ER Modeling

(Course Instructor: Bidur Devkota)

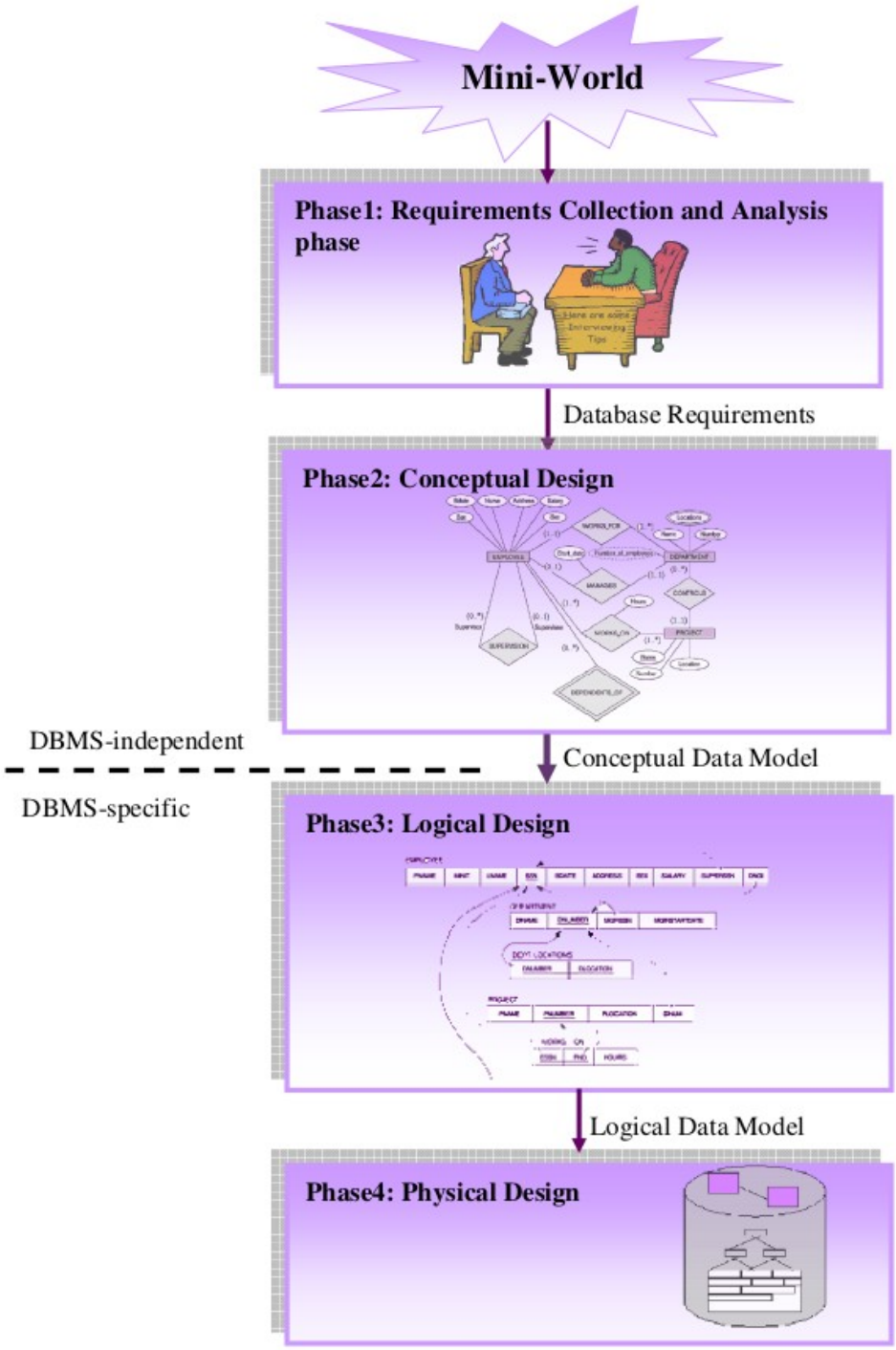


Figure: Design Phase

1. Databse Design Phases:

The main phases of database design are:

1. Phase 1: Requirements collection and analysis.
   - get a **description of the user requirements.**
   - **Interview** prospective database users to understand and document their data requirements
   - prepare **users' requirements** and **functional requirements** of the database application

2. Phase 2: Conceptual Design.
   - Develop conceptual schema for the database with High-Level Conceptual Data Model:
     - entity types,
     - relationship types,
     - constraints
   - do not include implementation details.
   - Act as a reference to ensure:
     - all data requirements are met
     - requirements do not include conflicts.

3. Phase 3: Logical design (data model mapping/Implementation)
   - Transform the high level conceptual schema into a lower- level implementation model.
   - Result of this phase is the schema as an implemented data model of the DBMS.

4. Phase 4: Physical design.
   - Decision about the internal storage structures, access paths and file organizations for the database files.

Example:

Design a database schema to fulfill the requirements for a company.

- Employees, departments, and projects
- Company is organized into departments
- Department controls several projects
- Employee: require each employee's name, Social Security number, address, salary, sex (gender), and birth date
- Keep track of the dependents of each employee

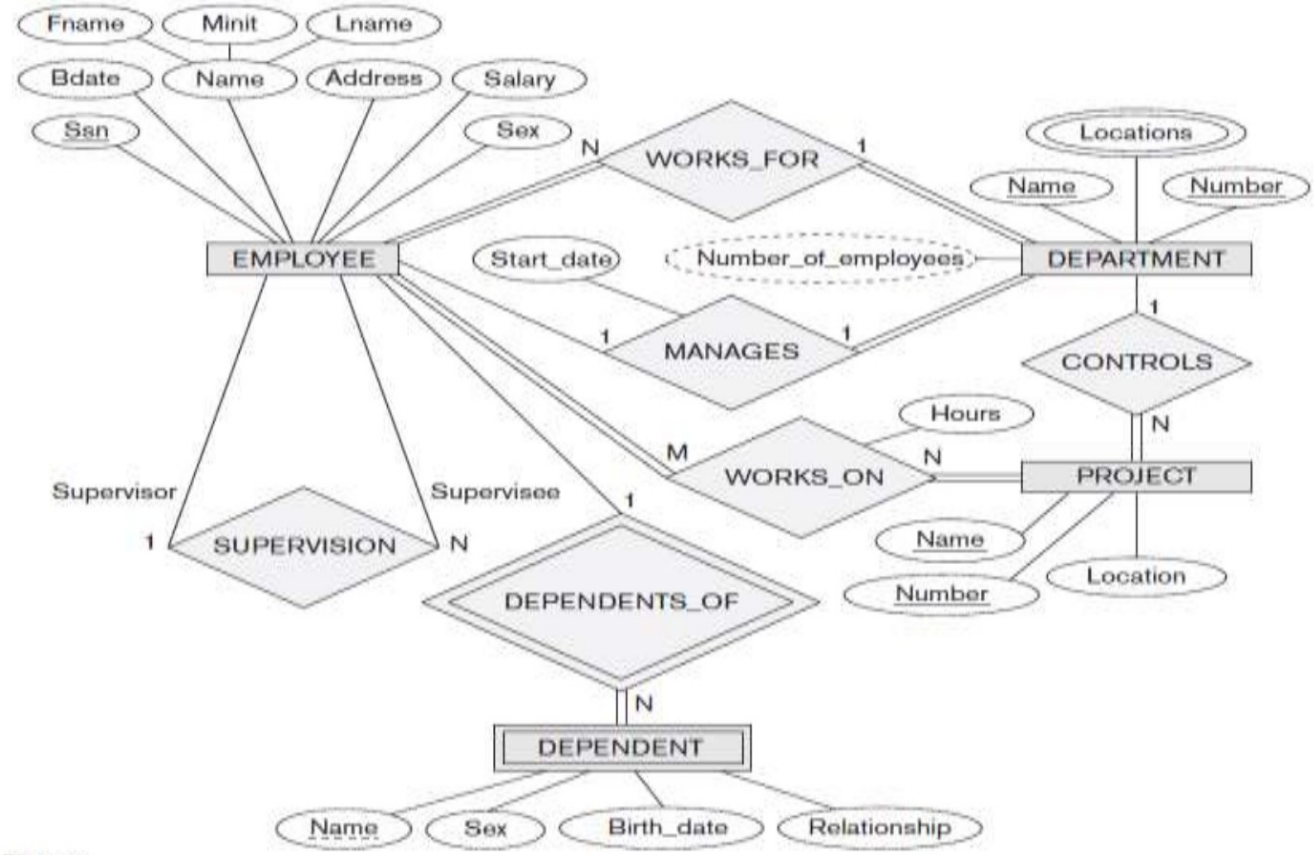


Figure: ER diagram for a company

Notations:

ER- Diagram is a visual representation of data that describe how data is related to each other.

Rectangles: This symbol represent entity types
Ellipses : Symbol represent attributes
Diamonds: This symbol represents relationship types
Lines: It links attributes to entity types and entity types with other relationship types
Primary key: attributes are underlined
Double Ellipses: Represent multi-valued attributes

2. **Entity-Relationship (ER) Model**

- a **high- level conceptual data model.**
- By Peter Chen in 1976
- **illustrates the real world situations** with concepts, used by general people.
- **Logically** describes a representation of the real world.
- Do **NOT** describe machine-related aspects.
- **grouping of data into entities**
- main concepts: **entity, attribute, and relationship.**

- **Entity:**
  - a "thing" (an object or an event) in the real world:
  - distinguishable from other things.
  - Entity types represent sets of objects and are pictured by rectangular nodes.
  - **Examples of entities:**
    - **Person:** Employee, Student, Patient
    - **Place:** Store, Building
    - **Object:** Machine, product, and Car
    - **Event:** Sale, Registration, Renewal
    - **Concept:** Account, Course

- **Characteristic of Entity:**
  - Concrete physical Existence like:
    - Employee, Collage, Car, Book, Dog, etc.
  - Described by its attributes:
    - an EMPLOYEE entity may have a Name, Age, Address, DOB,etc.
  - Determined by particular value of its attributes:
    - An Employee Name= “Hari”, Age=21, Address=Pokhara, DOB:1998-08-09
    - **Another** Employee Name= “Sita”, Age=20, Address=Baglung, DOB:1999-08-12
- **Entity sets may not be disjoint.**
  - E.g, it is possible to define the entity set of all employees of a Supermarket (employee) and the entity set of all customers of the Supermarket (customer). A person entity may be an employee entity, a customer entity, both, or neither.
- An **entity type** defines a set of entities with same attribute:
  - Eg. **Employee** entity type and **Company** entity type
  - The figure below shows two entity types, named **Employee** and **Company**, and their attribute list.
- **An entity is an instance of an entity type**.
- An **entity set is the collection of all entities of a particular entity type in the database** at any point in time.
  - Normally, **entity set** is referred by the same name as **entity type**.
    - E.g., **Employee** refers to both entity type and also the set of all employee entities.

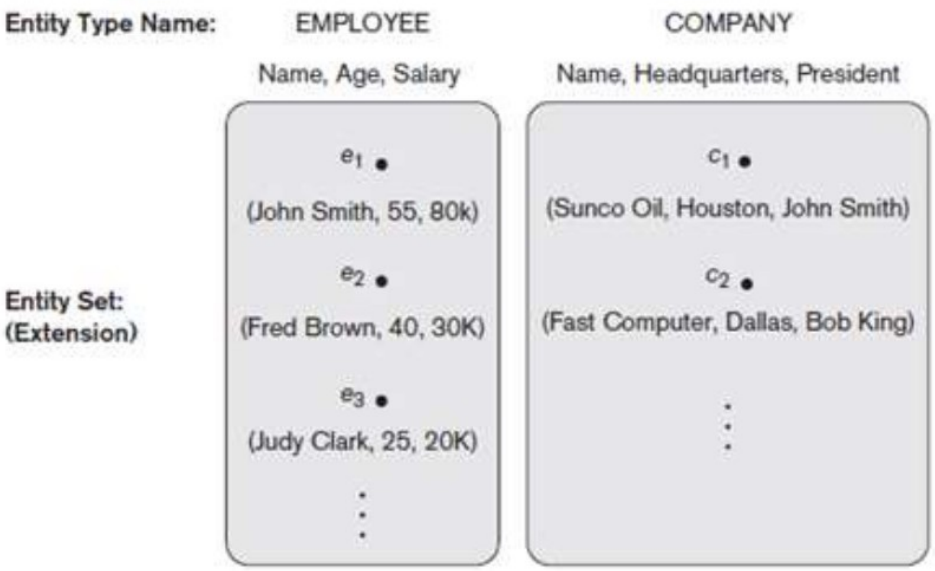
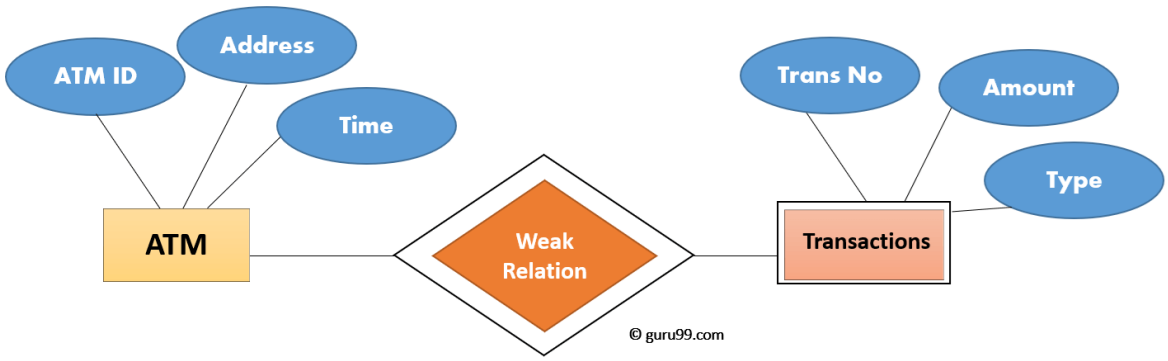


Figure
Two entity types, EMPLOYEE and COMPANY, and some member entities of each.

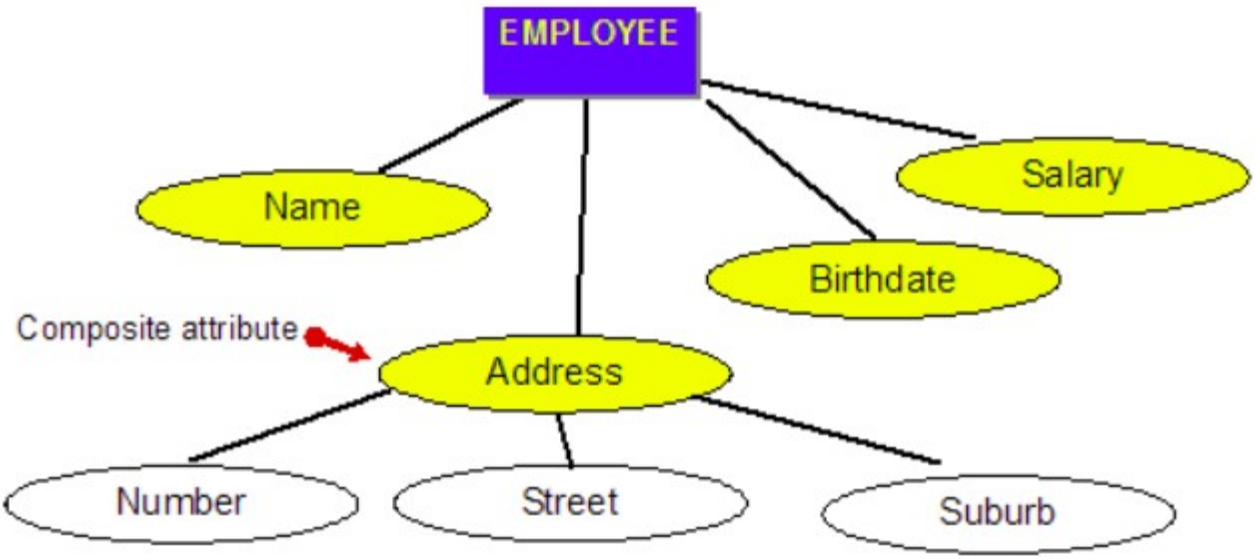
- **Weak Entities**
  - doesn't have its key attribute.
  - It can be identified uniquely by considering the primary key of another entity.
  - For that, weak entity sets need to have participation.
  - E.g. "Trans No" is a discriminator within a group of transactions in an ATM.



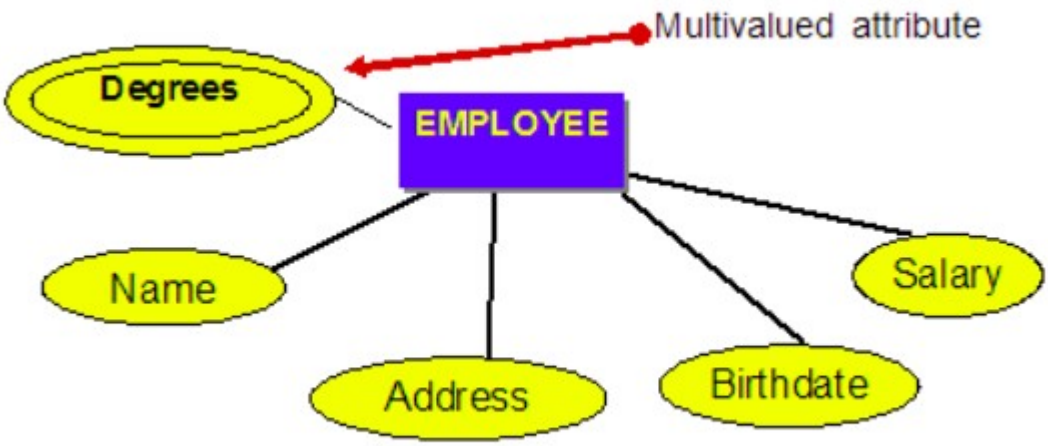
Strong Entity Set	Weak Entity Set
Strong entity set always has a primary key.	It does not have enough attributes to build a primary key.
It is represented by a rectangle symbol.	It is represented by a double rectangle symbol.
It contains a Primary key represented by the underline symbol.	It contains a Partial Key which is represented by a dashed underline symbol.
The member of a strong entity set is called as dominant entity set.	The member of a weak entity set called as a subordinate entity set.
Primary Key is one of its attributes which helps to identify its member.	In a weak entity set, it is a combination of primary key and partial key of the strong entity set.
In the ER diagram the relationship between two strong entity set shown by using a diamond symbol.	The relationship between one strong and a weak entity set shown by using the double diamond symbol.
The connecting line of the strong entity set with the relationship is single.	The line connecting the weak entity set for identifying relationship is double.

- **Attribute:**
  - **property** or **characteristic** of an entity type.
  - **Each attribute has a value drawn from some domain** (set of meaningful values).
  - Attributes name are written in an **ellipse** by **connecting it to its associated entity**
  - Attributes may also be associated with relationships
  - **An attribute is associated with exactly one entity or relationship.**
  - **Example:**
    - an employee entity may be described by the employee's name, age, address, dob, and job
    - An Employee Name= "Hari", Age=21, Address=Pokhara, DOB:1998-08-09, job: programmer

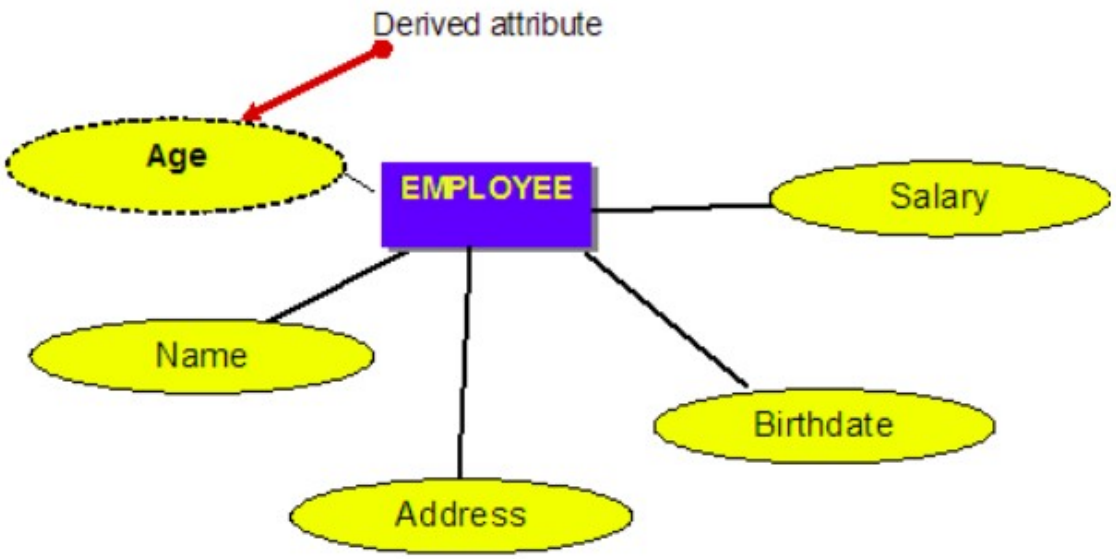
- **Types of attributes :**
  - Composite versus Simple(atomic):
    - can be divided into smaller subparts :
      - **Name** can First Name, Middle Name, Lastname.
      - **Address** can have Street_Address, city, state and postal_code



- Single-valued versus Multivalued:
  - Single Valued: Age is a single-valued attribute of person.
  - **multivalued** attribute may take on more than one value:
    - E.g. An EMPLOYEE may have more than one **degree**

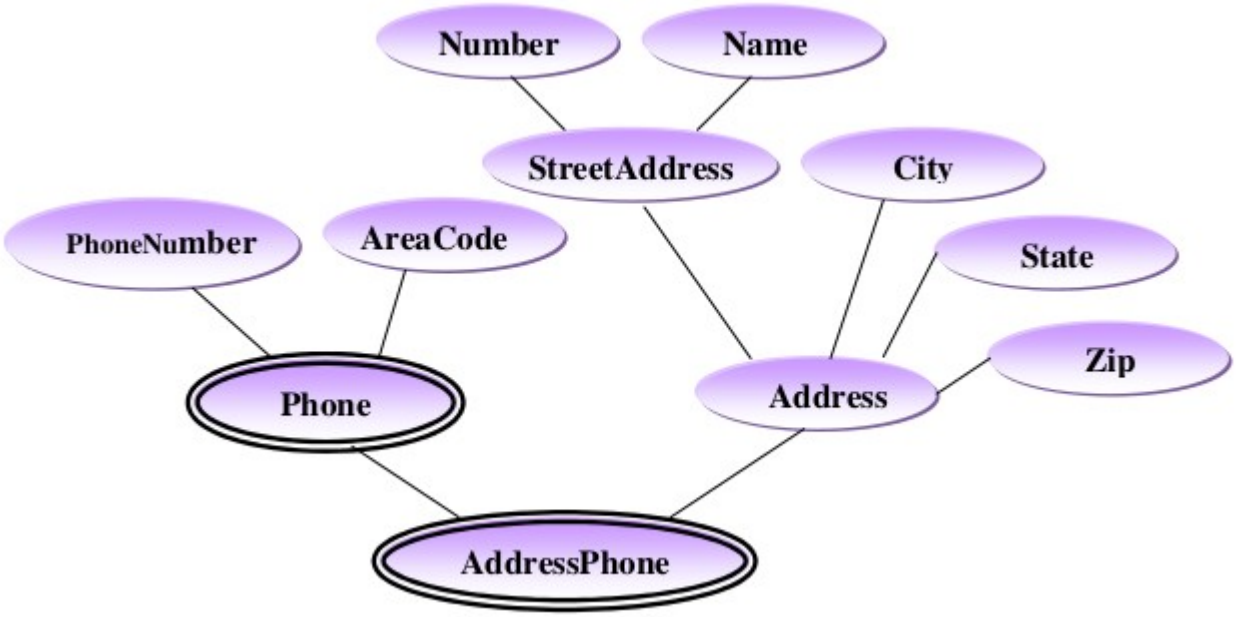


- Stored versus Derived.
  - Derived attributes: Attributes Contain values that are calculated from other attributes.
  - **Age** can be **derived** from attribute Birthdate.
  - **Birthdate** is **stored** attribute.



- Null values:
  - Some values may not be applicable e.g. **Deceased Date**

- Complex values:
  - Nested Composite and multi-valued attributes .
  - Example:
    - **PreviousDegrees** of a STUDENT is a composite multi-valued attribute denoted by {PreviousDegrees(College, Year, Degree, Field)}.
    - Address and Phone

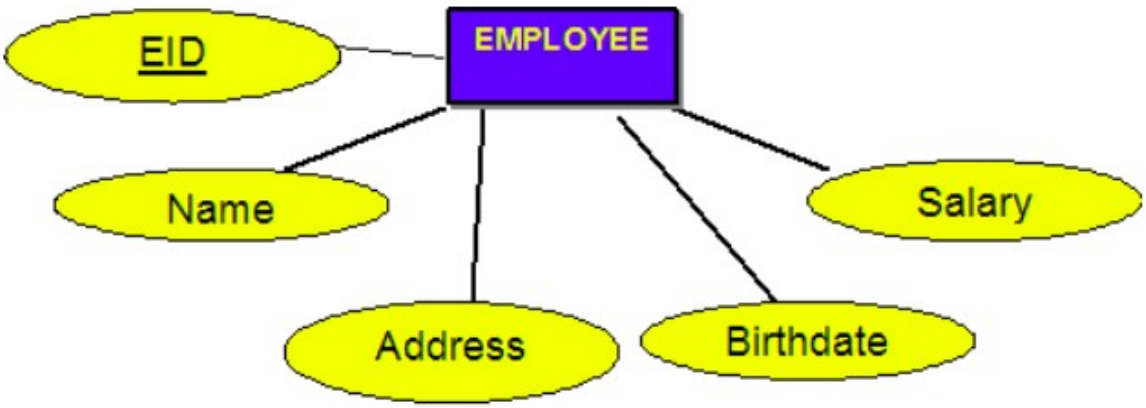


Student

Roll_no	Name	Class	Age
1	Andrew	5	12
2	Andrew	5	12
3	Augusto	5	11

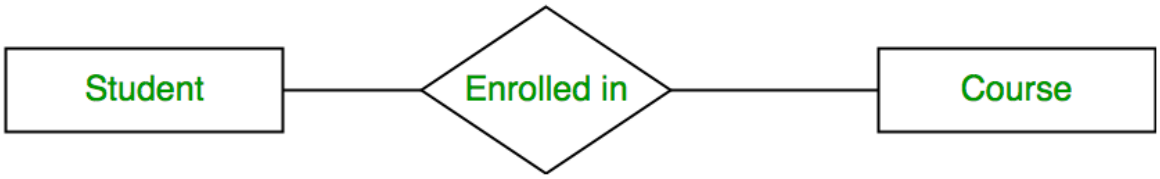
- **Key:**
  - **important constraint on the entities**
  - Key is an **attribute** or a group of attributes that **uniquely identify** individual entity in an entity set.
  - E.g.
    - Student's **Roll Number** is unique for each person.
    - Employee = {**EID**, Name, Address, Age, Salary}
      - Definite keys are any set that involving EID
      - Possible keys might be {Name, Address}
      - Unlikely keys: {Name}, {Age}

- **Primary Key:**
  - The primary key is the **minimal set of attributes which uniquely identifies** any row of a table.
  - key that is chosen as an identifying mechanism for the whole entity set
  - indicated by underlying attributes in the ER model.

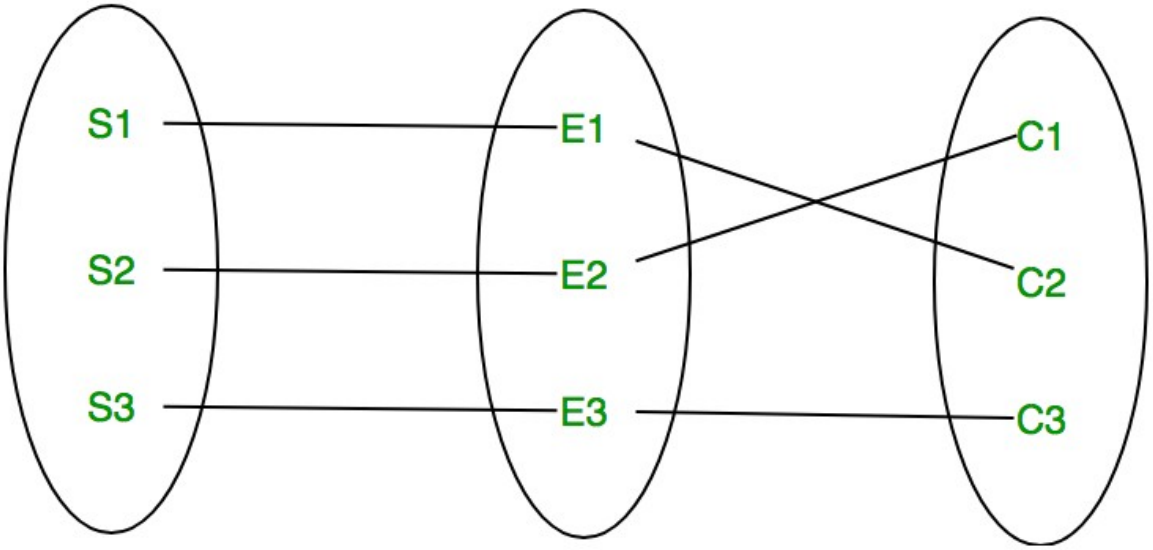


- **Relationship**
  - an **association** among two or more entities.
    - E.g. Ram **works** in the Chemistry department.
  - Entities take part in relationships.
  - Relationships: verbs or verb phrases.
  - Attributes of relationship (if present) → **descriptive attributes.**

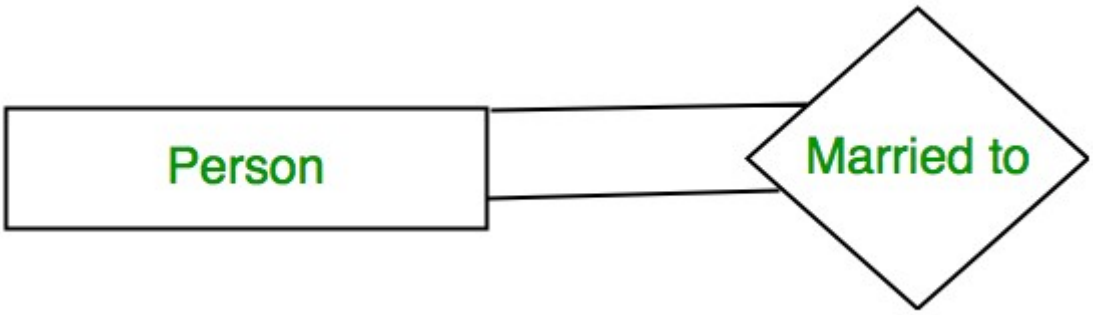
- **Relationship Type and Relationship Set:**
  - **Relationship Type:**
    - association between entity types.
    - E.g: **'Enrolled in'** is a relationship between entity type **Student** and **Course**.



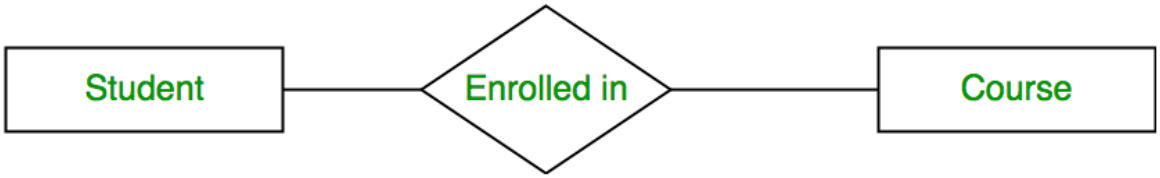
  - **Relationship Set:**
    - A set of relationships of same type.
    - E.g. Relationship Set in the figure below,
      - **S1** (student 1) is enrolled (**E1**) in **C2** (Course 2)
      - S2 is enrolled in C1.
      - S3 is enrolled in C3.



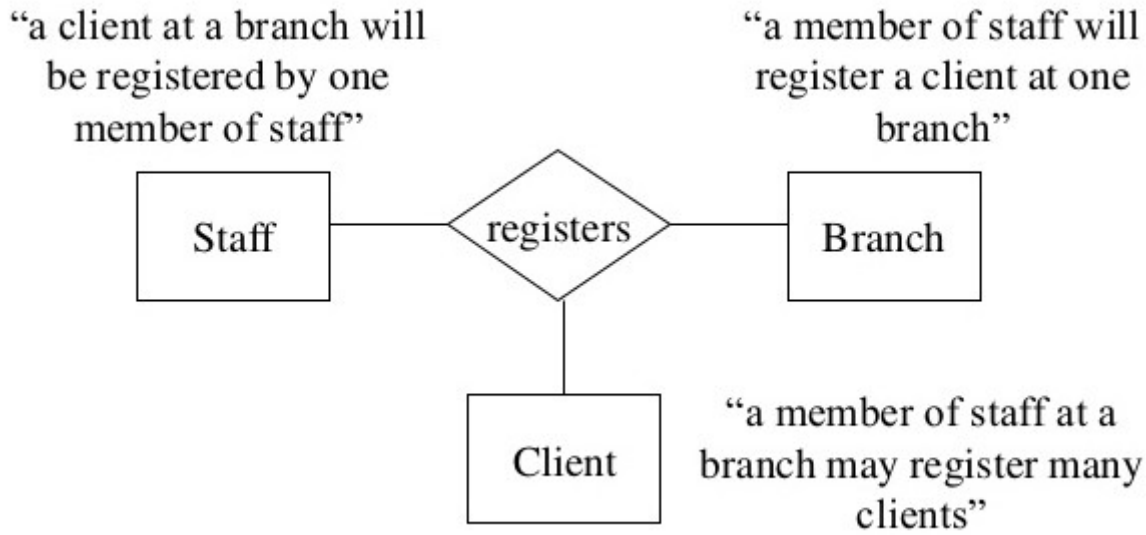
- **Relationship Set Degree:**
  - number of participating entity
  - **Unary** Relationship:
    - only ONE entity set participating in a relation



  - **Binary** Relationship :
    - With **2** entity sets (or degree two).
    - Most common relationship sets in a database system.
    - E.g.: **student** is enrolled in a **course**.

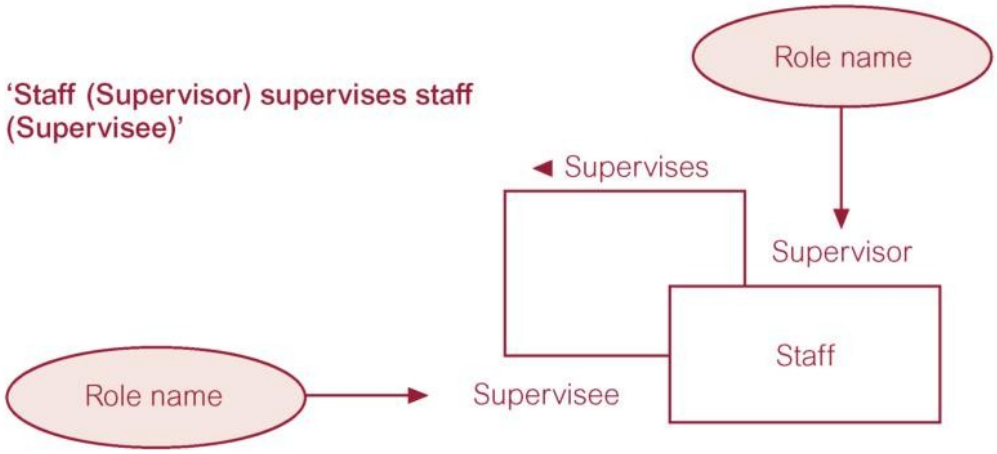


  - **Ternary** Relationship:
    - With **3** entities
    - E.g.: **Staff** registers **clients** at a **branch**.



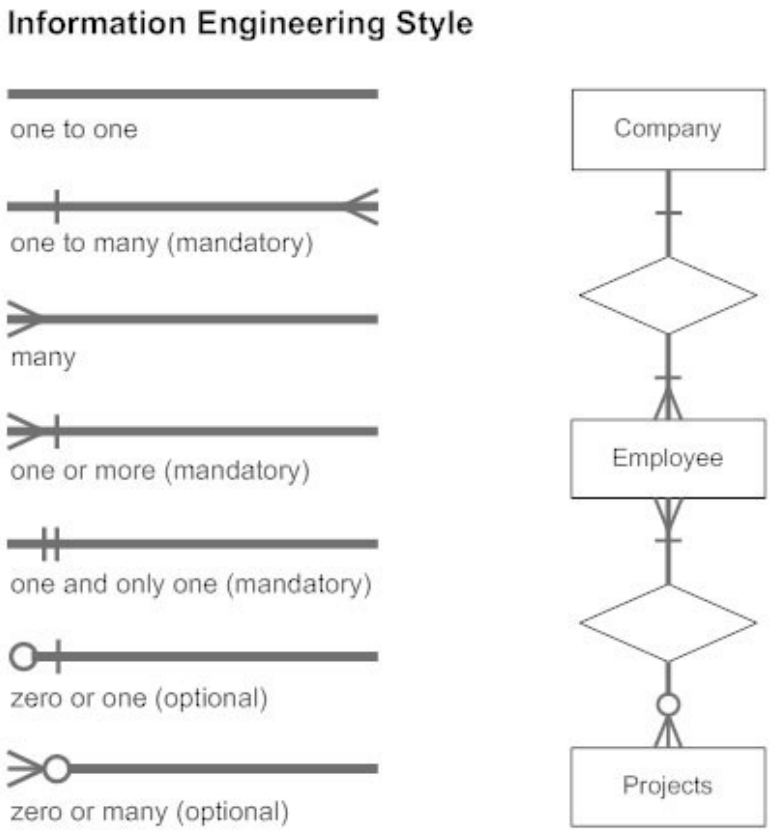
- **N-array** Relationship:
  - With **N** entities

- **Role Names and Recursive Relationship :**
  - **role name** illustrates the role of the entity type in a relationship
    - Tells what the relationship means.
  - In **Recursive Relationship** a single entity type involves in different roles.



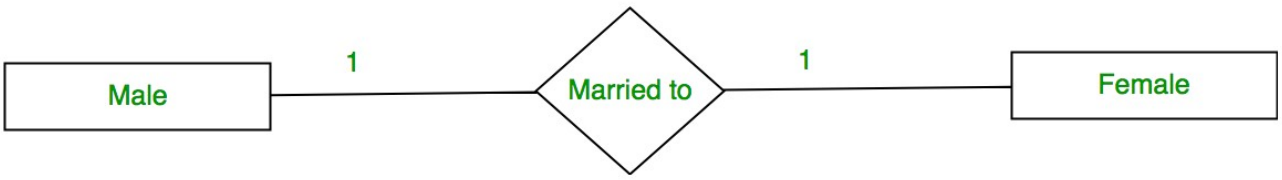
- **Constraints on Relationship Types**
  - constraints limit the possible combination of entities in a relationship.
    - Eg. For entities Doctor and Patient,
      - **Rule**: a patient cannot be seen by more than one doctor.
    - Such constraint should be illustrated in the schema.
  - There are 2 **main types of relationship constraints**
    - **Cardinality** ratio for binary relationship constraints
    - **Participation** constraints and Existence Dependencies.

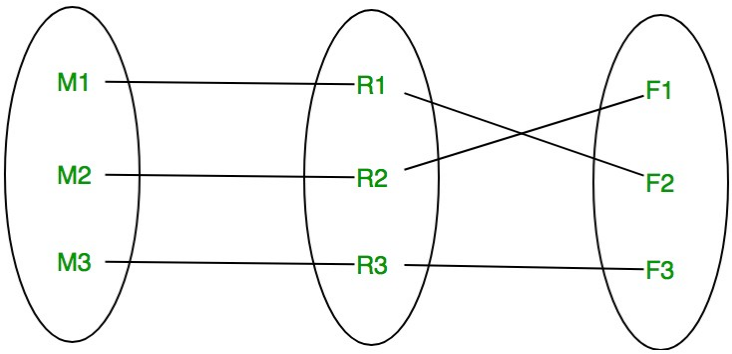
- **Cardinality**:
  - number of times an entity of an entity set participates in a relationship set.



One-One Relationship

each entity in each entity set can take part **only once in the relationship**

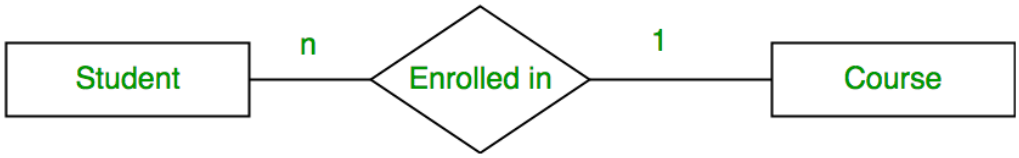


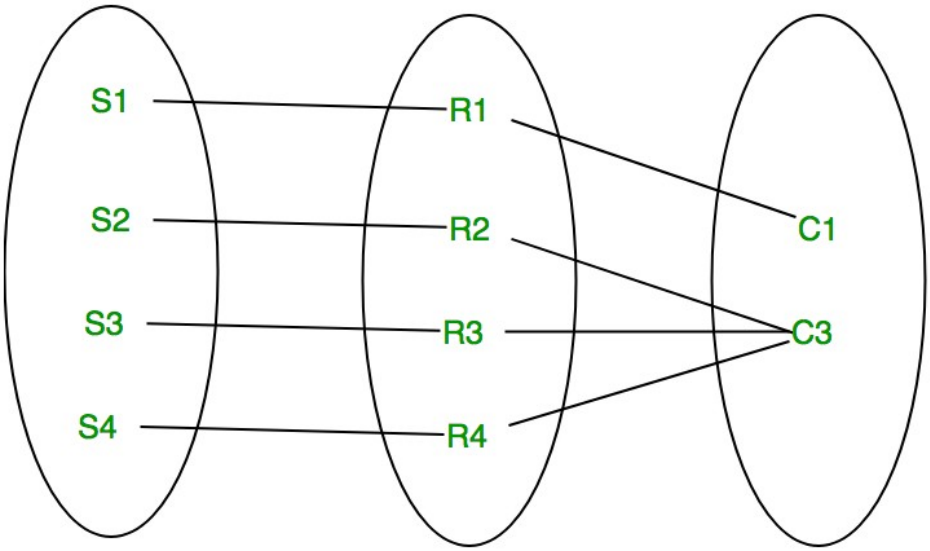


a male can marry to one female and a female can marry to one male

Many to one

entities in one entity set **can take part only once in the relationship set and entities in other entity set can take part more than once in the relationship set**

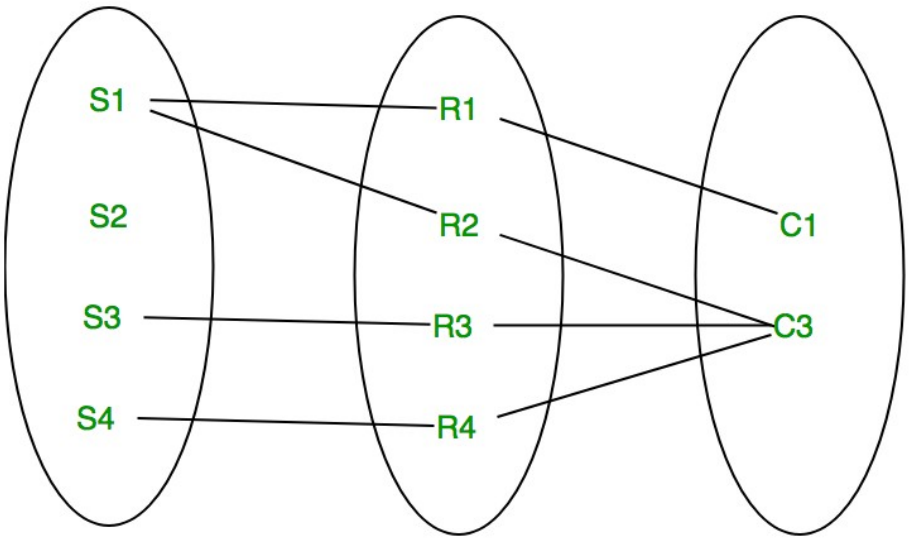




student can take only one course but one course can be taken by many students.

Many to many

entities in all entity sets can **take part more than once in the relationship**



a student can take more than one course and one course can be taken by many students.

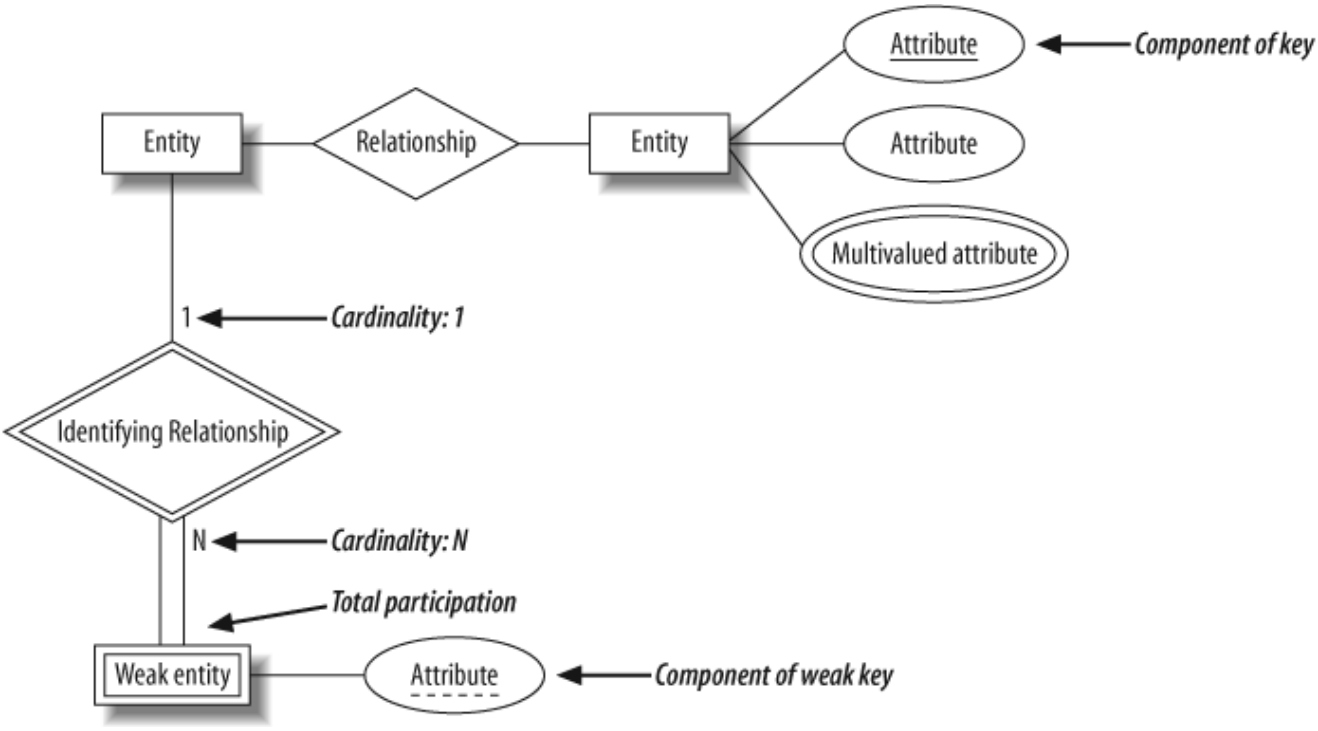
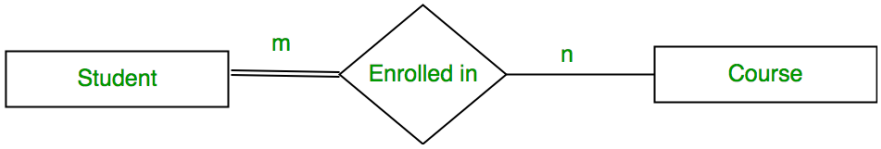
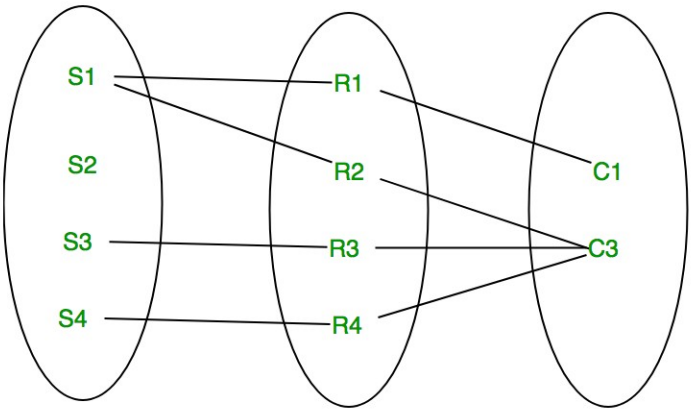


Figure: Quick summary of the ER diagram symbols

- **Participation Constraints and Existence Dependencies**
  - Defines the minimum number of relationship instances in which an entity must participate.
  - Types of participation constraints: (1)total Participation and (2) partial Participation
  - **Total Participation**
    - Each entity in the entity set **must participate** in the relationship.
    - E.g. Each **student must enroll in a course.**
    - Shown by **double** line in ER diagram.
    - **Double** line between the entity set "**Student**" and relationship set "**Enrolled** in" signifies total participation.

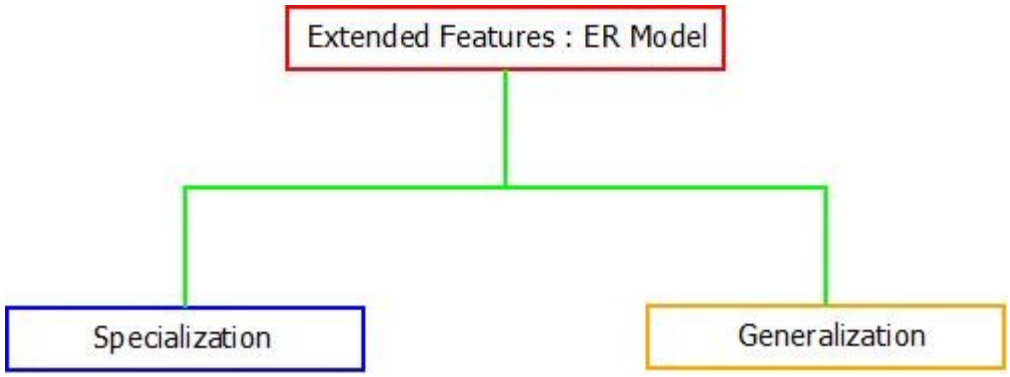




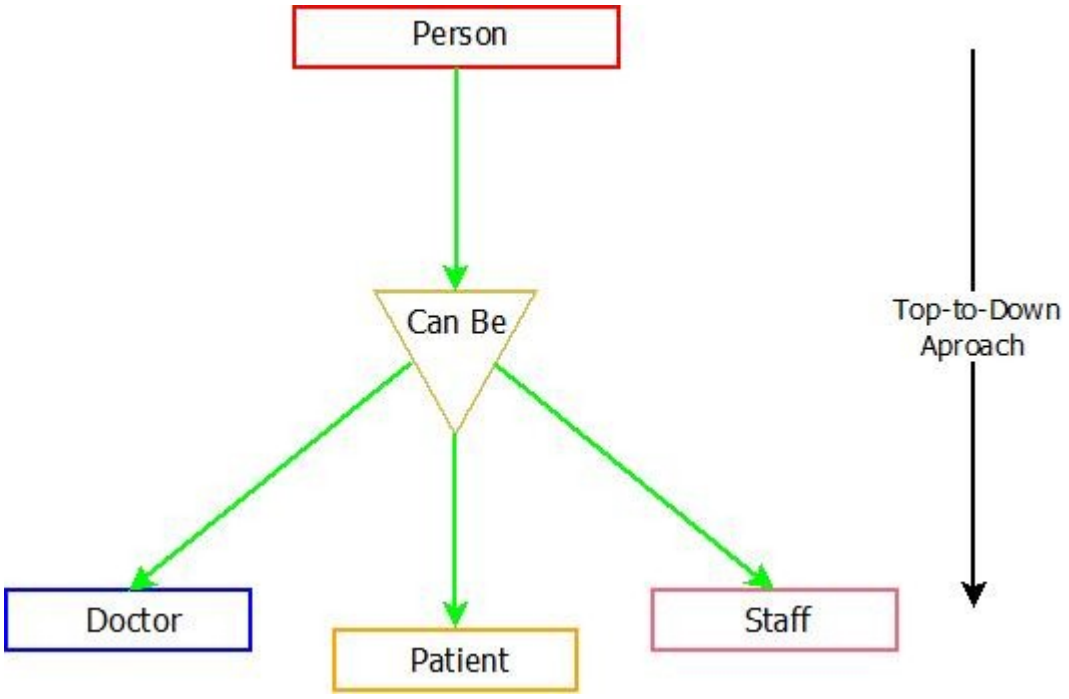
  - **Partial Participation**
    - **optional participation.**
    - Each entity in the entity set **may or may not participate** in the relationship.
    - Shown by **Single** line in ER diagram.
    - Single line between the entity set "**Course**" and relationship set "**Enrolled in**" shows partial participation.

Extended Features Of ER Model : Specialization & Generalization

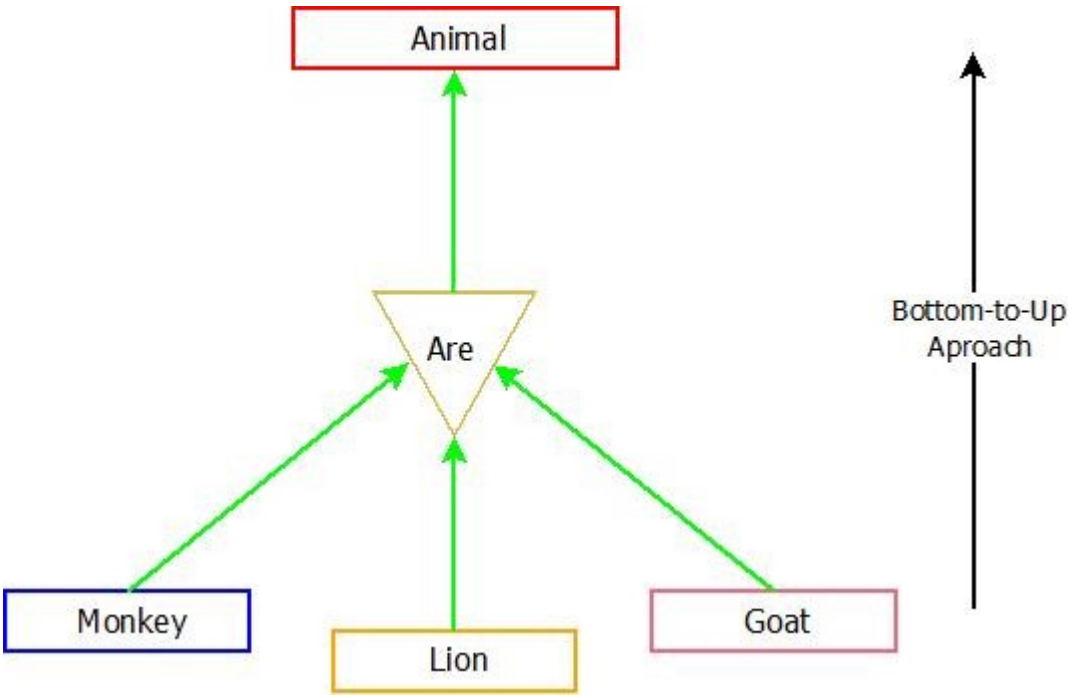
- Specialization and Generalization
- based upon the conceptual hierarchy
  - the manner in which ER Diagram is generated.

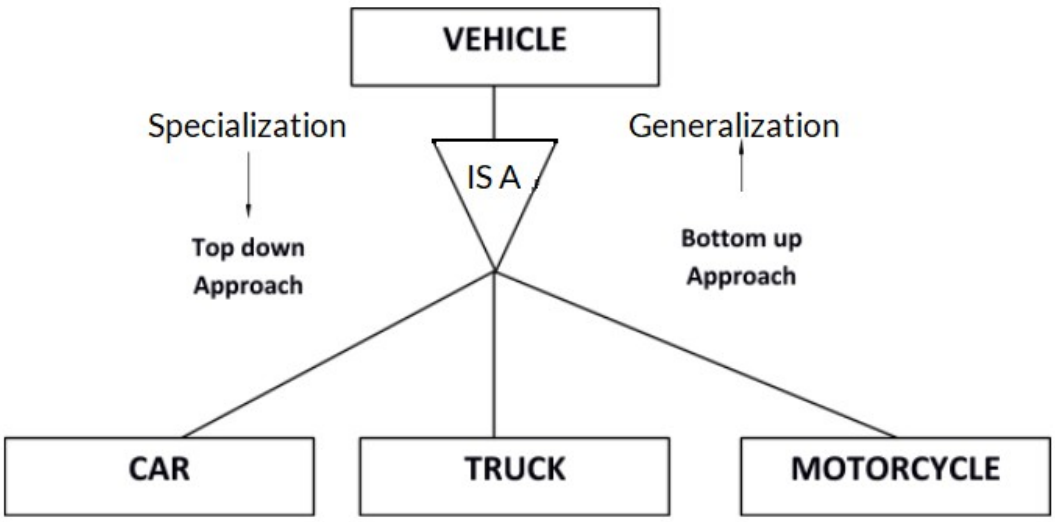


- **Specialization**
  - Top-to-Down approach.
  - process of **mastering** a specific domain.
  - In ER Model, specialization is the procedure to split up the entities into further sub entities on the basis of their functionalities and features.
  - E.g. : In a Hospital database , a person can either be a doctor, staff or patient which can be spirited according to their functions and specialties.

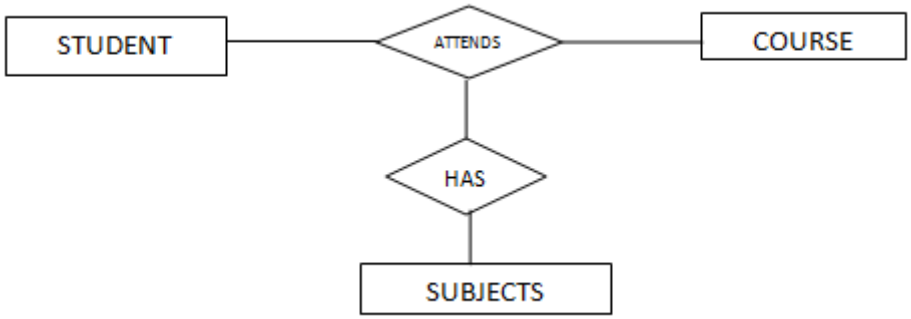


- **Generalization**
  - Bottom-to-Up approach.
  - Inverse of specialization.
  - The sub entities are **combined together into a super entity** set on the basis of some common features in such a way that the new entity thus formed contains all the features of the sub entities.
  - Example : Goat, Lion and Monkey have some come features of class animals.

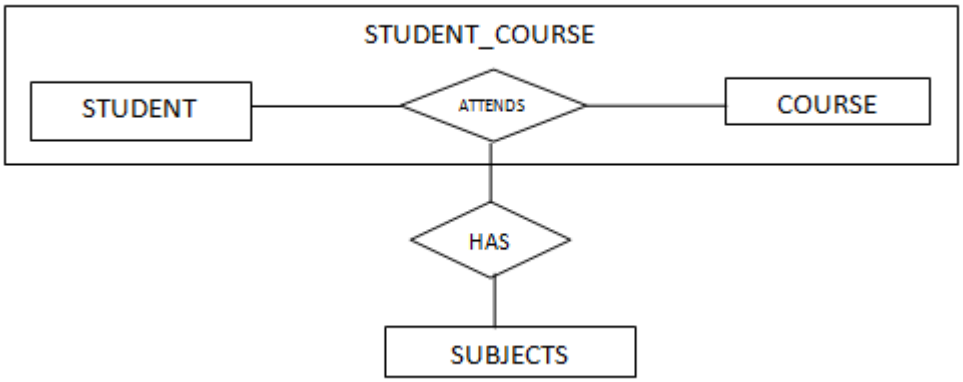




- **Aggregation**
  - Let us consider a scenario:
    - Student attends Course
    - Student has some subjects to study.
    - Course offers some subjects.



  - **It involves a relation over another relation.**
    - But ER Modeling does **NOT** support such relation. I
    - ER Model supports mapping **between entities, NOT** between **relations**.
    - **Aggregation** provides a solution in such scenarios.



  - From SUBJECT's offers its service to both STUDENT and COURSE.
  - So **merge** STUDENT and COURSE as one entity.
    - This Merging process → **Aggregation**.
  - STUDENT and COURSE are merged into one entity STUDENT_COURSE.
    - The new entity STUDENT_COURSE:
      - has two entities STUDENT and COURSE with 'Attends' relationship.
      - forms the mapping with SUBJECTS.
  - In summary:
    - Aggregation represents **relationship between a whole object and its component.**
    - It **expresses relationship among relationships.**
    - Aggregation involves merging entities but it is different than generalization.
      - Generalization: **merge entities of same domain** into one entity.
      - Aggregation: **merge related entities** into one entity.

References:

The Instructor do not own the contents (text and images) of this documents collected from various sources:

- A. Silberschatz, H.F. Korth, and S. Sudarshan, Database System Concepts, 4th Edition, McGraw Hill (ISBN: 0-07-120413)
- Raghu Ramakrishnan, and Johannes Gehrke, Database Management Systems, McGraw-Hill, 2003. (ISBN: 0-07-246563-8)
- Sawsan Abu_TaleB and Randa Al_Dallah (2009), Chapter 3: Data modeling using the Entity-Relationship Model, Available Online:https://www.kau.edu.sa/GetFile.aspx?id=144658&fn=Ch3_Data%20modeling%20using%20the%20Entity%20Relationship%20Model_2.pdf
- https://s3.amazonaws.com/ppt-download/dbmschap3-190406084105.pdf?response-content-disposition=attachment&Signature=k0wi6dlx2uVU9DmBhkrmAz0Avqo%3D&Expires=1597851105&AWSAccessKeyId=AKIAIA5TS2BVP74IAVEQ
- https://afteracademy.com/blog/what-are-super-key-primary-key-candidate-key-and-foreign-keys
- https://www.tutorialspoint.com/Generalization-Specialization-and-Aggregation-in-ER-Model
- https://www.guru99.com/er-diagram-tutorial-dbms.html
- https://cnx.org/contents/aM2VUeRT@1/Data-Modeling-Using-Entity-Relationship-Model
- https://www.slideshare.net/rupalirana07/ch-3-38604666
- https://www.geeksforgeeks.org/introduction-of-er-model/
- https://www.slideserve.com/xaviera-ferguson/database-systems
- https://www.smartdraw.com/entity-relationship-diagram/
- https://www.minigranth.com/dbms-tutorial/specialization-generalization/
- https://pudbmsnotes.blogspot.com/2016/11/convert-er-into-tablesrelation.html