

Using High-Level Conceptual Data Models for Database Design

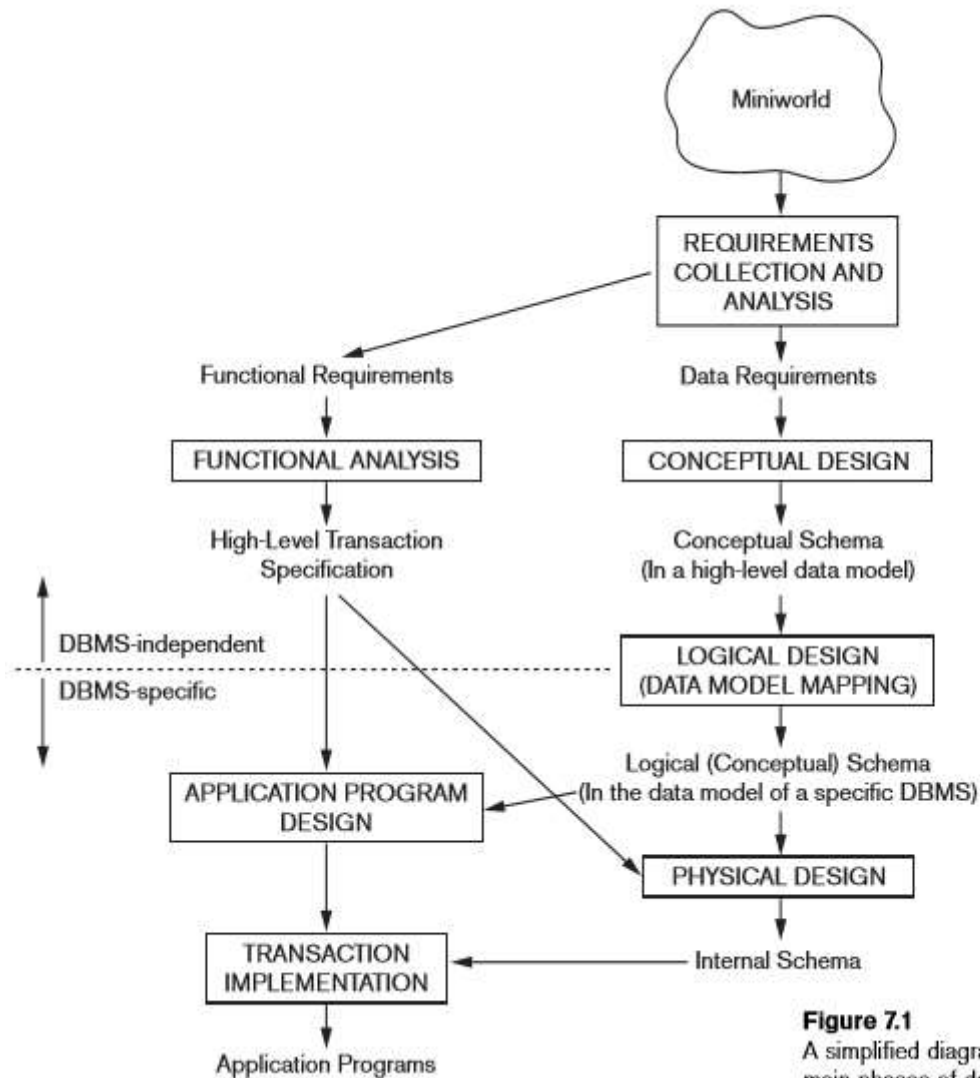


Figure 7.1

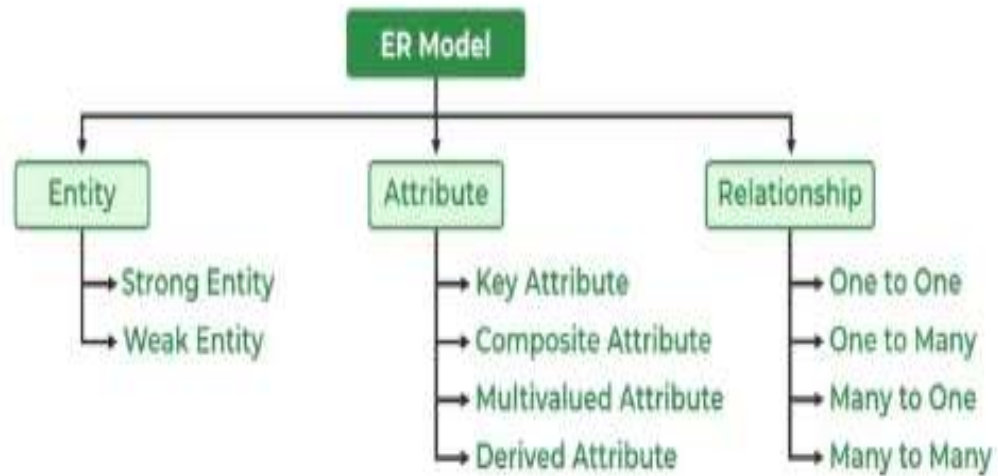
A simplified diagram to illustrate the main phases of database design.

- The first step shown is **requirements collection and analysis**.
- During this step, the database designers interview prospective database users to understand and document their data requirements
- Once the requirements have been collected and analyzed, the **next step is to create a conceptual schema** for the database , using a high-level conceptual data model.
- The conceptual schema is a concise description of the data requirements of the users these concepts do not include implementation details
- The next step in database design is the **actual implementation of the database**, using a commercial DBMS.
- Most current commercial DBMSs use an implementation data model—such as the relational or the object-relational database model
- So 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**
- 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

ER MODEL

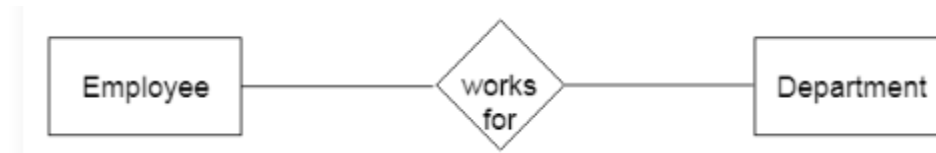
- ER model stands for an Entity- Relationship Model
- It is a **high level conceptual data model**
- This model is used to define the data elements and relationship for a specified system.
- It is used for conceptual data design of database applications
- It also develops a very simple and easy to design view of data.
- In ER modeling, the database structure is portrayed as a diagram called an **entity-relationship diagram**.
- The Entity Relationship Diagram explains the relationship among the entities present in the database.
- ER models are used to model real-world objects like a person, a car, or a company and the relation between these real-world objects.
- In short, the ER Diagram is the structural format of the database and for non technical users
- These diagrams are very easy to understand and easy to create even for a naive user. It gives a standard solution for visualizing the data logically.

COMPONENT OF ER DIAGRAM



ENTITY

- An Entity may be **an object with a physical existence** – a particular person, car, house, or employee
- Or it may be **an object with a conceptual existence** – a company, a job, or a university course.
- In the ER diagram, an entity can be represented as rectangles.
- Entity is distinguished from other objects on basis of attributes
- Entities can be tangible and intangible
- Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



Types of Entity

There are two types of entity:

1. Strong Entity

.

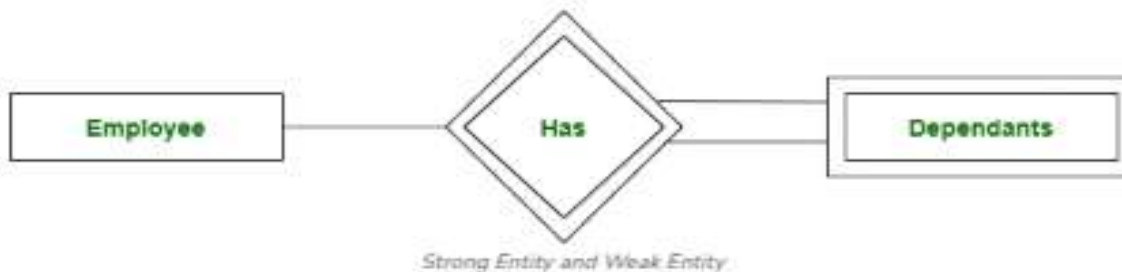
2. Weak Entity

1. STRONG ENTITY

- A type of entity that has a key Attribute.
- Strong Entity does not depend on other Entity in the Schema.
- It has a primary key, that helps in identifying it uniquely
- It is represented by a rectangle.

2. WEAK ENTITY

- An Entity type has a key attribute that uniquely identifies each entity in the entity set.
- A weak entity type is represented by a Double Rectangle.
- The relationship between the weak entity type and its identifying strong entity type is called **identifying relationship** and it is represented by a double diamond.



Examples for Strong and weak entity

1. In a banking system, have two entities : Transaction and Account

Transaction has their own properties such as Trans_id, Trans_date, Amount

Similarly, Account entity has their own properties : Acc_id, Acc_type, Balance

Here, Account is a strong entity and Transaction is a weak entity

ie, Transaction entities and their properties all are depends on other entity Account

Another eg :

2. In educational system , have two entities : Course and Module

Course have properties : Courseid, Coursename, Duration

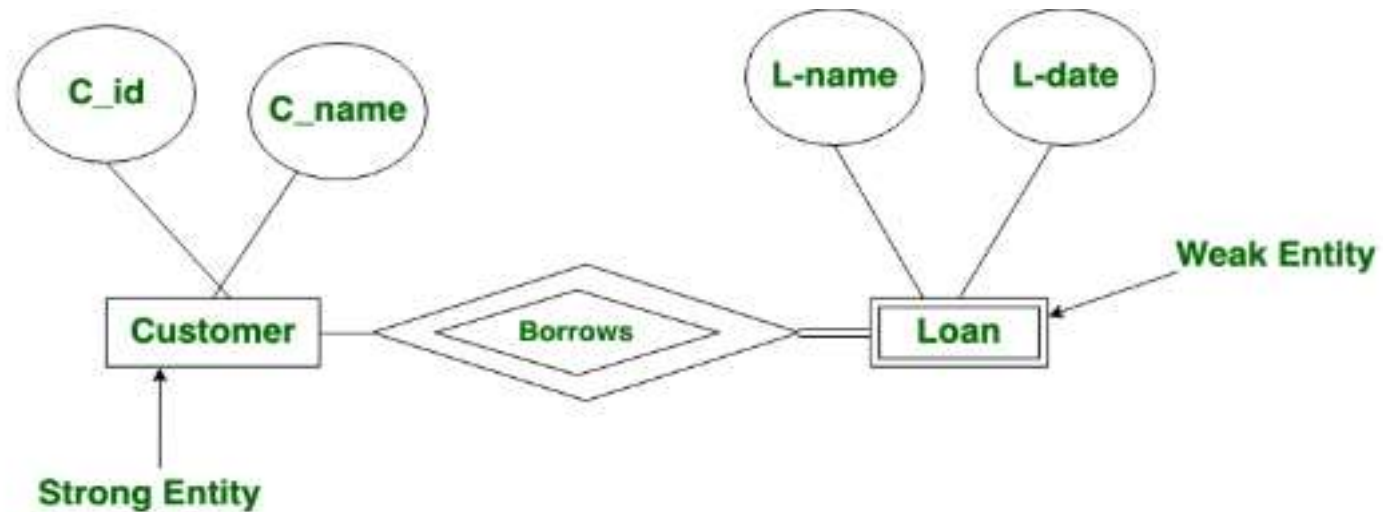
Module have properties : Moduleid, Title

Here, Course is a strong entity and Module is a weak entity

ie, Module entity that will depends on the other entity Course

Difference between Strong Entity & Weak Entity

Strong Entity	Weak Entity
Strong entity always has a primary key .	While a weak entity has a partial discriminator key.
Strong entity is not dependent on any other entity.	Weak entity depends on strong entity.
Strong entity is represented by a single rectangle.	Weak entity is represented by a double rectangle.
Two strong entity's relationship is represented by a single diamond.	While the relation between one strong and one weak entity is represented by a double diamond.
Strong entities have either total participation or partial participation.	A weak entity has a total participation constraint.

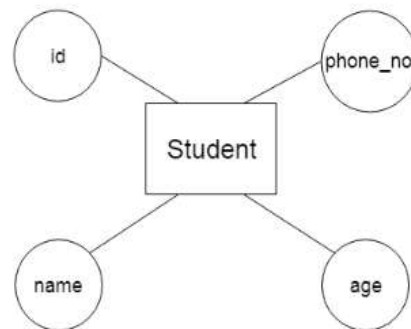


In ER models,

- **Strong entities can exist independently**
- **Weak entities depend on strong entities.**

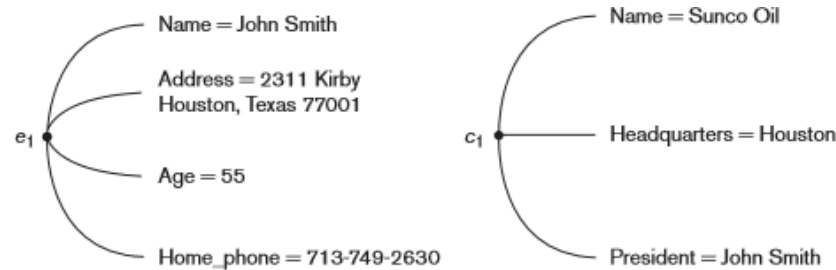
ATTRIBUTE

- Each entity has attributes
- The attribute is used to **describe the property of an entity**.
- Eclipse or Oval is used to represent an attribute.
- An entity may contain any number of attributes.
- Attributes can also be subdivided into another set of attributes.
- Attributes help define and organize the data, making it easier to retrieve and manipulate information within the database.
- For eg, id, age, contact number, name, etc. can be attributes of a student.
- Attributes define the properties of entities in an ER model
- A particular entity will have a value for each of its attributes.
- The attribute values that describe each entity become a major part of the data stored in the database.



For eg :

- The EMPLOYEE entity e1 has four attributes: Name, Address, Age, and Home_phone; their values are 'John Smith,' '2311 Kirby, Houston, Texas 77001', '55', and '713-749-2630', respectively.
- The COMPANY entity c1 has three attributes: Name, Headquarters, and President; their values are 'Sunco Oil', 'Houston', and 'John Smith', respectively.



Several types of attributes occur in the ER model

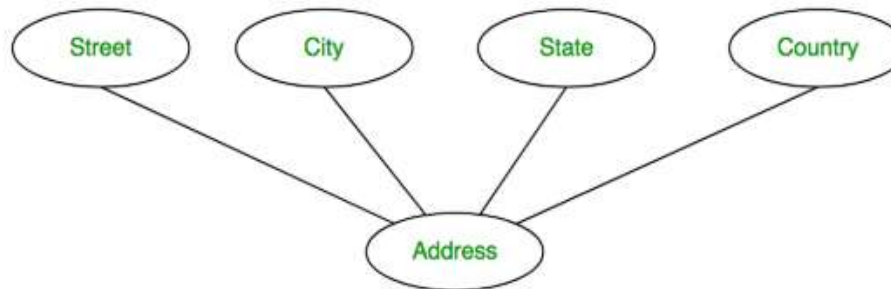
- Simple Attribute
- Composite Attribute
- Single valued Attribute
- Multi valued Attribute
- Stored Attribute
- Derived Attribute
- Key Attribute
- Null Attribute
- Descriptive Attribute

1. Simple Attribute

- Simple attributes are atomic values
- ie, an attribute that cannot be further subdivided into components is a simple attribute.
- **Eg:** The roll number of a student, the ID number of an employee, gender, and many more.

2. Composite Attribute

- An attribute composed of many other attributes is called a composite attribute.
- An attribute that can be split into components is a composite attribute.
- In ER diagram, the composite attribute is represented by an oval comprising of ovals
- **For eg,** the Address attribute of the student Entity type consists of Street, City, State, and Country.
- Composite attributes can form a hierarchy
- If the composite attribute is referenced only as a whole, there is no need to subdivide it into component attributes.

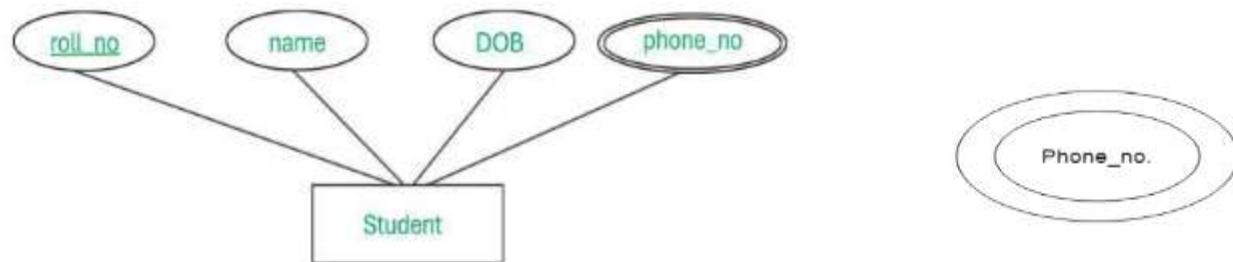


3. Single-Valued Attribute

- The attribute which takes up only a single value for each entity instance
- **Eg:** The age of a student, Aadhar card number , DOB of a person , Vehicle registration number

4. Multi-valued Attribute

- An attribute consisting of more than one value for a given entity.
- They represent a one to many relationship with an entity
- In ER diagram, a multi valued attribute is represented by a double oval.
- **For eg,** a student can have more than one phone number or email address

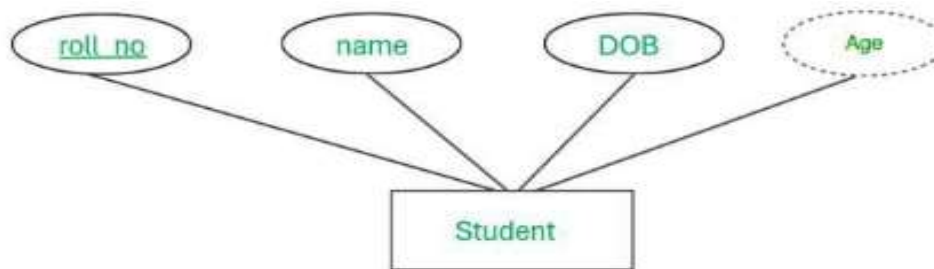


5. Stored Attribute

- The stored attribute are those attribute which doesn't require any type of further update since they are stored in the [database](#).
- ie, whose values are explicitly stored in the database
- **Eg:** DOB(Date of birth) is the stored attribute.

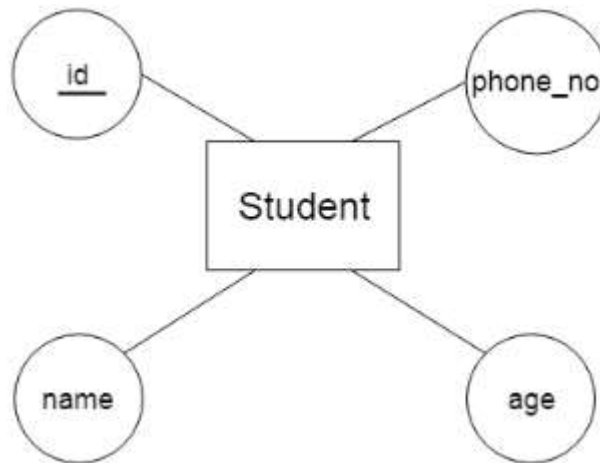
6. Derived Attribute

- An attribute that can be derived from other attributes of the entity type is known as a derived attribute.
- ie, they are not exist in the physical database, but their values are derived from other attributes
- So this avoids the data redundancy and save storage spaces
- **e.g.;** Age (can be derived from DOB). [Age = Current Date – DOB]
- In ER diagram, the derived attribute is represented by a dashed or dotted oval.



7. Key attribute

- Key attributes are those attributes that can uniquely identify the entity in the [entity set](#).
- The key attribute is used to represent the main characteristics of an entity.
- The key attribute is represented by an ellipse with the text underlined.
- It ensures the data integrity and avoid the duplication of data
- **Eg:** Roll-No is the key attribute because it can uniquely identify the student.



8. Complex Attribute

- In general, composite and multi-valued attributes can be nested arbitrarily.
- We can represent arbitrary nesting by grouping components of a composite attribute between parentheses () and separating the components with commas, and by displaying multi-valued attributes between braces { }. Such attributes are called complex attributes.
- These attributes are rarely used in DBMS.
- **Eg:** Address because address contain composite value like street, city, state, PIN code and also multi valued because one people has more that one house address.





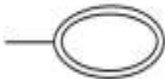
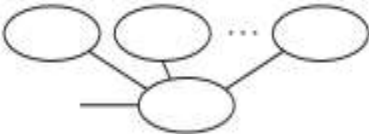

```
{Address_phone( (Phone(Area_code,Phone_number)),Address(Street_address  
(Number,Street,Apartment_number),City,State,Zip) )}
```

Figure
A complex attribute:
Address_phone.

9. Null Attribute / Null Values

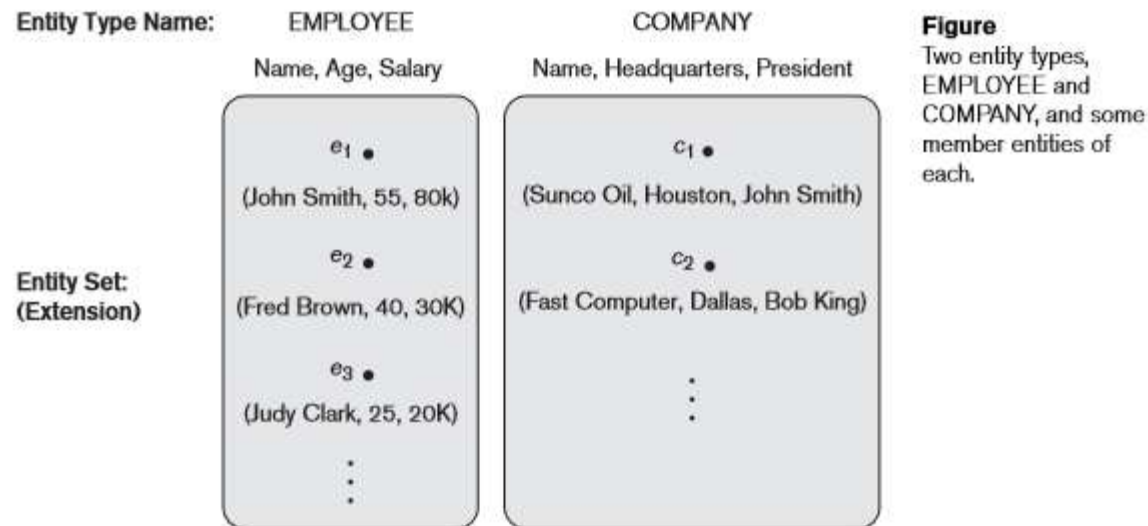
- In some cases, a particular entity may not have an applicable value for an attribute.
- For eg, in a Person entity, attribute Spouse_Name may not have a value if a person is not married
- Then , we can give a special value called 'NULL' is created
- Null value is a value which is not inserted but it does not hold zero value.
- The attributes which can have a null value called null valued attributes.
- NULL can also be used if we do not know the value of an attribute for a particular entity
- **For eg**, if we do not know the home phone number of a person

Symbols Used in Entity Model

Symbol	Meaning
	Entity
	Weak Entity
	Attribute
	Key Attribute
	Multivalued Attribute
	Composite Attribute
	Derived Attribute

ENTITY TYPE

- A database usually contains groups of entities that are similar.
- For eg., a company employing hundreds of employees may want to store similar information concerning each of the employees. These employee entities share the same attributes, but each entity has its own values for each attribute.
- **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.

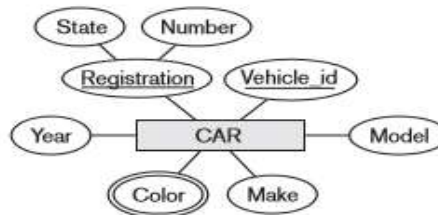


Entity set

- The collection of all entities of a particular entity type in the database at any point in time is called an entity set
- In other words, the collection of same type of entities that is their attributes are same is called entity set
- The entity set is usually referred to using the same name as the entity type
- We can say that entity type is a superset of the entity set as all the entities are included in the entity type
- **An entity type is represented in ER diagrams as a rectangular box enclosing the entity type name.**
- An entity type describes the schema or intension for a set of entities that share the same structure.
- The collection of entities of a particular entity type is grouped into an entity set, which is also called the extension of the entity type.

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

(a)



(b)

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

CAR₁
((ABC 123, TEXAS), TK629, Ford Mustang, convertible, 2004 {red, black})

CAR₂
((ABC 123, NEW YORK), WP9872, Nissan Maxima, 4-door, 2005, {blue})

CAR₃
((VSY 720, TEXAS), TD729, Chrysler LeBaron, 4-door, 2002, {white, blue})

⋮

Domain or Value Sets of an attribute

- Each simple attribute of an entity type is associated with a value set (or domain of values), which specifies the set of values that may be assigned to that attribute for each individual entity
- Domain of an attribute is the allowed set of values of that attribute.
- **For eg.,** if attribute is ‘grade’, then its allowed values are A,B,C,F.

Ie, Grade = {A, B,C,F}

- Value sets are not displayed in ER diagrams
- And are typically specified using the basic data types available in most programming languages, such as integer, string, Boolean, float, enumerated type, subrange, and so on.
- Additional data types to represent common database types such as date, time, and other concepts are also employed.
- Mathematically, an attribute A of entity set E whose value set is V can be defined as a function from E to the power set P(V) of V:

$$A : E \rightarrow P(V)$$

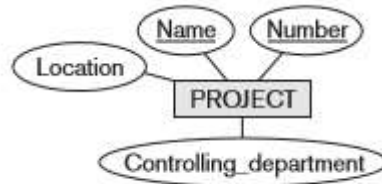
- We refer to the value of attribute A for entity ‘e’ as A(e).

Define the entity types for the COMPANY database, based on the requirements :

1. An entity type DEPARTMENT with attributes Name, Number, Locations, Manager, and Manager_start_date. Locations is the only multi-valued attribute. We can specify that both Name and Number are (separate) key attributes because each was specified to be unique.
2. An entity type PROJECT with attributes Name, Number, Location, and Controlling_department. Both Name and Number are (separate) key attributes.
3. An entity type EMPLOYEE with attributes Name, Ssn, Sex, Address, Salary, Birth_date, Department, and Supervisor. Both Name and Address may be composite attributes; however, this was not specified in the requirements. We must go back to the users to see if any of them will refer to the individual components of Name—First_name, Middle_initial, Last_name—or of Address.
4. An entity type DEPENDENT with attributes Employee, Dependent_name, Sex, Birth_date, and Relationship (to the employee).



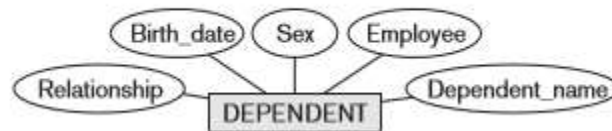
(1)



(2)



(3)



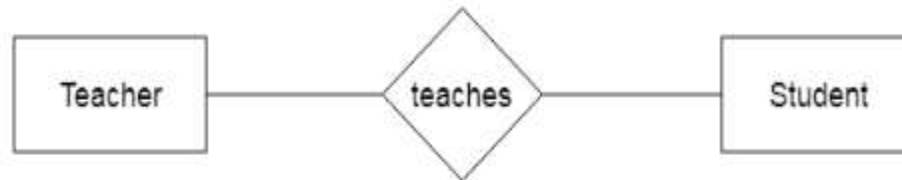
(4)

Relationship

- An attribute of one entity type refers to another entity type, some relationship exists.
- Relates two or more distinct entities with a specific meaning.
- It is an association between two or more entities of same or different entity set
 - For example, EMPLOYEE John works on the PROJECT

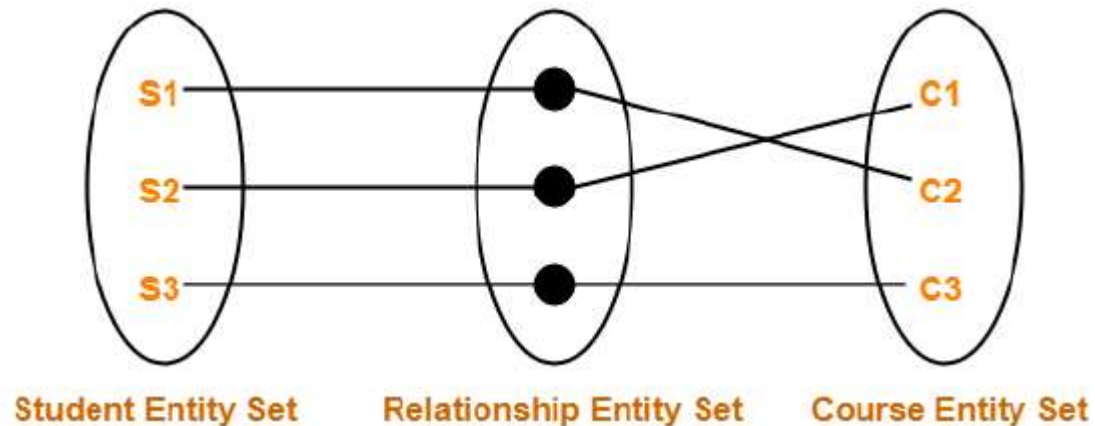
Relationship Type

- A Relationship Type represents the association between entity types.
- ie, A relationship is used to describe the relation between entities
- **In ER diagram, the relationship type is represented by a diamond or rhombus and connecting the entities with lines.**
- For example, 'Teaches' is a relationship type that exists between entity type Student and Teacher.



Relationship Set

- A relationship set is a set of relationships of the same type.
- Characteristics of Relationship Set :
 - **Degree** : Indicates to the number of properties related with the relationship set.
 - **Arity** : Arity of a relationship set indicates the number of taking part relations.
 - **Cardinality** : The number of occurrences or records that can be related with each substance on both sides of the relationship.



- A relationship type R among n entity types E_1, E_2, \dots, E_n defines a set of associations or a relationship
- The relationship set R is a set of relationship instances r_i , where each r_i associates n individual entities (e_1, e_2, \dots, e_n) , and each entity e_j in r_i is a member of entity set E_j
- Each of the entity types E_1, E_2, \dots, E_n is said to participate in the relationship type R ; similarly, each of the individual entities e_1, e_2, \dots, e_n is said to participate in the relationship instance $r_i = (e_1, e_2, \dots, e_n)$.
- **For eg :** Consider a relationship type **WORKS_FOR** between the two entity types **EMPLOYEE** and **DEPARTMENT**, which associates each employee with the department for which the employee works in the corresponding entity set. Each relationship instance in the relationship set **WORKS_FOR** associates one **EMPLOYEE** entity and one **DEPARTMENT** entity.

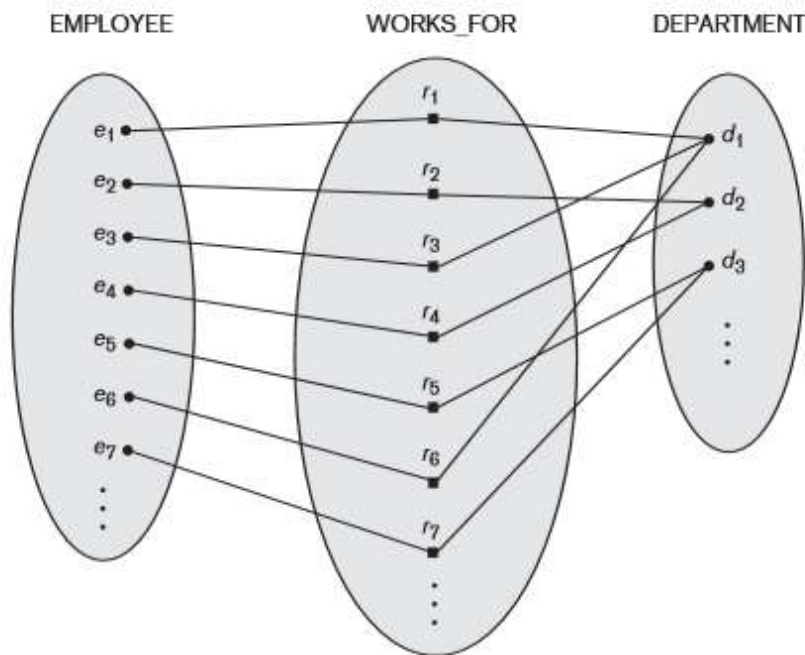
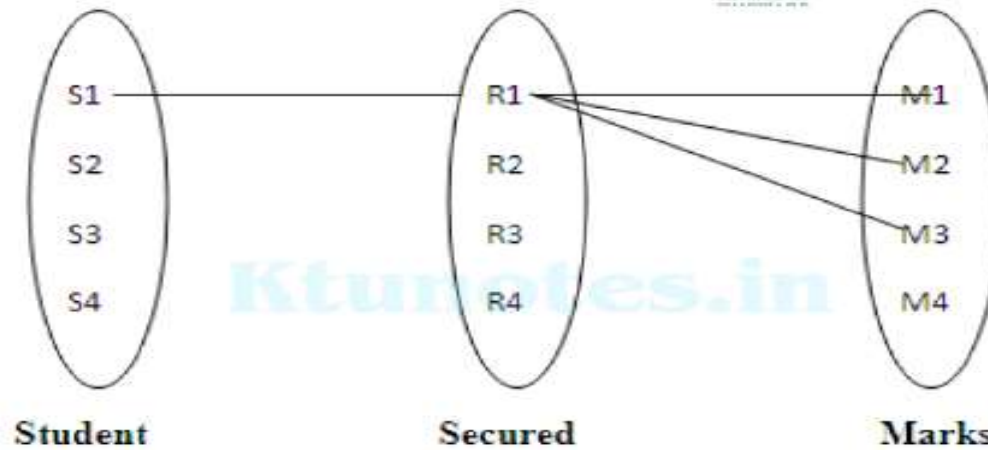


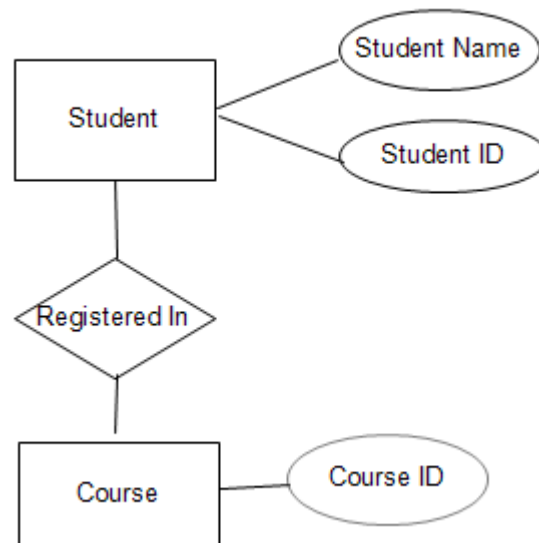
Figure
Some instances in the **WORKS_FOR** relationship set, which represents a relationship type **WORKS_FOR** between **EMPLOYEE** and **DEPARTMENT**.

Eg : 2



Relationship type: secured
Relationship set: {R1, R2, R3, R4}
Relationship instances: R1

GRAPHICAL REPRESENTATION OF RELATION SETS

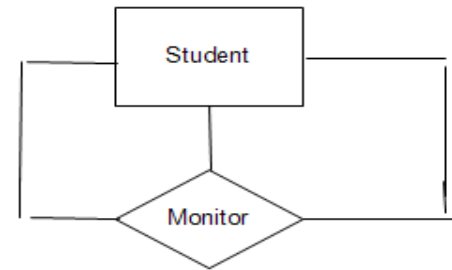


Degree of a Relationship Set

- The number of different entity sets participating in a relationship set.
- ie, the number of participating entity type is called as a degree of relationship type
- Relationships can generally be of any degree, but the ones most common are binary relationships.
- Higher degree relationships are generally more complex than binary relationships

1. Unary Relationship:

- When there is only **ONE** entity set participating in a relation
- For example, one person is married to only one person.
- Also called as ***recursive relationship***



2. Binary Relationship:

- When there are **TWO** entities set participating in a relation
- For example, a Student is enrolled in a Course.



3. N-ary Relationship:

- When there are n entities set participating in a relationship
- When there are **THREE** entities set participating in a relationship is called as **Ternary Relationship**

What is Cardinality?

- The number of times an entity of an entity set participates in a relationship set
- Cardinality can be of different types:

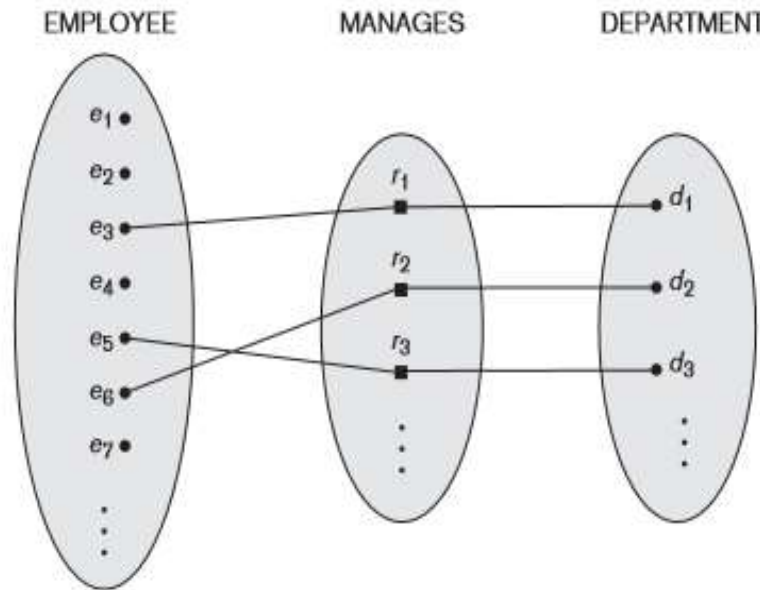
1. One-to-One:

- When only one instance of an entity is associated with the relationship, then it is known as one to one relationship.
- Let us assume that a male can marry one female and a female can marry one male. So the relationship will be one-to-one.



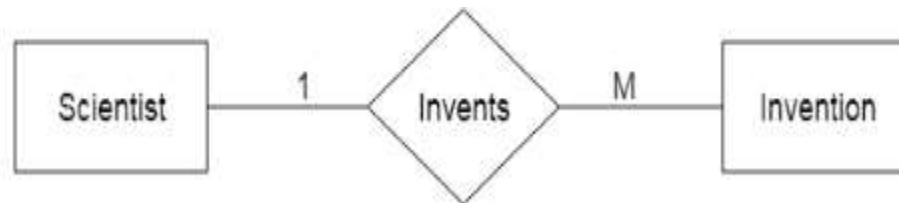
Another example of a 1:1 binary relationship is MANAGES which relates a department entity to the employee who manages that department. This represents the mini world constraints - an employee can manage one department only and a department can have one manager only.

Figure
A 1:1 relationship,
MANAGES.



2. One-to-Many:

- When only one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then this is known as a one-to-many relationship.
- **For eg,** Scientist can invent many inventions, but the invention is done by the only specific scientist.



3. Many-to-One:

- When more than one instance of the entity on the left, and only one instance of an entity on the right associates with the relationship then it is known as a many-to-one relationship.
- **For eg,** Student enrolls for only one course, but a course can have many students.

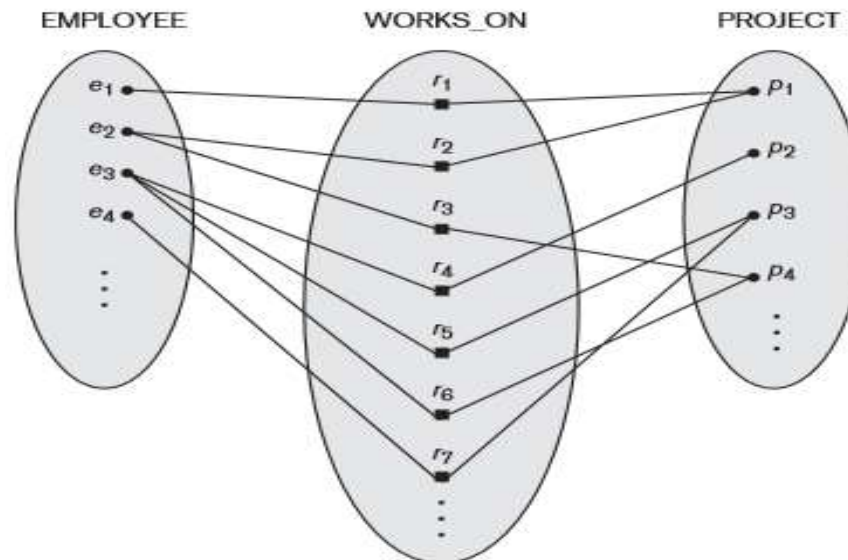


4. Many-to-Many:

- When more than one instance of the entity on the left, and more than one instance of an entity on the right associates with the relationship then it is known as a many-to-many relationship.
- **For eg,** Employee can assign by many projects and project can have many employees.



Eg 2: The relationship type WORKS_ON is of cardinality ratio M:N, because the mini world rule is that an employee can work on several projects and a project can have several employees.



Constraints on ER Model

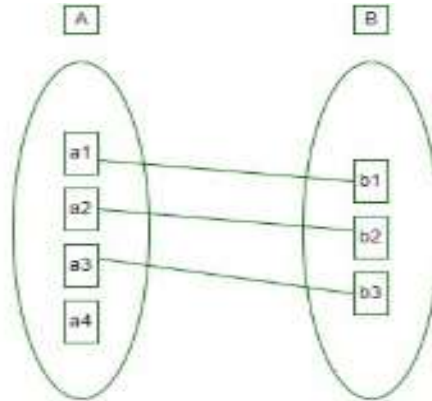
- Constraints are used for modeling limitations on the relations between entities.
- There are two types of constraints on the Entity Relationship (ER) model –
 - Mapping Cardinality or Cardinality ratio
 - Participation Constraints

1. Mapping Cardinality

- It is expressed as the number of entities to which another entity can be associated via a relationship set.
- For the binary relationship set there are entity set A and B then the mapping cardinality can be one of the following :
 - One – to - One
 - One – to - Many
 - Many – to - One
 - Many – to - Many

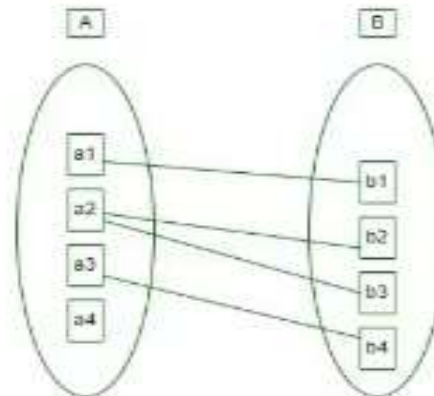
One-to-one relationship

An entity set A is associated with at most one entity in B and an entity in B is associated with at most one entity in A.



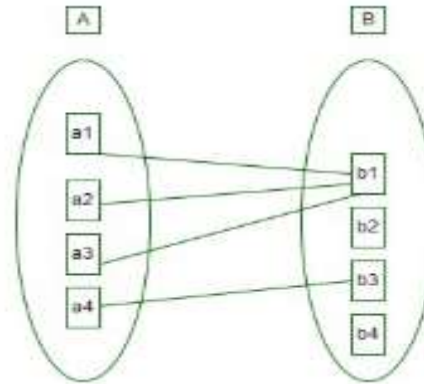
One-to-many relationship

An entity set A is associated with any number of entities in B with a possibility of zero and an entity in B is associated with at most one entity in A.



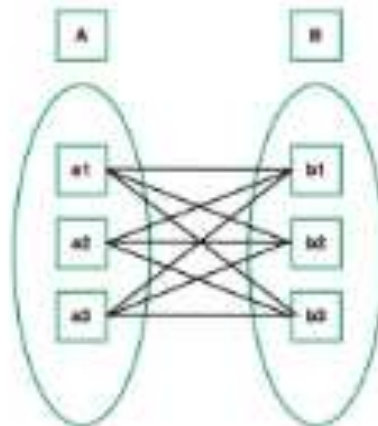
Many-to-one relationship

An entity set A is associated with at most one entity in B and an entity set in B can be associated with any number of entities in A with a possibility of zero.



Many-to-many relationship

An entity set A is associated with any number of entities in B with a possibility of zero and an entity in B is associated with any number of entities in A with a possibility of zero.

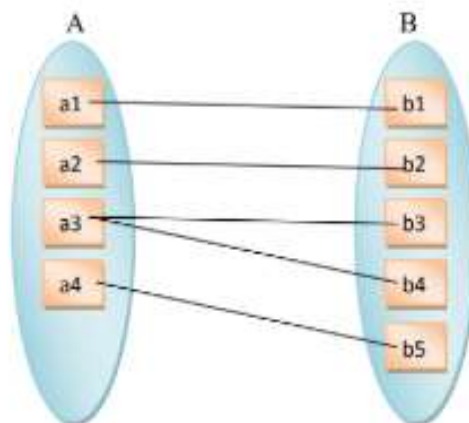


2. Participation Constraints

- It refers to rules that *determine the minimum and maximum participation* of entities or relationships in a given relationship set.
- It specifies whether the existence of an entity depends on being related to another entity through relationship types
- These constraints define the max and min number of relationship instances that each entity can participate in
 - **Maximum cardinality** : Maximum no. of times an entity can participate in a relationship
 - **Minimum Cardinality** : Minimum no. of times an entity can participate in a relationship
- For example, in a College Database, partial participation would permit courses with no enrolled students, while entire participation might require all students to be enrolled in at least one course.
- There are two types of participation constraints :
 - Total Participation
 - Partial Participation

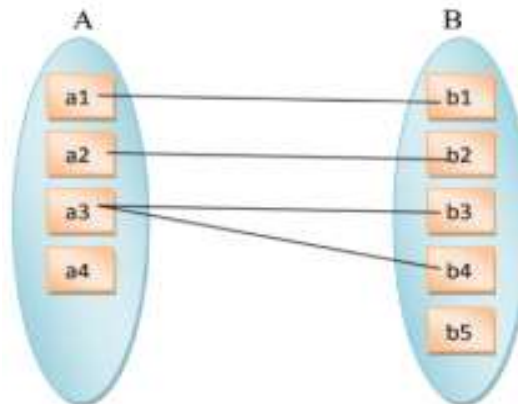
Total Participation

- The participation of an entity set E in a relationship set R is said to be total , *if every entity in E participates in at least one relationship in R*
- Total participation is also called *existence dependency*
- In ER diagrams, total participation (or existence dependency) is displayed as a double line connecting the participating entity type to the relationship
- Eg : In a university database, for instance, total participation between courses and students indicates that each student is required to be registered in a minimum of one course. It follows that no student can be excluded from a course.
- In below digram, the participation of entity set A in the relationship set is total because every entity and the participation of entity set B in the relationship set is also total because every entity of B also participates in the relationship set.



Partial Participation

- In database design, partial participation also known as *optional participation*
- *If only some of the entities in E participate in relationship R*, then the participation of E in R is said to be partial participation
- In ER diagrams, partial participation is represented by a single line
- Consider a database at a university, for instance, Partial participation can mean that some students are enrolled in classes but not all students are registered in them. While some objects are connected to one another, others may stand alone.
- In below diagram, The participation of entity set A in the relationship set is partial because only some entities of A participate in the relationship set, while the participation of entity set B in the relationship set is total because every entity of B participates in the relationship set.



- *Total Participation which is represented by double lines*
- *Partial Participation which is represented by single lines*

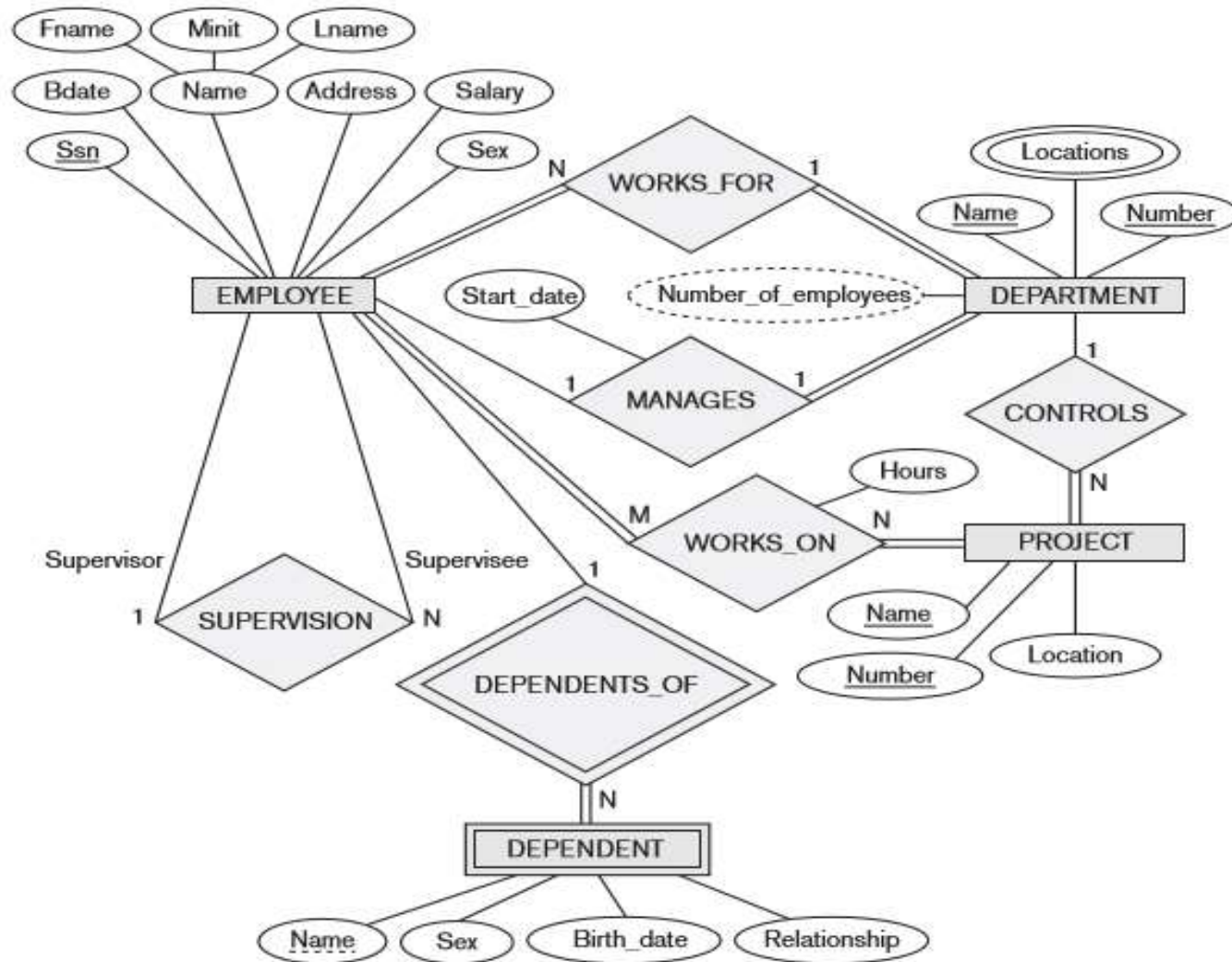
Suppose an entity set Student related to an entity set Course through Enrolled relationship set.

- The participation of entity set course in enrolled relationship set is **partial** because a course may or may not have students enrolled in. It is possible that only some of the course entities are related to the student entity set through the enrolled relationship set.
- The participation of entity set student in enrolled relationship set is **total** because every student is expect to relate at least one course through the enrolled relationship set.



Participation in Enrolled relationship set: **Partial** Course
Total Student

Sample ER Schema Diagram for COMPANY database



In our example, we specify the following relationship types:

1. MANAGES, a 1:1 relationship type between EMPLOYEE and DEPARTMENT.
EMPLOYEE participation is partial
2. WORKS_FOR, a 1:N relationship type between DEPARTMENT and EMPLOYEE.
Both participations are total.
3. CONTROLS, a 1:N relationship type between DEPARTMENT and PROJECT.
The participation of PROJECT is total, whereas that of DEPARTMENT is determined to be partial
4. SUPERVISION, a 1:N relationship type between EMPLOYEE (in the supervisor role) and EMPLOYEE (in the supervisee role).
Both participations are determined to be partial
5. WORKS_ON, determined to be an M:N relationship type with attribute Hours
Both participations are determined to be total
6. DEPENDENTS_OF, a 1:N relationship type between EMPLOYEE and DEPENDENT, which is also the identifying relationship for the weak entity type DEPENDENT.
The participation of EMPLOYEE is partial, whereas that of DEPENDENT is total.

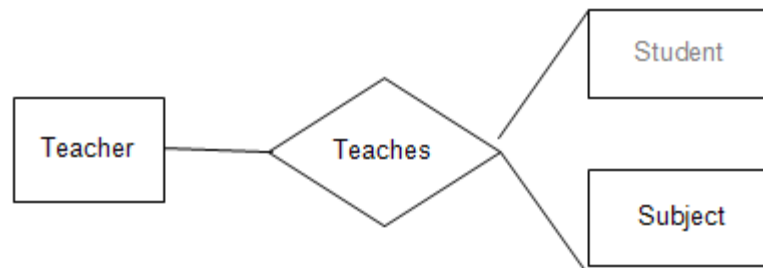
RELATIONSHIP OF DEGREE 3

- A relationship of degree 3 is also known as a **ternary relationship**, which is when **three entities are associated with each other**.
- We define a ternary relationship among three entities only when the concept cannot be represented by several binary relationships among those entities.
- The **degree of a relationship** refers to the number of entity types involved in it.

ie, Relationship Degree = 3

- **For Example,**

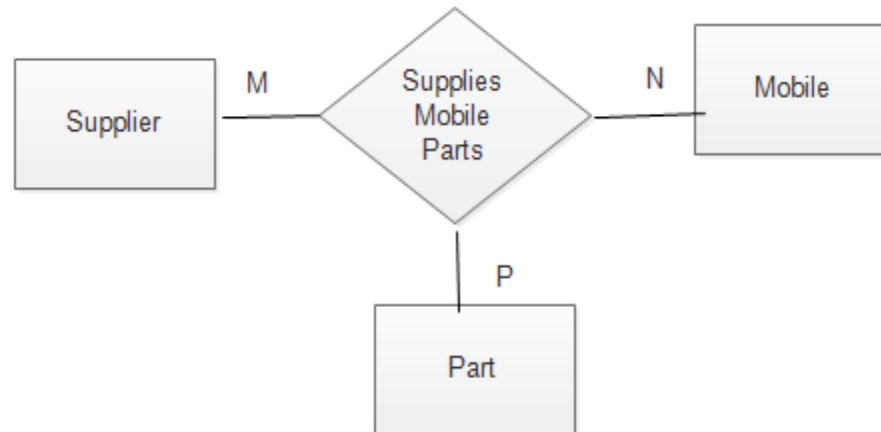
In a data model for an educational institution, we could define a single relationship that associates the entities Subject , Teacher , and Student .



- For Example : Consider a Mobile manufacture company.

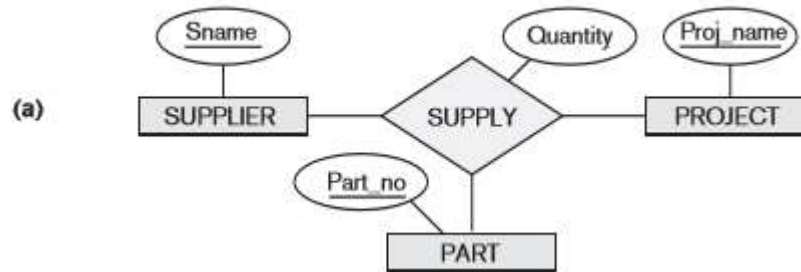
Three different entities involved:

- Mobile - Manufactured by company.
 - Part - Mobile Part which company get from Supplier.
 - Supplier - Supplier supplies Mobile parts to Company.
- Mobile, Part and Supplier will participate simultaneously in a relationship, because of this fact when we consider cardinality we need to consider it in the context of two entities simultaneously relative to third entity.

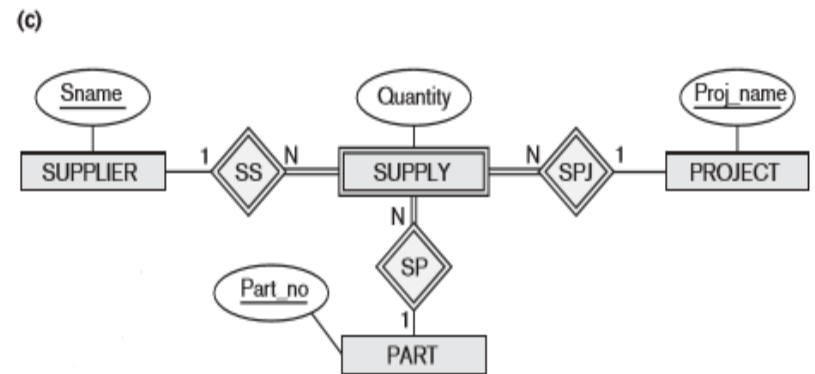
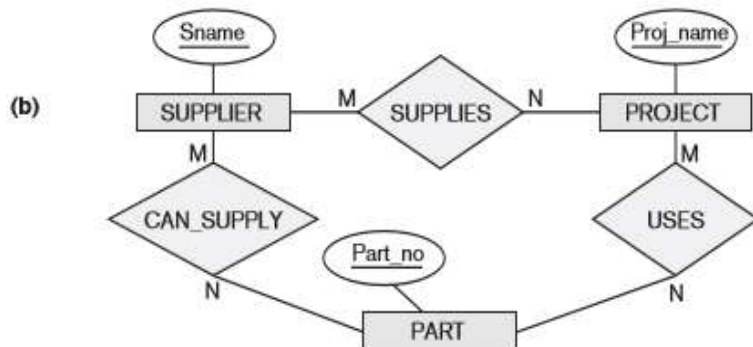


Choosing between Binary and Ternary (or Higher-Degree) Relationships

- Figure(a), which displays the schema for the SUPPLY relationship type that was displayed at the entity set or relationship set
- The relationship set of **SUPPLY** is a set of relationship instances (s, j, p) , where s is a **SUPPLIER** who is currently supplying a **PART** p to a **PROJECT** j .



- Figure(b) shows an ER diagram for three binary relationship types CAN_SUPPLY, USES, and SUPPLIES.
- Suppose that CAN_SUPPLY, between SUPPLIER and PART, includes an instance (s, p) whenever supplier s can supply part p
- USES, between PROJECT and PART, includes an instance (j, p) whenever project j uses part p, and SUPPLIES, between SUPPLIER and PROJECT, includes an instance (s, j) whenever supplier s supplies some part to project j.
- The existence of three relationship instances (s,p),(j,p),and (s,j) in CAN_SUPPLY, USES, and SUPPLIES, respectively, does not necessarily imply that an instance (s, j, p) exists in the ternary relationship SUPPLY, because the meaning is different.
- Some database design tools are based on variations of the ER model that permit only binary relationships.
- In this case, a ternary relationship such as SUPPLY must be represented as a weak entity type, with no partial key and with three identifying relationships.
- The three participating entity types SUPPLIER, PART, and PROJECT are together the owner entity types in Figure(c).
- Hence, an entity in the weak entity type SUPPLY in Figure 7.17(c) is identified by the combination of its three owner entities from SUPPLIER, PART, and PROJECT.



Cardinality in Ternary Relationship

- In a ternary relationship, **"cardinality"** refers to the specific number of instances of one entity that can be associated with a given combination of instances from the other two entities involved in the relationship
- ie, essentially defining how many "links" can exist between each pair of entities within the three-way connection
- It can be expressed as a combination of
 - one-to-one
 - one-to-many
 - many-to-many
- Often represented as a notation like "**M:N:P**" where M, N, and P represent the potential multiplicity of each entity involved.
- The cardinality constraint of an entity in a ternary relationship is defined by a pair of two entity instances associated with the other single entity instance.

Say for a given instance of Supplier and an Instance of Part, can that supplier supply that particular part for multiple Mobile models.

Example – Consider a Supplier S1 that supplies a Processor P1 to the company and the uses the Processor P1 supplied by Supplier S1 in its multiple Models in that case the cardinality of Mobile relative to Supplier and Part is N (many).

In case of Supplier's cardinality we can say for a given instance of Mobile one of its Part can be supplied by multiple Suppliers.

Example – Consider a Mobile M1 that has a Part P1 and it is being supplied by multiple Suppliers in that case the cardinality of Supplier relative to Mobile and Part is M (many).

Similarly, for a given instance of Supplier and an instance for Mobile does the Supplier supply multiple Parts.

Example – Consider a Supplier S1 supplying parts for Mobile M1 like screen, Processor etc. in that case the cardinality of Part relative to Supplier and Mobile is P (many).