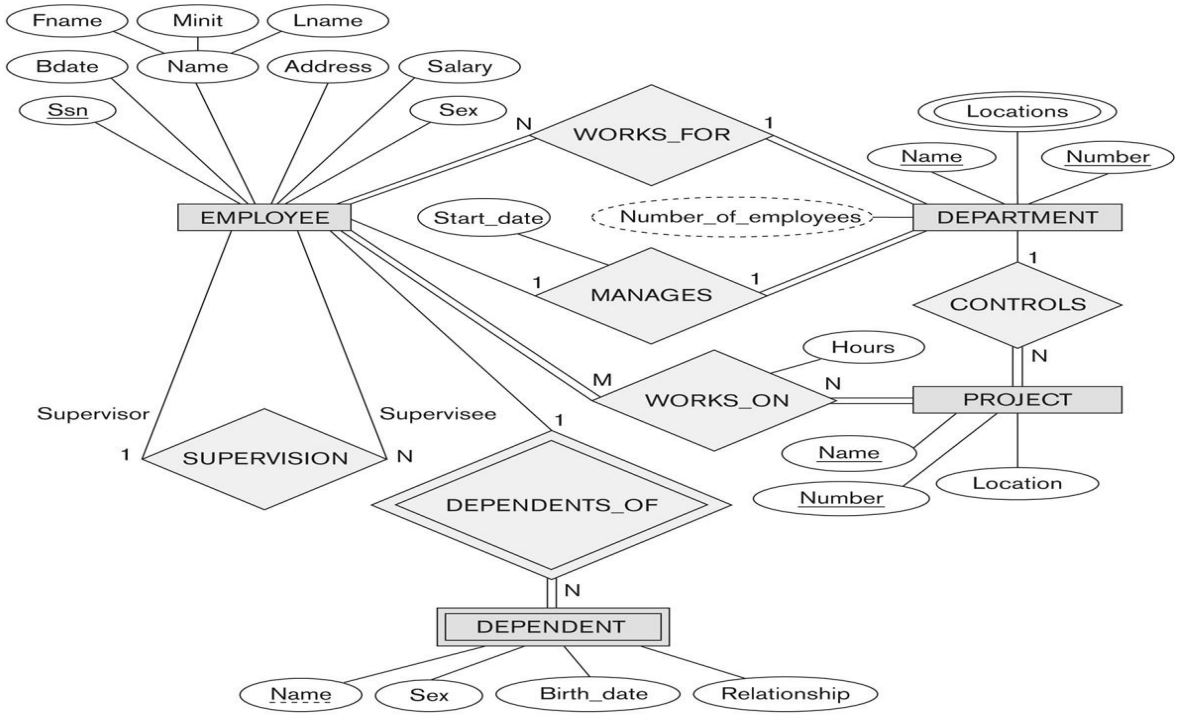
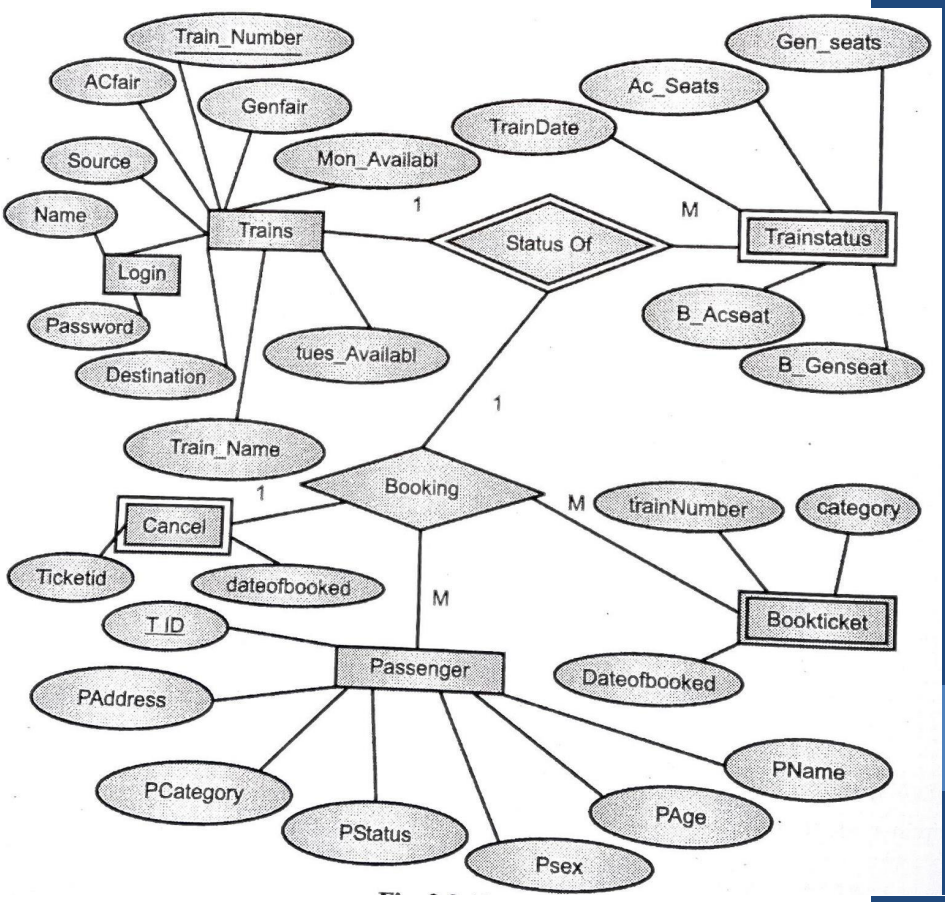


Figure 3.2
An ER schema diagram for the COMPANY database. The diagrammatic notation is introduced gradually throughout this chapter.

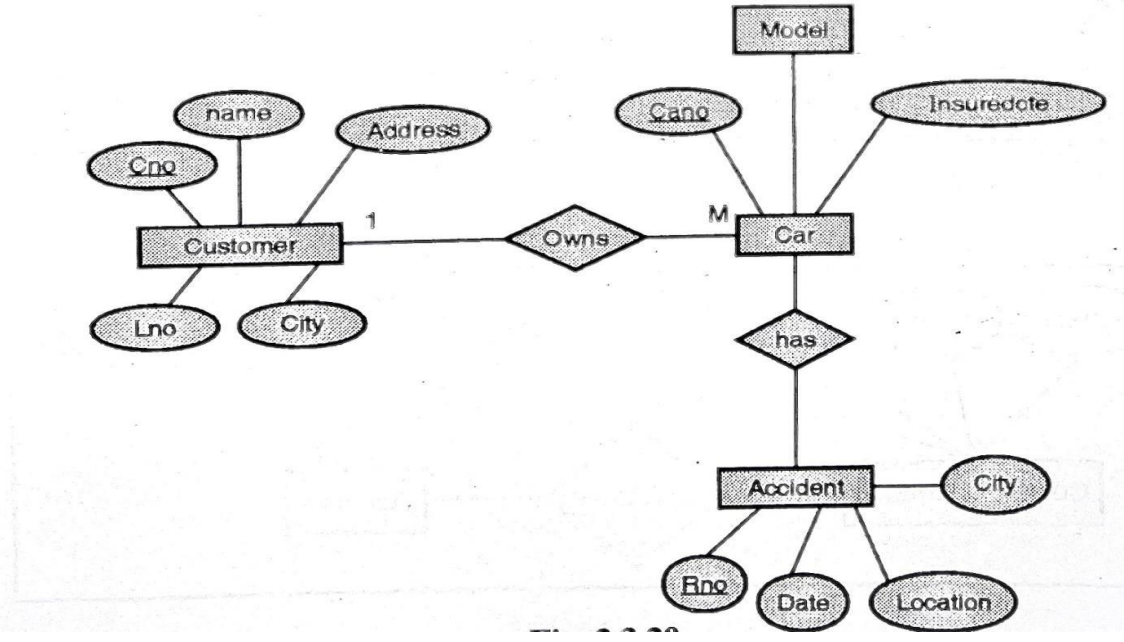
Examples: ER Diagram

- **Draw E-R diagram for online railway reservation system**



Examples: ER Diagram

- **Construct an E-R diagram for a car insurance company that has a set of customers each of whom owns one or more cars. Each car has associated with it zero to any number of recorded accidents.**



Examples: ER Diagram

Following information is maintained for online bookstore

(i) Books(ISBN, price, title,year)

(i) Author(name, address, url)

(il) Publisher(name,address,phone, url)

(iv)Customer(name, address, email, phone) (name is discriminating attribute)

(v)Shopping basket(basketID)

- Construct ER diagram with following constraints:
  - Each book should have author and publisher.
  - Book may have more than one author.
  - Each customer have a dedicated shopping basket.
  - Books can further be categorized as books, music, cassette , compact disks.

ER Model: Drawbacks

- **Loss of information content** : Some information be lost or hidden in ER model
- **Limited relationship representation** : ER model represents limited relationship as compared to another data models like relational model etc.
- **No representation of data manipulation**: It is difficult to show data manipulation in ER model.
- **Popular for high level design** : ER model is very popular for designing high level design

Unit 2.2

- Extended Entity-Relationship (EER) Model
  - Generalization
  - Specialization
  - Aggregation

Extended E-R Model (EER)

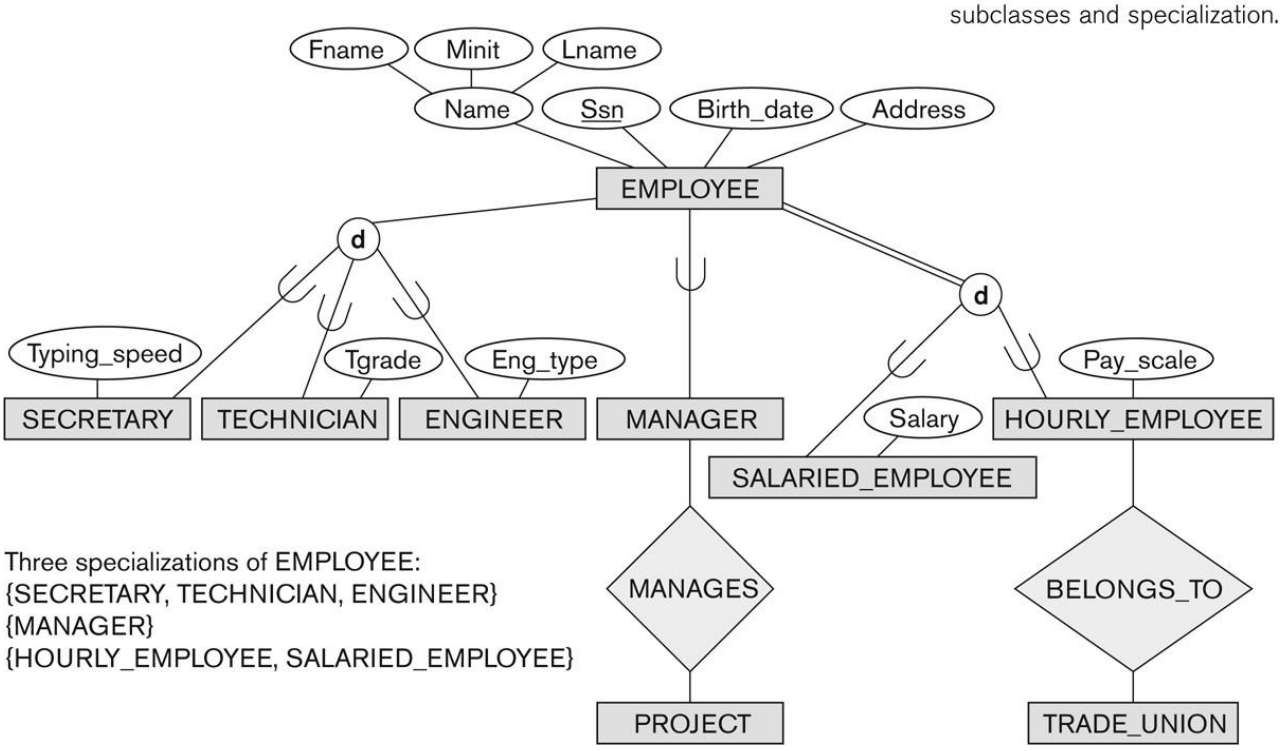
- EER creates a design more accurate to database schemas.
- It reflects the data properties and constraints more precisely.
- It includes all modeling concepts of the ER model.
- Diagrammatic technique helps for displaying the EER schema.
- It includes the concept of specialization and generalization.
- It is used to represent a collection of objects that is union of objects of different entity types.

Subclasses and Superclasses

- An entity type may have additional meaningful subgroupings of its entities
  - Example: **EMPLOYEE** may be further grouped into:
    - **SECRETARY, ENGINEER, TECHNICIAN, …**
      - Based on the EMPLOYEE's Job
    - **MANAGER**
      - EMPLOYEEs who are managers
    - **SALARIED_EMPLOYEE, HOURLY_EMPLOYEE**
      - Based on the EMPLOYEE's method of pay
- EER diagrams extend ER diagrams to represent these additional subgroupings, called *subclasses* or *subtypes*

Subclasses and Superclasses(contd..)

Figure 4.1
EER diagram notation to represent subclasses and specialization.



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

Subclasses and Superclasses (contd..)

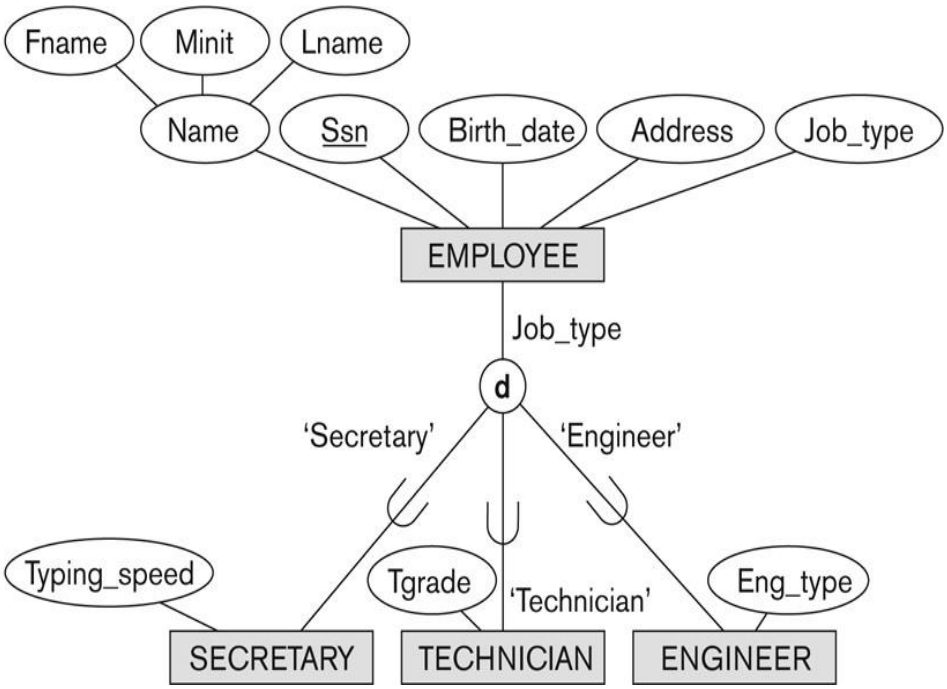
- Each of these subgroupings is a subset of EMPLOYEE entities
- Each is called a subclass of EMPLOYEE
- EMPLOYEE is the superclass for each of these subclasses
- These are called superclass/subclass relationships:
  - EMPLOYEE/SECRETARY
  - EMPLOYEE/TECHNICIAN
  - EMPLOYEE/MANAGER
- These are also called **IS-A relationships**
  - SECRETARY IS-A EMPLOYEE, TECHNICIAN IS-A EMPLOYEE, ….

Subclasses and Superclasses (contd..)

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

Attribute Inheritance in Superclass / Subclass Relationships

Figure 4.4
EER diagram notation for an attribute-defined specialization on Job_type.

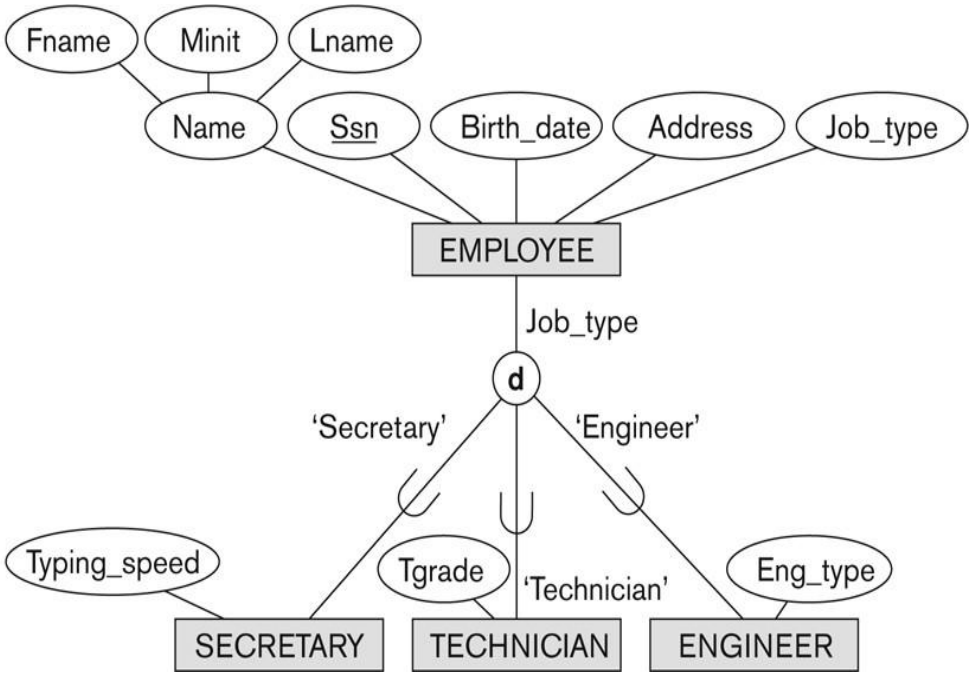


Specialization

- Specialization 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

Representing Specialization in EER Diagrams

Figure 4.4
EER diagram notation for an attribute-defined specialization on Job_type.

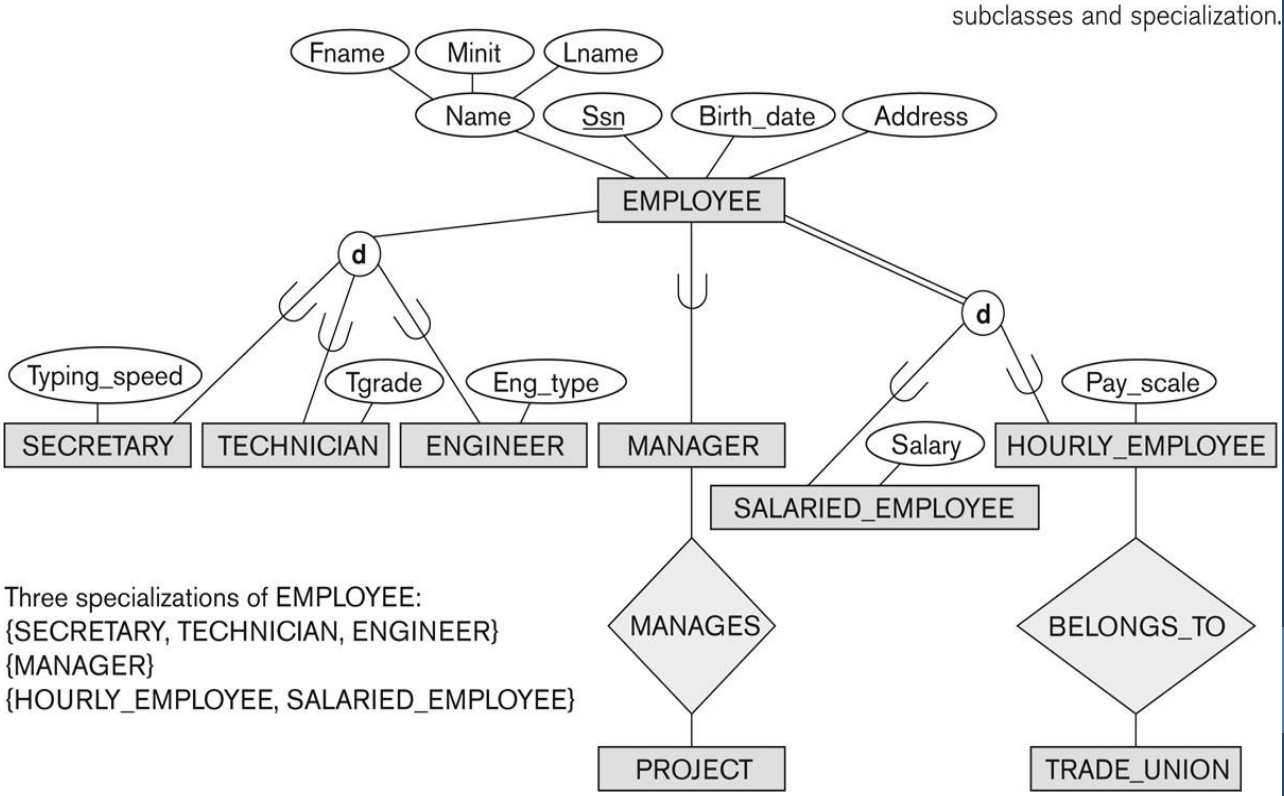


Specialization (contd..)

- Example: Another specialization of EMPLOYEE based on *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* or *local* attributes.**
    - For example, the attribute Typing Speed of SECRETARY
  - The subclass can also participate in specific relationship types.
    - For example, a relationship BELONGS_TO of HOURLY_EMPLOYEE

Specialization (contd..)

Figure 4.1
EER diagram notation to represent subclasses and specialization.



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

Generalization

- Generalization is the reverse of the specialization process
- Several classes with common features are generalized into a superclass;
  - original classes become its subclasses
- 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 (contd..)

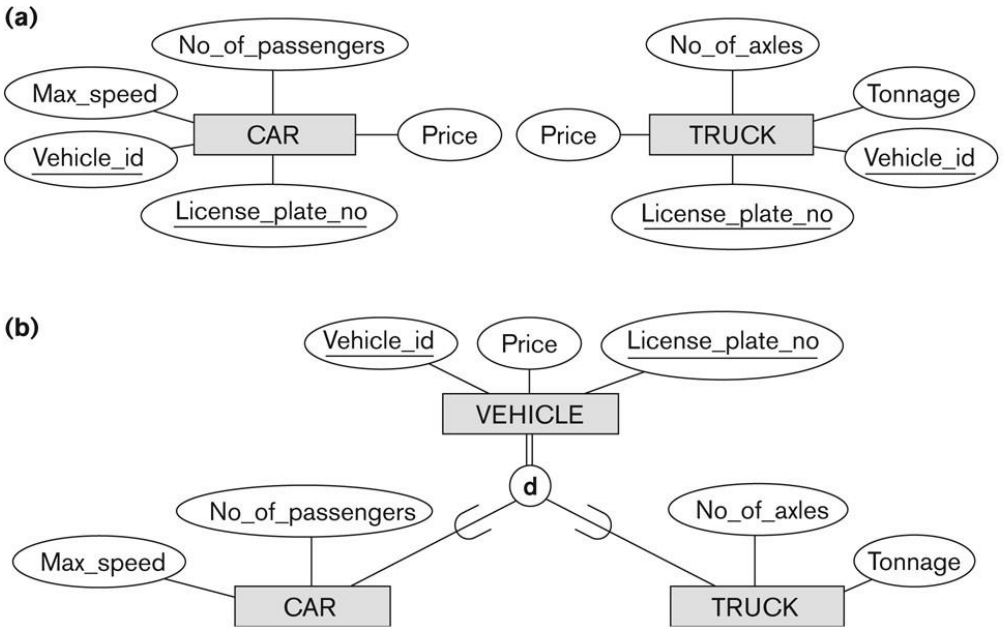


Figure 4.3
Generalization. (a) Two entity types, CAR and TRUCK.
(b) Generalizing CAR and TRUCK into the superclass VEHICLE.

Generalization and Specialization

- Diagrammatic notation are 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

Generalization and Specialization (contd..)

- Data Modeling with Specialization and Generalization
  - A superclass or subclass represents a collection (or set or grouping) of entities
  - It also represents a particular *type of entity*
  - Shown in rectangles in EER diagrams (as are entity types)
  - We can call all entity types (and their corresponding collections) ***classes***, whether they are entity types, superclasses, or subclasses

Constraints on Specialization and Generalization

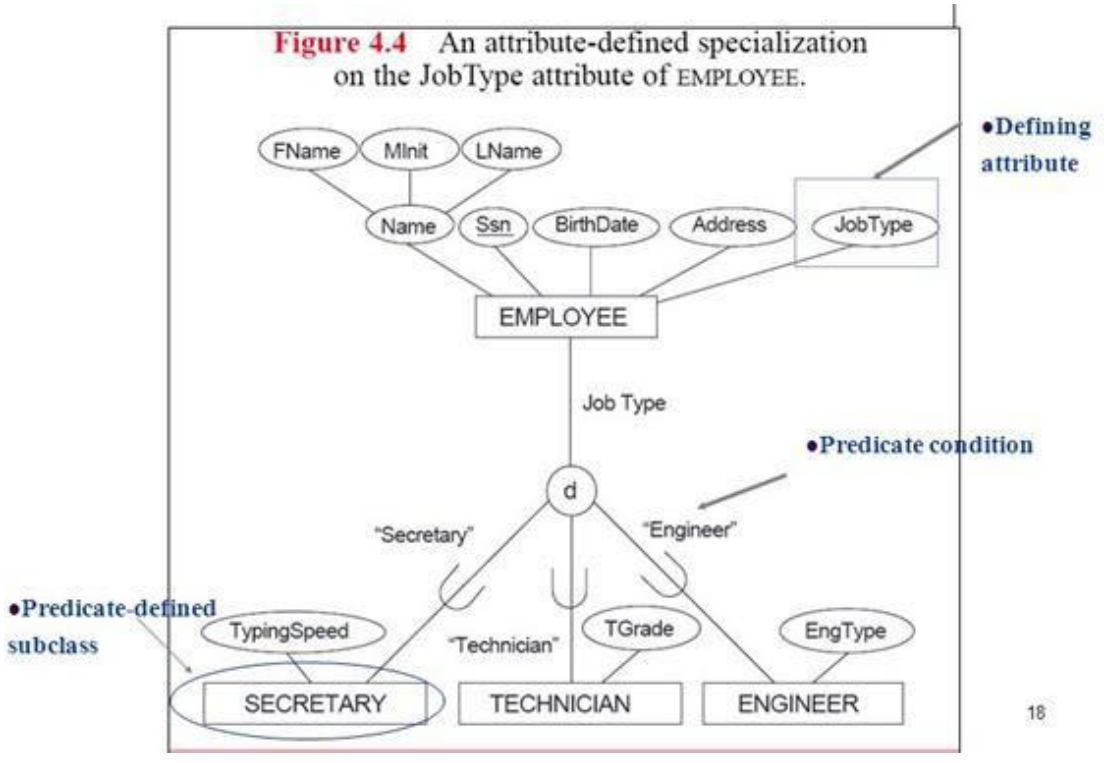
- If we can determine exactly those entities that will become members of subclass by a condition, the subclasses are called **predicate-defined (or condition-defined)** subclasses
  - Condition is a constraint that determines subclass members
  - Display a predicate-defined subclass by writing the predicate condition next to the line attaching the subclass to its superclass

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

Constraints on Specialization and Generalization (contd..)

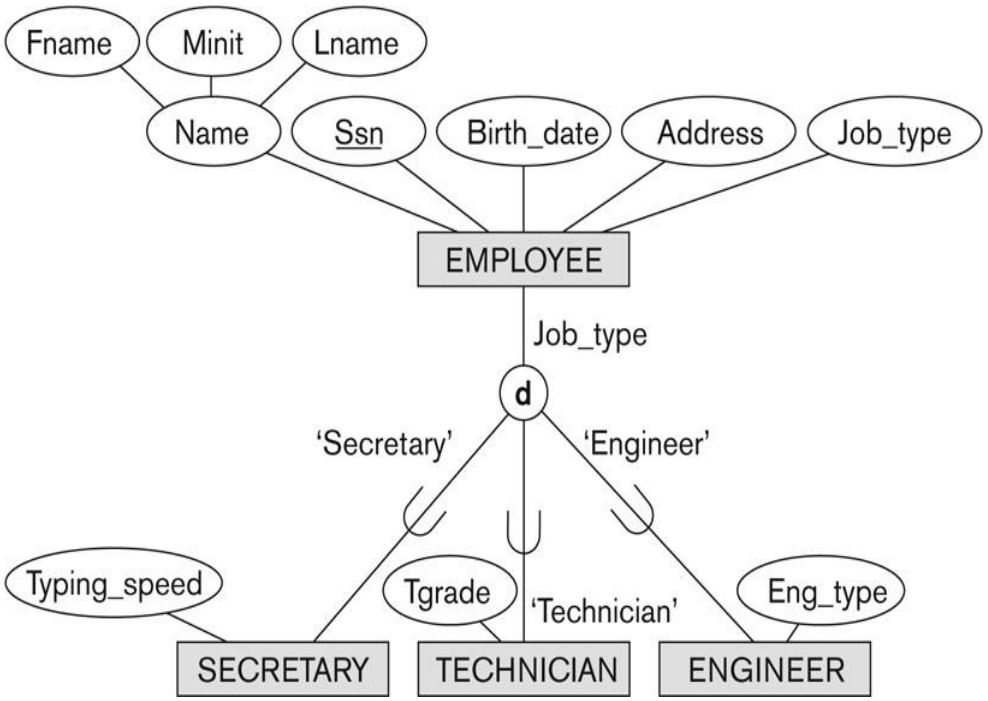
- If no condition determines membership, the subclass is called **user-defined**
  - Membership in a subclass is determined by the database users by applying an operation to add an entity to the subclass
  - Membership in the subclass is specified individually for each entity in the superclass by the user

Constraints on Specialization and Generalization (contd..)



Displaying an attribute-defined specialization in EER diagrams

Figure 4.4
EER diagram notation for an attribute-defined specialization on Job_type.



Constraints on Specialization and Generalization (contd..)

- Two basic constraints can apply to a specialization / generalization:
  - Disjointness Constraint
  - Completeness Constraint

Constraints on Specialization and Generalization (contd..)

- **Disjointness Constraint:**
  - 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
  - Specified by ***d*** in EER diagram
  - If not disjoint, specialization is ***overlapping***:
    - that is the same entity may be a member of more than one subclass of the specialization
  - Specified by ***o*** in EER diagram

Constraints on Specialization and Generalization (contd..)

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

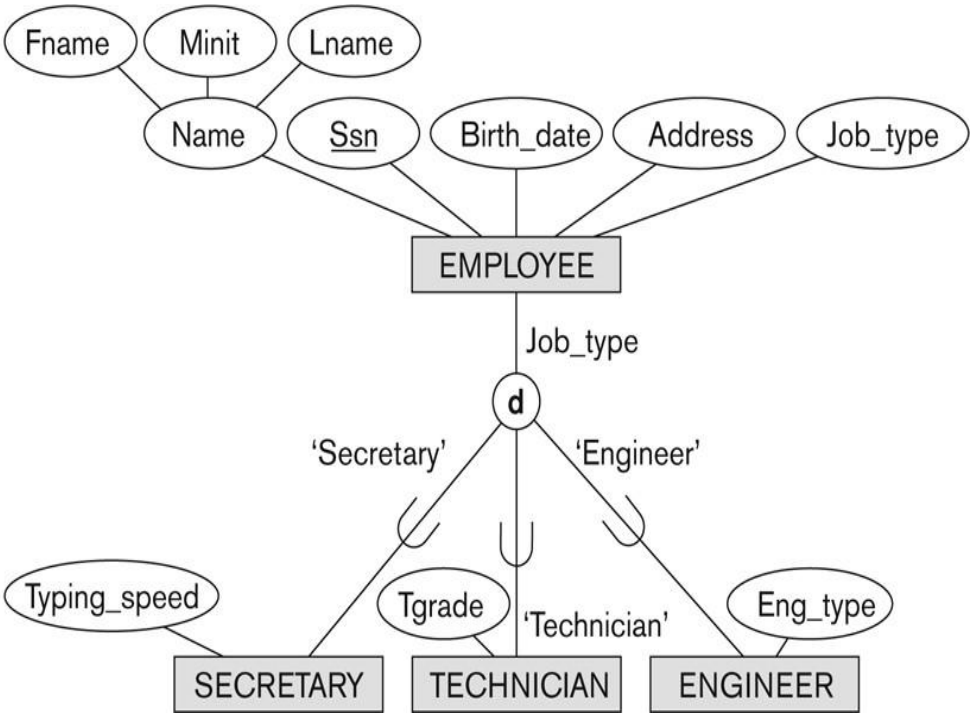
Constraints on Specialization and Generalization (contd..)

- 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 disjoint partial Specialization

Figure 4.4
EER diagram notation for an attribute-defined specialization on Job_type.



Example of overlapping total Specialization

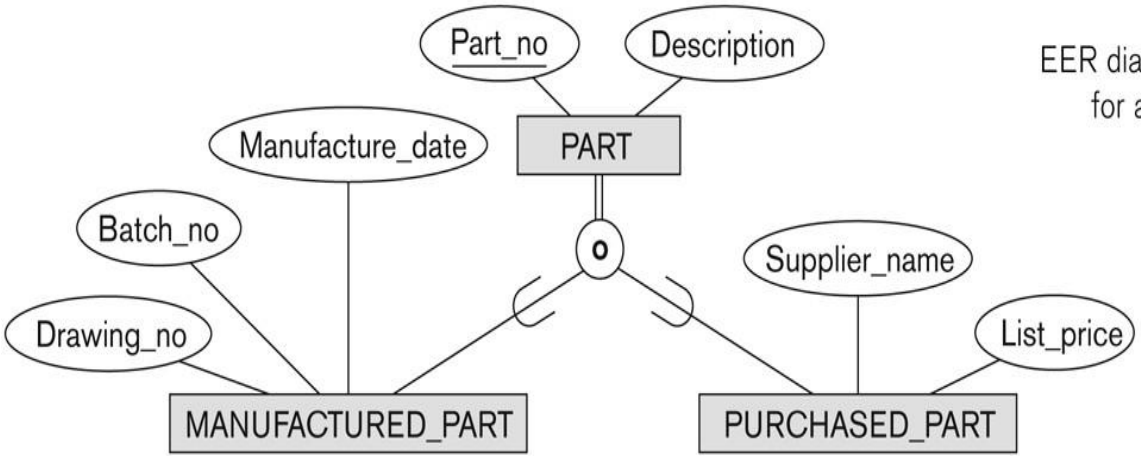
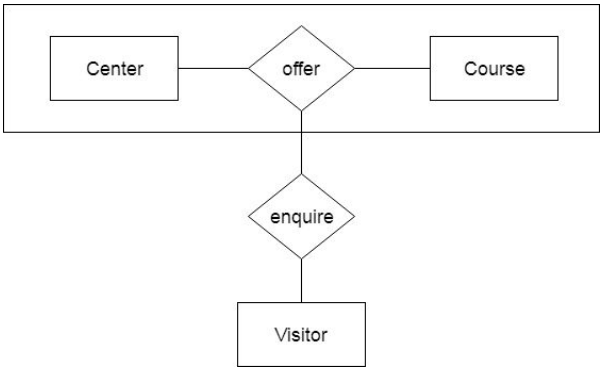


Figure 4.5
EER diagram notation for an overlapping (nondisjoint) specialization.

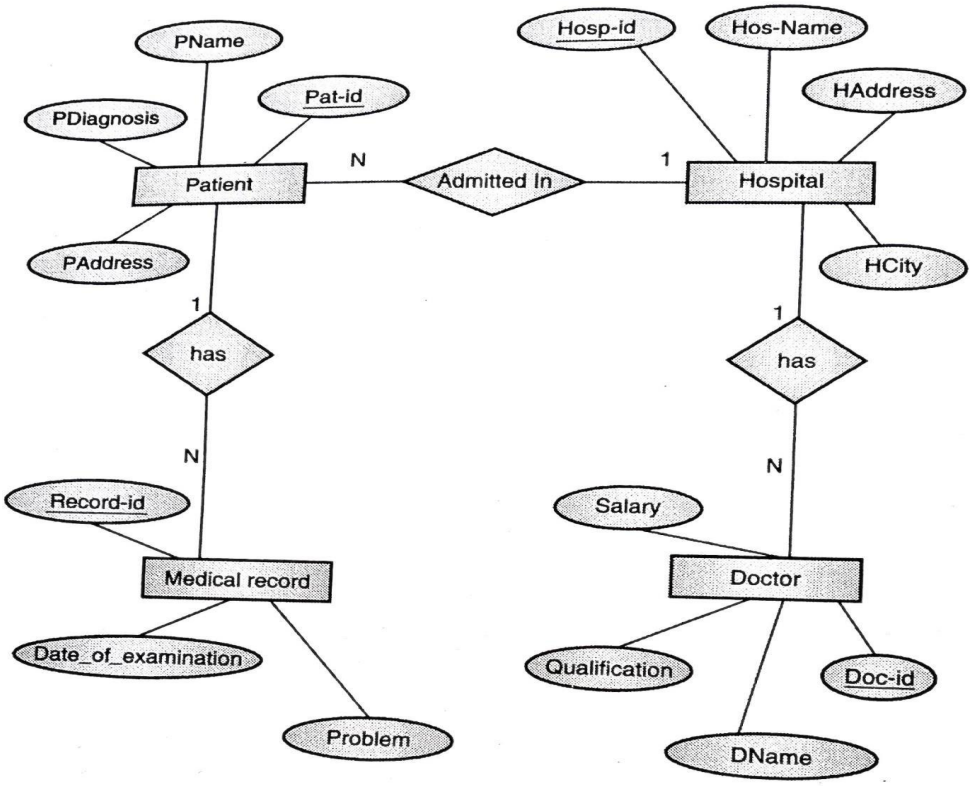
Aggregation

- In aggregation, the relation between two entities is treated as a single entity.
- In aggregation, relationship with its corresponding entities is aggregated into a higher level entity.
- **For example:** Center entity offers the Course entity act as a single entity in the relationship which is in a relationship with another entity visitor. In the real world, if a visitor visits a coaching center then he will never enquiry about the Course only or just about the Center instead he will ask the enquiry about both.



Examples: ER Diagram

- **Draw an E-R diagram of hospital management system.**



Exercise

- For library management system following information is maintained
- Books(Accession_no, Title, Author, Price, Booktype, Publisher)
- Borrower(Membership_no, Name, Address, Category, max_no_of_books_issued, Accession_no)
- Draw ER diagram for above taking into consideration following constraints and by making use of atleast one EER feature:
    1. A book may have more than one author
    2. There maybe more than one copy of book
    3. Borrower can be staff or a student. Depending on this category max number of books that can be issued will vary(student can ask max 3 books and staff can ask 10 books)

Practice Questions

1. What are the different types of attributes in ER diagram?
2. Define entity type, entity set, weak entity, regular entity, total participation, partial participation.
3. Define primary key, foreign key, candidate key, super key with example
4. Explain with example specialization and Generalization.
5. Explain overlapping, disjoint.
6. List and explain three main components of ER Diagram
7. Difference between strong and weak entity
8. Draw E-R diagram for Library management system. Students and faculties are members who borrow and return the books. Books have title, author, publication, price and number of books.
9. Define with example primary key, foreign key, disjoint, overlapping, total participation.
10. List and explain different types of keys available in DBMS
11. Construct an E-R diagram and create relational schema for a hospital with a set of patients and a set of medical doctors. Associated with each patient a log of the various test and examinations conducted.

Practice Questions

1. What are the different types of Cardinality Constraints?
2. Explain features of EER Diagram
3. Explain Aggregation with example.
4. Design a database for an airline. The database must keep track of customers and their reservations, flights and their status, seat assignments on individual flights, and the schedule and routing of future flights. Your design should include an E-R diagram, a set of relational schemas, and a list of constraints, including primary-key and foreign-key constraints
5. Draw E-R diagram and create Relational schema. Classroom has no. of benches. Each bench has its unique no., width, height, length & capacity. It is made up with different material. Bench must be located in the classrooms. Classroom may or may not have benches. When the bench placed in the particular CR is recorded.
6. Compare a weak and a strong entity set with some examples and diagram.
7. Explain the distinction between disjoint and overlapping constraints.
8. Explain the distinction between total and partial constraints.

Practice Questions

1. Illustrate in detail Specialization with Disjoint, overlapping constraints with example.
    a. Draw ER diagram for banking enterprise.ER diagram should include
    b. Different types of attributes
    c. Relations
    d. Cardinality Constraint
    e. Participation Constraint
    f. Specialization/ Generalization/ Aggregation
2. What are the basic constructs of the ER model ?Explain it.
3. Distinguish between simple and composite attributes
4. Distinguish between strong entity and weak entity
5. What is aggregation? Explain with an example?
6. Write a short note on extended features of ERD.
7. What is mean by key? Explain its types.
8. Explain shared subclasses with examples.

Thank You!!