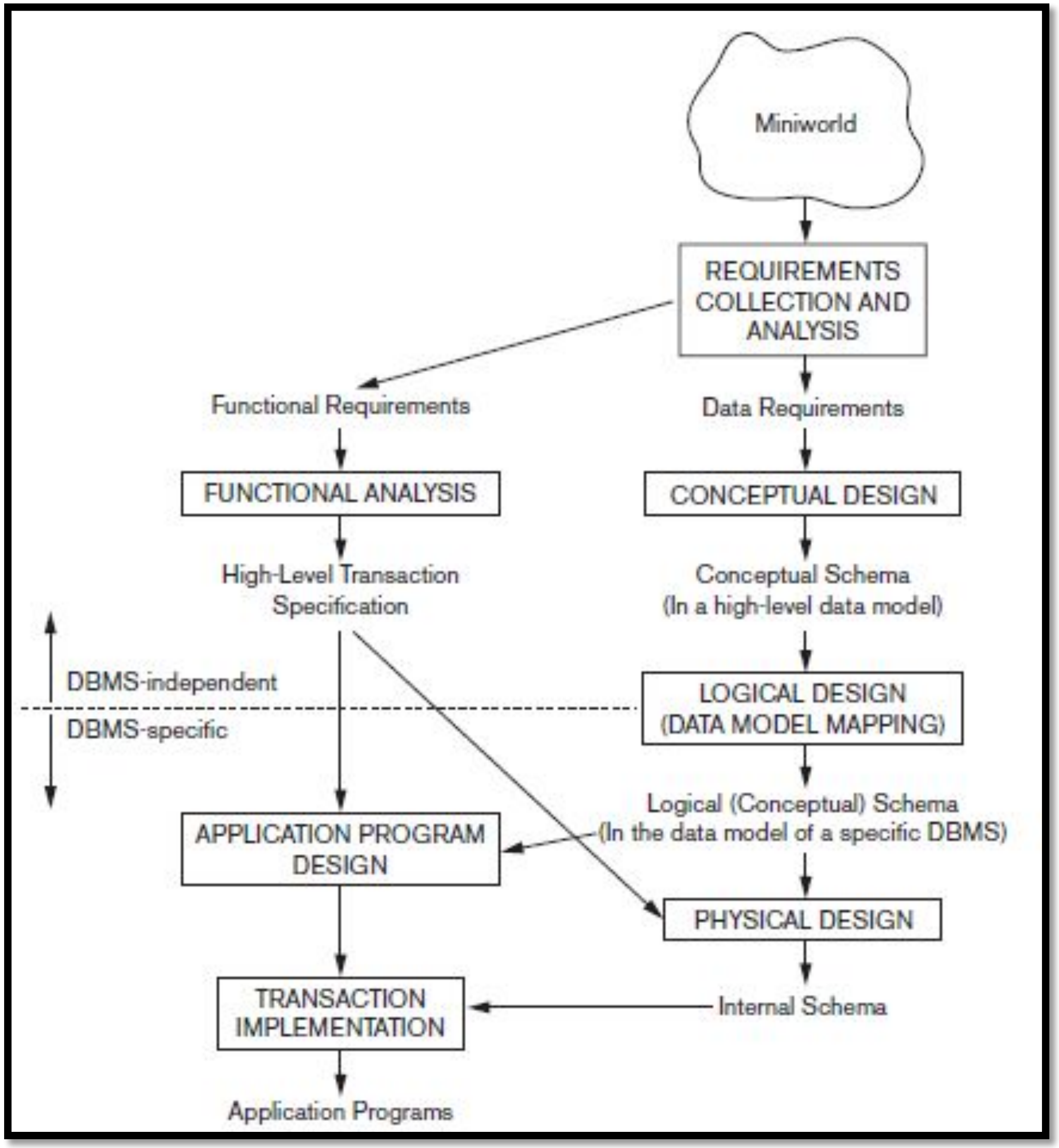
Chapter - 2
Data Modeling Using the Entity–Relationship (ER) Model

Conceptual modeling is a very important phase in designing a successful database application.

Using High-Level Conceptual Data Models for Database Design

A simplified diagram to illustrate the main Phases of Database Design


Miniworld
REQUIREMENTS COLLECTION AND ANALYSIS
Functional Requirements
Data Requirements
FUNCTIONAL ANALYSIS
CONCEPTUAL DESIGN
High-Level Transaction Specification
Conceptual Schema (In a high-level data model)
DBMS-independent
DBMS-specific
LOGICAL DESIGN (DATA MODEL MAPPING)
Logical (Conceptual) Schema (In the data model of a specific DBMS)
APPLICATION PROGRAM DESIGN
PHYSICAL DESIGN
TRANSACTION IMPLEMENTATION
Internal Schema
Application Programs

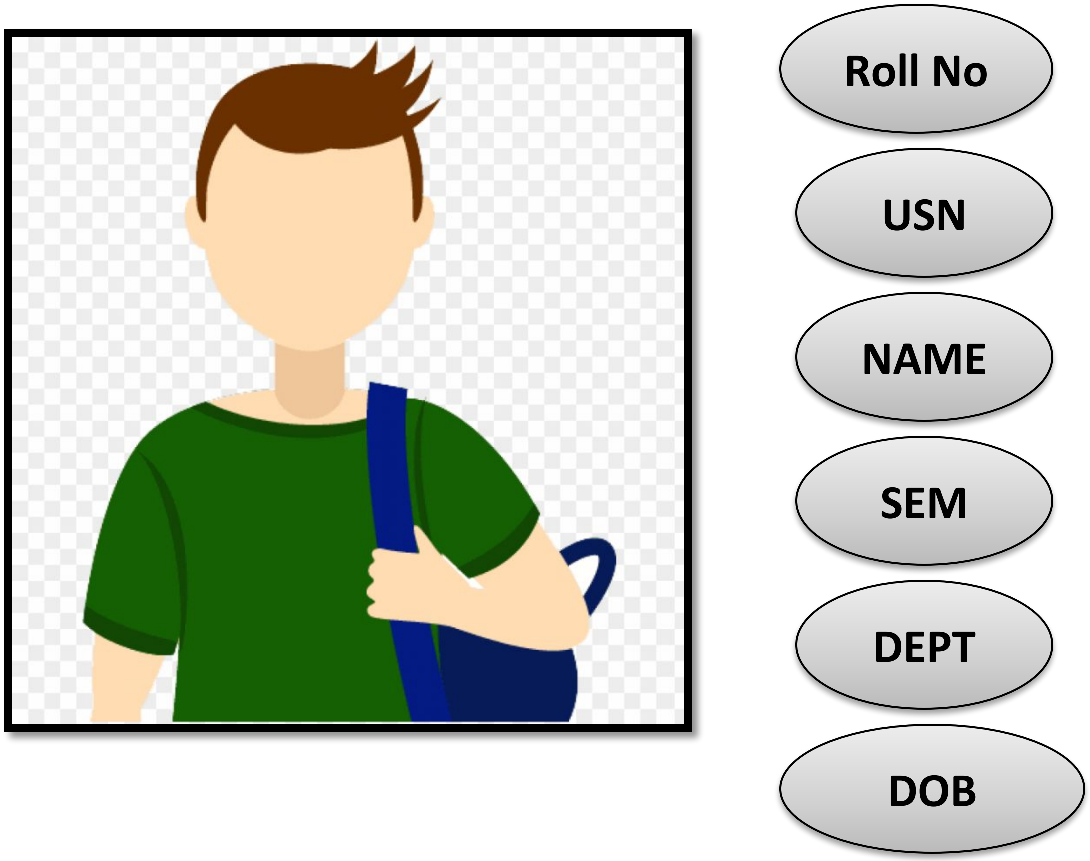
1. The first step is **requirements collection and analysis** in this step, the database designers interview prospective database users to understand and document their data requirements along with **functional requirements** of the application, which consist of operations(or transactions). In software design it is common to use data flow, sequence diagrams to specify these requirements.
2. The second step is to create a **conceptual schema for the database, using a high-level conceptual data model. It consists of** detailed descriptions of the **entity types**, **relationships**, and **constraints**. These are easier to understand and can be used to communicate with nontechnical users.
3. The third step in **database design** is the actual implementation of the database, using a commercial DBMS such as the **relational (SQL) model**. the conceptual schema is transformed from the high-level data model into the implementation data model. This step is called **logical design or data model mapping.**
4. The last step is the **physical design phase, during which the internal storage structures,** file organizations, indexes, access paths, and physical design parameters for the database files are specified.
5. In parallel with these activities, **application programs are designed** and implemented **as database transactions** corresponding to the high-level transaction specifications.

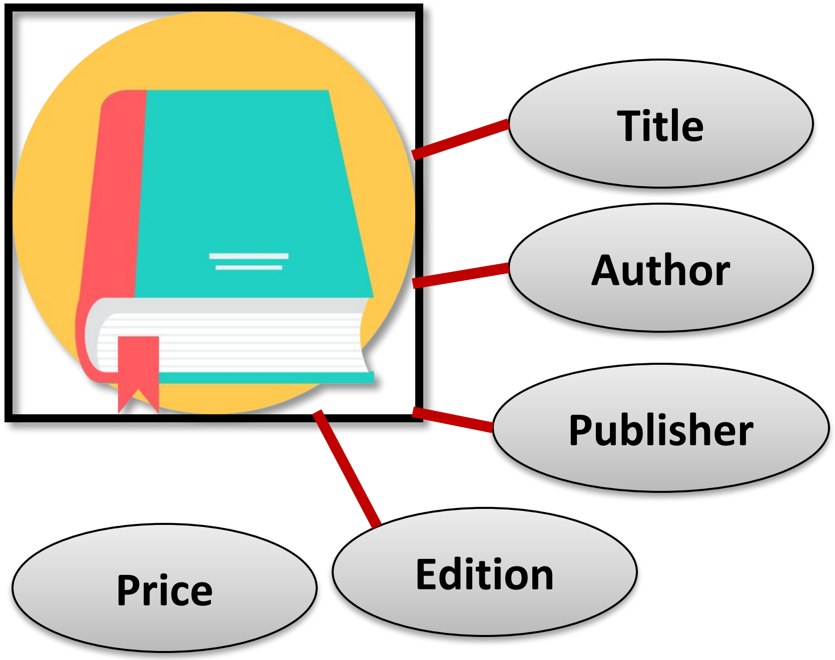
Entity Types, Entity Sets, Attributes, and Keys

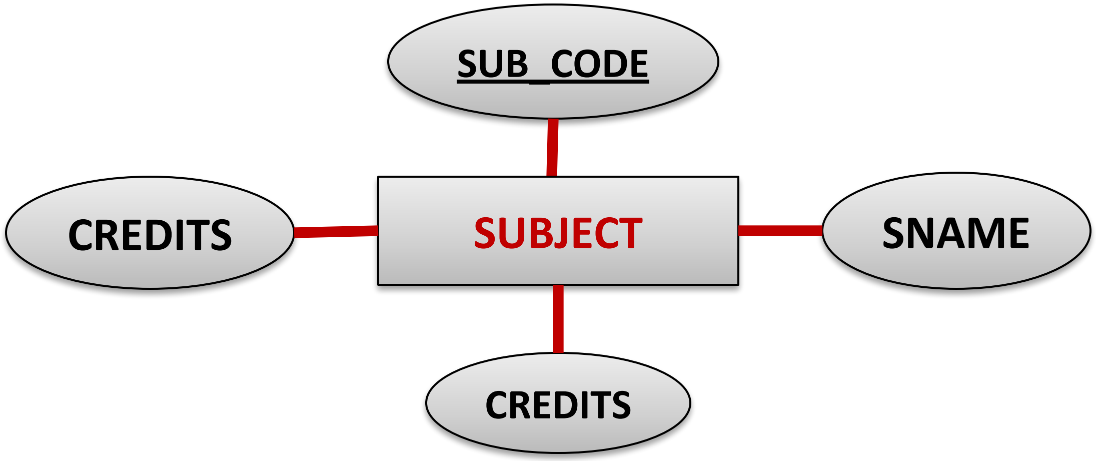
Entity : is an *object in the real world with an independent existence.* It may have physical existence for example, a particular person, car, house, or employee Or it may have a conceptual existence for instance, a company, a job, or a subject.

Attributes : property or characteristic that describes an entity.

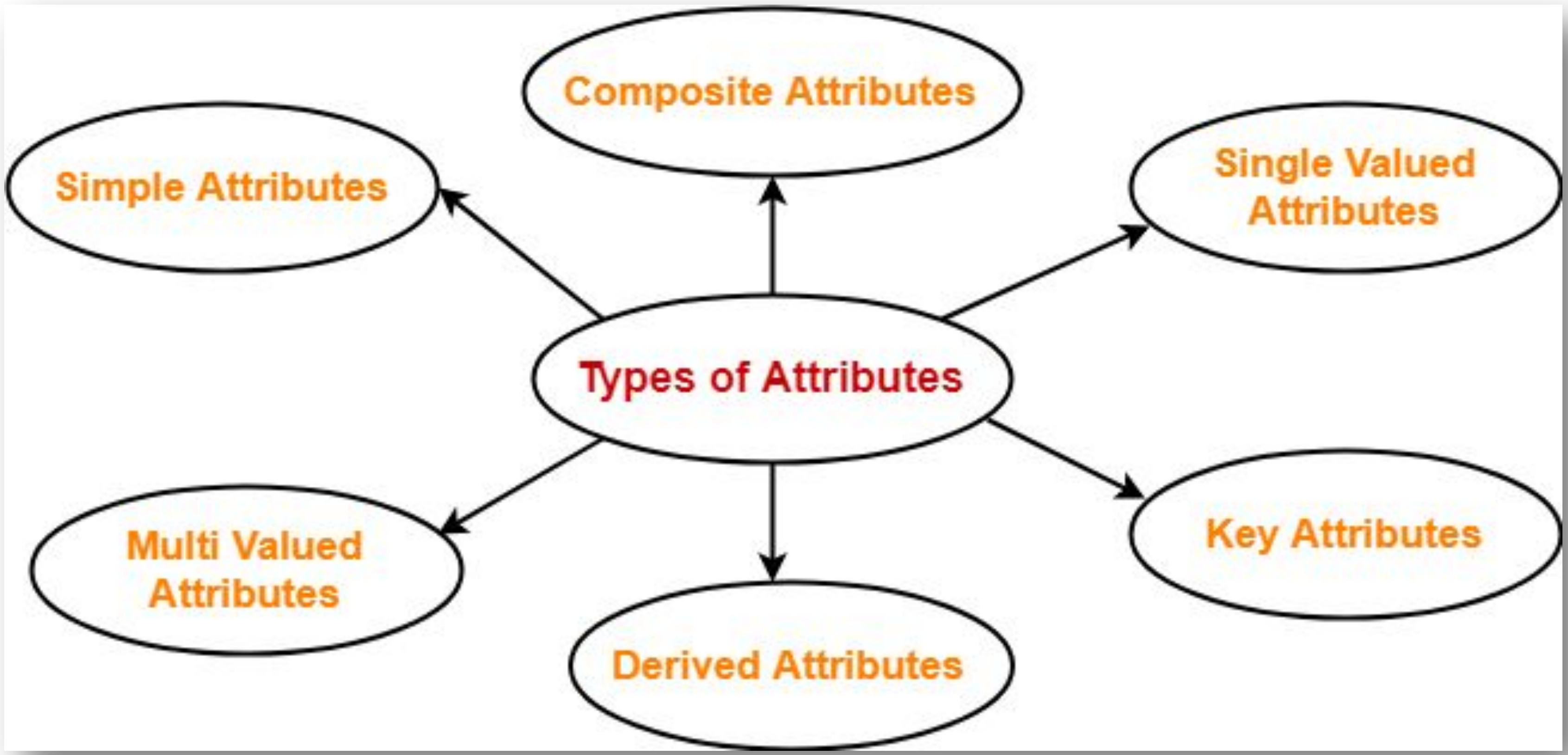
Entity **Attributes**




Title
Author
Publisher
Edition
Price

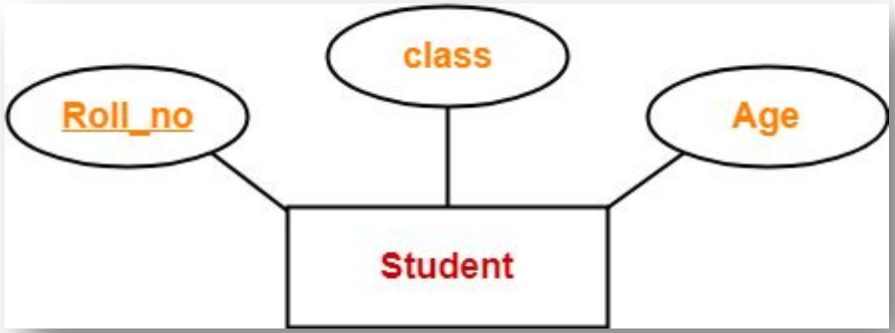

SUB_CODE
CREDITS
SUBJECT
SNAME
CREDITS

Different Types of Attributes in ER Model

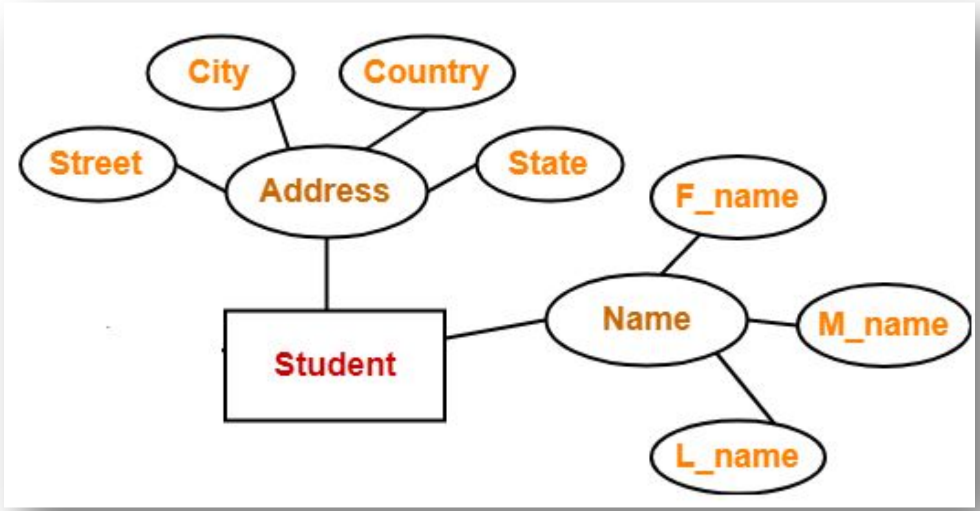


Different Types of Attributes in ER Model

1. **Simple Attribute :** Simple attributes are those attributes which can not be divided into sub parts. They are also known as atomic attributes.

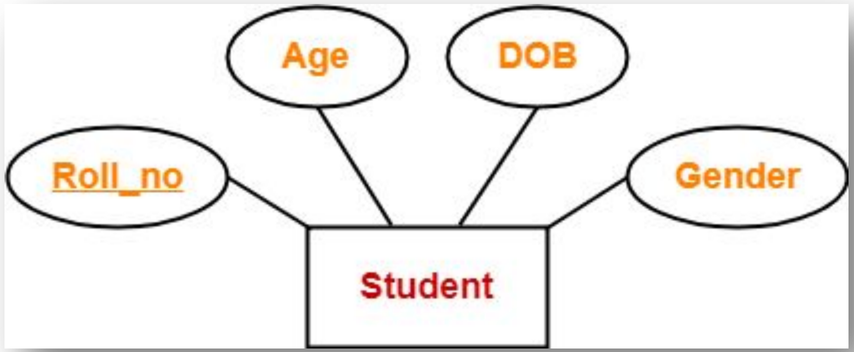


2. **Composite Attributes :** Composite attributes are those attributes which can divided into smaller sub-parts. The value of a composite attribute is the concatenation of the values of its component simple attributes.

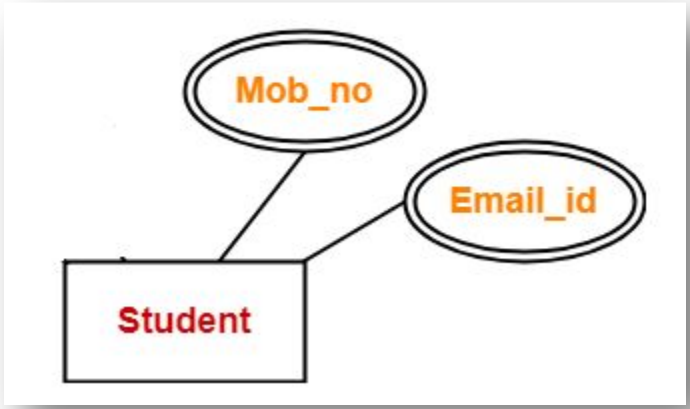


Different Types of Attributes in ER Model

3. **Single Valued :** Most attributes have a single value for a particular entity; such attributes are called single-valued.

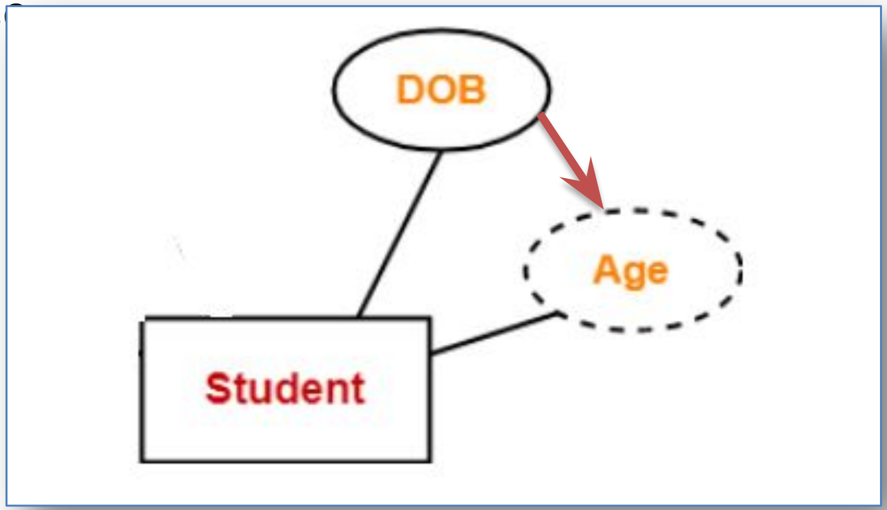


4. **Multi-Valued :** Multi valued attributes are those attributes which can take more than one value for a given entity from an entity set.



Different Types of Attributes in ER Model

5. **Stored attribute** : Stored attributes are the attributes that exist in the physical databas



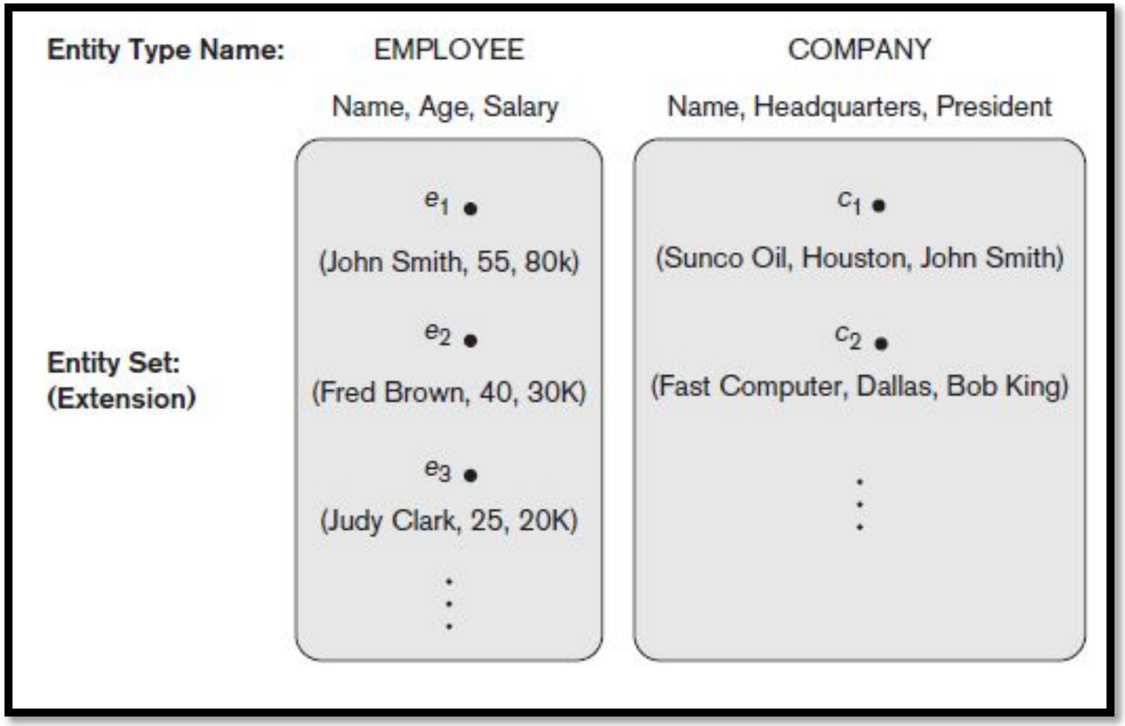
6. **Derived attribute** : Derived attributes are the attributes that do not exist in the physical database, but their values are derived from other attributes present in the database.
7. **Complex Attributes :** is formed by nesting composite attributes and multi-valued attributes in arbitrary way.

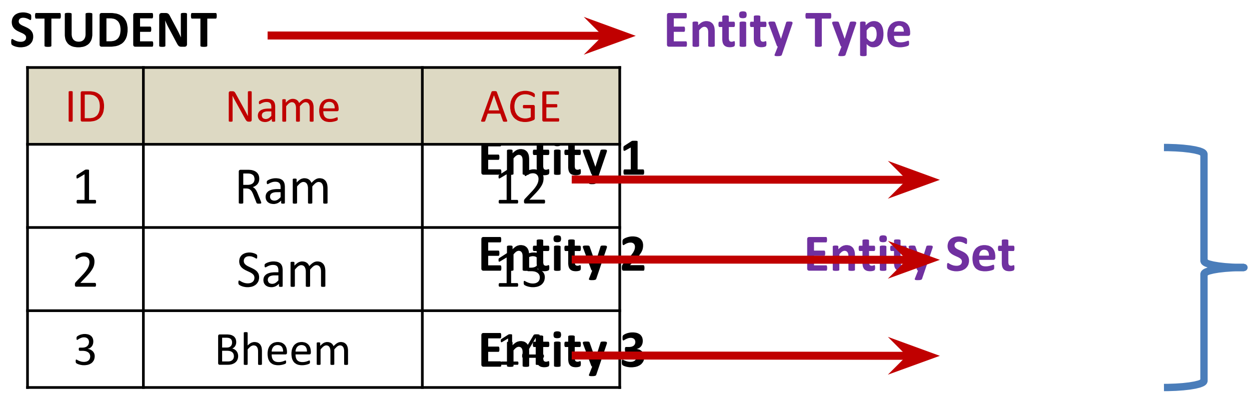
   For example: A person can have many phone numbers, many e-mail addresses, home addresses etc. So, if there is a attribute in the name of 'Contact detail',it can be a complex attribute.

Entity Types, Entity Sets, Keys, and Value Sets

Entity Types : an entity type defines a *collection (or set) of entities that have the same attributes. Each* entity type in the database is described by its name and attributes.

Entity Sets : The collection of all entities of a particular entity type in the database at any point in time is called an **entity set or entity collection;** the entity set is usually referred to using the same name as the entity type, even though they are two separate concepts.





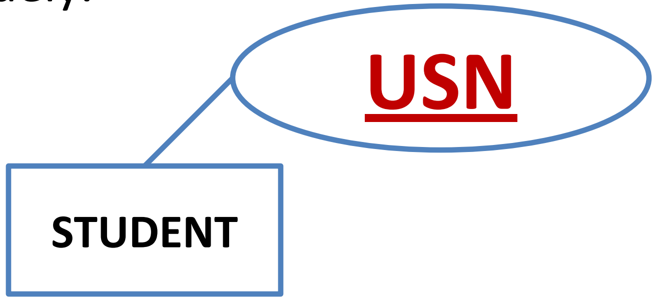
ID	Name	AGE
1	Ram	12
2	Sam	13
3	Bheem	14

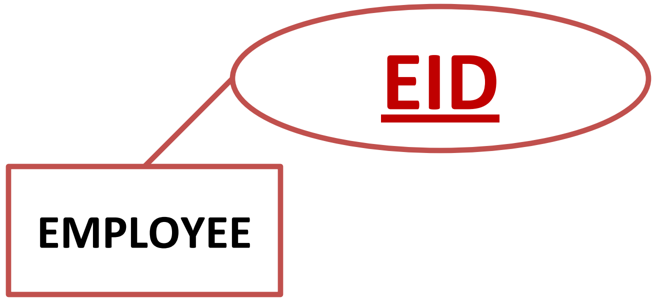
Entity : It is something which has real existence. Like ROW1 contains information about Ram(id, name and Age) which has existence in real world . So the ROW1 is an entity. So we may say **each ROW is an entity**.

Entity Type : It is collection of entity having common attribute. As in Student table each row is an entity and have common attributes. So **STUDENT** is an entity type which contains entities having attributes id, name and Age. Also each entity type in a database is described by a name and a list of attribute. So we may say a table is an entity type.

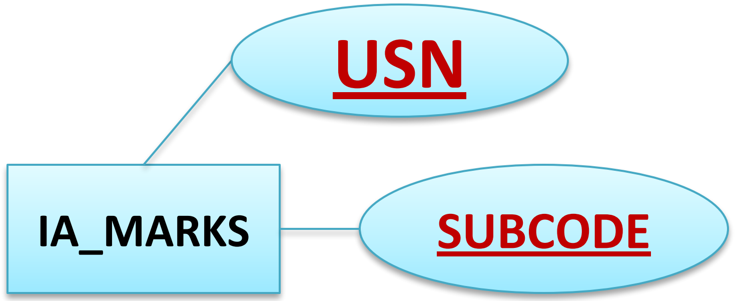
Entity Set : It is a set of entities of same entity type. so a set of one or more entities of Student Entity type is an Entity Set.

Key Attributes : An important constraint on the entities of an entity type is the **key or uniqueness constraint on attributes. An entity type usually** has one or more attributes whose values are distinct for each individual entity in the entity set. Such an attribute is called a **key attribute, and its values can be used to** identify each entity uniquely.



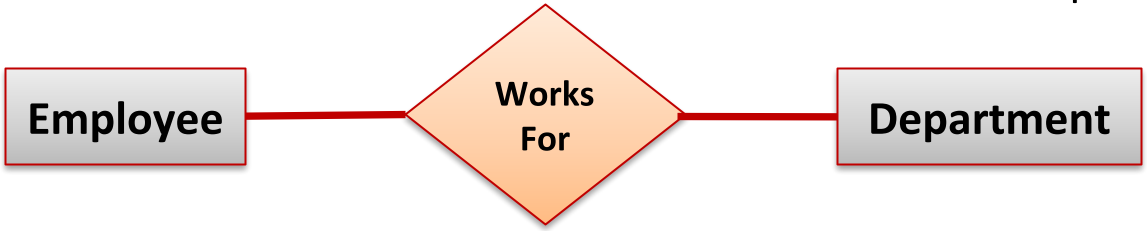


Composite key : Some entity types have *more than one key attribute.*



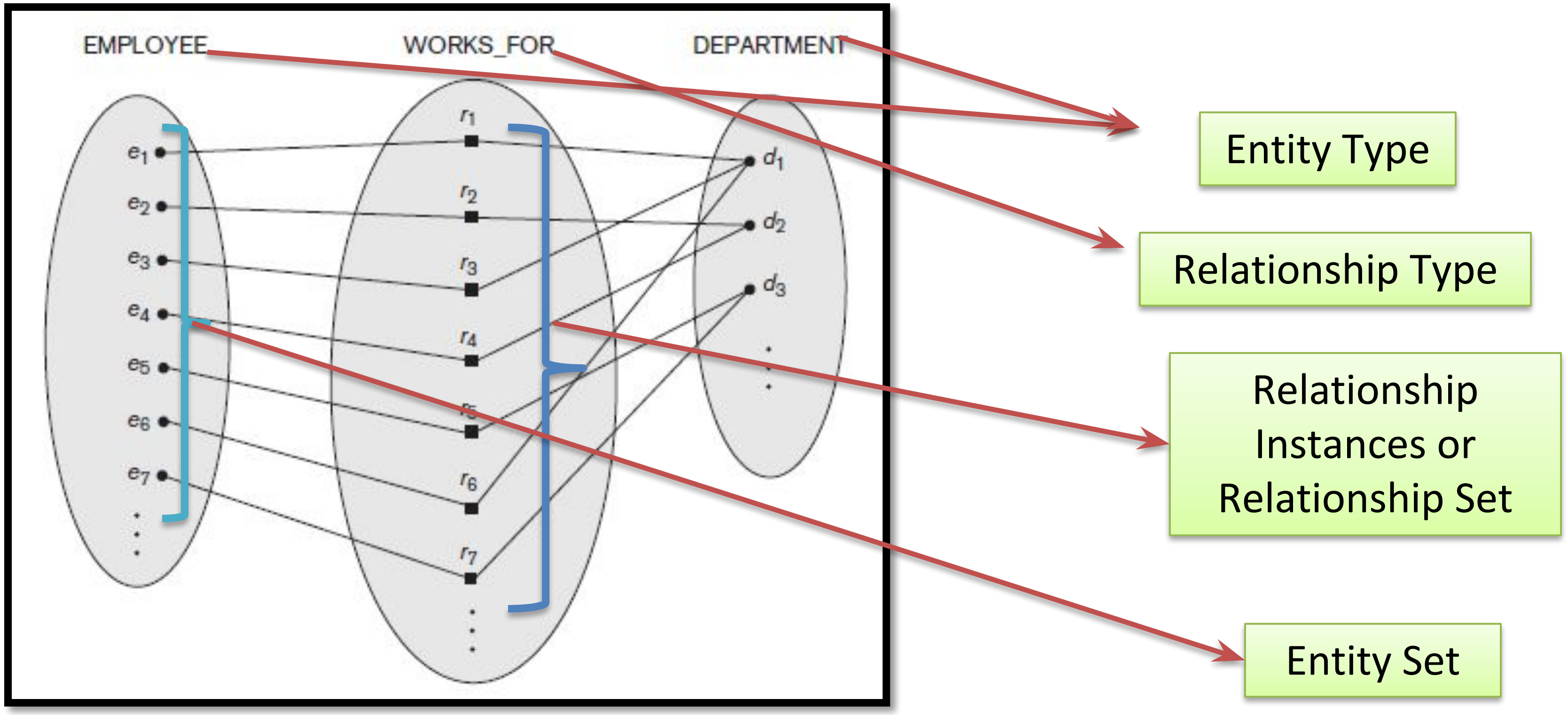
<u>USN</u>	<u>SUB-CODE</u>	IA 1	IA 2	IA 3
1	18CS51	30	40	30
1	18CS52	30	40	30
1	18CS53	30	40	30

Relationship : an association between entities is called a relationship.



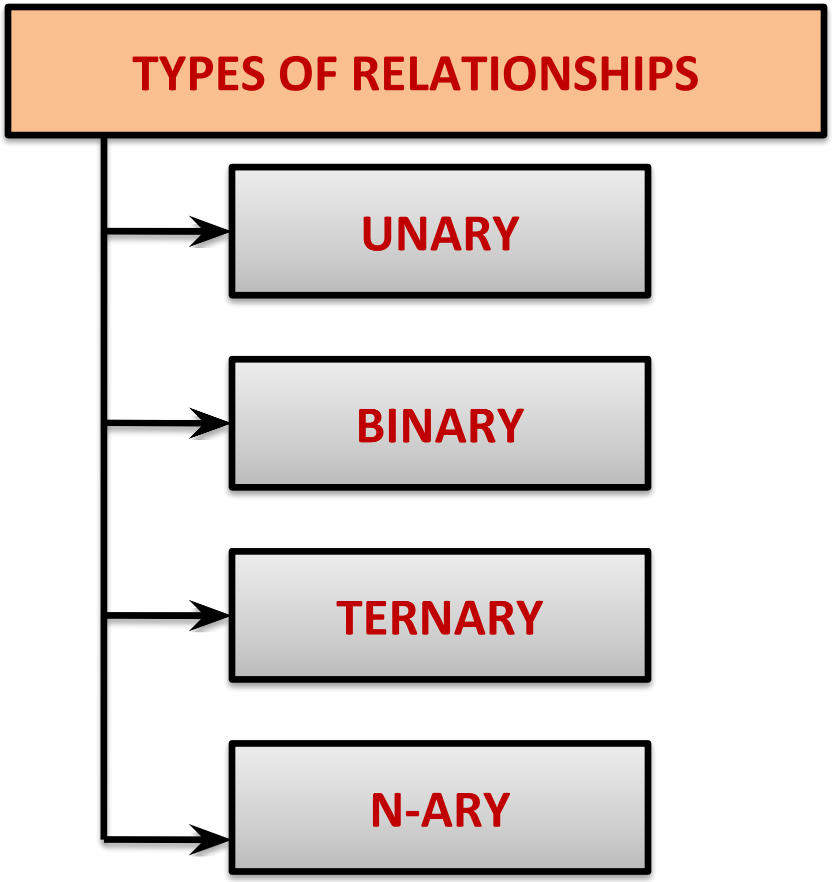
Relationship type : A relationship type *R among n entity types E1, E2, . . . , En* ***defines a set of associations***— or a relationship set—**among entities** from these entity types.

Relationship set : Mathematically, the relationship set *R is a set of* ***relationship instances ri, where each ri associates n*** individual entities (*e1, e2, . . . , en), and each entity ej in ri is a member of entity set Ej,* 1 ≤ *j* ≤ *n.*

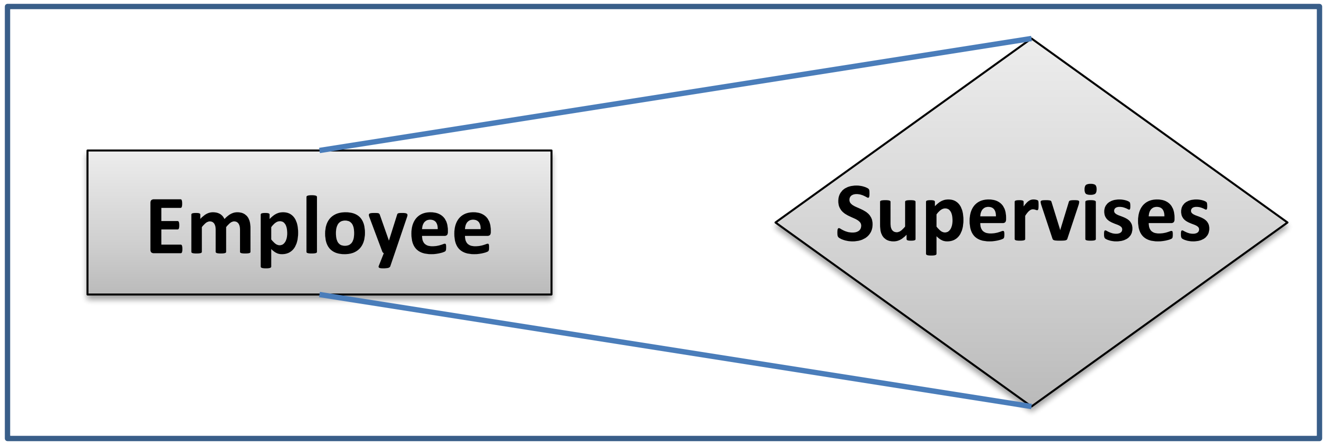


Degree of a Relationship Type

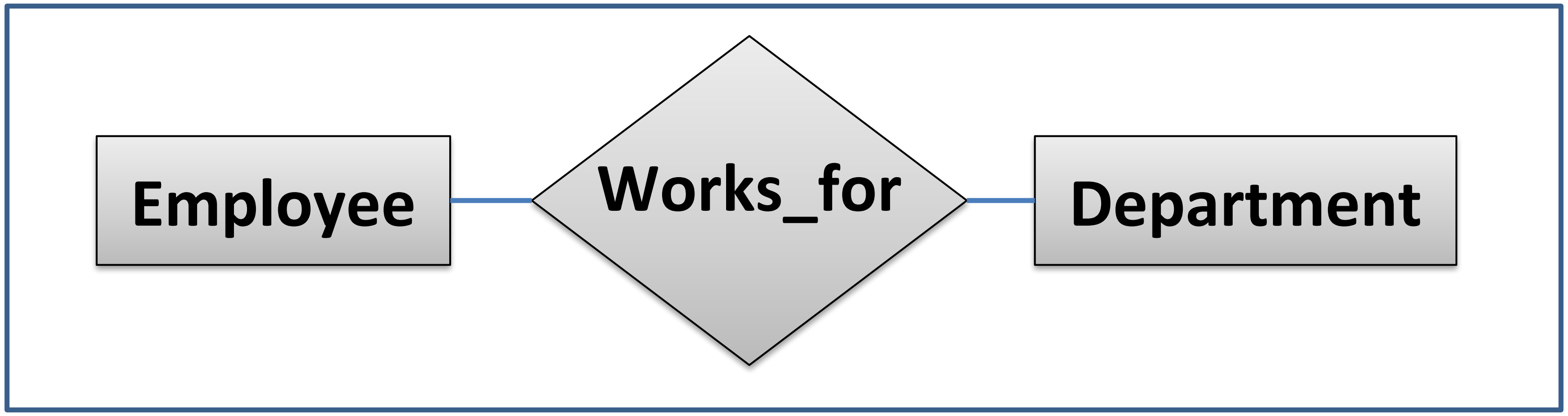
Degree of a Relationship Type : The *degree* of a relationship is the **number of entity types** that **participate** in the **relationship**.

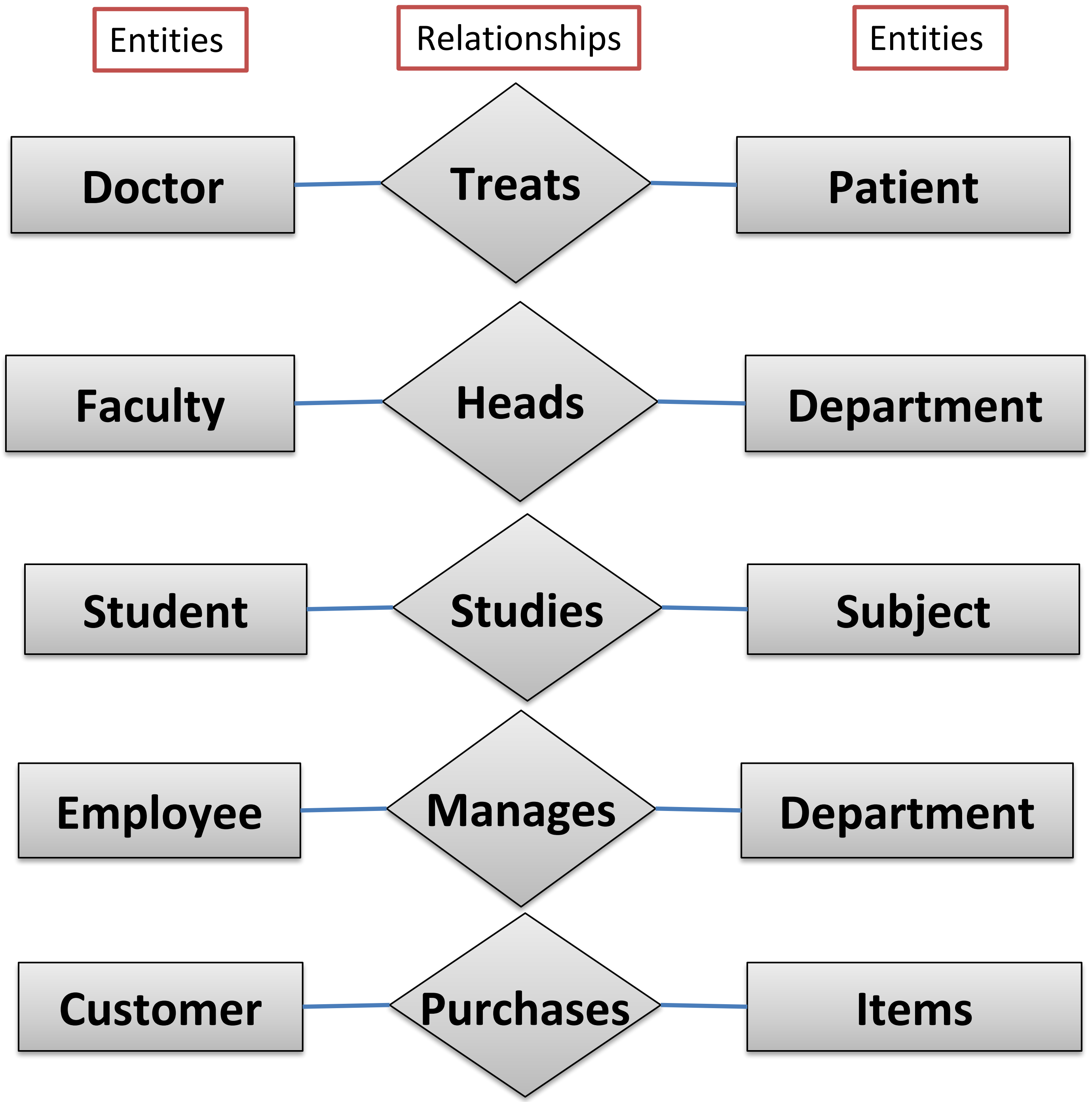


1. **Unary Relationship Type:** A **unary relationship** is when both participants in the relationship is the same entity.



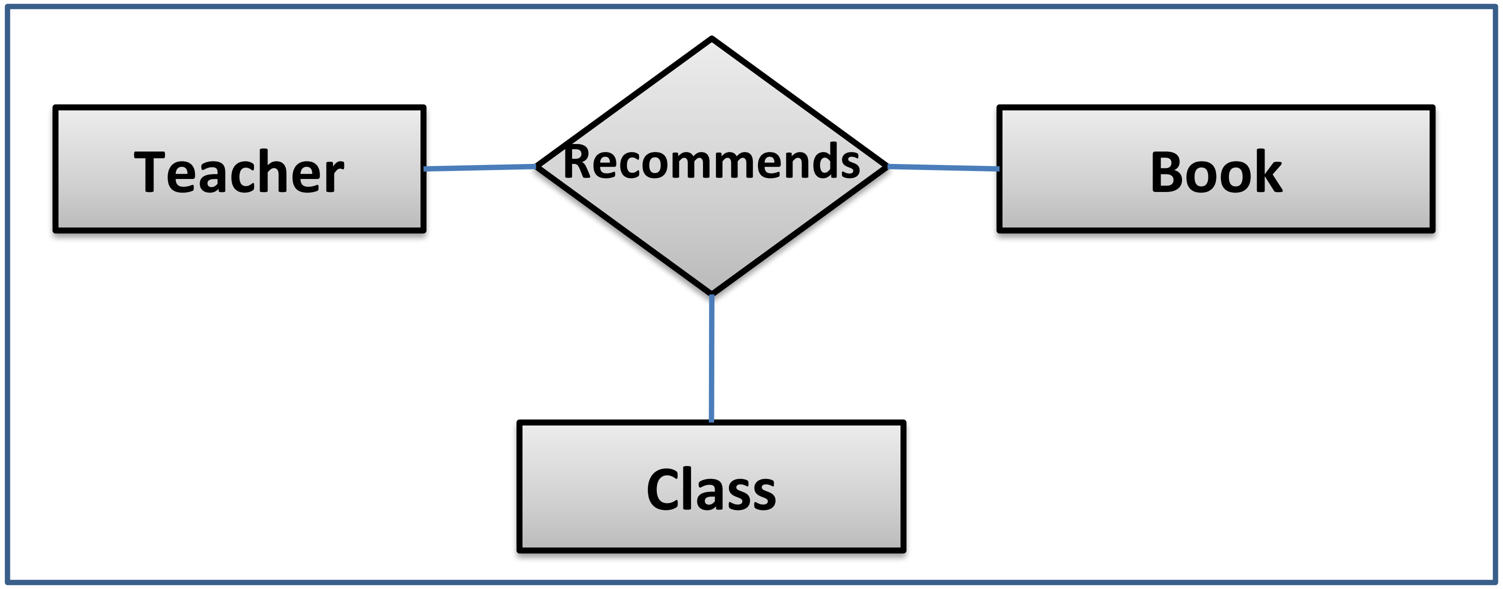
2. **Binary Relationship Type:** A **binary relationship** is when two entities participate, and is the most common relationship degree.

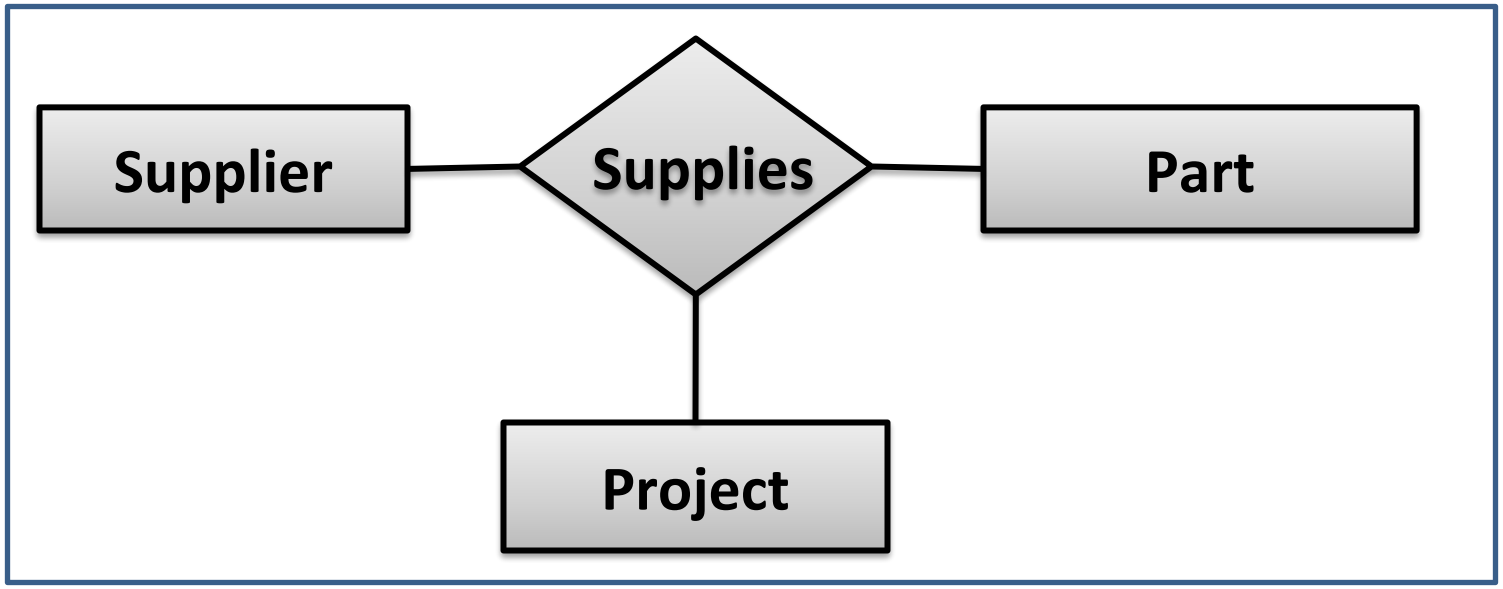




Binary Relationship Type Examples

3. Ternary Relationship Type: A **ternary relationship** is when three entities participate in the relationship.



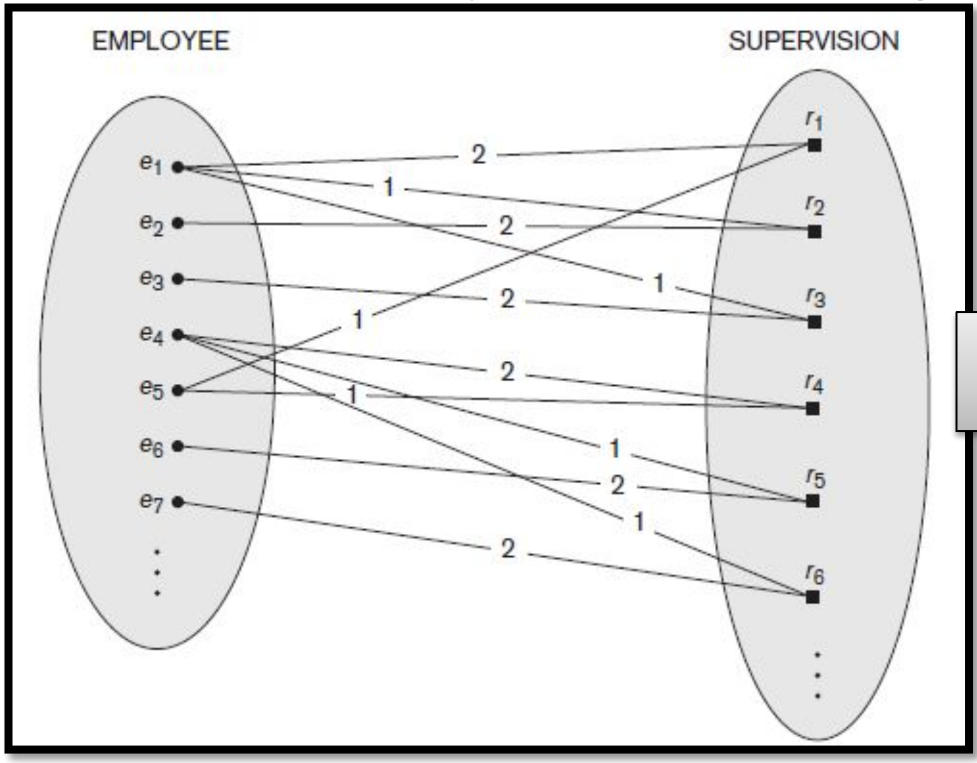


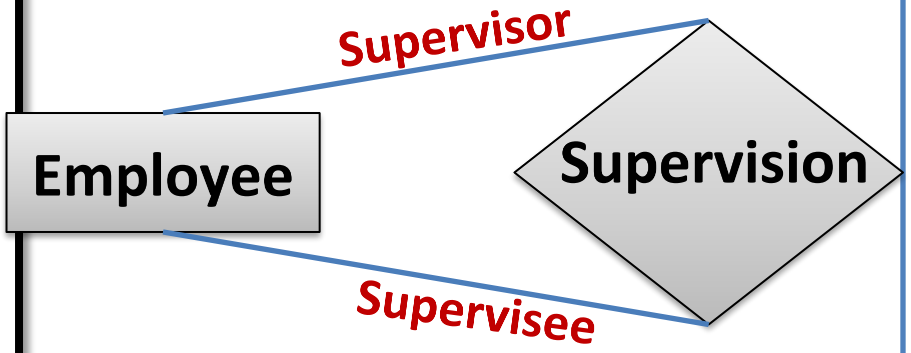
Role Names and Recursive Relationships

- Each entity type that participates in a relationship type plays a particular role in the relationship.

For example, in the WORKS_FOR relationship type, EMPLOYEE plays the role of *employee or worker* and DEPARTMENT plays the role of *department or employer.*

- *Same entity type participating more than* once in a relationship type in *different roles. In such cases the role name becomes* essential for distinguishing the meaning of the role that each participating entity plays. Such relationship types are called **recursive relationships or self-referencing relationships.**





Constraints on Binary Relationship Types

We can distinguish two main types of binary relationship constraints:

1. **Cardinality ratio :** The cardinality ratio for a **binary relationship** specifies the ***maximum number of relationship instances that an entity* can participate in**.

   These are the different Cardinality ratios,

   I. 1:1

   II. 1:N

   III. N:1

   IV. M:N

2. **Participation Constraint:** It specifies the **minimum number of relationship instances that each entity can participate in** and is sometimes called the minimum cardinality constraint.

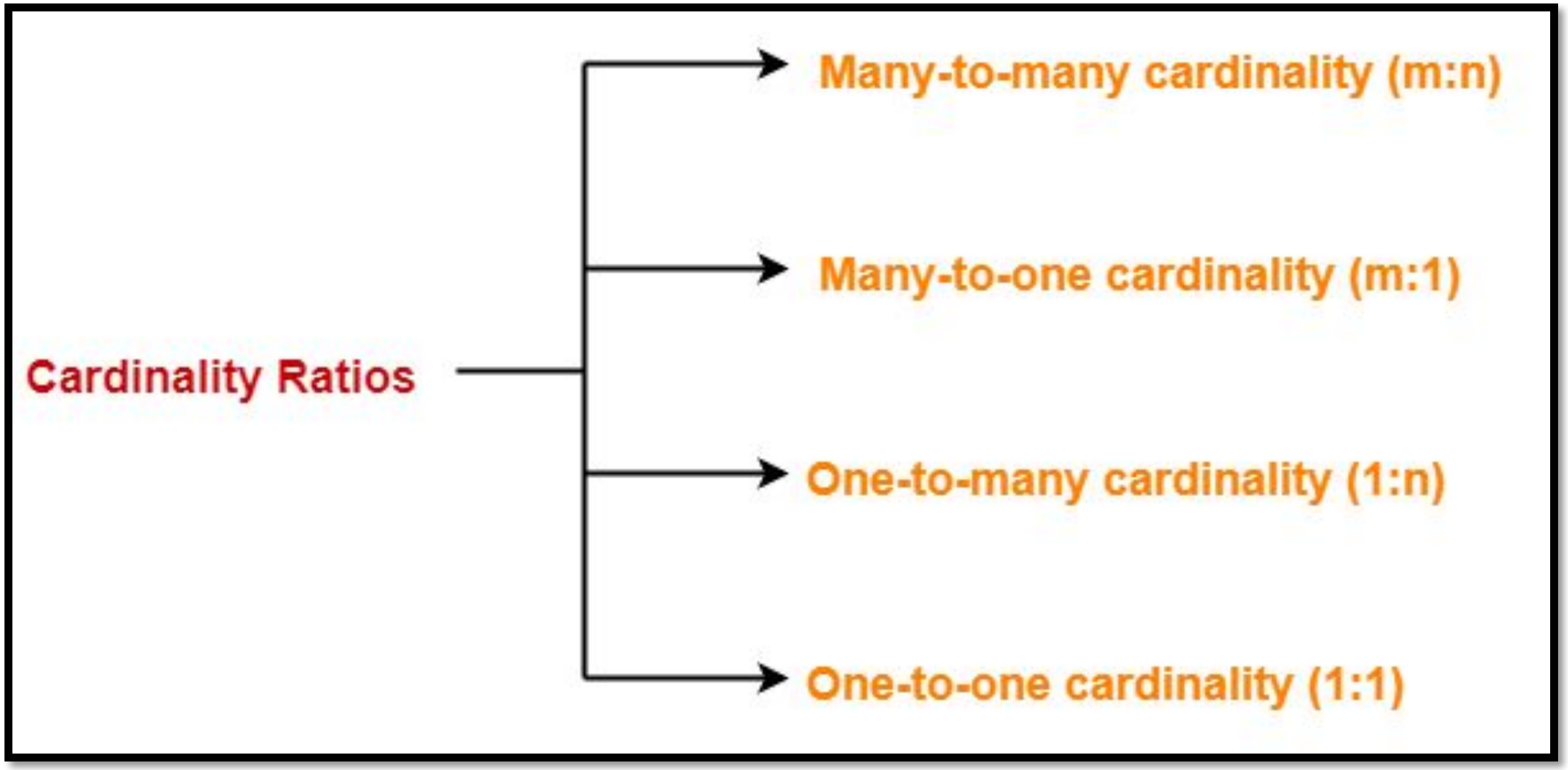
   There are two types of participation constraints

   II. Total Participation

   III. Partial Participation

Constraints on Binary Relationship Types : Cardinality ratio

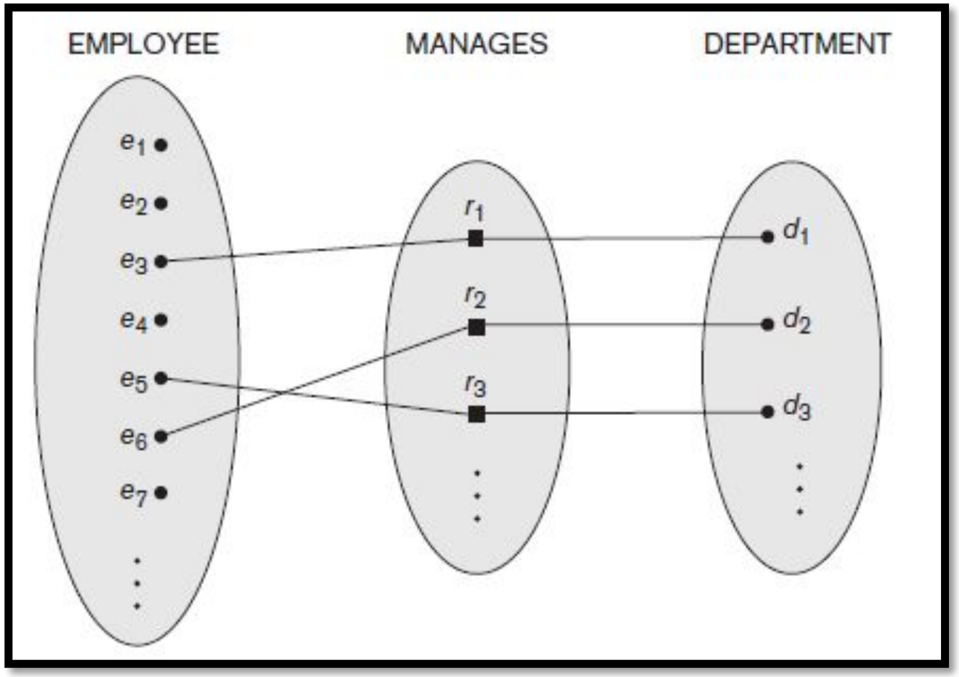
1. **Cardinality ratio:** The cardinality ratio for a **binary relationship** specifies the ***maximum number of relationship instances that an entity*** **can participate in**.
   For example, in the **WORKS_FOR** binary relationship type, **DEPARTMENT:EMPLOYEE** is of cardinality ratio **1:N**, meaning that each department can be related to (that is, employs) any number of employees (N).

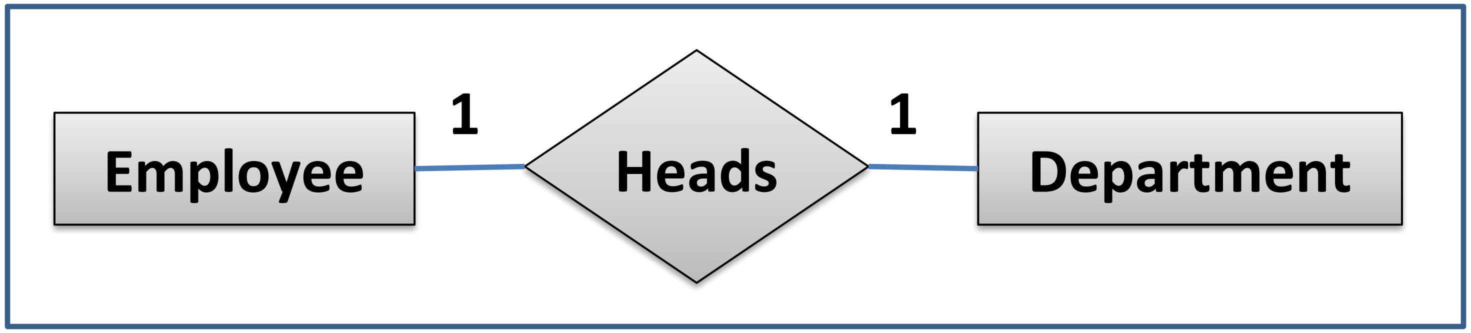

Cardinality Ratios
Many-to-many cardinality (m:n)
Many-to-one cardinality (m:1)
One-to-many cardinality (1:n)
One-to-one cardinality (1:1)

1. One to One Cardinality ratio (1 : 1) :-

When only one instance of an entity is associated in a relationship with another entity is known as 1 : 1 cardinality ratio.

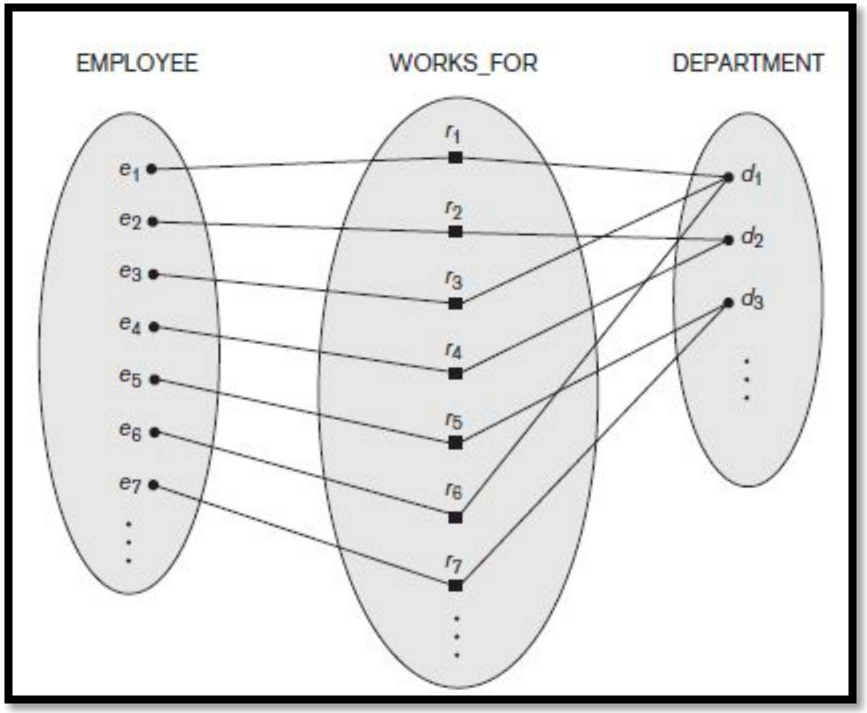
Both Entities in a binary relationship, participating in zero or maximum of one relationship instance is 1 : 1 cardinality ratio.

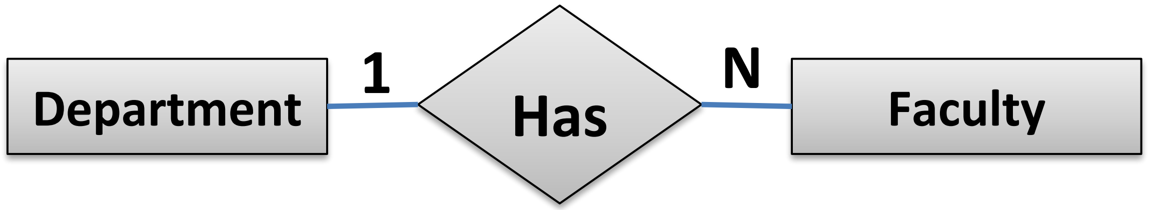


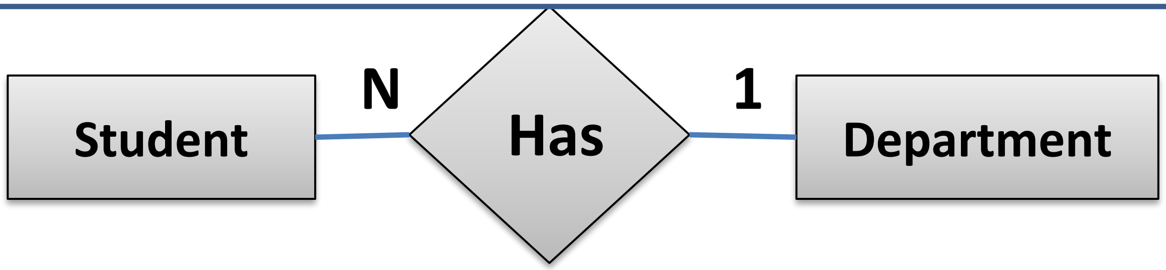


2. Many to One Cardinality ratio (N : 1) / One to Many Cardinality ratio (1 : N)

Either of the two Entities in a binary relationship, One entity participates in more than one relationship instances, and another entity participates in only one relationship instance is 1 : N or N : 1 cardinality ratio.

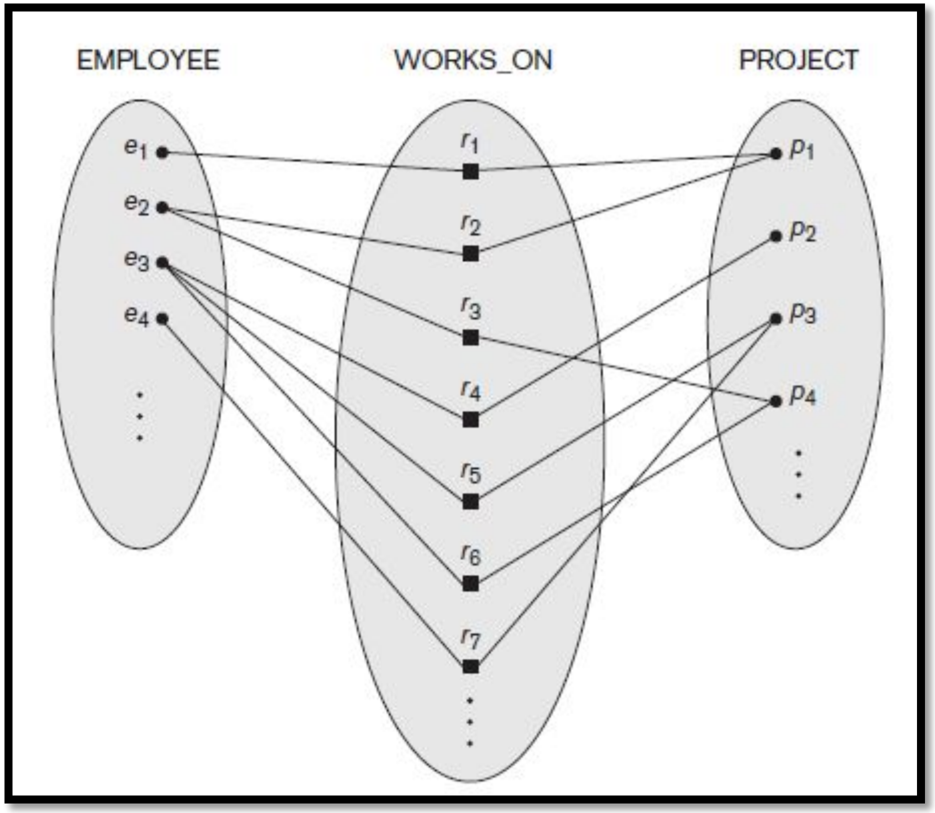


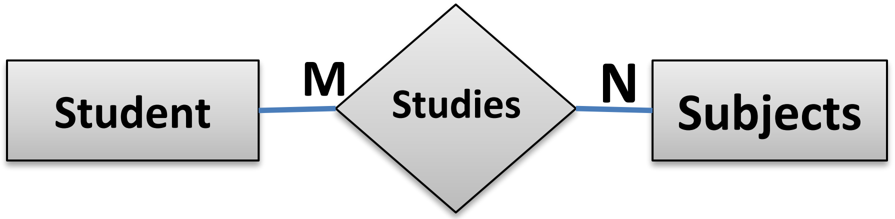


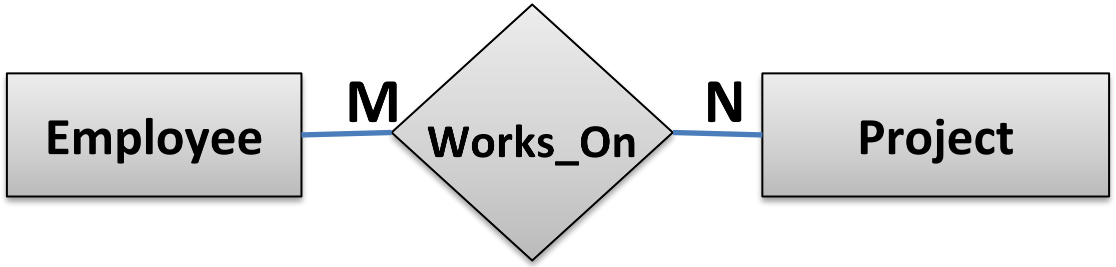


3. Many to Many Cardinality ratio (M : N) :-

Both Entities in a binary relationship, participating in more than one relationship instances M : N cardinality ratio.



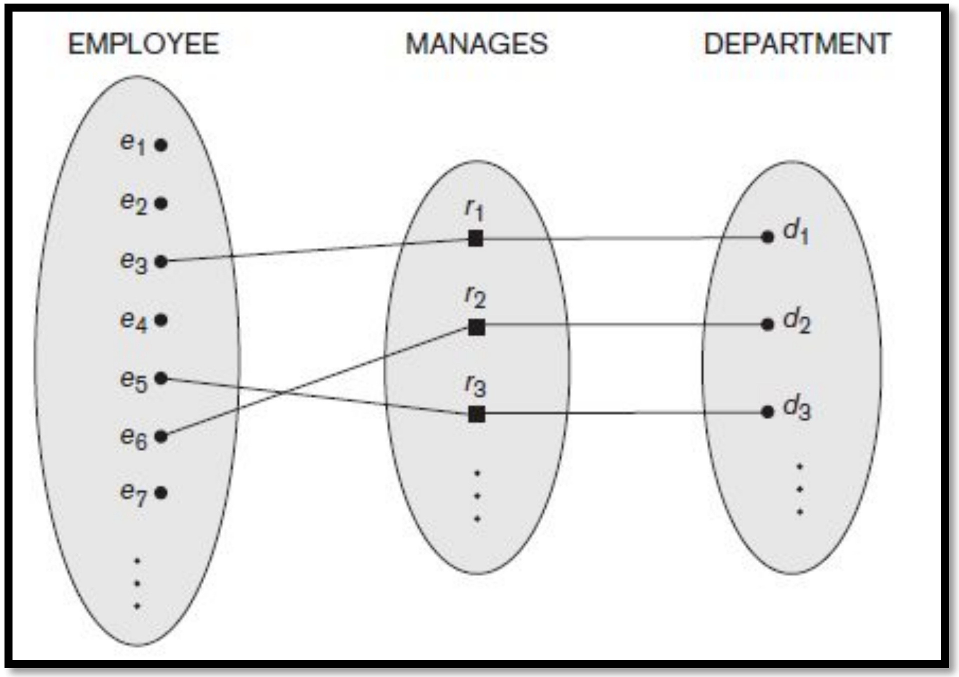


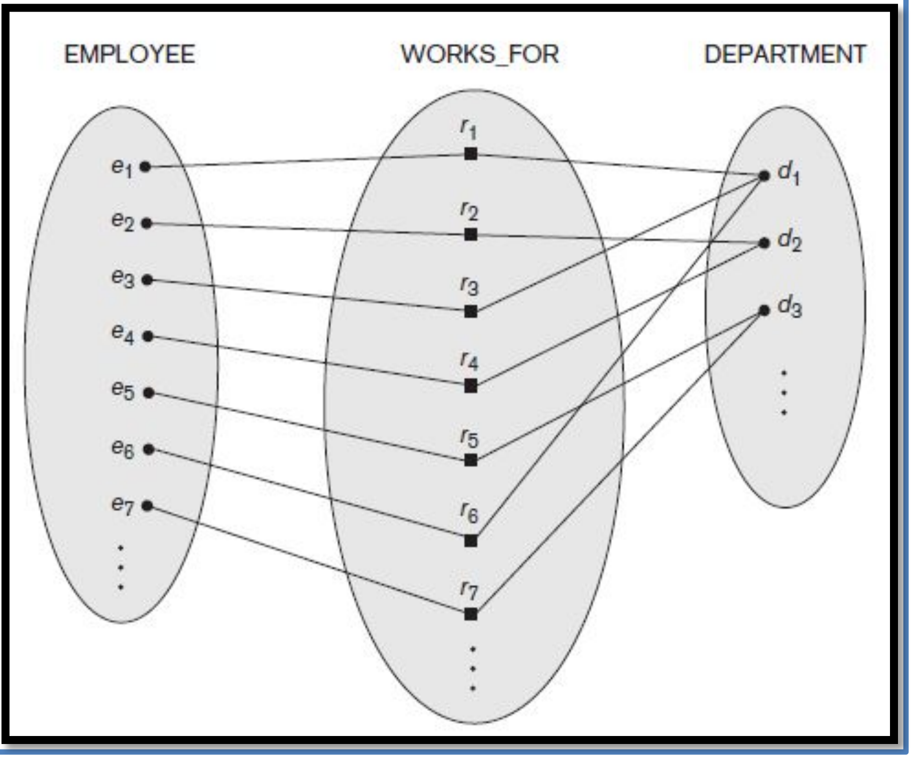


Participation Constraints and Existence Dependencies

The participation constraint specifies whether **the existence of an entity depends on its being related** to **another entity** via the relationship type. This constraint specifies the ***minimum number of relationship instances that each entity can participate in*** and is sometimes called the **minimum cardinality constraint.**

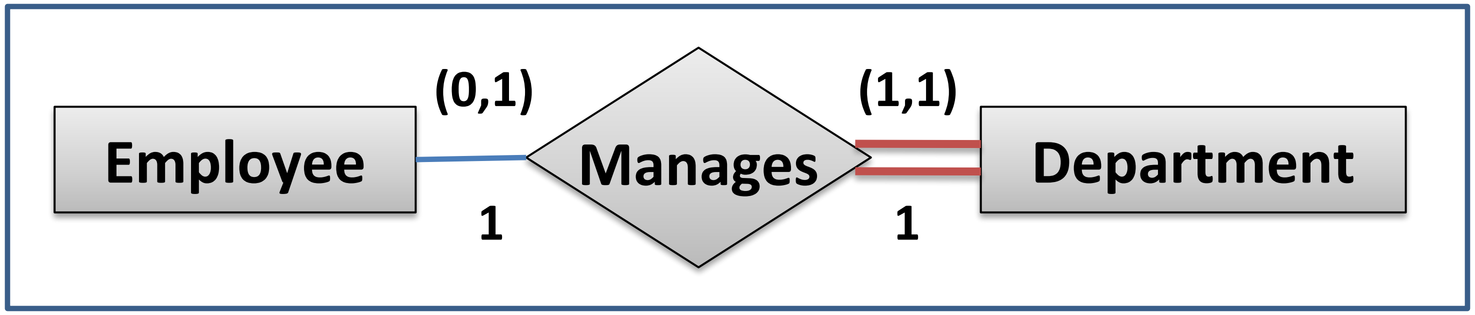
1. **Total Participation Constraint :** also known as existence dependence, every entity should participate in at least one relationship instance for its existence.
2. **Partial Participation Constraint** : an entity may or may not participate in any relationship instance(s)



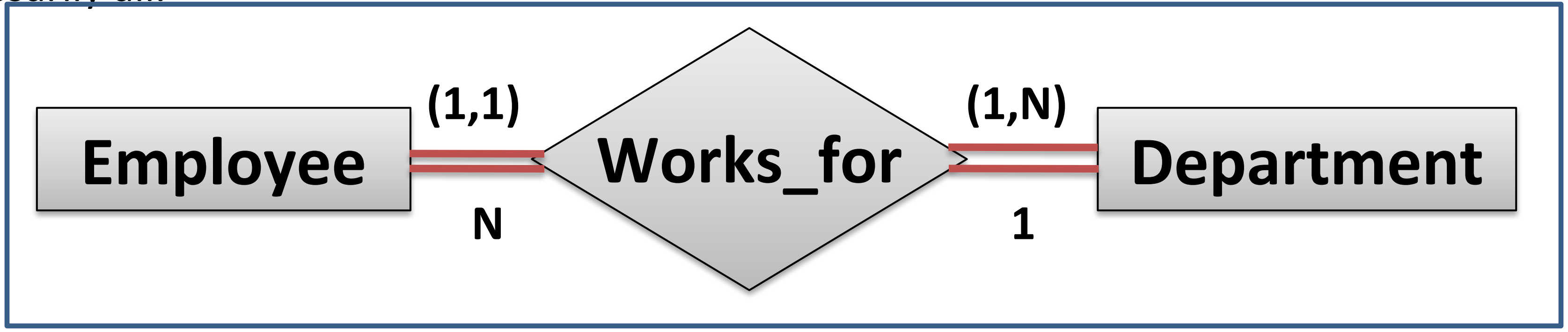


Total Participation is represented by double line in ER Model ═

Partial Participation is represented by single line in ER Model —



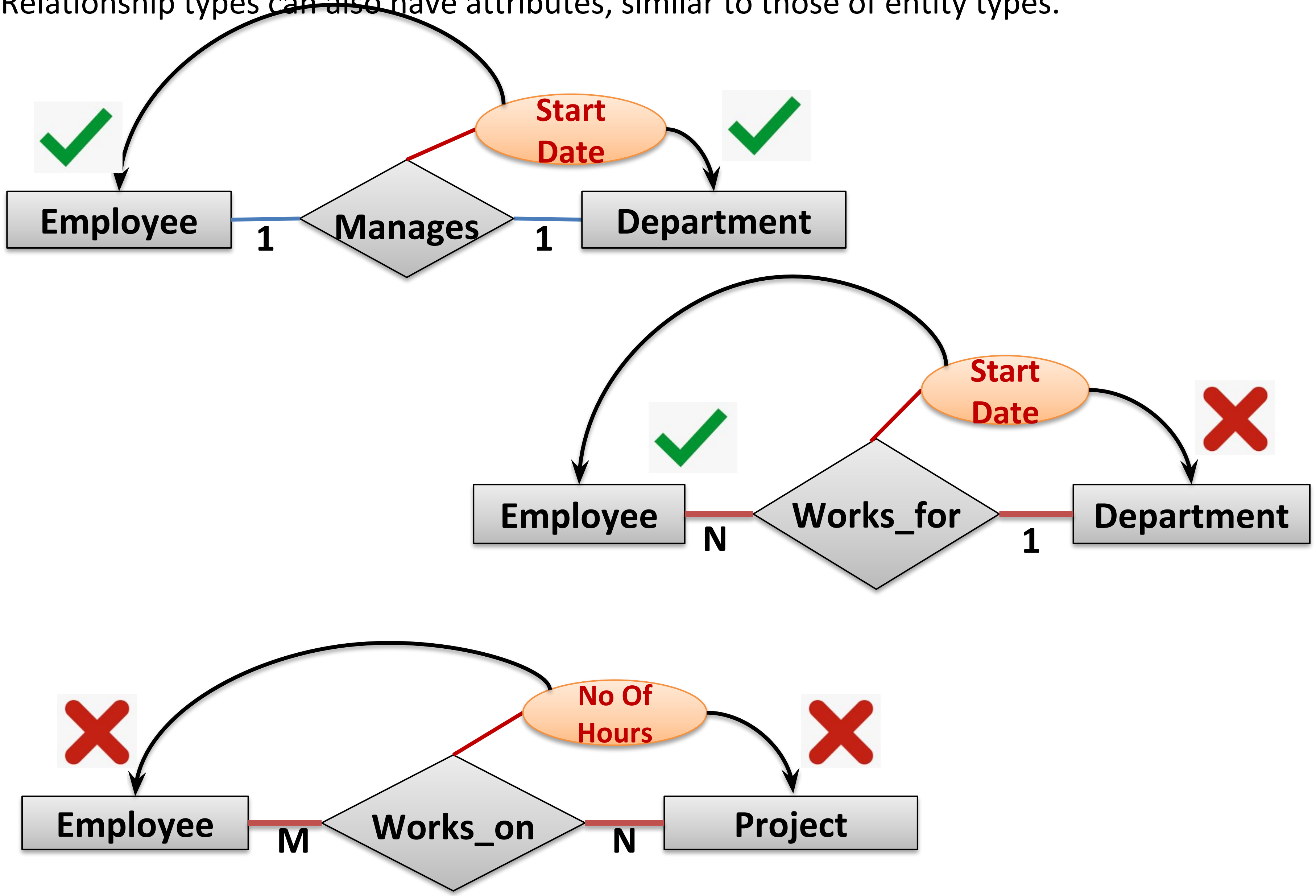
- Every employee is not expected/eligible to manage a department, so the participation of EMPLOYEE in the MANAGES relationship type is **partial, meaning that *some or part of the set of*** employee entities are related to some department entity via MANAGES, but not necessarily all.



- Thus, the participation of EMPLOYEE in WORKS_FOR is called **total participation, meaning that every entity in *the total set of employee*** entities must be related to a department entity via WORKS_FOR.

Attributes of Relationship Types

Relationship types can also have attributes, similar to those of entity types.



Employee

EID	EName	AGE	Salary	Start_Date
1	Ram	28	35000	01-08-2018
2	Sam	29	45000	01-08-2016
3	Bheem	30	30000	01-08-2018

Department

DID	Dname	Intake	Dhead	Start_Date
1	CS	120	Dr. A	01-08-2018
2	ME	120	Dr. B	01-08-2016
3	EC	120	Dr. C	01-08-2018

Project

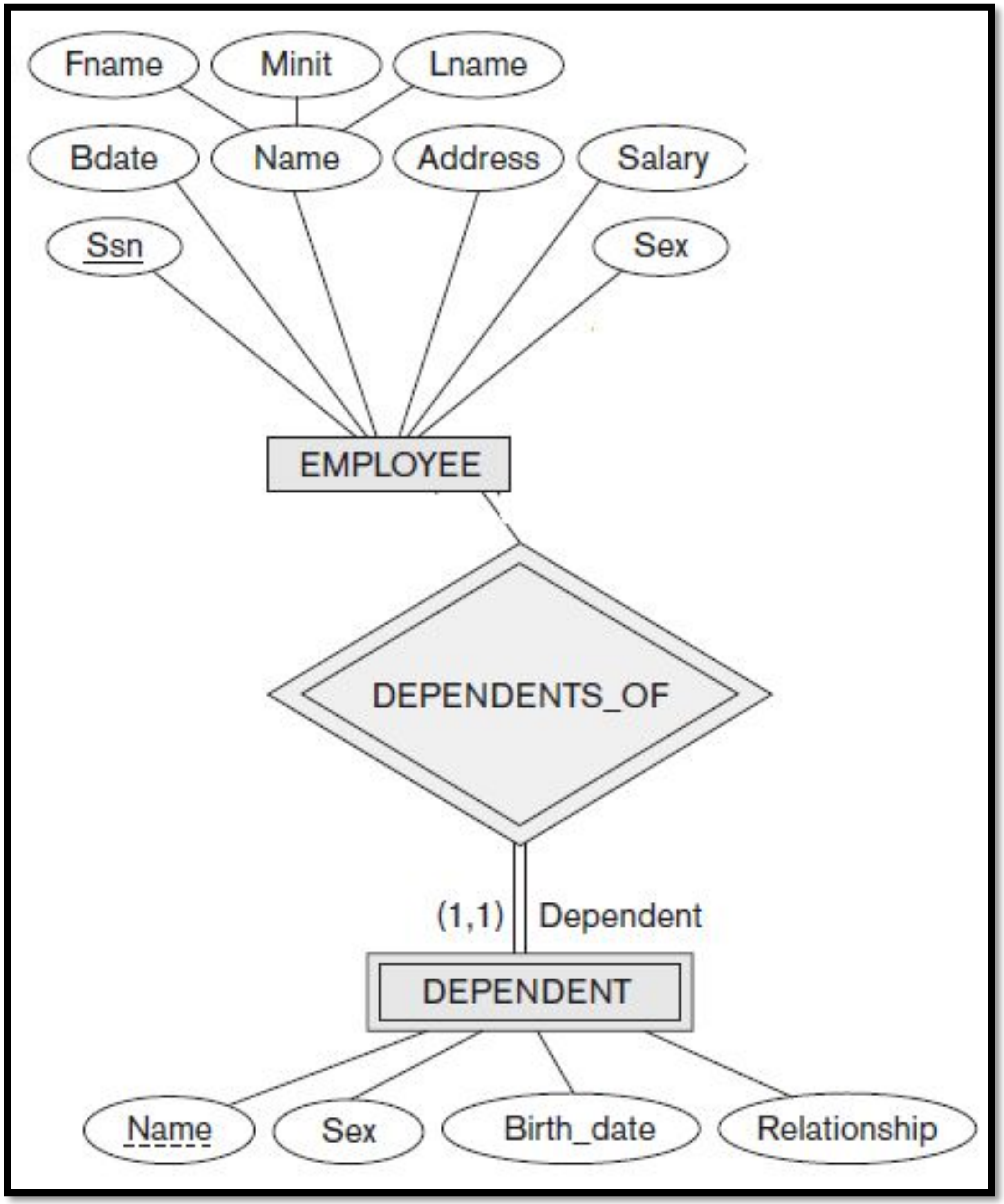
PID	Pname	Plocation
1	X	BGM
2	Y	BGLR
3	Z	DELHI

WorksOn

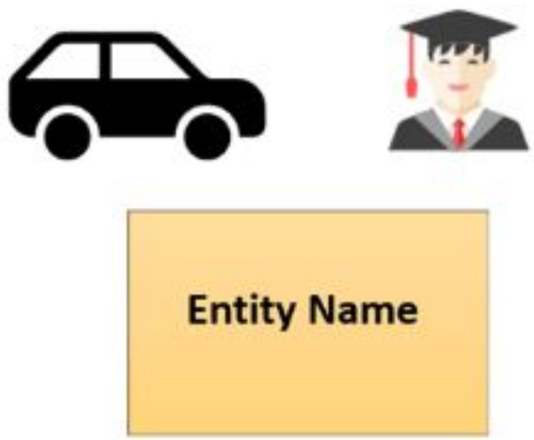
EID	PID	Noofhours
1	1	10
1	2	20
2	1	15

Weak Entity Types

- Entity types that **do not have key attributes** of their own are called **weak entity types.**
- They are identified by being related to specific entities from another entity type in combination with one of their attribute values. We call this other entity type the **identifying or owner entity type** and we call the relationship type that relates a weak entity type to its owner the **identifying relationship** of the weak entity type.
- In ER diagrams, both a weak entity type and its identifying relationship are distinguished by surrounding their **boxes and diamonds with double lines**. The partial key attribute is underlined with a dashed or dotted line.


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
EMPLOYEE
DEPENDENTS_OF
(1,1)
Dependent
DEPENDENT
Name
Sex
Birth_date
Relationship

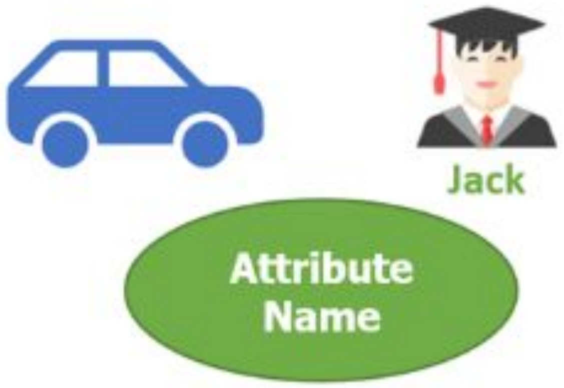
Recall



Entity

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

Example: Car, Student



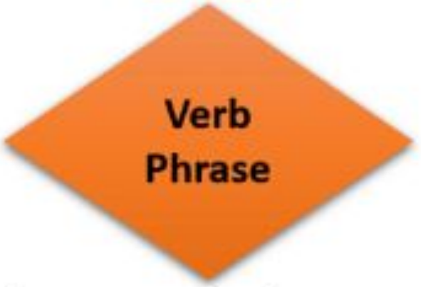
Attribute

Property or characteristic of an entity

Example: Color of car Entity

Name of Student Entity

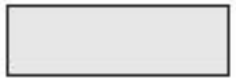
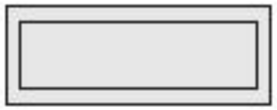
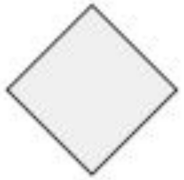
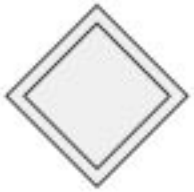
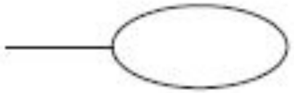
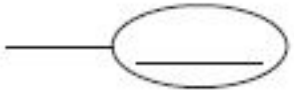
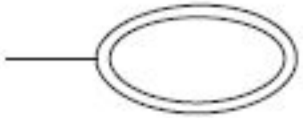
Relation

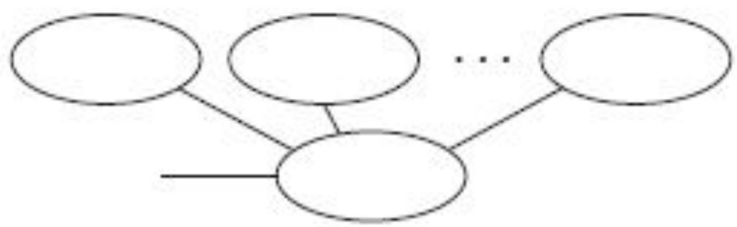


Association between the instances of one or more entity types

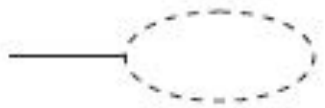
Example: Blue Car Belongs to Student Jack

Symbols Used In ER Diagram

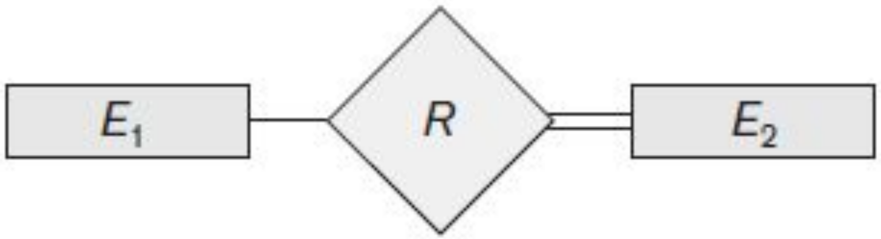
Symbol	Meaning
	Entity
	Weak Entity
	Relationship
	Indentifying Relationship
	Attribute
	Key Attribute
	Multivalued Attribute



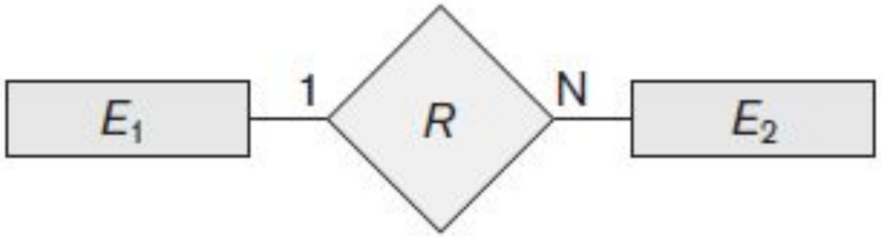
Composite Attribute



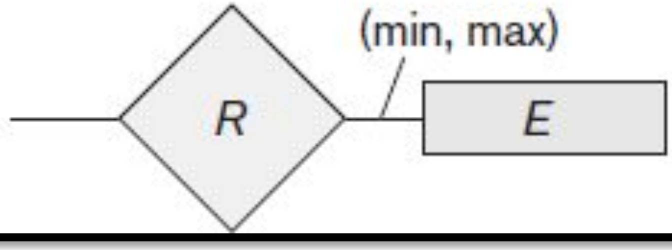
Derived Attribute



Total Participation of E_2 in R



Cardinality Ratio 1: N for $E_1 : E_2$ in R



Structural Constraint (min, max)
on Participation of E in R

Steps Involved in Drawing an ER Diagram

- Identify the Entities
- Identify the Relationship among the entities
- Identify Various Attributes of each entity
- Identify key Attributes
- Identify Cardinality ratio
- Represent Participation Constraints

Design an ER Diagram for a Cricket Tournament

Scenario :

BCCI wants to have a database application to store all the information related to IPL 2022.

Consider a Cricket Tournament “IPL 2022” to be organized by BCCI. In the tournament there are many teams contesting, each having a Teamid, Team_Name, City, a coach. Each team is uniquely identified by using Teamid. A team can have many Players and a captain. Each player is uniquely identified by Playerid, having a Name, and multiple phone numbers, age. A player represents only one team. There are many Stadiums to conduct matches. Each stadium is identified using Stadiumid, having a stadium_name, Address (involves city, area_name, pincode). A team can play many matches. Each match played between the two teams in the scheduled date and time in the predefined Stadium. Each match is identified uniquely by using Matchid. Each match won by any of the one team needs to be recorded in the database. For each match man_of_the match award given to a player.

Draw ER Diagram for Election Database

Scenario :

A country wants to conduct an election for the parliament and wants to have a database application to store all the information generated.

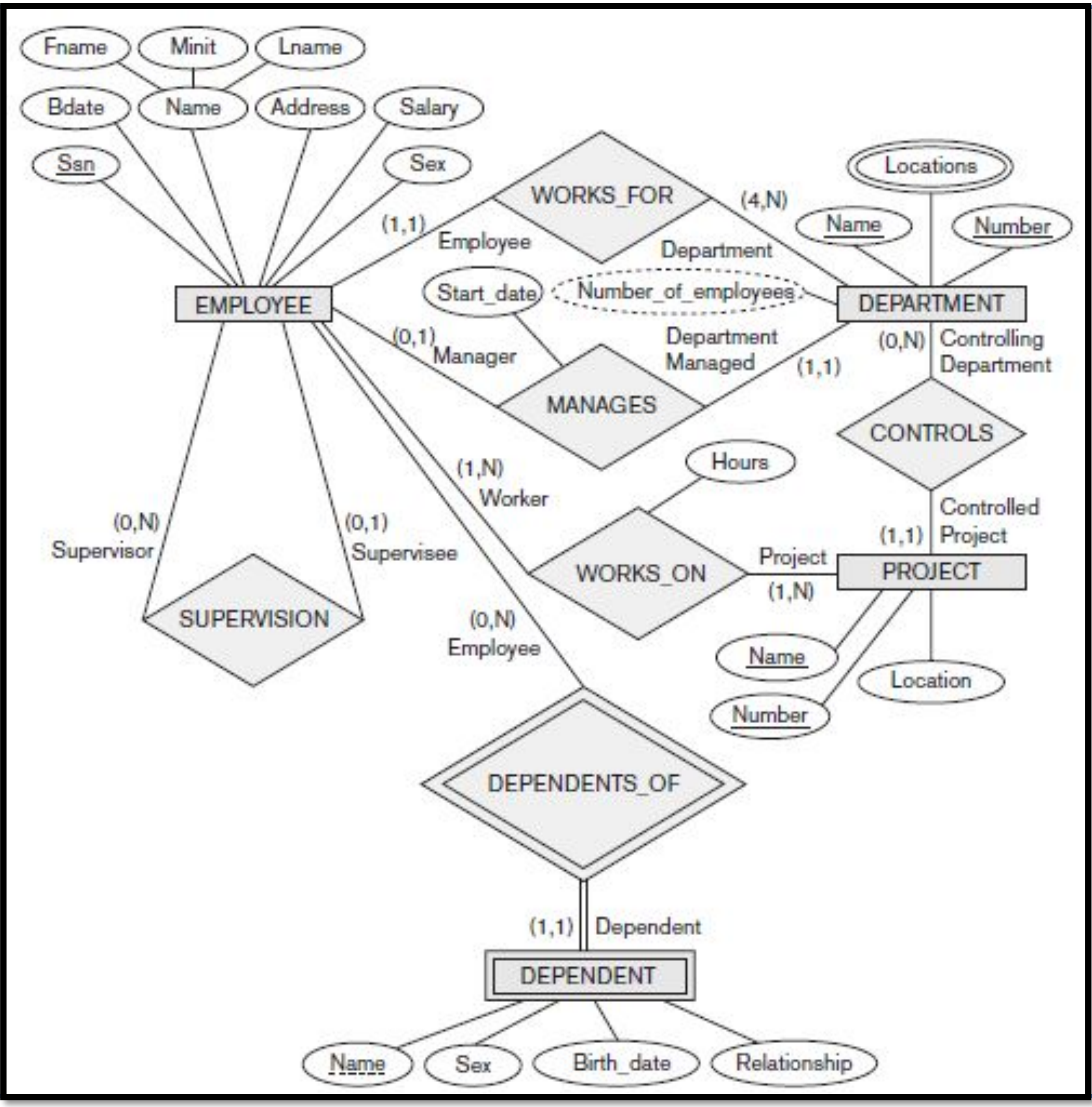
A country having many constituencies. Each constituency is identified uniquely by Constituency_id, having the Name, belongs to a state, Number_of_voters. A constituency can have many voters. Each voter is uniquely identified by using Voter_id, having the Name, age, address (involves Houseno, city, state, pincode). Each voter belongs to only one constituency. There are many candidates contesting in the election. Each candidate is uniquely identified by using candidate_id, having Name, phone_no, age, state. A candidate belongs to only one party. There are many parties. Each party is uniquely identified by using Party_id, having Party_Name, Party_symbol. A candidate can contest from many constituencies under a same party. A party can have many candidates contesting from different constituencies. No constituency having the candidates from the same party. A constituency can have many contesting candidates belongs to different parties. Each voter votes only one candidate of his/her constituency.

Draw ER Diagram for a Company Database

Scenario :

Tata Steels Private Limited wants to have a database application to store all the information generated in the company.

The company has 4 Departments in it, each Department has many Employees working in it. Each Department has a manager, there can be only one manager for a Department. Every Employee should work for one or the other Department. Each Department controls many Projects. An employee works in more than one Project. Each Employee has Dependents. Every Employee is supervised by one or the other supervisor. A Department can control minimum of 1 and maximum of 5 projects.



Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
WORKS_FOR
(4,N)
Locations
(1,1)
Employee
Department
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
(0,1)
Manager
Department Managed
(1,1)
(0,N)
Controlling Department
MANAGES
CONTROLS
(1,N)
Worker
Hours
(0,N)
Supervisor
(0,1)
Supervisee
WORKS_ON
Controlled Project
(1,1)
Project
PROJECT
(1,N)
SUPERVISION
(0,N)
Employee
Name
Location
Number
DEPENDENTS_OF
(1,1)
Dependent
DEPENDENT
Name
Sex
Birth_date
Relationship

Draw ER Diagram for a Movie Database

Scenario :

Sony Pictures Motion Picture Group wants to have a database application to store all the information generated in the company.

Sony Pictures Produces many Movies every year. Each Movie is contributed by various people like Actors, Producers, Directors, Choreographers, Screenplay Writers, etc involved in it. There can be more than one actor/actress in each Movie. Each movie is rated by various rating agencies. A Movie can be any language.

Draw ER Diagram for a Hospital Database

Scenario :

Apollo Hospital wants to have a database application to store all the information generated in the hospital.

Apollo Hospital wants to maintain each and every information that is generated in the hospital which includes Employees (Doctors, Lab Technicians, Nurses, Security Personnel, Administrators etc), Patients, Departments. There are many doctors who treat many patients. Each Doctor belongs to a particular department depending on his/her expertise. There are many departments in the hospital. Each department has a head. Nurses work for a particular department to serve the patients. Each department has operation rooms, patient rooms (private, semi-private, general).

Draw ER Diagram for a Train Reservation System

Scenario :

Indian Railways wants to have a database application to store all the records for data analysis.

There are many Trains in Indian Railways, each train has details like Train No, Train Name, Source, Destination and No of SL, AC1,AC2,AC3 seats. Each train is driven by Driver. Customer can book train seats after registration into the Indian Railways by giving basic details. Customer can book any number of seats from any source to destination on the availability of the seats. There are different types of seats Sleeper class, Ac Tier1, Ac Tier2, Ac Tier3. All the Seat Booking information should be maintained properly.

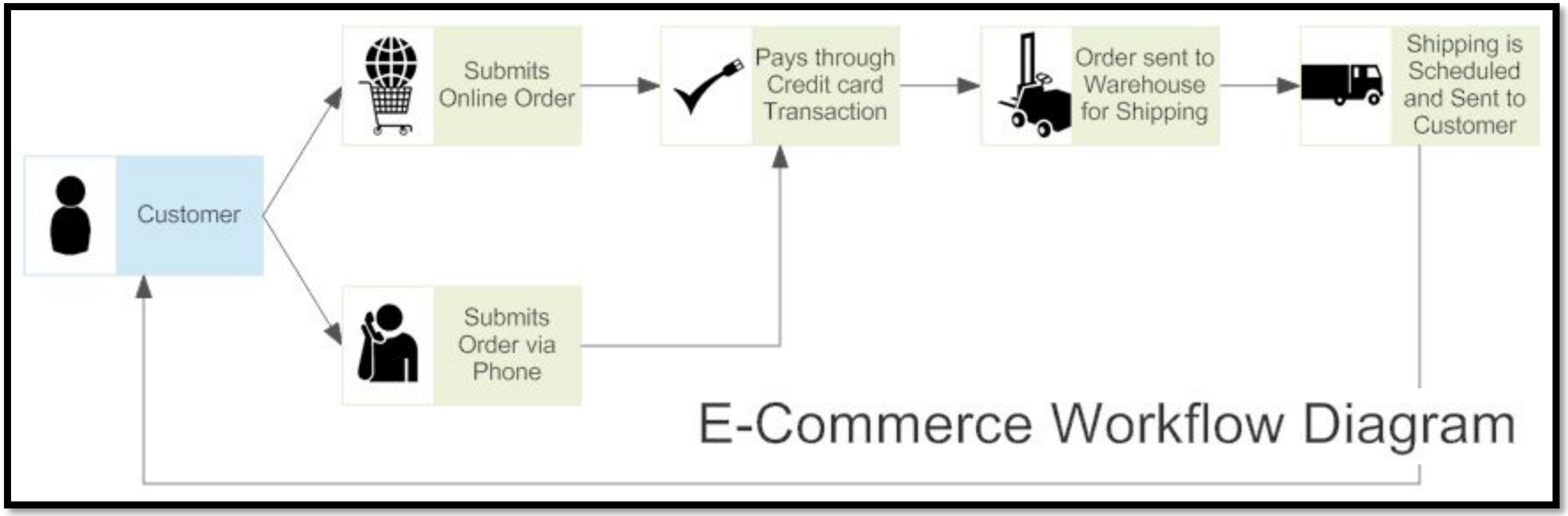
Consider the following details for Drawing ER Diagram,

Entities: TRAIN, DRIVER, BOOKING_INFO, CUSTOMER_INFO, TRAIN_FARES

Relationships : Drives, Fares_from_each_src_to_Dest, Books_Seat, Seat_Enquiries

Draw ER Diagram for the Following,

1. Online Food Ordering System (Zomato, Swiggy)
2. Retail Outlets
3. Online Shopping System (Flipkart, Amazon, Myntra)



Customer
Submits Online Order
Submits Order via Phone
Pays through Credit card Transaction
Order sent to Warehouse for Shipping
Shipping is Scheduled and Sent to Customer
E-Commerce Workflow Diagram

Relational Database Design Using ER-to-Relational Mapping

- In this section we describe the steps of an **algorithm for ER-to-relational mapping**,

Step 1: Mapping of Regular Entity Types.

Step 2: Mapping of Weak Entity Types.

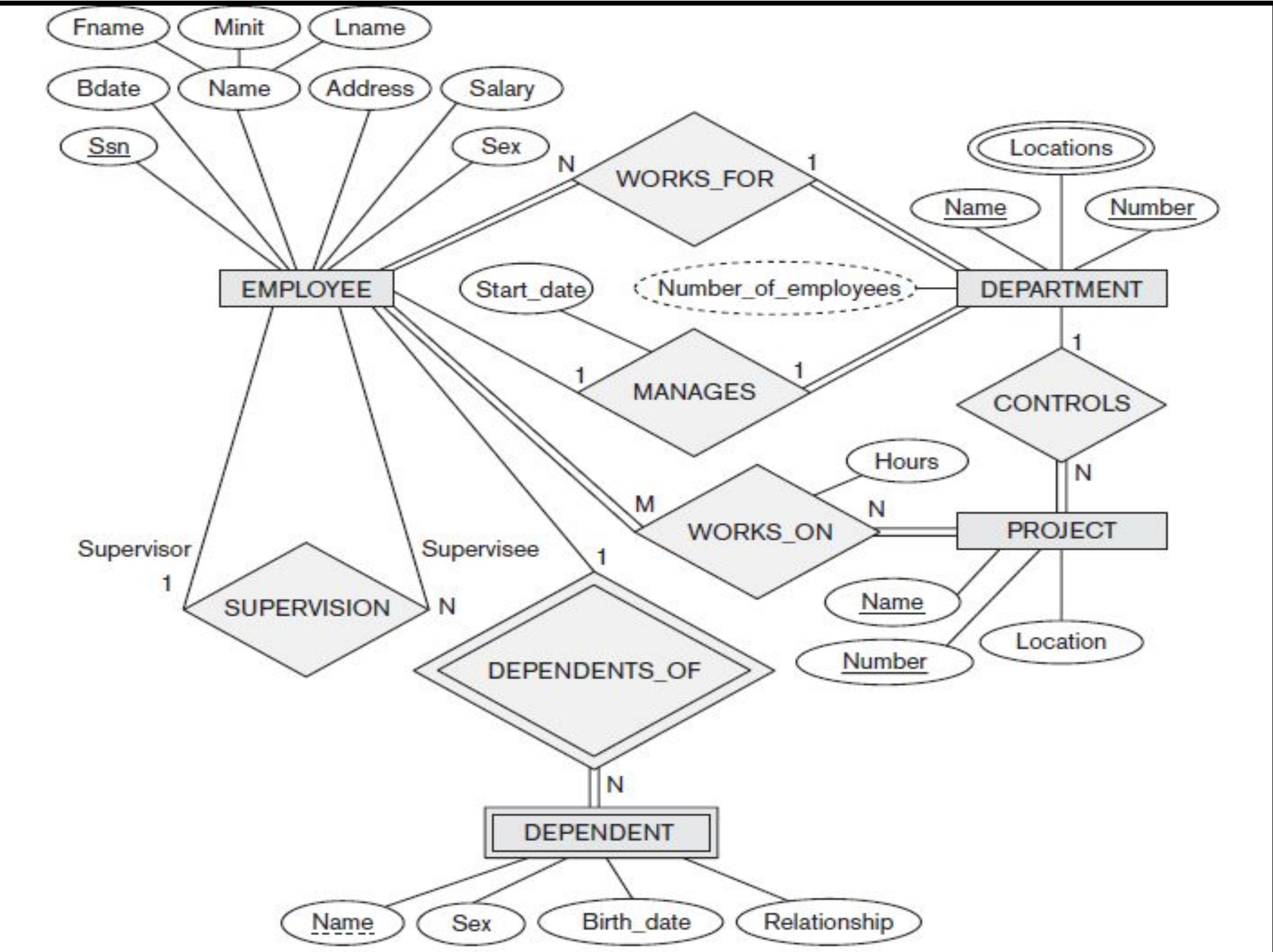
Step 3: Mapping of Binary 1:1 Relationship Types.

Step 4: Mapping of Binary 1:N Relationship Types

Step 5: Mapping of Binary M:N Relationship Types.

Step 6: Mapping of Multivalued Attributes.

Step 7: Mapping of N- ary Relationship *Types.*


Fname
Minit
Lname
Bdate
Name
Address
Salary
Ssn
Sex
N
WORKS_FOR
1
Locations
Name
Number
EMPLOYEE
Start_date
Number_of_employees
DEPARTMENT
1
1
MANAGES
1
CONTROLS
Hours
N
M
N
WORKS_ON
PROJECT
Supervisor
Supervisee
1
1
SUPERVISION
N
Name
Number
Location
DEPENDENTS_OF
N
DEPENDENT
Name
Sex
Birth_date
Relationship

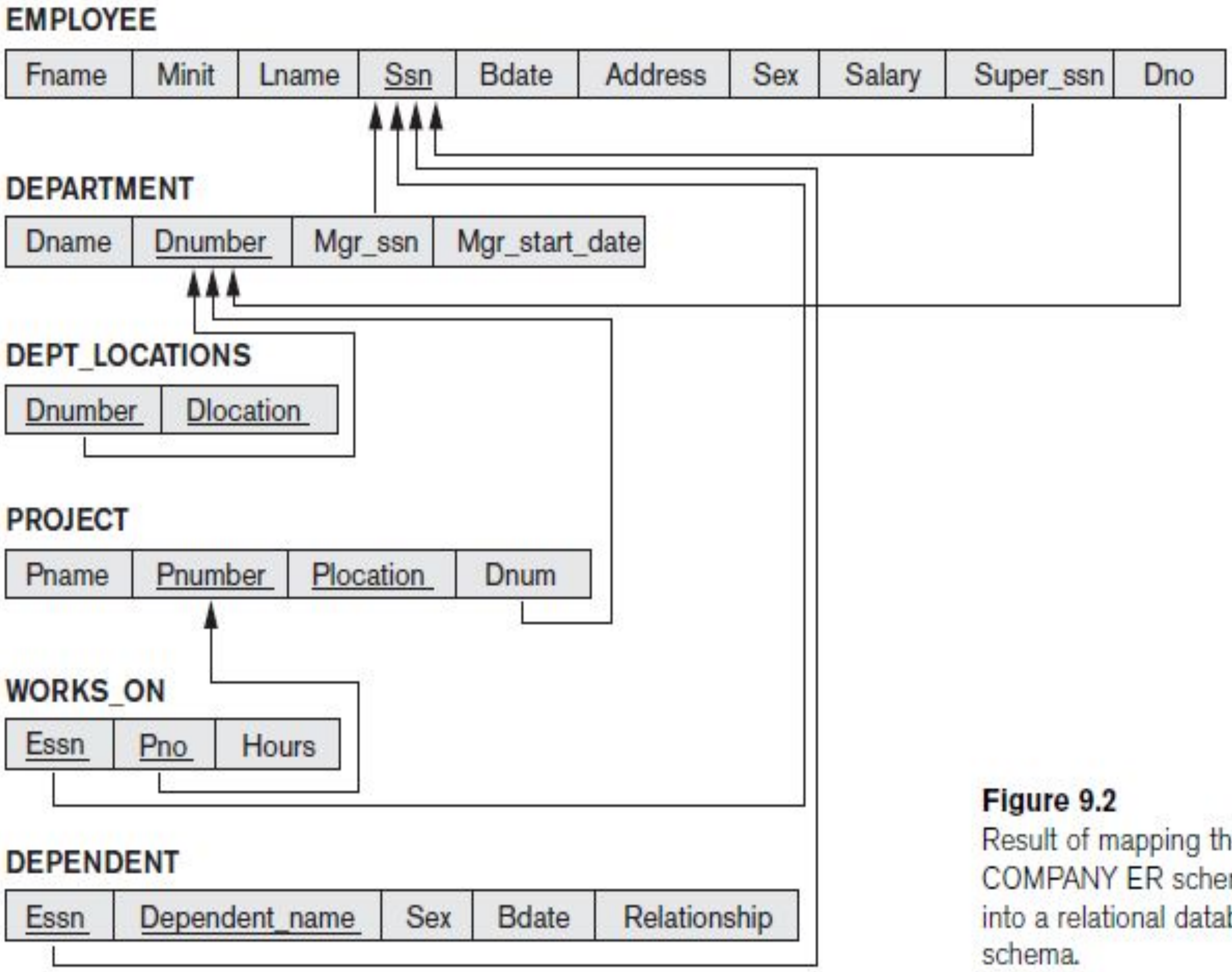


Figure 9.2
Result of mapping the COMPANY ER schema into a relational database schema.

- **Step 1: Mapping of Regular Entity Types.**

❑ **For each regular (strong) entity type E** *in the ER schema, create a relation R that includes all the simple attributes of E.* Include only the simple component attributes of a composite attribute.

❑ Choose one of the key attributes of *E as the primary key for R. If the chosen key of E is a composite,* then the set of simple attributes that form it will together form the primary key of *R.*

❑ If multiple keys were identified for *E during the conceptual design, the information* describing the attributes that form each additional key is kept in order to specify additional (unique) keys of relation *R.*

- **Step 2: Mapping of Weak Entity Types.**

- ❑ **For each weak entity type W** in the ER schema with owner entity type E, create a relation R and include all simple attributes (or simple components of composite attributes) of W as attributes of R.
- ❑ In addition, include as foreign key attributes of R, the primary key attribute(s) of the relation(s) that correspond to the owner entity type(s); this takes care of mapping the identifying relationship type of W.
- ❑ The primary key of R is the combination of the primary key(s) of the owner(s) and the partial key of the weak entity type W, if any.

- **Step 3: Mapping of Binary 1:1 Relationship Types.**

  For each binary 1:1 relationship type *R in the ER schema, identify the relations S and T that correspond to* the entity types participating in *R. There are three possible approaches:*

- ❑ *The foreign* key approach
  - Choose one of the relations say, S—and include as a foreign key in S the primary key of T. It is better to choose an entity type with total participation in R in the role of S.
- ❑ The merged relationship approach
  - This is possible when *both participations are total.*
- ❑ The cross reference or relationship relation approach
  - This approach is required for binary M:N relationship

- **Step 4: Mapping of Binary 1:N Relationship Types.**

❑ **For each regular binary** 1:N relationship type *R, identify the relation S that represents the participating entity* type at the *N-side of the relationship type. Include as foreign key in S the primary key* of the relation *T that represents the other entity type participating in R; we do this* because each entity instance on the N-side is related to at most one entity instance on the 1-side of the relationship type.

❑ Include any simple attributes (or simple components of composite attributes) of the 1:N relationship type as attributes of *S.*

- **Step 5: Mapping of Binary M:N Relationship Types.**

- **For each binary M:N** relationship type *R, create a new relation S to represent R. Include as foreign key* attributes in *S the primary keys of the relations that represent the participating* entity types; their *combination will form the primary key of S.*
- *Also include any simple* attributes of the M:N relationship type (or simple components of composite attributes) as attributes of *S.*
- *Notice that we cannot represent an M:N relationship* type by a single foreign key attribute in one of the participating relations (as we did for 1:1 or 1:N relationship types) because of the M:N cardinality ratio; we must create a separate *relationship relation S.*

- **Step 6: Mapping of Multivalued Attributes.**

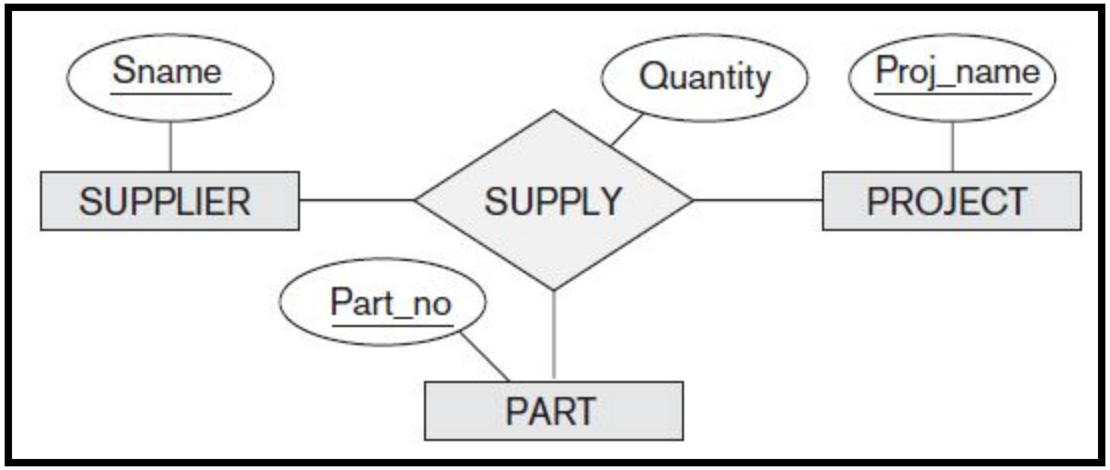
- **For each multivalued attribute *A*,** create a new relation *R. This relation R will include an attribute corresponding to A,* plus the primary key attribute *K—as a foreign key in R—of the relation that represents* the entity type or relationship type that has *A as a multivalued attribute.*
- *The* primary key of *R is the combination of A and K. If the multivalued attribute is composite,* we include its simple components.

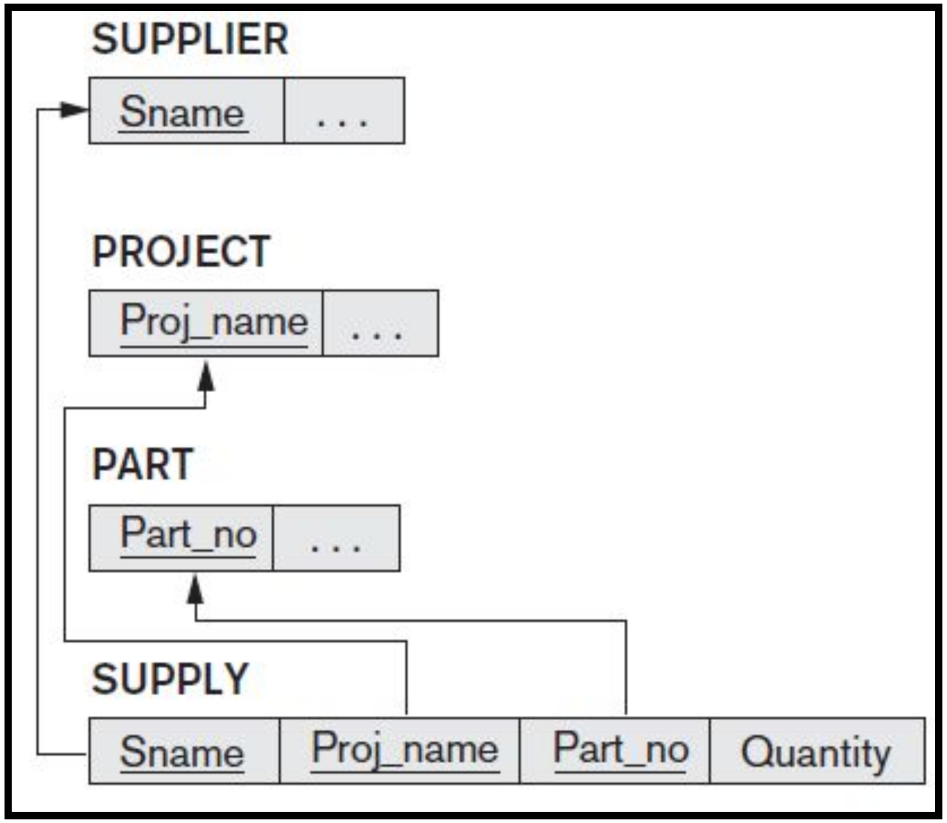
- **Step 7: Mapping of N-ary Relationship Types.**

❑ ***For each n-ary relationship*** type *R, where n > 2, create a new relation S to represent R. Include as foreign key* attributes in *S the primary keys of the relations that represent the participating* entity types.

❑ Also include any simple attributes of the *n-ary relationship type (or* simple components of composite attributes) as attributes of *S. The primary key of S* is usually a combination of all the foreign keys that reference the relations representing the participating entity types.

❑ However, if the cardinality constraints on any of the entity types *E participating in R is 1, then the primary key of S should not* include the foreign key attribute that references the relation *E corresponding to E*


Sname
Quantity
Proj_name
SUPPLIER
SUPPLY
PROJECT
Part_no
PART


SUPPLIER
Sname
. . .
PROJECT
Proj_name
. . .
PART
Part_no
. . .
SUPPLY
Sname
Proj_name
Part_no
Quantity