Chapter Three

# Database Design

Database design is the process of coming up with different kinds of specification for the data to be stored in the database. The database design part is one of the middle phases we have in information systems development where the system uses a database approach. Design is the part on which we would be engaged to describe how the data should be perceived at different levels and finally how it is going to be stored in a computer system.

## Information System with Database application consists of several tasks which include:

- *Planning of Information systems Design*
- *Requirements Analysis,*
- ***Design (Conceptual, Logical and Physical Design)***
- *Implementation*
- *Testing and deployment*
- *Operation and Support*

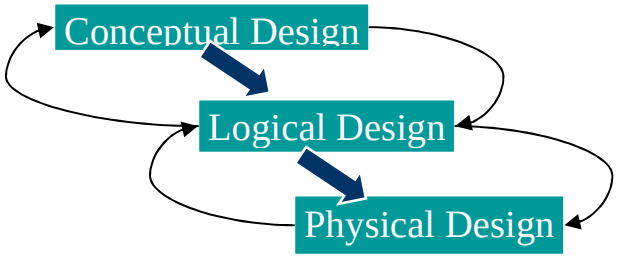
From these different phases, the prime interest of a database system will be the Design part which is again sub divided into other three sub-phases. These sub-phases are:

1. ***Conceptual Design***
2. ***Logical Design, and***
3. ***Physical Design***

- In general, one has to go back and forth between these tasks to refine a database design, and decisions in one task can influence the choices in another task.
- In developing a good design, one should answer such questions as:
    - What are the relevant Entities for the Organization
    - What are the important features of each Entity

- What are the important Relationships
- What are the important queries from the user
- What are the other requirements of the Organization and the Users

# The Three levels of Database Design




## Conceptual Database Design

- Conceptual design is the process of constructing a model of the information used in an enterprise, ***independent of any physical considerations.***
  - ❖ It is the source of information for the logical design phase.
  - ❖ Mostly uses an Entity Relationship Model to describe the data at this level.
- After the completion of Conceptual Design one has to go for refinement of the schema, which is verification of Entities, Attributes, and Relationships

## Logical Database Design

- Logical design is the process of constructing a model of the information used in an enterprise based on a specific data model (e.g. relational, hierarchical or network or object), but independent of a particular DBMS and other physical considerations.
  - ❖ Normalization process
    - Collection of Rules to be maintained
    - Discover new entities in the process
    - Revise attributes based on the rules and the discovered Entities

## Physical Database Design

- Physical design is the process of producing a description of the implementation of the database on secondary storage. -- defines specific storage or access methods used by database
  - Describes the storage structures and access methods used to achieve efficient access to the data.

- Tailored to a specific DBMS system -- Characteristics are function of DBMS and operating systems
- Includes estimate of storage space

# Conceptual Database Design

- Conceptual design revolves around discovering and analyzing organizational and user data requirements
- The important activities are to identify
  - Entities
  - Attributes
  - Relationships
  - Constraints
- And based on these components develop the ER model using
  - ER diagrams

# The Entity Relationship (E-R) Model

- Entity-Relationship modeling is used to represent conceptual view of the database
- The main components of ER Modeling are:
  - **Entities**
    - Corresponds to entire table, not row
    - Represented by Rectangle
  - **Attributes**
    - Represents the property used to describe an entity or a relationship
    - Represented by Oval
  - **Relationships**
    - Represents the association that exist between entities
    - Represented by Diamond
  - **Constraints**
    - Represent the constraint in the data
      - Cardinality and Participation Constraints

**Before working on the conceptual design of the database, one has to know and answer the following basic questions.**

- What are the ***entities*** and ***relationships*** in the enterprise?
- What information about these entities and relationships should we store in the database?
- What are the ***integrity constraints*** that hold? Constraints on each data with respect to update, retrieval and store.

- Represent this information pictorially in ***ER diagrams,*** then map ER diagram into a relational schema.

# Developing an E-R Diagram

- Designing conceptual model for the database is not a one linear process but an iterative activity where the design is refined again and again.
- To identify the entities, attributes, relationships, and constraints on the data, there are different set of methods used during the analysis phase. These include information gathered by…
  - Interviewing end users individually and in a group
  - Questionnaire survey
  - Direct observation
  - Examining different documents
- Analysis of requirements gathered
  - Nouns --→ prospective entities
  - Adjectives--→prospective attributes
  - Verbs/verb phrases-→prospective relationships

- The basic E-R model is graphically depicted and presented for review.
- The process is repeated until the end users and designers agree that the E-R diagram is a fair representation of the organization's activities and functions.
- Checking for Redundant Relationships in the ER Diagram. Relationships between entities indicate access from one entity to another - it is therefore possible to access one entity occurrence from another entity occurrence even if there are other entities and relationships that separate them - this is often referred to as *Navigation*' of the ER diagram
- The last phase in ER modeling is validating an ER Model against requirement of the user.

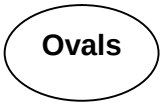
# Graphical Representations in ER Diagramming

- Entity is represented by a **RECTANGLE** containing the name of the entity.

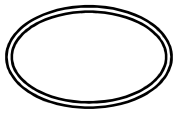


- Connected entities are called relationship participants

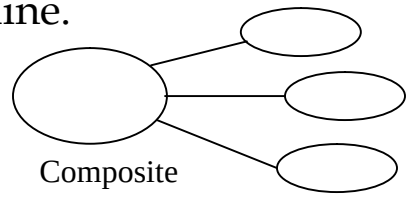
- Attributes are represented by **OVALS** and are connected to the entity by a line.




Attribute



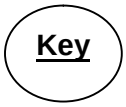
Multi-valued Attribute



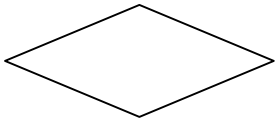
Composite Attribute

- A derived attribute is indicated by a **DOTTED LINE**. (……..)

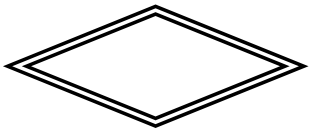
- **PRIMARY KEYS** are underlined.




- Relationships are represented by **DIAMOND** shaped symbols
  - Weak Relationship is a relationship between Weak and Strong Entities
  - Strong Relationship is a relationship between two strong Entities



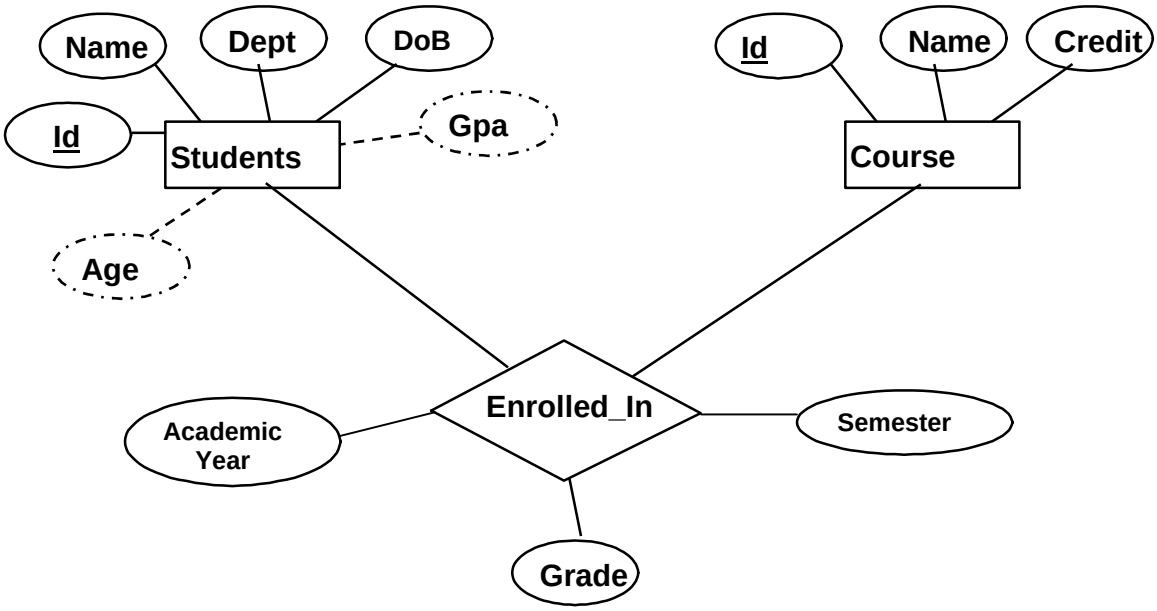
Strong Relationship



Weak Relationship

**Example 1:** Build an ER Diagram for the following information:

- A student record management system will have the following two basic data object categories with their own features or properties: Students will have an Id, Name, Dept, Age, GPA and Course will have an Id, Name, Credit Hours
    - Whenever a student enroll in a course in a specific Academic Year and Semester, the Student will have a grade for the course




**Example 2:** Build an ER Diagram for the following information:

- A Personnel record management system will have the following two basic data object categories with their own features or properties: Employee will have an Id, Name, DoB, Age, Tel and Department will have an Id, Name, Location
    - Whenever an Employee is assigned in one Department, the duration of his stay in the respective department should be registered.

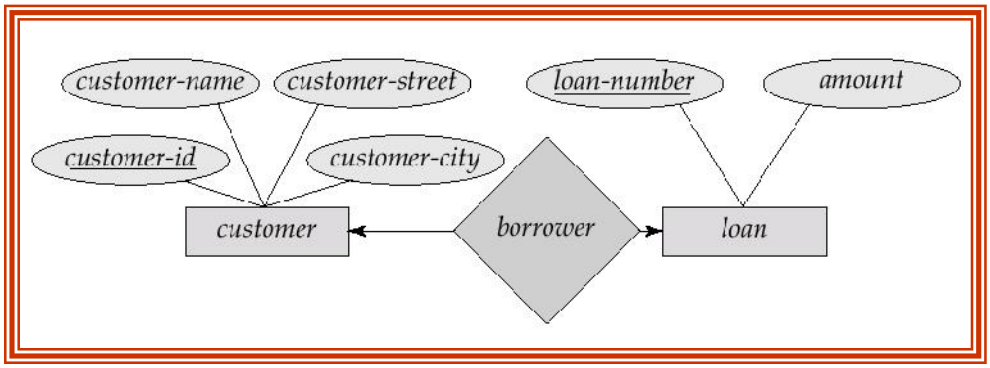
# Structural Constraints on Relationship

1. **Constraints on Relationship / Multiplicity/ Cardinality Constraints**

- Multiplicity constraint is the number or range of possible occurrence of an entity type/relation that may relate to a single occurrence/tuple of an entity type/relation through a particular relationship.
- Mostly used to insure appropriate enterprise constraints.

## One-to-one relationship:

- A customer is associated with at most one loan via the relationship *borrower*
- A loan is associated with at most one customer via *borrower*

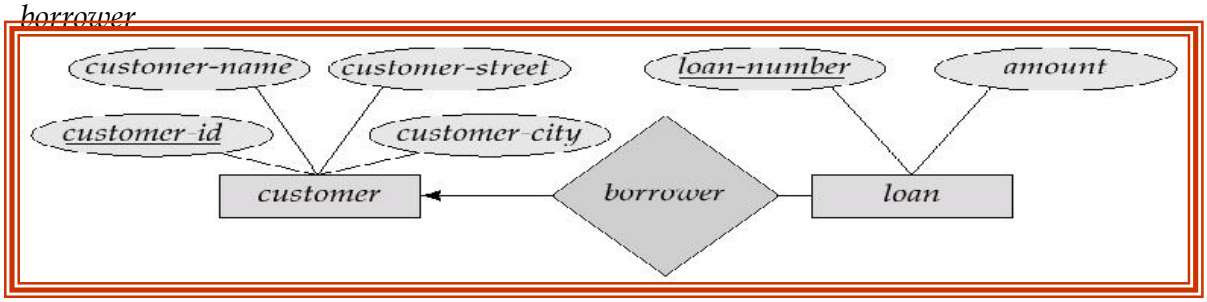



**E.g.:** Relationship ***Manages*** between ***STAFF*** and ***BRANCH***
***The multiplicity of the relationship is:***

- *One branch can only have one manager*
- *One employee could manage either one or no branches*

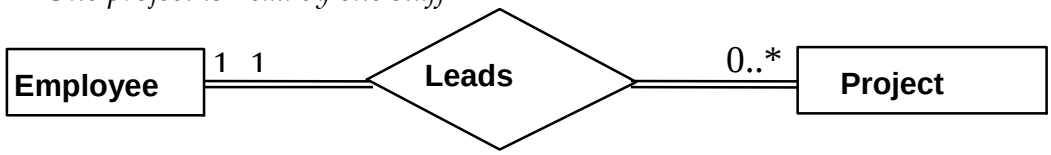
## One-To-Many Relationships

- In the one-to-many relationship a loan is associated with at most one customer via *borrower*, a customer is associated with several (including 0) loans via *borrower*



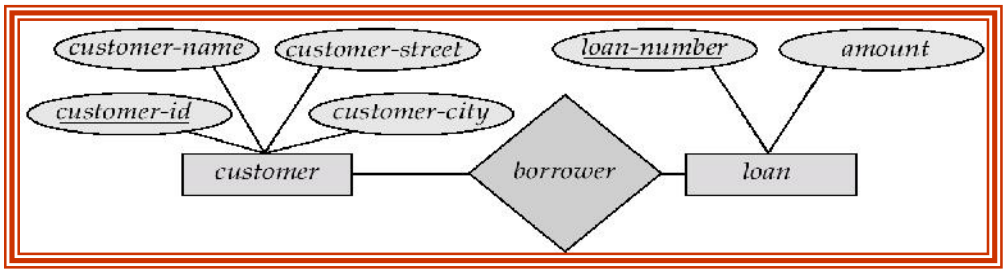

**E.g.:** Relationship ***Leads*** between ***STAFF*** and ***PROJECT***
***The multiplicity of the relationship***

- *One staff may Lead one or more project(s)*
- *One project is Lead by one staff*



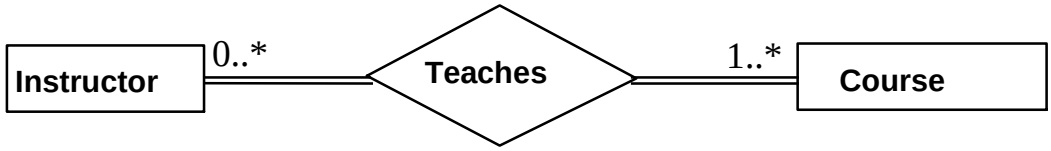

## Many-To-Many Relationship

- A customer is associated with several (possibly 0) loans via borrower
- A loan is associated with several (possibly 0) customers via borrower




**E.g.:** Relationship ***"Teaches" between INSTRUCTOR*** and ***COURSE***
***The multiplicity of the relationship***

- *One Instructor Teaches one or more Course(s)*
- *One Course Thought by Zero or more Instructor(s)*



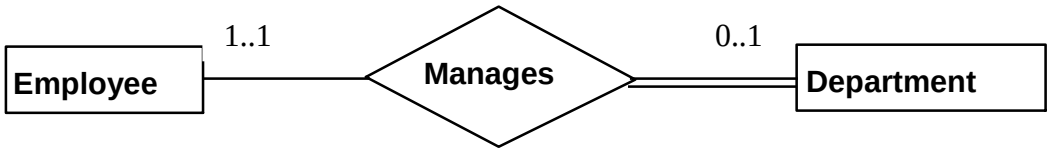
# Participation of an Entity Set in a Relationship Set

Participation constraint of a relationship is involved in identifying and setting the mandatory or optional feature of an entity occurrence to take a role in a relationship. There are two distinct participation constraints with this respect, namely: ***Total Participation*** and ***Partial Participation***

- **Total participation**: every tuple in the entity or relation participates in at least one relationship by taking a role. This means, every tuple in a relation will be attached with at least one other tuple. The entity with total participation in a relationship will be connected to the relationship using a double line.
- **Partial participation**: some tuple in the entity or relation may not participate in the relationship. This means, there is at least one tuple from that Relation not taking any role in that specific relationship. The entity with partial participation in a relationship will be connected to the relationship using a single line.

  - E.g. 1: Participation of EMPLOYEE in "belongs to" relationship with DEPARTMENT is total since every employee should belong to a department.
    Participation of DEPARTMENT in "belongs to" relationship with EMPLOYEE is total since every department should have more than one employee.


  - E.g. 2: Participation of EMPLOYEE in "manages" relationship with DEPARTMENT, is partial participation since not all employees are managers.
    Participation of DEPARTMENT in "Manages" relationship with EMPLOYEE is total since every department should have a manager.



# Problem in ER Modeling

The Entity-Relationship Model is a conceptual data model that views the real world as consisting of entities and relationships. The model visually represents these concepts by the Entity-Relationship diagram. The basic constructs of the ER model are entities, relationships, and attributes. Entities are concepts, real or abstract, about which information is collected. Relationships are associations between the entities. Attributes are properties which describe the entities.

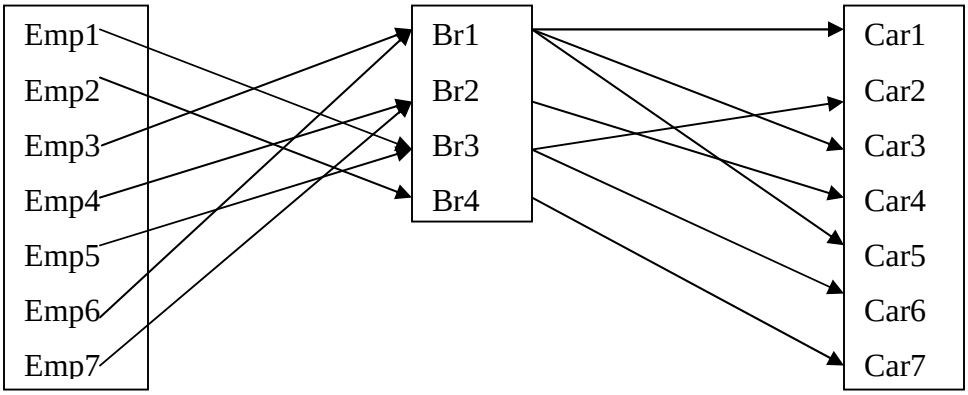
While designing the ER model one could face a problem on the design which is called a connection traps. **Connection traps** are problems arising from misinterpreting certain relationships

There are two types of connection traps;

1. **Fan trap:**
   Occurs where a model represents a relationship between entity types, but the pathway between certain entity occurrences is ambiguous.
   May exist where two or more one-to-many (1:M) relationships fan out from an entity. The problem could be avoided by restructuring the model so that there would be no 1:M relationships fanning out from a singe entity and all the semantics of the relationship is preserved.

   Example:


Semantics description of the problem;

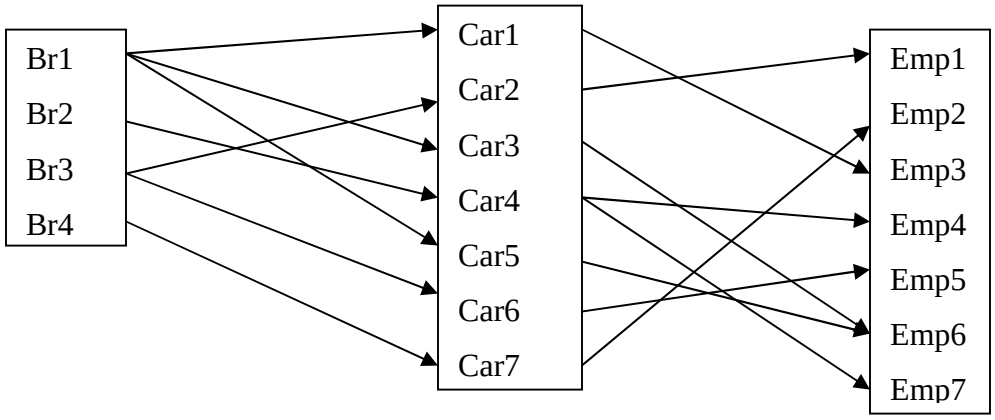



**Problem**: Which car (Car1 or Car3 or Car5) is used by Employee 6 Emp6 working in Branch 1 (Br1)? Thus from this ER Model one can not tell which car is used by which staff since a branch can have more than one car and also a branch is

populated by more than one employee. Thus we need to restructure the model to avoid the connection trap.

To avoid the Fan Trap problem we can go for restructuring of the E-R Model. This will result in the following E-R Model.


Semantics description of the problem;



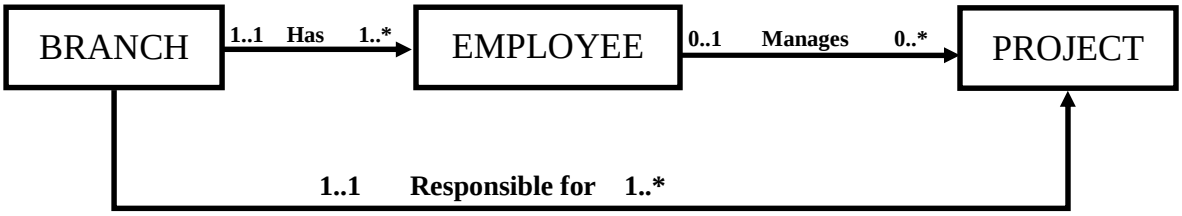
2. **Chasm Trap:**
   Occurs where a model suggests the existence of a relationship between entity types, but the path way does not exist between certain entity occurrences.
   Chasm trap may exist when there are one or more relationships with a minimum multiplicity on cardinality of zero forming part of the pathway between related entities.

   Example:


   If we have a set of projects that are not active currently then we can not assign a project manager for these projects. So there are project with no project manager making the participation to have a minimum value of zero.

   **Problem**:
   How can we identify which BRANCH is responsible for which PROJECT? We know that whether the PROJECT is active or not there is a responsible BRANCH. But which branch is a question to be answered, and since we have a minimum participation of zero between employee and PROJECT we can't identify the BRANCH responsible for each PROJECT.

   The solution for this Chasm Trap problem is to add another relation ship between the extreme entities (BRANCH and PROJECT)



# Enhanced E-R (EER) Models

- Object-oriented extensions to E-R model
- EER is important when we have a relationship between two entities and the participation is partial between entity occurrences. In such cases EER is used to reduce the complexity in participation and relationship complexity.
- ER diagrams consider entity types to be primitive objects
- EER diagrams allow refinements within the structures of entity types

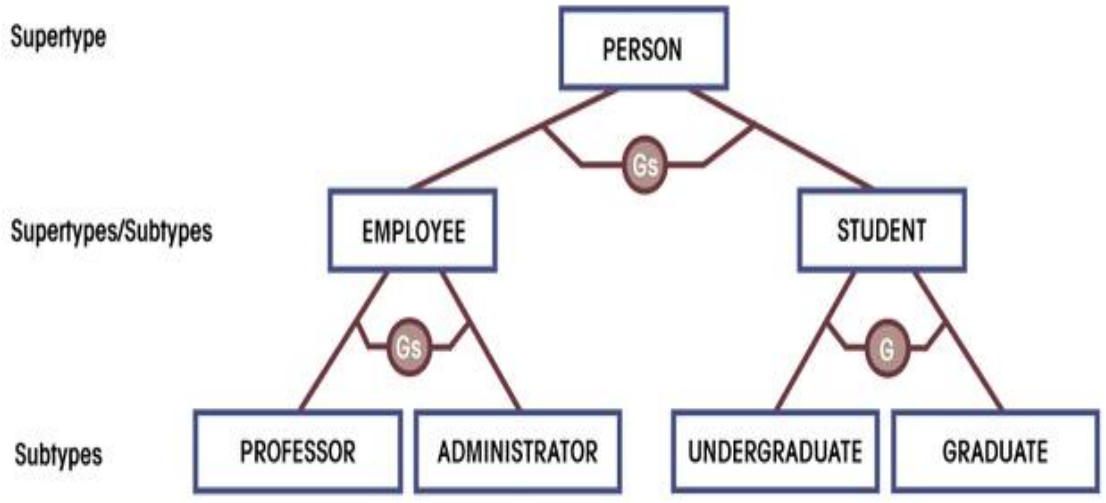
- EER Concepts
  - Generalization
  - Specialization
  - Sub classes
  - Super classes
  - Attribute Inheritance
  - Constraints on specialization and generalization

## ■ Generalization

- Generalization occurs when two or more entities represent categories of the same real-world object.
- Generalization is the process of defining a more general entity type from a set of more specialized entity types.
- A generalization hierarchy is a form of abstraction that specifies that two or more entities that share common attributes can be generalized into a higher level entity type.
- Is considered as bottom-up definition of entities.
- Generalization hierarchy depicts relationship between higher level superclass and lower level subclass.

Generalization hierarchies can be nested. That is, a subtype of one hierarchy can be a supertype of another. The level of nesting is limited only by the constraint of simplicity.

***Example***: Account is a generalized form for saving and Current Accounts



## ■ Specialization

- Is the result of subset of a higher level entity set to form a lower level entity set.
- The specialized entities will have additional set of attributes (distinguishing characteristics) that distinguish them from the generalized entity.
- Is considered as Top-Down definition of entities.
- Specialization process is the inverse of the Generalization process. Identify the distinguishing features of some entity occurrences, and specialize them into different subclasses.
- Reasons for Specialization
  - Attributes only partially applying to superclasses
  - Relationship types only partially applicable to the superclass
- In many cases, an entity type has numerous sub-groupings of its entities that are meaningful and need to be represented explicitly. This need requires the representation of each subgroup in the ER model. The generalized entity is a superclass and the set of specialized entities will be subclasses for that specific Superclass.
  - **Example**: Saving Accounts and Current Accounts are Specialized entities for the generalized entity Accounts. Manager, Sales, Secretary: are specialized employees.

## ■ Subclass/Subtype

- An entity type whose tuples have attributes that distinguish its members from tuples of the generalized or Superclass entities.
- When one generalized Superclass has various subgroups with distinguishing features and these subgroups are represented by specialized form, the groups are called subclasses.
- Subclasses can be either mutually exclusive (disjoint) or overlapping (inclusive).
- A single subclass may inherit attributes from two distinct superclasses.
- A mutually exclusive category/subclass is when an entity instance can be in only one of the subclasses.

  E.g.: An EMPLOYEE can either be SALARIED or PART-TIMER but not both.
- An overlapping category/subclass is when an entity instance may be in two or more subclasses.

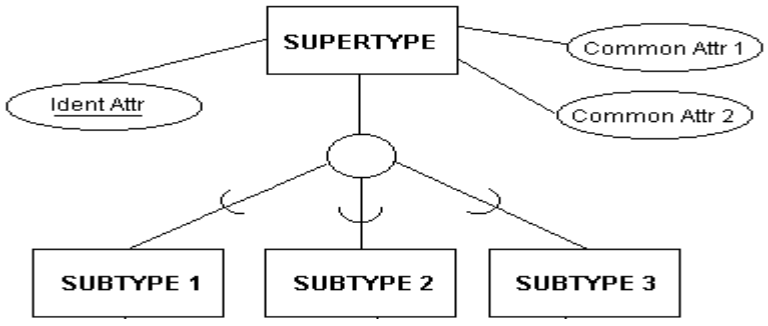
  E.g.: A PERSON who works for a university can be both EMPLOYEE and a STUDENT at the same time.

## ■ Superclass /Supertype

- An entity type whose tuples share common attributes. Attributes that are shared by all entity occurrences (including the identifier) are associated with the supertype.
- Is the generalized entity

## ■ Relationship Between Superclass and Subclass

- The relationship between a superclass and any of its subclasses is called a superclass/subclass or class/subclass relationship
- An instance can not only be a member of a subclass. i.e. Every instance of a subclass is also an instance in the Superclass.
- A member of a subclass is represented as a distinct database object, a distinct record that is related via the key attribute to its super-class entity.
- An entity cannot exist in the database merely by being a member of a subclass; it must also be a member of the super-class.
- An entity occurrence of a sub class not necessarily should belong to any of the subclasses unless there is full participation in the specialization.
- The relationship between a subclass and a Superclass is an "IS A" or "IS PART OF" type.
    - ***Subclass IS PART OF Superclass***
    - ***Manager IS AN Employee***
- All subclasses or specialized entity sets should be connected with the superclass using a line to a circle where there is a subset symbol indicating the direction of subclass/superclass relationship.

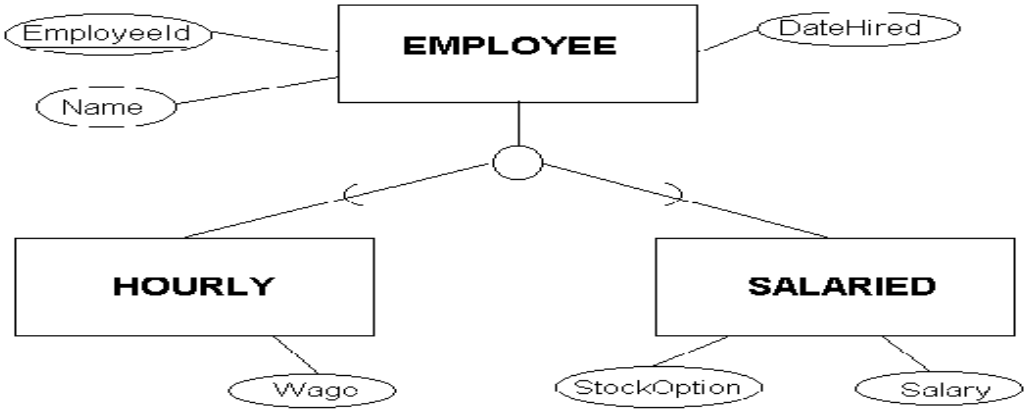



- We can also have subclasses of a subclass forming a hierarchy of specialization.

- Superclass attributes are shared by all subclasses of that superclass
- Subclass attributes are unique for the subclass.

## ■ Attribute Inheritance

- An entity that is a member of a subclass inherits all the attributes of the entity as a member of the superclass.
- The entity also inherits all the relationships in which the superclass participates.
- An entity may have more than one subclass categories.
- All entities/subclasses of a generalized entity or superclass share a common unique identifier attribute (primary key). i.e. The primary key of the superclass and subclasses are always identical.



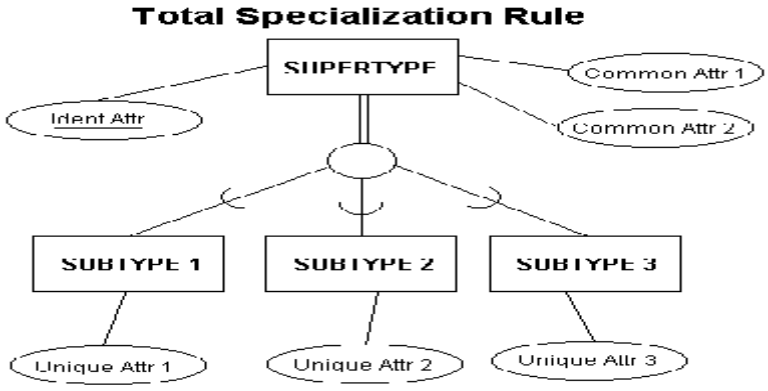

- Consider the **EMPLOYEE** supertype entity shown above. This entity can have several different subtype entities (for example: **HOURLY** and **SALARIED**), each with distinct properties not shared by other subtypes. But whether the employee is **HOURLY** or **SALARIED**, same attributes (EmployeeId, Name, and DateHired) are shared.
- The Supertype **EMPLOYEE** stores all properties that subclasses have in common. And **HOURLY** employees have the unique attribute Wage (hourly wage rate), while **SALARIED** employees have two unique attributes, StockOption and Salary.

# Constraints on specialization and generalization

## ■ Completeness Constraint.

- The **Completeness Constraint** addresses the issue of whether or not an occurrence of a Superclass must also have a corresponding Subclass occurrence.
- The completeness constraint requires that all instances of the subtype be represented in the supertype.
- The **Total Specialization Rule** specifies that an entity occurrence should at least be a member of one of the subclasses. Total Participation of superclass instances on subclasses is diagrammed with a **double line** from the Supertype to the circle as shown below.

  E.g.: If we have EXTENTION and REGULAR as subclasses of a superclass STUDENT, then it is mandatory that each student to be either EXTENTION or REGULAR student. Thus the participation of instances of STUDENT in EXTENTION and REGULAR subclasses will be total.




- The **Partial Specialization Rule** specifies that it is not necessary for all entity occurrences in the superclass to be a member of one of the subclasses. Here we have an optional participation on the specialization. Partial Participation of superclass instances on subclasses is diagrammed with a **single line** from the Supertype to the circle.

  E.g.: If we have MANAGER and SECRETARY as subclasses of a superclass EMPLOYEE, then it is not the case that all employees are either manager or secretary. Thus the participation of instances of employee in MANAGER and SECRETARY subclasses will be partial.

## ■ Disjointness Constraints.

- Specifies the rule whether one entity occurrence can be a member of more than one subclasses. i.e. it is a type of business rule that deals with the situation where an entity occurrence of a Superclass may also have more than one Subclass occurrence.
- The **Disjoint Rule** restricts one entity occurrence of a superclass to be a member of only one of the subclasses. Example: a EMPLOYEE can either be SALARIED or PART-TIMER, but not the both at the same time.
- The **Overlap Rule** allows one entity occurrence to be a member f more than one subclass. Example: EMPLOYEE working at the university can be both a STUDENT and an EMPLOYEE at the same time.
- This is diagrammed by placing either the letter "**d**" for disjoint or "**o**" for overlapping inside the circle on the Generalization Hierarchy portion of the E-R diagram.

The two types of constraints on generalization and specialization (Disjointness and Completeness constraints) are not dependent on one another. That is, being disjoint will not favour whether the tuples in the superclass should have Total or Partial participation for that specific specialization.

From the two types of constraints we can have four possible constraints

- ■ Disjoint AND Total
- ■ Disjoint AND Partial
- ■ Overlapping AND Total
- ■ Overlapping AND Partial