UNIT - I

Introduction to Database Management System and ER Model

Indira College of Engineering Management, Pune

INTRODUCTION

Data: Collection of Information.

Database: The database is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently. It is also used to organize the data in the form of a table, schema, views, and reports, etc.

For example: The college Database organizes the data about the admin, staff, students and faculty etc. Using the database, you can easily retrieve, insert, and delete the information.

Database Management System: Database management system is a software which is used to manage the database. For example: MySQL, Oracle, etc are a very popular commercial database which is used in different applications.

INTRODUCTION

- DBMS provides an interface to perform various operations like database creation, storing data in it, updating data, creating a table in the database and a lot more.
- It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.

PURPOSE

- The purpose of database systems is to make the database user-friendly and do easy operations. Users can easily insert, update, and delete. Actually, the main purpose is to have more control of the data.
- The purpose of database systems is to manage the following insecurities:
    - data redundancy and inconsistency
    - difficulty in accessing data
    - data isolation
    - atomicity of updates
    - concurrent access
    - security problems
    - supports multiple views of data

- **Avoid data redundancy and inconsistency:**

  If there are multiple copies of the same data, it just avoids it. It just maintains data in a single repository. Also, the purpose of database systems is to make the database consistent.

- **Difficulty in accessing data:**

  A database system can easily manage to access data. Through different queries, it can access data from the database.

- **Data isolation:**

  Data are isolated in several fields in the same database.

- **Atomicity of updates:**

  In case of power failure, the database might lose data. So, this feature will automatically prevent data loss.

- **Concurrent access:**

  Users can have multiple access to the database at the same time.

- **Security problems:**

  Database systems will make the restricted access. So, the data will not be vulnerable.

- **Supports multiple views of data:**

  It can support multiple views of data to give the required view as their needs. Only database admins can have a complete view of the database. We cannot allow the end-users to have a view of developers.

Advantages of DBMS -

- Controlling Data Redundancy
- Data Consistency
- Sharing of Data
- Data Control
- Security
- Control over concurrency
- Data Modelling of Real World

Disadvantages -

- Increase Cost
- Complexity
- Size
- Higher impact of failure

Difference in between the File Processing System and DBMS -

FILE SYSTEM	**DBMS**
Redundant data is present	No presence of redundant data
Data consistency is low	Due to the process of normalization, the data consistency is high
Difficult to share data in traditional file system.	Data can be easily shared by different applications
The File System does not have centralized data control.	DBMS provides centralized data storage. Keeping control on data is very much easy.

Difference in between the File Processing System and DBMS -

FILE SYSTEM	DBMS
Less complex, does not support complicated transactions	More complexity in managing the data, easier to implement complicated transactions
Less security	Supports more security mechanisms
The file system approach is cheaper to design.	The database system is expensive to design.
The file system doesn't have a crash mechanism, i.e., if the system crashes while entering some data, then the content of the file will be lost.	DBMS provides a crash recovery mechanism, i.e., DBMS protects the user from system failure.

Difference in between the File Processing System and DBMS -

FILE SYSTEM	DBMS
In the file system approach, there is no concept of data models exists.	In the database approach, 3 types of data models exist: Hierarchal data models Network data models Relational data models
Cobol, C++ etc.	Oracle, SQL Server, MySQL etc.

Database-System Applications

Sr No.	Sector	Use of DBMS
1	Banking	For customer information, account activities, payments, deposits, loans, etc.
2	Airlines	For reservations and schedule information.
3	Universities	For student information, course registrations, colleges and grades.
4	Telecommunication	It helps to keep call records, monthly bills, maintaining balances, etc.

Database-System Applications

Sr No.	Sector	Use of DBMS
5	Education System	Schools, Colleges, teachers, exam schedules, accounts etc.
6	Sales	Use for storing customer, product & sales information.
7	HR Management	For information about employees, salaries, payroll, deduction, generation of paychecks, etc.

View of Data

Abstraction: **Data abstraction** allow developers to keep complex data structures away from the users. The developers achieve this by hiding the complex data structures through **levels of abstraction**.

Data abstraction is **hiding the complex data structure** in order to **simplify the user’s interface** of the system. It is done because many of the users interacting with the database system are not that much computer trained to understand the complex data structures of the database system.

Levels of Abstraction -

- Physical level (internal level)
- Logical level (conceptual level)
- View level (external level)

Physical Level -

- The physical or the internal level schema describes **how the data is stored in the hardware**. It also describes how the data can be accessed. The physical level shows the data abstraction at the lowest level and it has **complex data structures**. Only the database administrator operates at this level.

Logical Level -

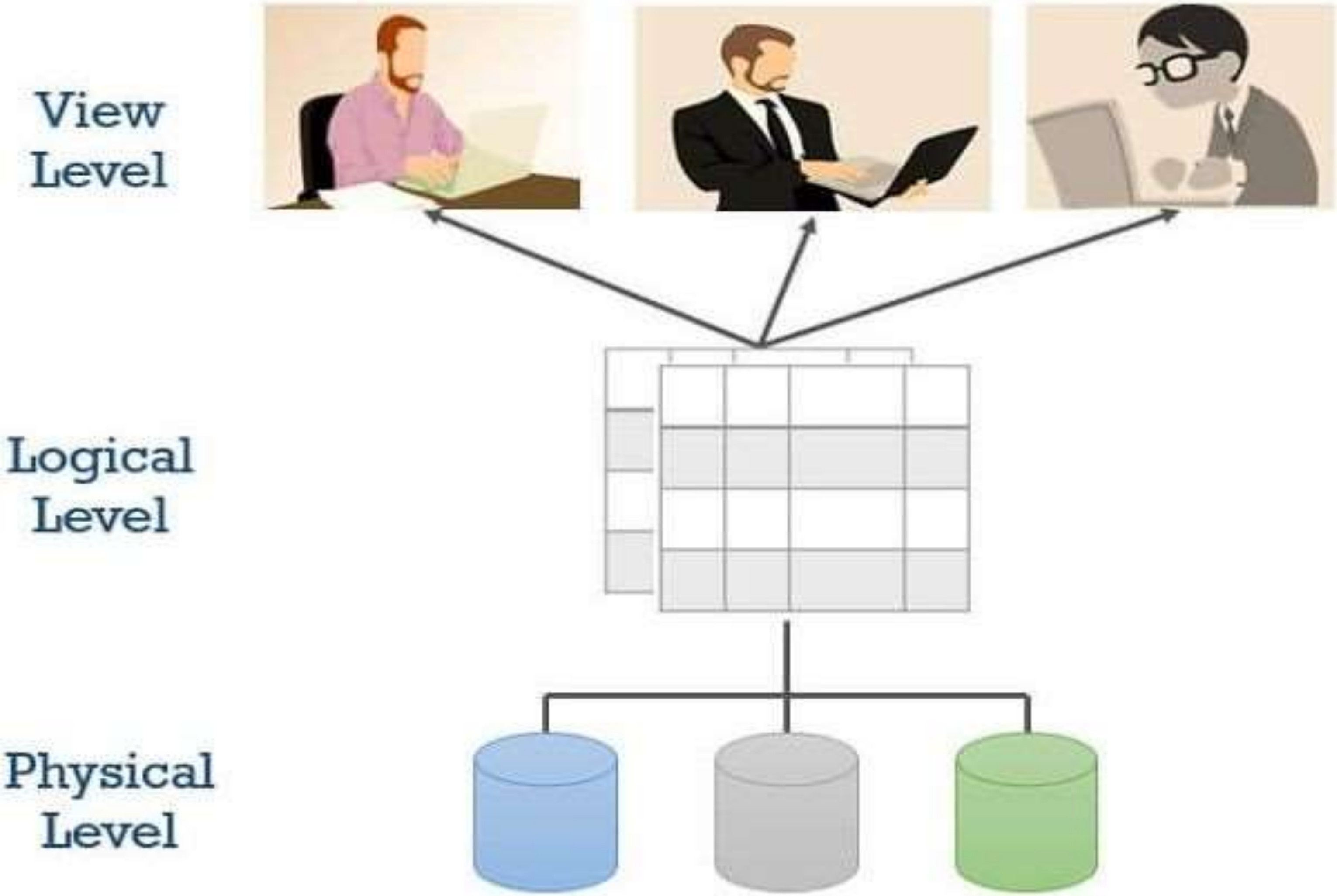
- It is a level above the physical level. Here, the data is stored in the form of the **entity set**, **entities**, their **data types**, the **relationship** among the entity sets, **user operations** performed to retrieve or modify the data and certain **constraints on the data**. Well adding constraints to the view of data adds the security. As users are restricted to access some particular parts of the database.
- It is the developer and database administrator who operates at the logical or the conceptual level.

View Level -

- It is the highest level of data abstraction and exhibits only a part of the whole database. It exhibits the data in which the user is interested. The view level can describe many views of the same data. Here, the user retrieves the information using different application from the database.

- The figure below describes the three-schema architecture of the database:

View of Data



Instances and Schemas -

What is an instance?

- We can define an instance as the information stored in the database at a particular point of time.
- The data in the database keeps on changing with time. As we keep inserting or deleting the data to and from the database.

What is schema?

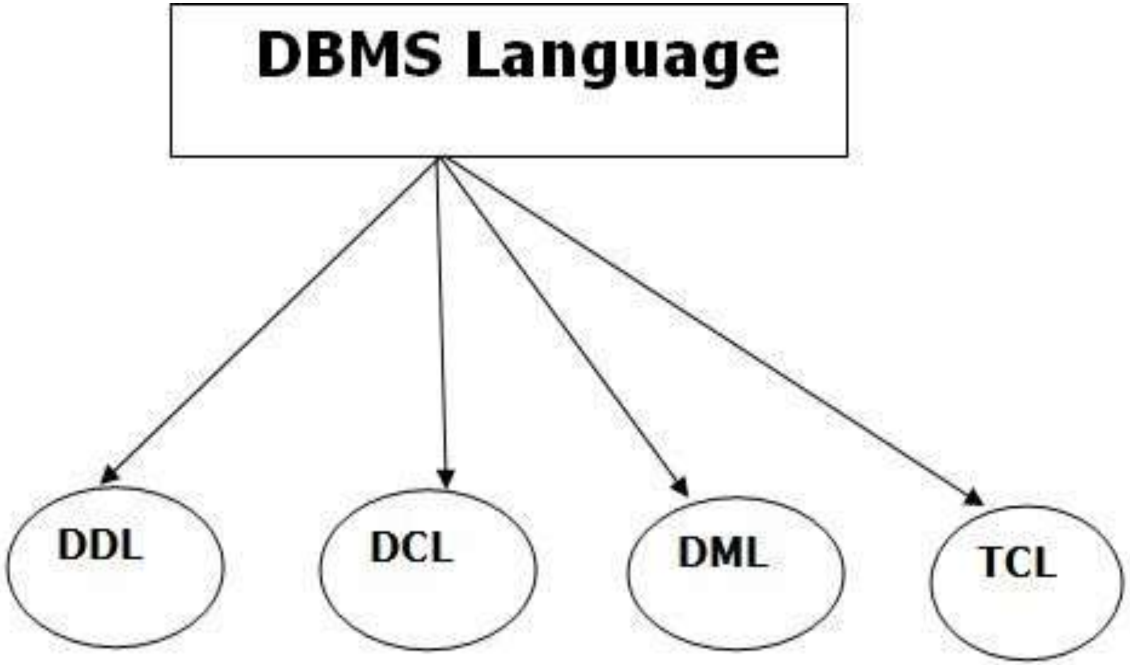
- The definition of a database comprises of the description of what data it would contain what would be the relationship between the data. This definition is the database schema.

- View of data in DBMS describes the abstraction of data at three-level i.e. **physical** level, **logical** level, **view** level.
- The physical level of abstraction defines how data is **stored** in the storage.
- Abstraction at the logical level describes **what data** would be stored in the database? what would be the **relation** between the data?
- The view level or external level of abstraction describes the **application** which the users use to retrieve the information from the database.
- An **instance** is the retrieval of information from the database at a certain point of time. An instance in a database keeps on **changing with time.**
- **Schema** is the overall design of the entire database. Schema of the database is not changed frequently.

Database Languages -

- A DBMS has appropriate languages and interfaces to express database queries and updates.
- Database languages can be used to read, store and update the data in the database.

Types of Database Languages -



Data Definition Language(DDL) -

DDL stands for **D**ata **D**efinition **L**anguage. It is used to define database structure or pattern.

It is used to create schema, tables, indexes, constraints, etc. in the database.

Using the DDL statements, you can create the skeleton of the database.

Data definition language is used to store the information of metadata like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.

Data Definition Language(DDL) -

Here are some tasks that come under DDL:

- **Create:** It is used to create objects in the database.
- **Alter:** It is used to alter the structure of the database.
- **Drop:** It is used to delete objects from the database.
- **Truncate:** It is used to remove all records from a table.
- **Rename:** It is used to rename an object.
- **Comment:** It is used to comment on the data dictionary.

These commands are used to update the database schema that's why they come under Data definition language.

Data Manipulation Language(DML) -

DML stands for **D**ata **M**anipulation **L**anguage. It is used for accessing and manipulating data in a database. It handles user requests.

Here are some tasks that come under DML:

- **Select:** It is used to retrieve data from a database.
- **Insert:** It is used to insert data into a table.
- **Update:** It is used to update existing data within a table.
- **Delete:** It is used to delete all records from a table.

Data Control Language(DCL) -

DCL stands for **D**ata **C**ontrol **L**anguage. It is used to retrieve the stored or saved data.

The DCL execution is transactional. It also has rollback parameters.

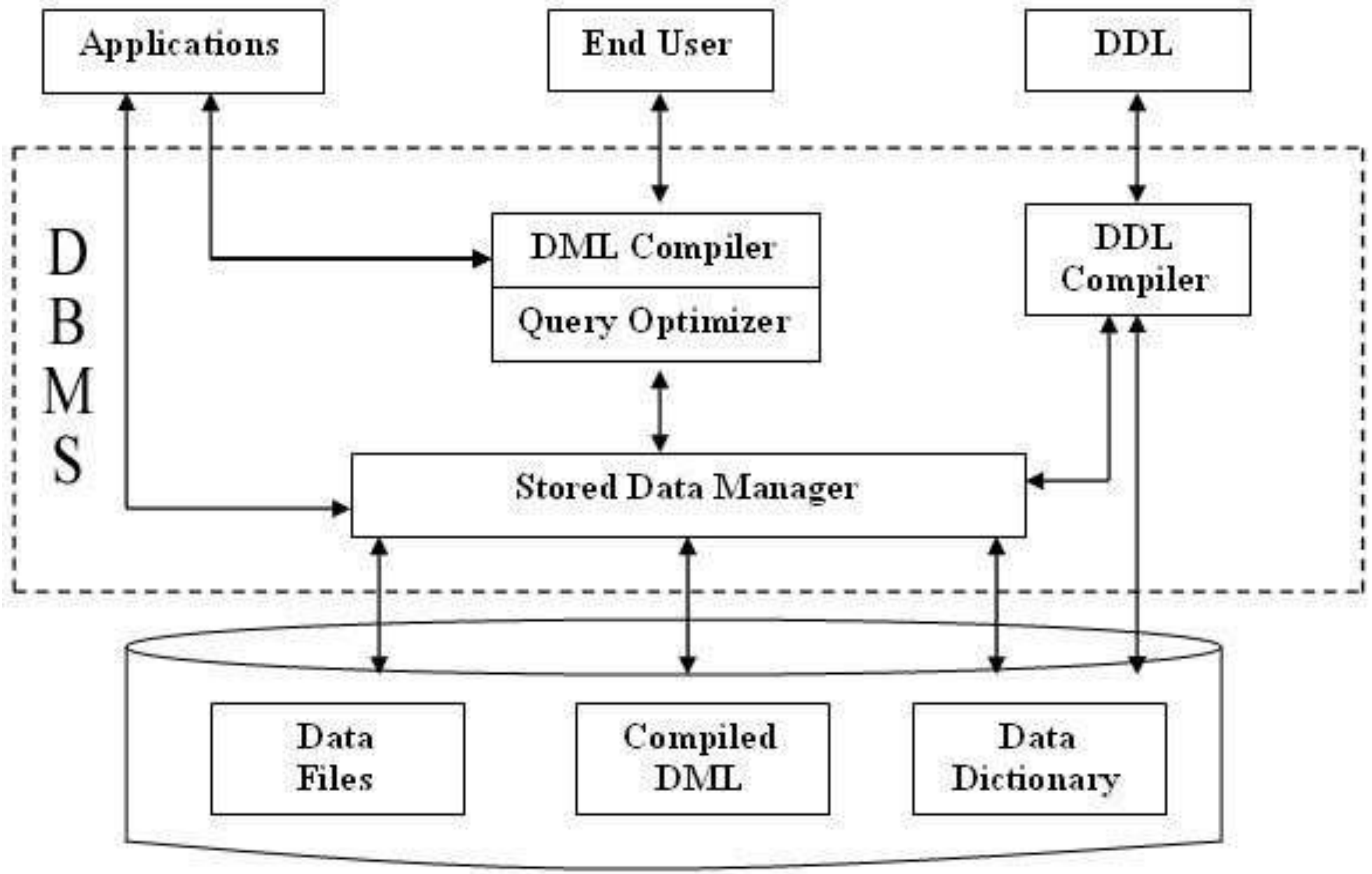
Here are some tasks that come under DCL:

- **Grant:** It is used to give user access privileges to a database.
- **Revoke:** It is used to take back permissions from the user.

There are the following operations which have the authorization of Revoke:

- CONNECT, INSERT, USAGE, EXECUTE, DELETE, UPDATE and SELECT.

Database System Structure -



Database System Structure -

- **Application –**

  It can be considered as a user-friendly web page where the user enter the requests. Here he simply enters the details that he needs and presses buttons to get the data.

- **End User –**

  They are the real users of the database. They can be developers, designers, administrators, or the actual users of the database.

Database System Structure -

- **DDL –**

  Data Definition Language is a query fired to create database, schemas, tables, etc in the database. They Create the structure of database.

- **DDL Compiler -**

  This part of the database is responsible for processing the DDL commands. That means this compiler actually breaks down the command into machine-understandable codes. It is also responsible for storing the metadata information like table name, space used by it, number of columns in it etc.

Database System Structure -

- **DML Compiler -**

  When the user inserts, deletes, updates or retrieve the record from the database, he will be sending requests which he understands by pressing some buttons. But for the database to work/understand the requests, it should be broken down to object code. This is done by this compiler.

- **Query Optimizer -**

  When a user fires some requests, whatever be the request, it should be efficient enough to fetch, insert, update, or delete the data from the database. The query optimizer decides the best way to execute the user request which is received from the DML compiler.

Database System Structure -

- **Stored Data Manager** –
  - This is also known as Database Control System. It is one of the main central systems of the database.
  - It convers the requests received from query optimizer to machine-understandable form. It makes actual requests inside the database.
  - It helps to maintain consistency and integrity by applying the constraints.
  - It controls concurrent access.
  - It guarantees that there is no data loss or data mismatch happens between the transaction of multiple users.
  - It helps to back up the database and recovers data whenever required.

Database System Structure -

- **Data Files** – It has the real data stored in it.
- **Compiled DML** – Some of the processed DML statements (insert, update, delete) are stored in it so that if there are similar requests, it will be re-used.
- **Data Dictionary** – It contains all the information about the database. As the name suggests, it is the dictionary of all the data items. It contains a description of all the tables, view, ,materialized views, constraints, indexes, triggers, etc.

Data Models -

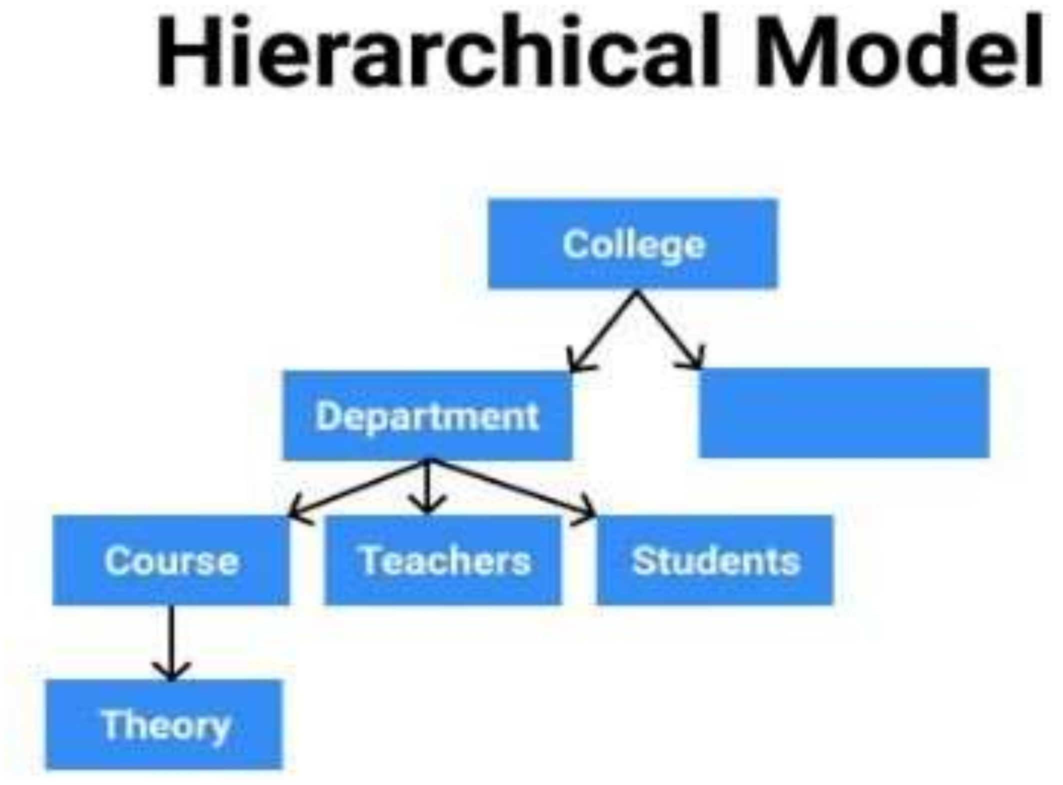
Types of Data Models in DBMS -

- Hierarchical database model
- Relational model
- Network model
- Entity-relationship model
- Object-oriented database model

Hierarchical database model -

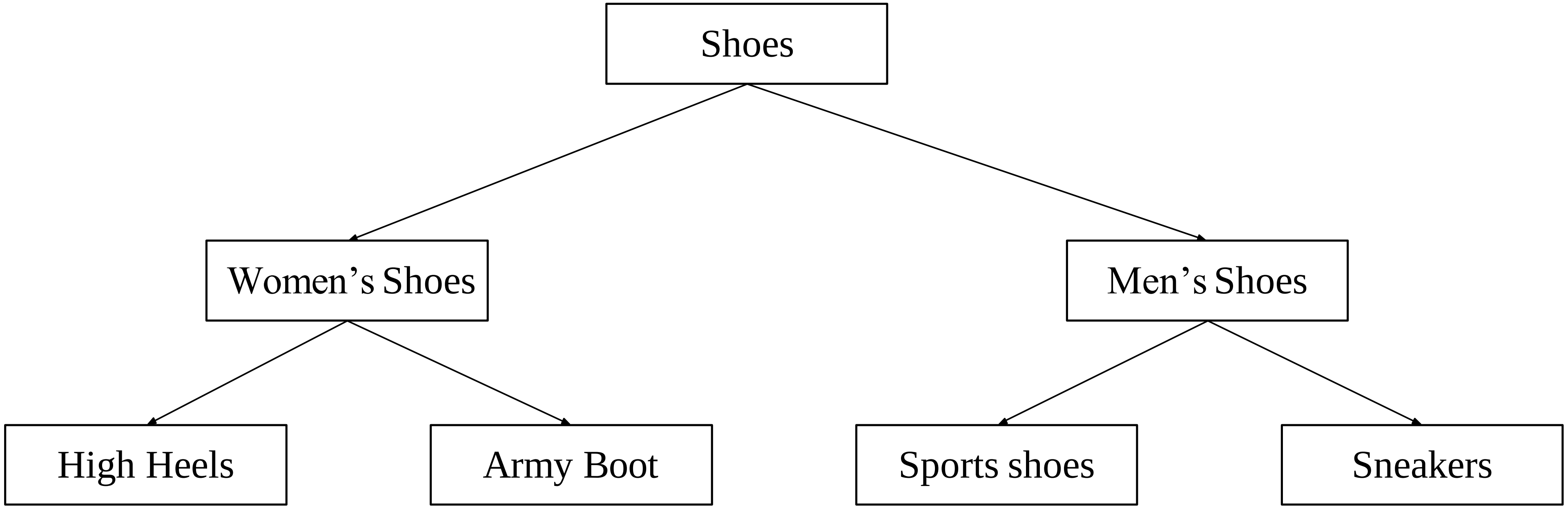
- Data is organized in tree structure.
- It contains Parent-Child relationship where root of the tree is a parent which then branches into its children. It is another type of record based data model.
- The data is stored in the form of records. These records are connected to one another.
- A record is a collection of fields; each field contains only one value. The hierarchical data models represent data according to its hierarchy.

Hierarchical database model -



Hierarchical database model -

- Example – We can represent the relationship between shoes present on a shopping website in the following way -



Shoes

Women’s Shoes

Men’s Shoes

High Heels

Army Boot

Sports shoes

Sneakers

Hierarchical database model -

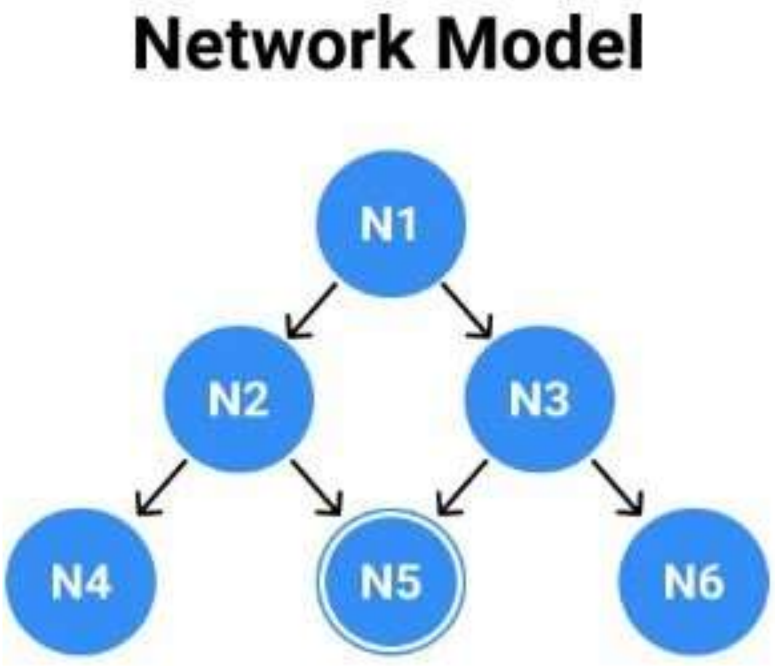
Advantages –

- Simple to use
- Database integrity
- Efficient

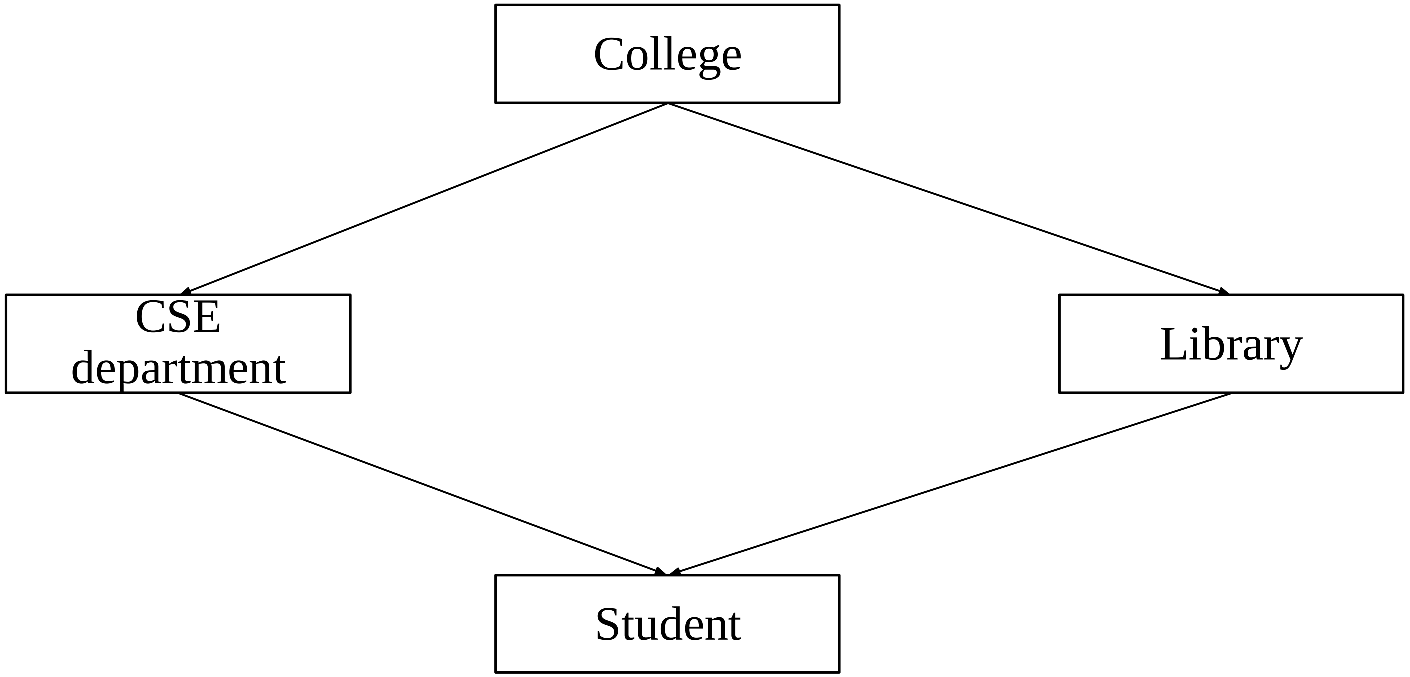
Network model -

- The network model was created to represent complex data relationships more effectively when compared to hierarchical models, to improve database performance and standards.
- It has entities which are organized in a graphical representation and some entities are accessed through several paths.
- A User perceives the network model as a collection of records in 1:M relationships.

Network model -



Network model -


College

CSE department

Library

Student

Network model -

The features of a Network Model are as follows −

- **Ability to Merge Relationships** − It has an ability to manage one-to-one relationships as well as many-to-many relationships.
- **Many paths** − There can be more than one path to the same record because of more relationships. It makes data access fast and simple.

Network model -

Advantages –

- Network models represent complex data relationships better than the hierarchical models.
- It handles so many relationship types.
- Data access is more flexible than hierarchical models.
- Improved database performance.
- It includes Data Definition Language (DDL) and Data Manipulation Language (DML) commands.

Entity – Relationship Model -

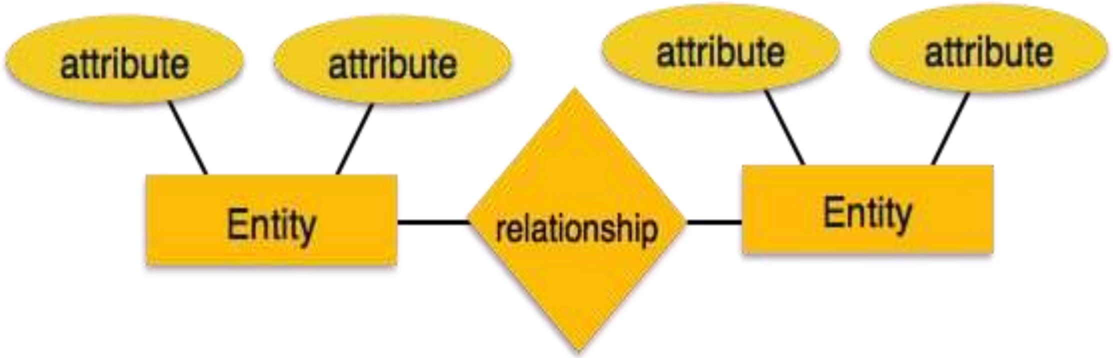
Entity-Relationship (ER) Model is based on the notion of real-world entities and relationships among them. While formulating real-world scenario into the database model, the ER Model creates entity set, relationship set, general attributes and constraints.

ER Model is best used for the conceptual design of a database.

ER Model is based on −

- **Entities** and their *attributes.*
- **Relationships** among entities.

Entity – Relationship Model -

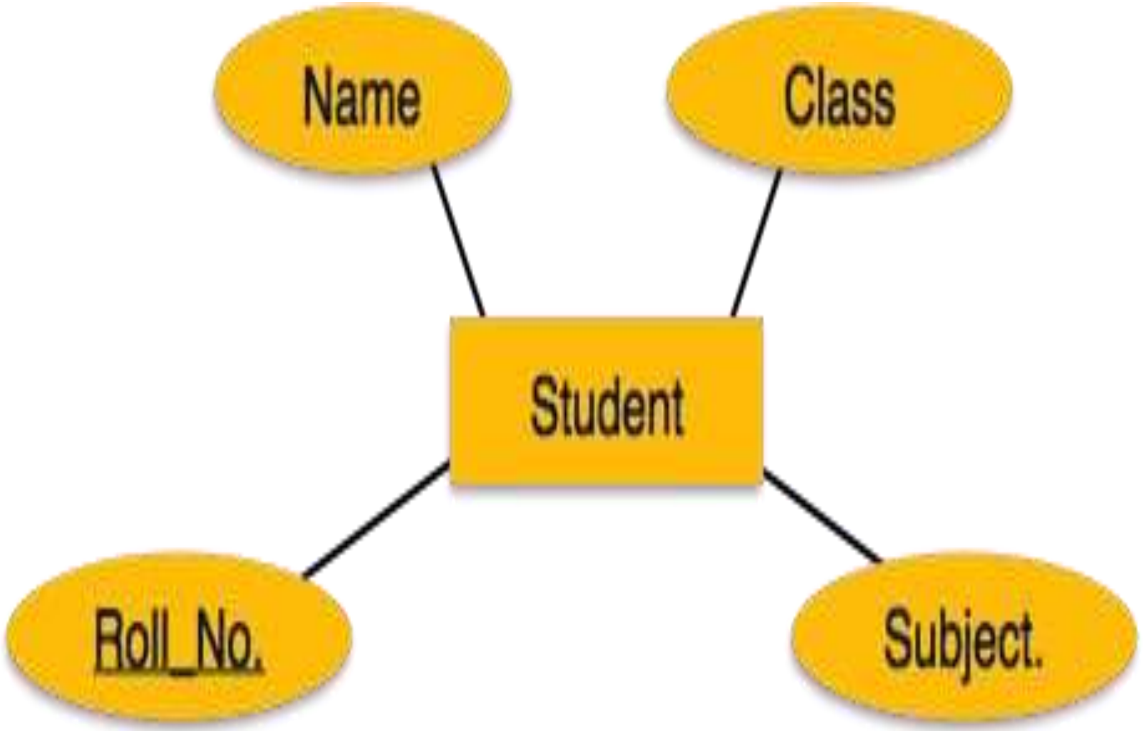


Entity − An entity in an ER Model is a real-world entity having properties called **attributes**. Every **attribute** is defined by its set of values called **domain**. For example, in a school database, a student is considered as an entity. Student has various attributes like name, age, class, etc.

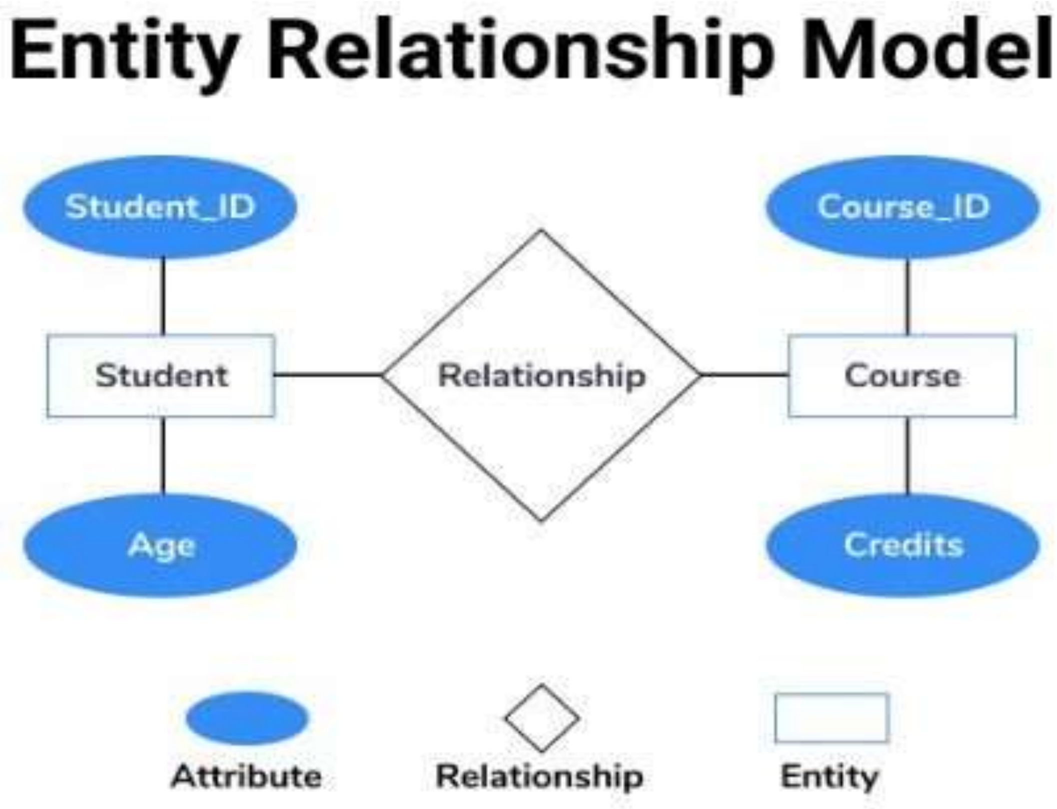
Relationship − The logical association among entities is called ***relationship***. Relationships are mapped with entities in various ways. Mapping cardinalities define the number of association between two entities. Mapping cardinalities −

- one to one
- one to many
- many to one
- many to many

Entity – Relationship Model -



Entity – Relationship Model -



Relational Model -

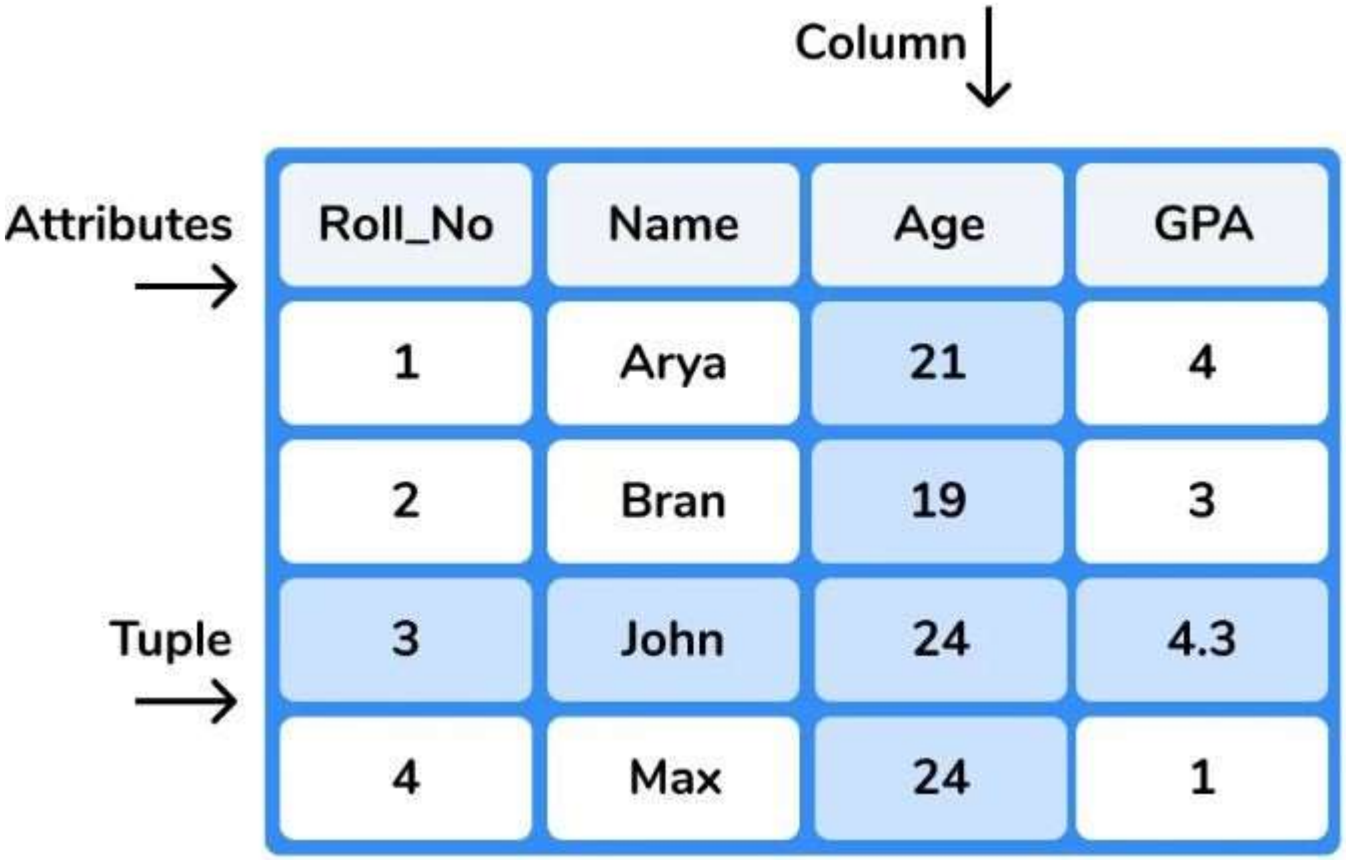
Relational database is an attempt to simplify the data structure by making use of tables. Tables are used to represent the data and their relationships. Table is a collection of rows and columns. Tables are also known as relations.

Records known as tuples and fields known as attributes.

The relational model is called as record based model because the database is structured in fixed format records of different types. A record is consists of fields or attributes.

In the relational model, every record must have a unique identification or key based on the data.

Relational Model -


Column
Attributes
Tuple

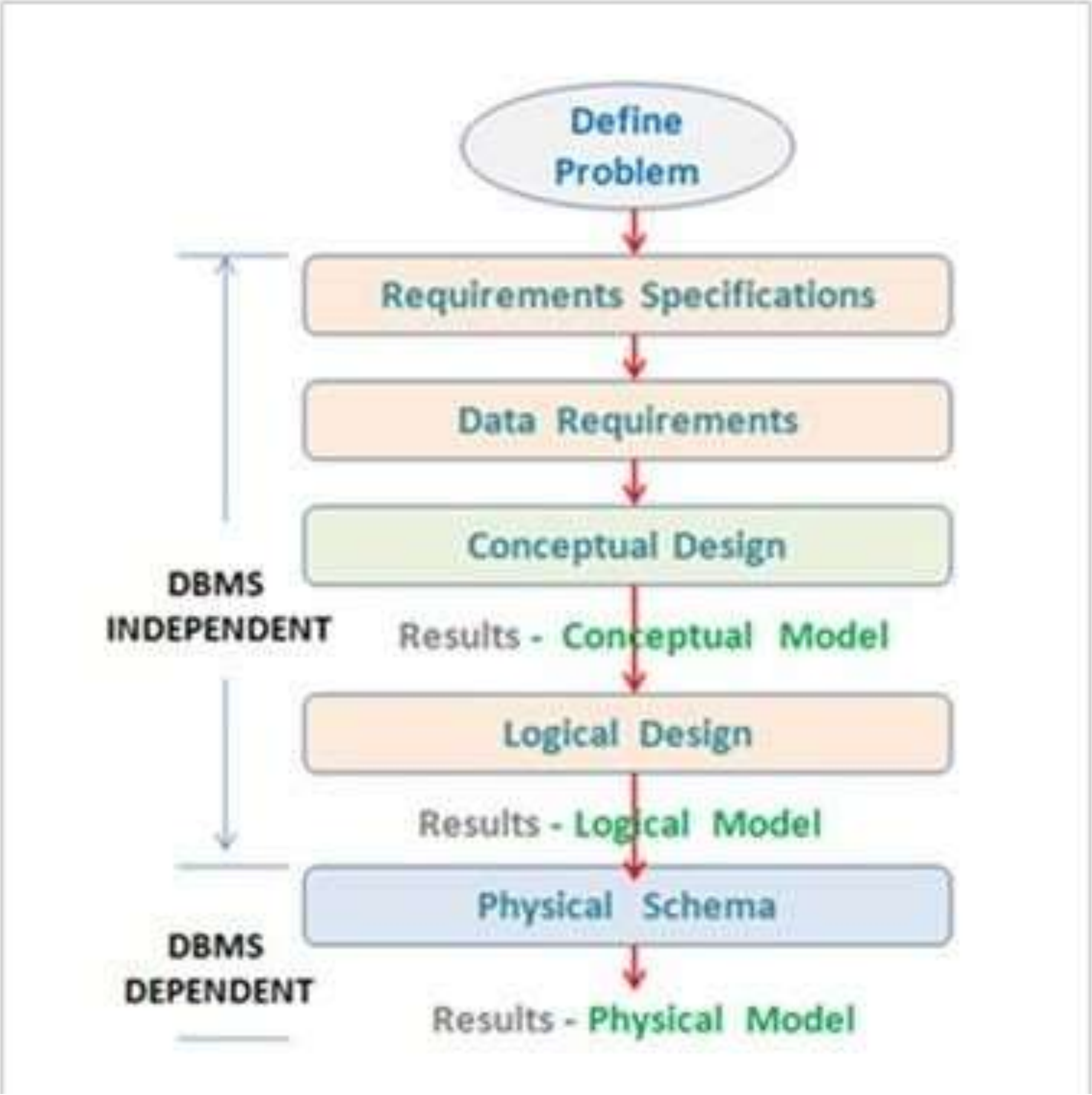
Roll_No	Name	Age	GPA
1	Arya	21	4
2	Bran	19	3
3	John	24	4.3
4	Max	24	1

Database Design and ER Model -

Database Design Process -

- **Database Design**: The overall success of the system is completely depends upon the design of the database. The design is mainly focused on the security and access of the data by the application program. The requirements of user plays an important role in database design.
- The process of doing database design generally consists of a number of steps which will be carried out by the database designer.

Database Design -



Database Design -

1. Requirement Analysis

- Planning has to be done on what are the basic requirements of the project under which the design of the database has to be taken forward.
- Database designers understand and document the data requirement of the database users.

2. Conceptual Design:

- Detailed description of system.
- Concise description of the data requirements and detailed description of the entity types, relationships, attributes and constraints.

Database Design -

3. Logical Design –

- Actual implementation of the database, using commercial DBMS.
- Most Commercial DBMS uses implementation data model.

4. Physical Design –

- The internal storage structures, indexes, access paths – Specified.

Database Design -

- Example – Database Application COMPANY.

1. Requirements Gathered –

   Company is organized into department.

   A department controls no. of Projects.

   Department needs Employees.

2. Conceptual Design –

   Department (Dept_Name, Dept_No., Location, Manager)

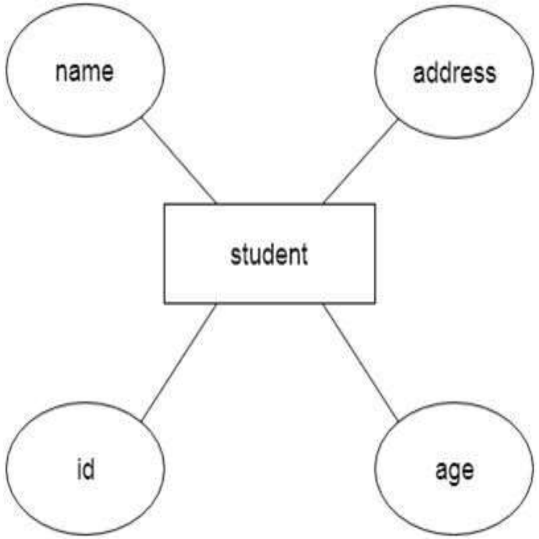
   Project (Pro_Name, Project_No.)

   Employee (Emp_id, Emp_Name, Salary, Address)

Database Design and ER Model -

- **ER Model:** In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram.

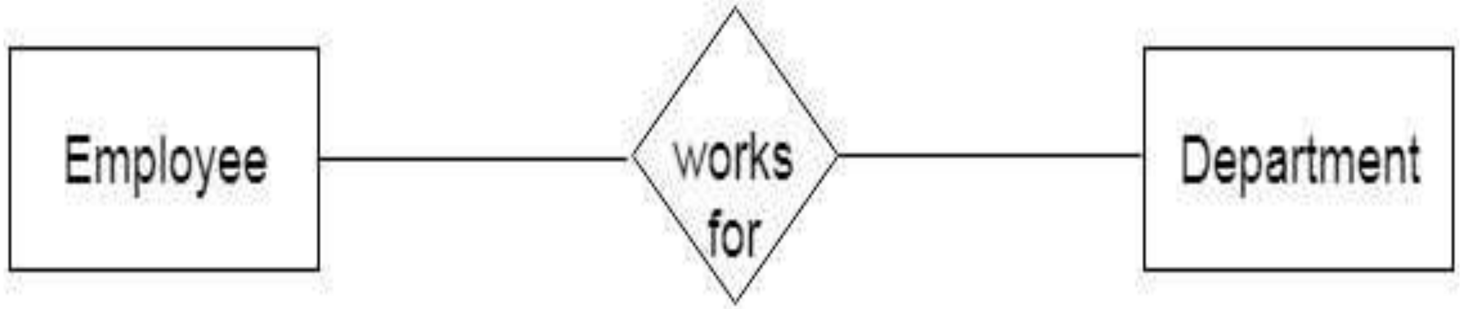
  For example, Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age, etc. The address can be another entity with attributes like city, street name, pin code, etc and there will be a relationship between them.



Database Design and ER Model -

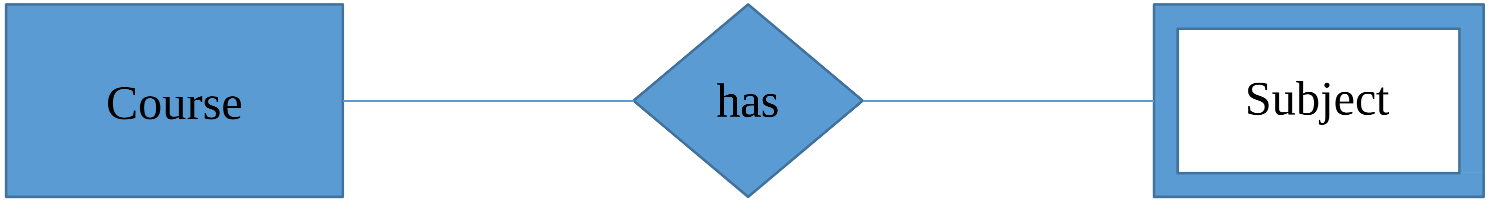
1. Entity:

- An entity may be any object, class, person or place. In the ER diagram, an entity can be represented as rectangles.
- Consider an organization as an example- manager, product, employee, department etc. can be taken as an entity.



Database Design and ER Model -

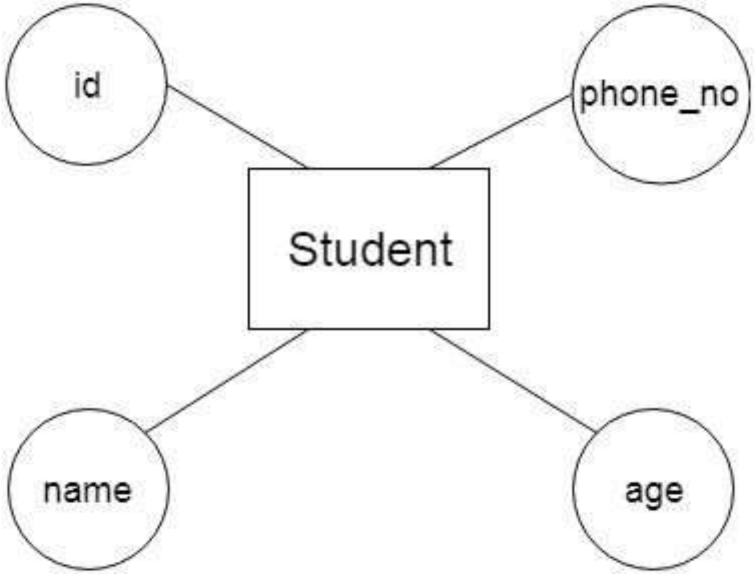
- **Weak Entity-** An entity that depends on another entity called a weak entity. The weak entity doesn't contain any key attribute of its own. The weak entity is represented by a double rectangle.



Database Design and ER Model -

2. Attribute: The attribute is used to describe the property of an entity. Oval is used to represent an attribute.

For example, id, age, contact number, name, etc. can be attributes of a student.

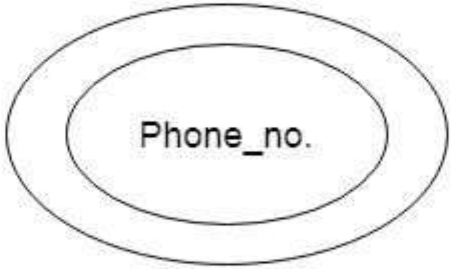


Database Design and ER Model -

i. **Single-valued Attribute** – A single attribute is the attribute which can hold a single value for the single entity.

ii. **Multi-valued Attribute** - An attribute can have more than one value. These attributes are known as a multivalued attribute. The double oval is used to represent multivalued attribute.

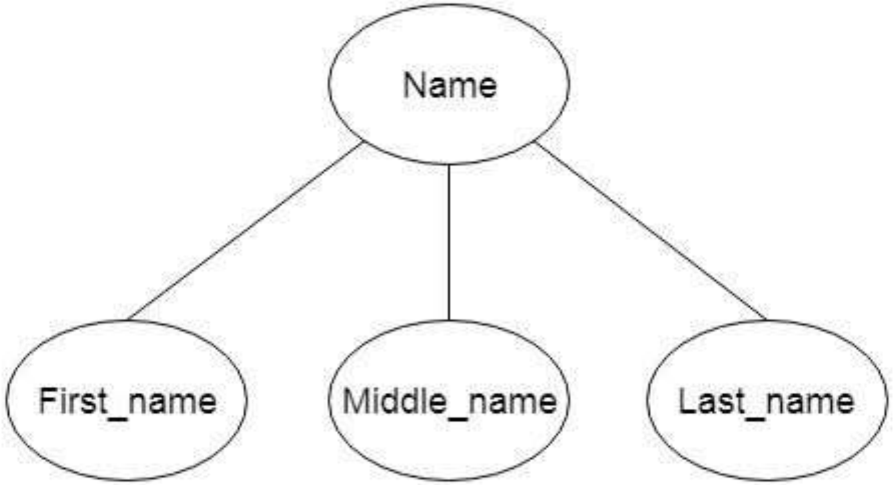
For example, a student can have more than one phone number.



Database Design and ER Model -

iii. Composite Attribute

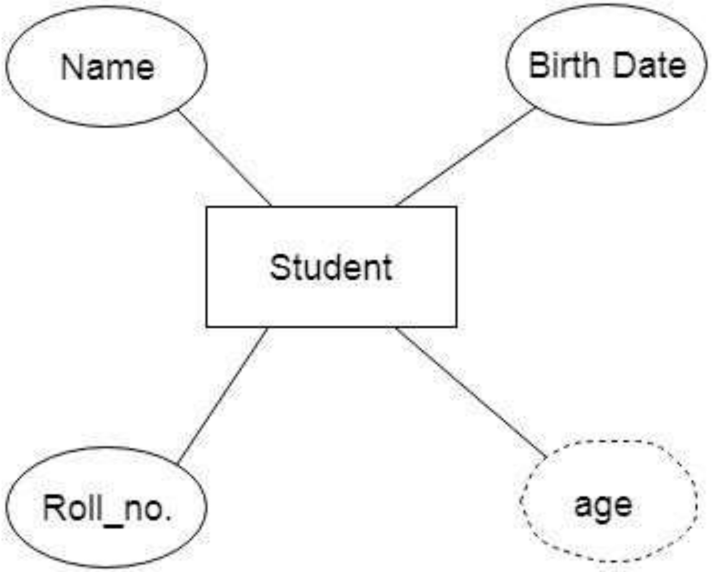
An attribute that composed of many other attributes is known as a composite attribute. The composite attribute is represented by an oval, and those ovals are connected with an ovals.



Database Design and ER Model -

iv. Derived Attribute

- An attribute that can be derived from other attribute is known as a derived attribute. It can be represented by a dashed ellipse.
- **For example,** A person's age changes over time and can be derived from another attribute like Date of birth.



Database Design and ER Model -

3. Relationships – The association between two different entities is called as relationship.

For example – An employee works at a department, a students enrolls for a course.

The degree of Relationships – It refers to numbers of entities participated in the relation.

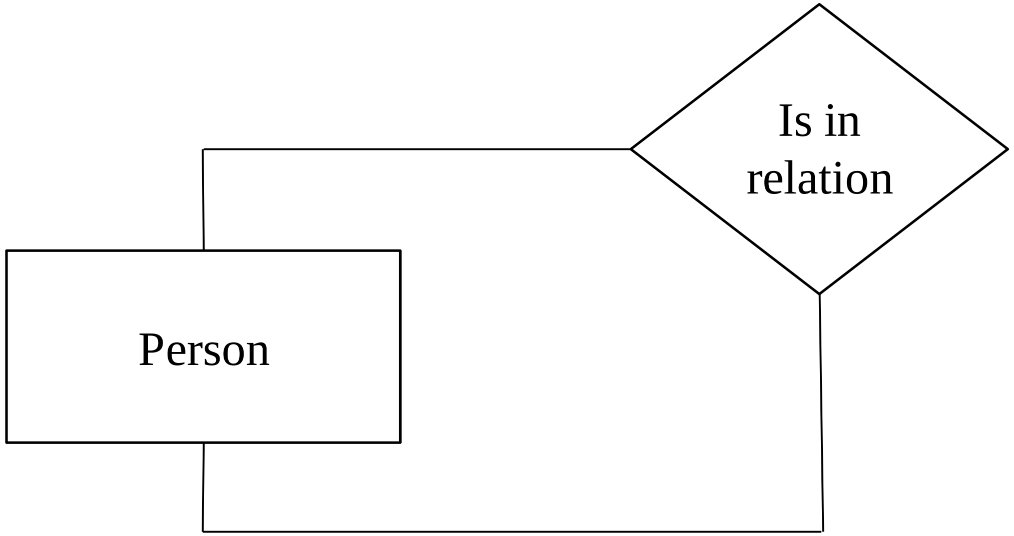
1. Unary Relationship
2. Binary Relationship
3. Ternary Relationship
4. Quaternary Relationship

1. **Unary Relationship** – A unary relationship exists when there is relation between single entity.

Example – A person can be in the relationship with another person.

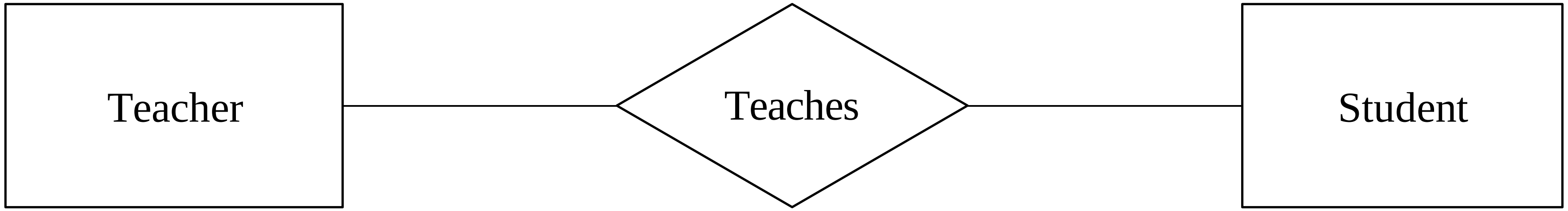
A woman who can be someone’s mother.

A person that is a someone’s child.



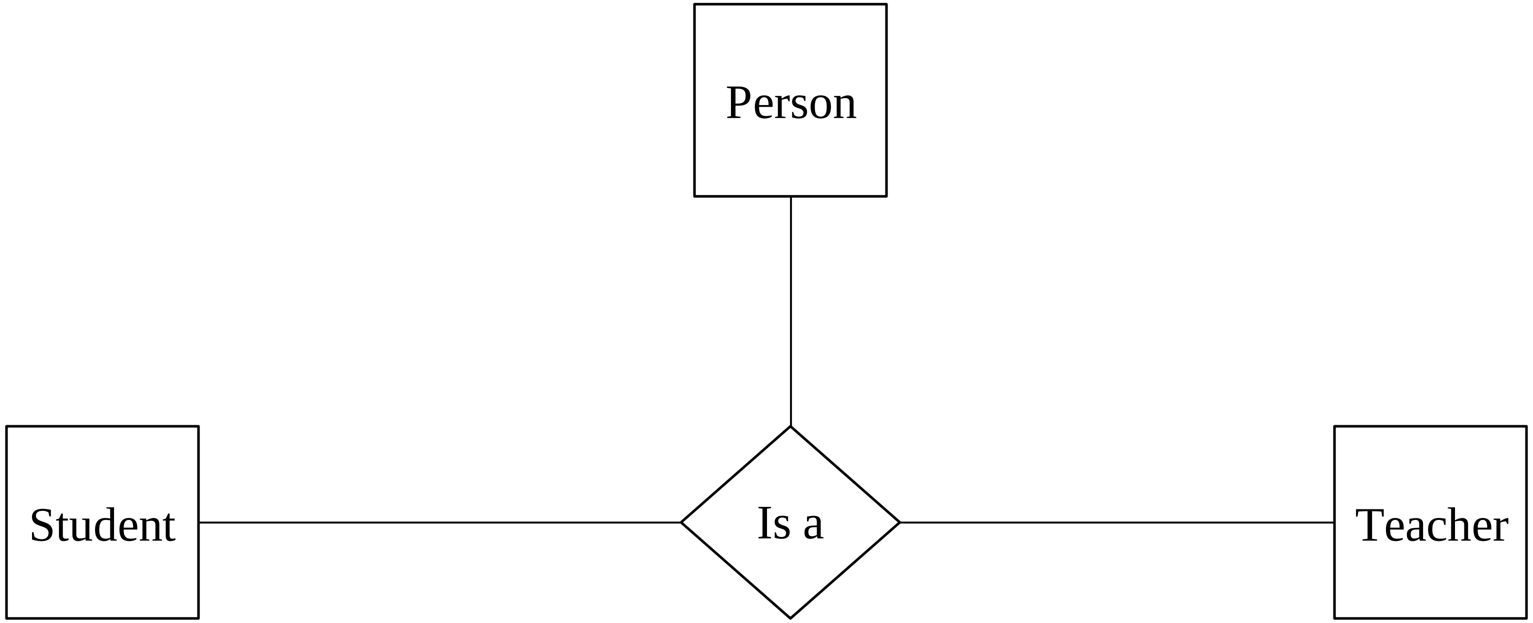
2. **Binary Relationship** – A binary relationship exists when there is relation between only two entities.

Example – A teacher teaches student.



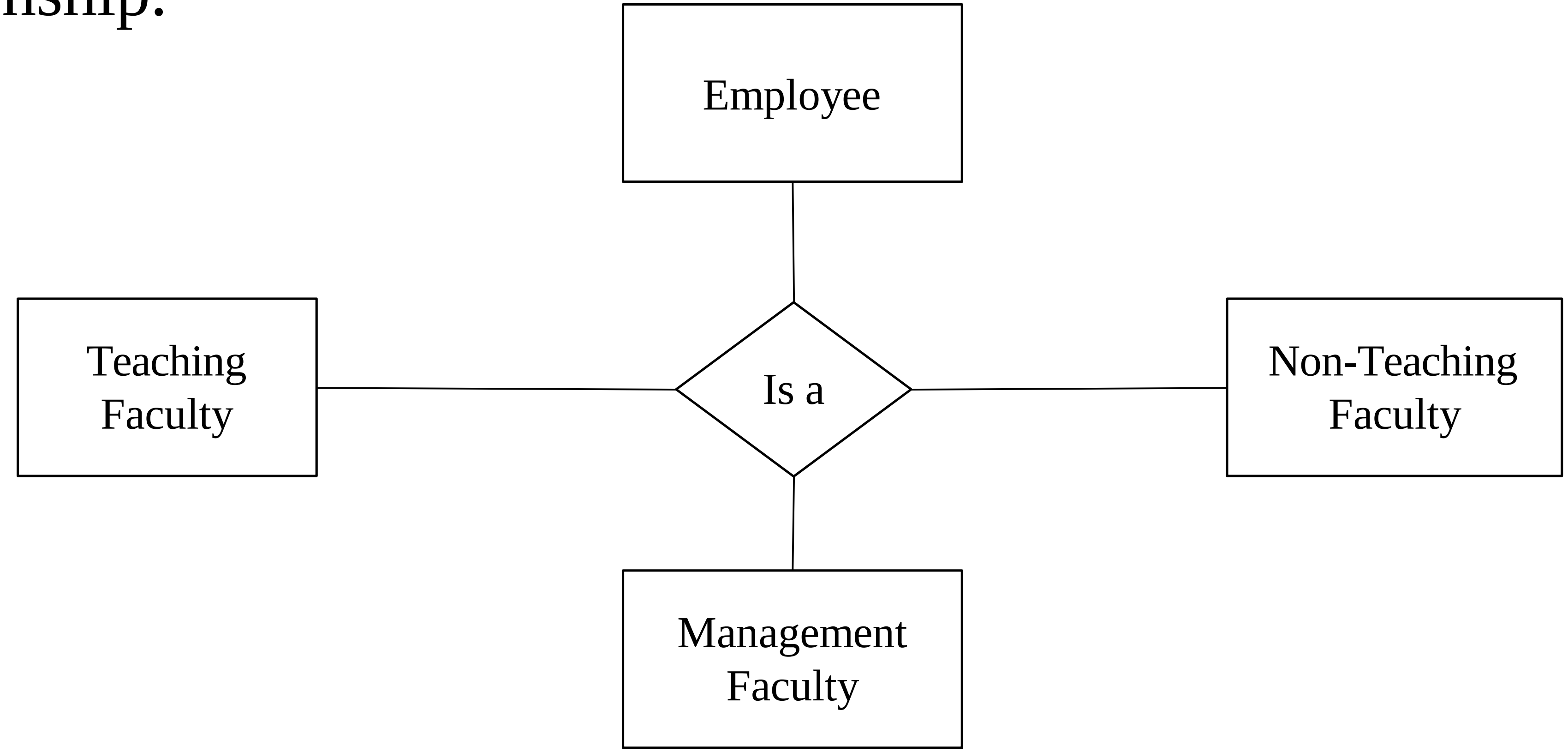
3. **Ternary Relationship** – A ternary relationship exists when there are relations between three entities.

Example – A person can be a student and a person also can be teacher.



4. **Quaternary Relationship** – A quaternary relationship exists when there are relations between four entities.

Example – Employee, Management Faculty, Teaching Faculty, and Non-Teaching Faculty are connected with each other via is a relationship.



Database Design and ER Model -

4. Constraints: Constraints are **the rules enforced on the data columns of a table**. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database. Constraints could be either on a column level or a table level.

Database Design and ER Model -

Relational Constraints

- These are the restrictions or sets of rules imposed on the database contents. It validates the quality of the database. It validates the various operations like data insertion, updation, and other processes that have to be performed without affecting the integrity of the data. It protects us against threats/damages to the database.
- Mainly Constraints on the relational database are of 4 types –

1. Domain constraints
2. Key constraints or Uniqueness Constraints
3. Entity Integrity constraints
4. Referential integrity constraints

Database Design and ER Model -

Relational Constraints

1. Domain Constraints

Every domain must contain atomic values(smallest indivisible units) which means composite and multi-valued attributes are not allowed.

We perform a datatype check here, which means when we assign a data type to a column we limit the values that it can contain. Eg. If we assign the datatype of attribute age as int, we can’t give it values other than int datatype.

Database Design and ER Model -

Relational Constraints

2. Key Constraints or Uniqueness Constraints

- These are called uniqueness constraints since it ensures that every tuple in the relation should be unique.
- A relation can have multiple keys or candidate keys(minimal superkey), out of which we choose one of the keys as the primary key, we don't have any restriction on choosing the primary key out of candidate keys, but it is suggested to go with the candidate key with less number of attributes.
- Null values are not allowed in the primary key, hence Not Null constraint is also part of the key constraint.

Database Design and ER Model -

Relational Constraints

3. Entity Integrity Constraints

Entity Integrity constraints say that no primary key can take a NULL value, since using the primary key we identify each tuple uniquely in a relation.

Database Design and ER Model -

Relational Constraints

4. Referential Integrity Constraints

The Referential integrity constraint is specified between two relations or tables and used to maintain the consistency among the tuples in two relations.

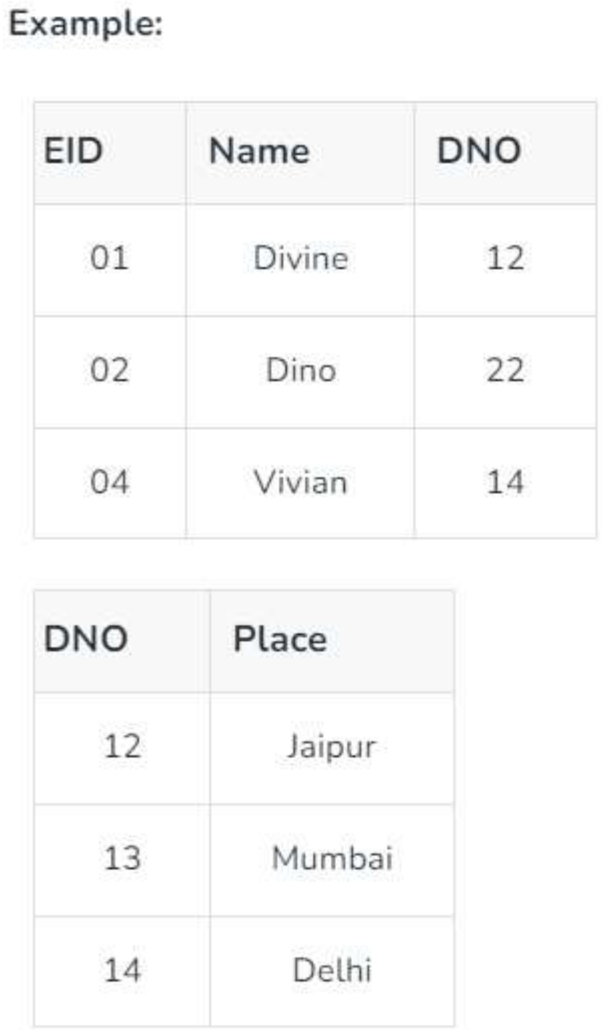
This constraint is enforced through a foreign key, when an attribute in the foreign key of relation R1 has the same domain(s) as the primary key of relation R2, then the foreign key of R1 is said to reference or refer to the primary key of relation R2.

The values of the foreign key in a tuple of relation R1 can either take the values of the primary key for some tuple in relation R2, or can take NULL values, but can’t be empty.

Database Design and ER Model -

Relational Constraints

4. Referential Integrity Constraints



Example:

EID	Name	DNO
01	Divine	12
02	Dino	22
04	Vivian	14

DNO	Place
12	Jaipur
13	Mumbai
14	Delhi

1. Mapping Cardinalities: -

Cardinality in DBMS –

Cardinality refers to the uniqueness of data values contained in a column.

High Cardinality – Column contains large percentage of totally unique values.

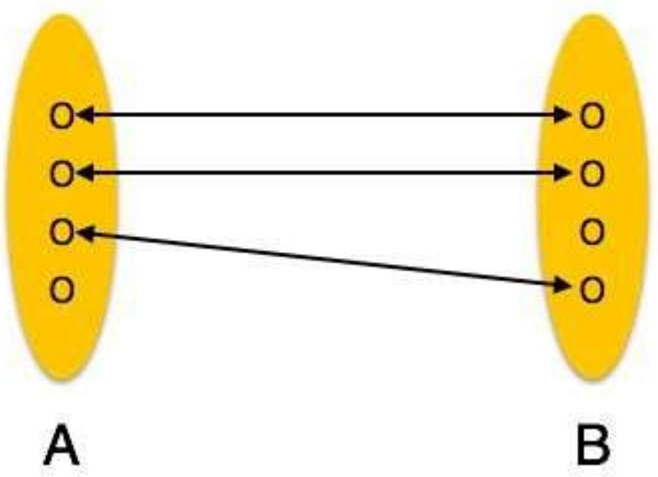
Low Cardinality – Column contains a lot of repeat in its data range.

Cardinality refers to the relationship between tables. Cardinality between tables can be One-to-one, One to Many, Many-to-One, Many-to-Many.

A relationship where two entities are participating called a binary relationship.

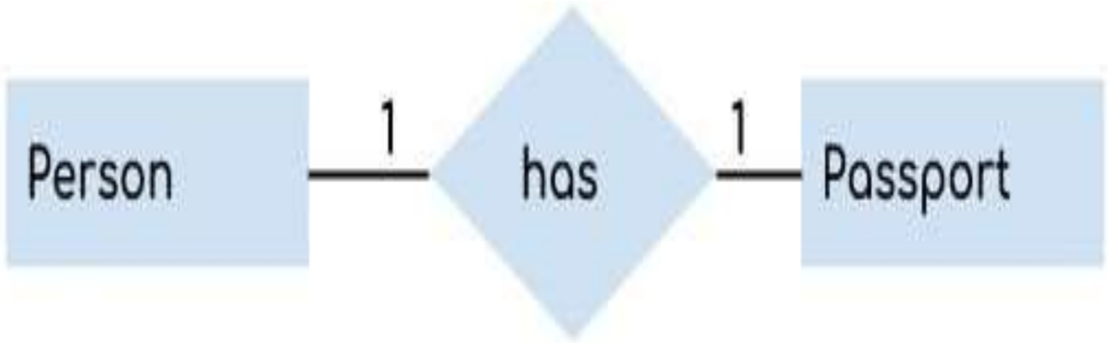
One-to-One

- When a single instance of an entity is associated with a single instance of another entity then it is called one to one relationship.
- **For example,** A female can marry to one male, and a male can marry to one female.



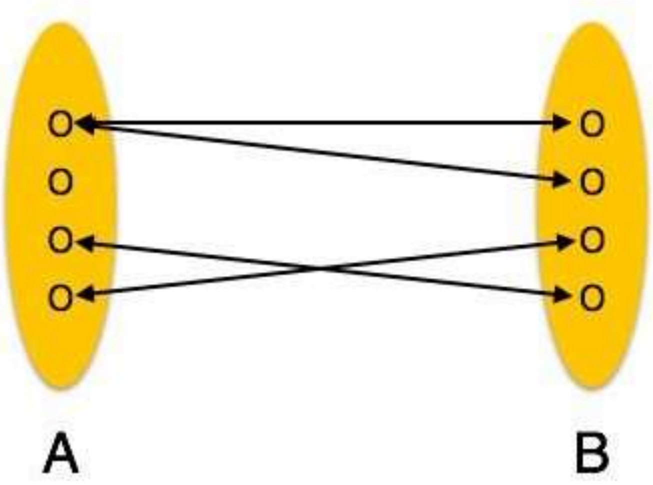
One-to-One

For example, a person has only one passport and a passport is given to one person.



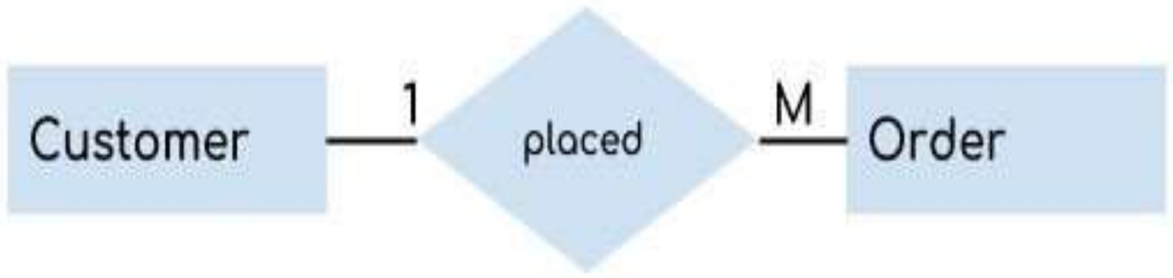
One-to-Many

When a single instance of an entity is associated with more than one instances of another entity then it is called one to many relationship.



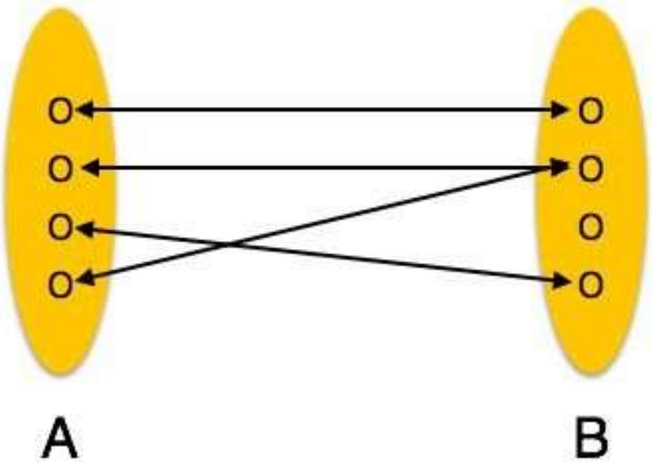
One-to-Many

For example – a customer can place many orders but a order cannot be placed by many customers.



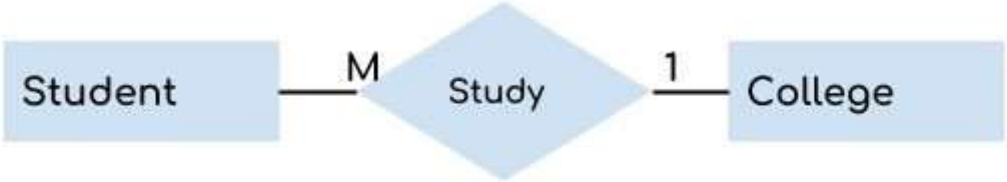
Many-to-One

When more than one instances of an entity is associated with a single instance of another entity then it is called many to one relationship.



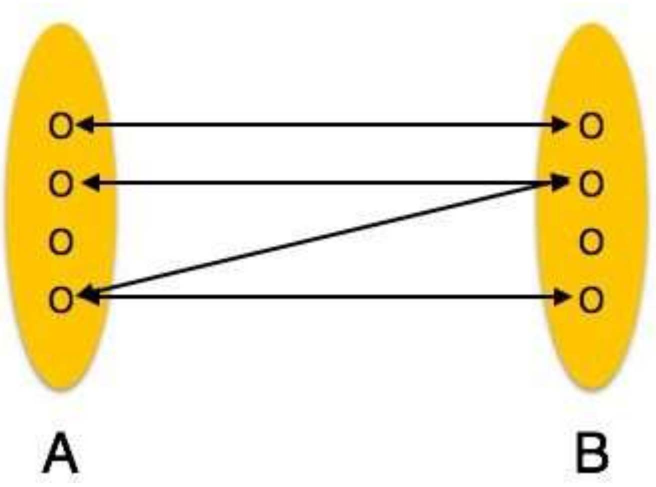
Many-to-One

For example – many students can study in a single college but a student cannot study in many colleges at the same time.



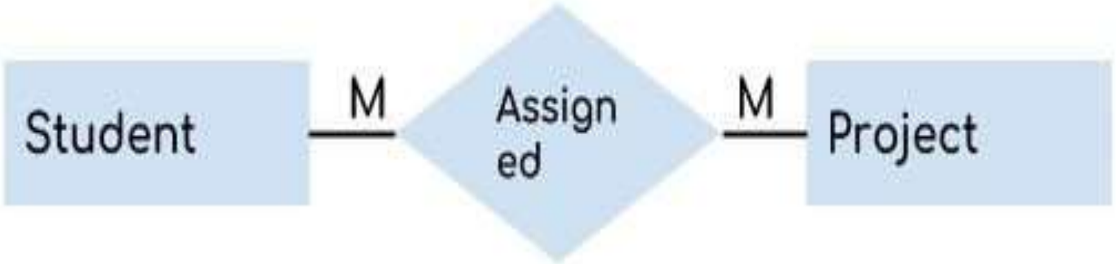
Many-to-Many

When more than one instances of an entity is associated with more than one instances of another entity then it is called many to many relationship.



Many-to-Many

For example, Students can be assigned to many projects and a project can be assigned to many students.

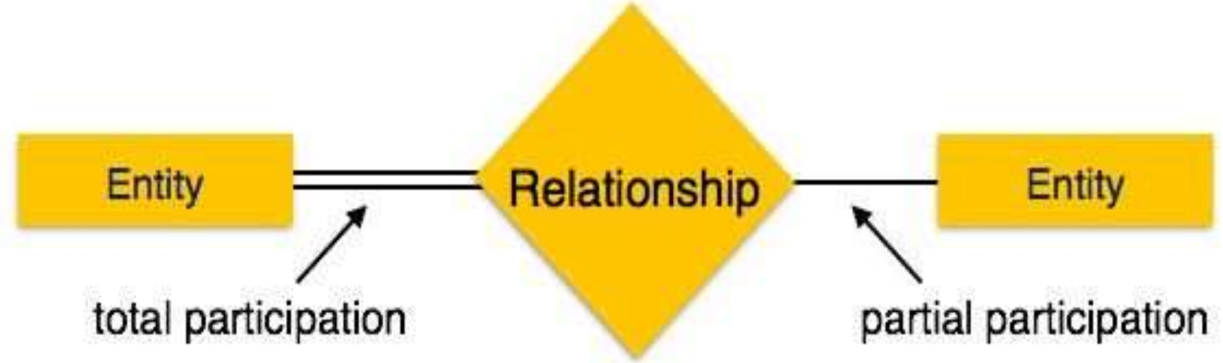


2. **Participation Constraints:**

i. **Total Participation - Each entity is involved in the relationship**. If each student must enroll in a course, the participation of student will be total. Total participation is represented by double lines.

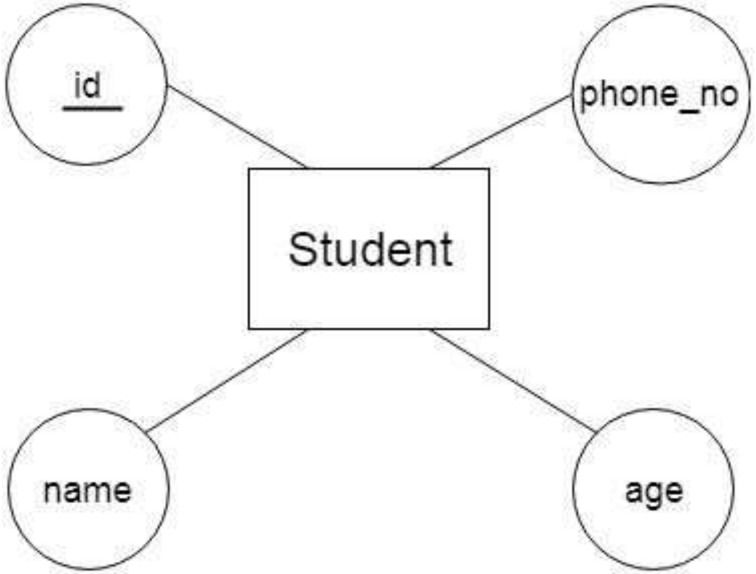
ii. **Partial Participation -** Not all entities are involved in the relationship. Partial participation is represented by single lines.

If some courses are not enrolled by any of the student, the participation of course will be partial.



5. Keys Attribute

Key Attribute is **an attribute or a set of attributes that help to uniquely identify a tuple (or row) in a relation (or table)**. Keys are also used to establish relationships between the different tables and columns of a relational database. Individual values in a key are called key values. The key attribute is represented by an oval with the text underlined.



There are some types of keys:

i. Primary Key
ii. Super Key
iii. Alternate Key
iv. Candidate Key
v. Foreign Key

i) Primary Key – A primary key is **the column or columns that contain values that uniquely identify each row in a table**. A database table must have a primary key for Optim to insert, update, restore, or delete data from a database table. Optim uses primary keys that are defined to the database.

Example – In Employee database, the Employee_id should be primary key. Because this field cannot be kept NULL as well as no Employee_id should be repeated.

ii) Super Key - **Super Key** is the one or more attributes of the entity, which uniquely identifies the record in the database.

Example – In the Employee database having attributes Employee Name, Employee_id, Employee_DOB.

iii) Candidate Key – A candidate key is a subset of a super key set where the key which contains no redundant attribute is none other than a **Candidate Key**. A candidate key is a set of attributes that recognizes the tuples in relation or table.

The role of a candidate key is to identify a table row or column uniquely. Also, the value of a candidate key cannot be Null. The description of a candidate key is "no redundant attributes" and being a "minimal representation of a tuple," according to the Experts.

In student database with attributes Student_reg_id, Roll_No, Name, Address, Cont_No.

Candidate key are : Student_reg_id, Roll_No.

iv) Alternate Key – From all candidate keys, only one key gets selected as primary key, remaining keys are known as alternate keys.

Example – In Employee table Employee_address, Employee_no, DOB, are the alternative keys.

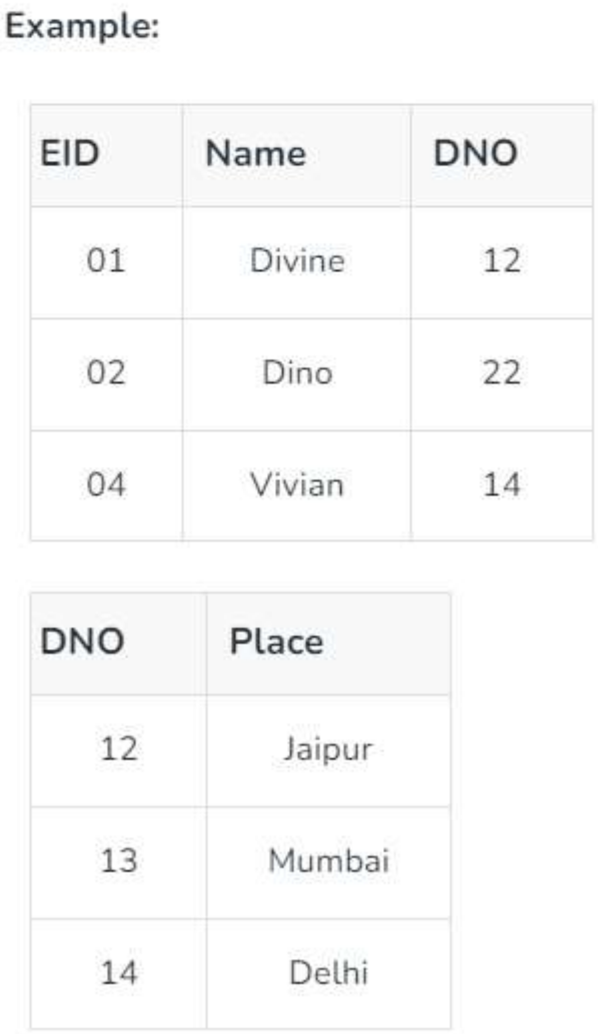
v) Foreign Key(FK) –

Foreign key is a single attribute or collection of attributes in one table that refers to the primary key of other table.

Thus foreign keys refers to primary key.

The table containing the primary key is called parent table and the table containing foreign key is called child table.

v) Foreign Key(FK) –



Example:

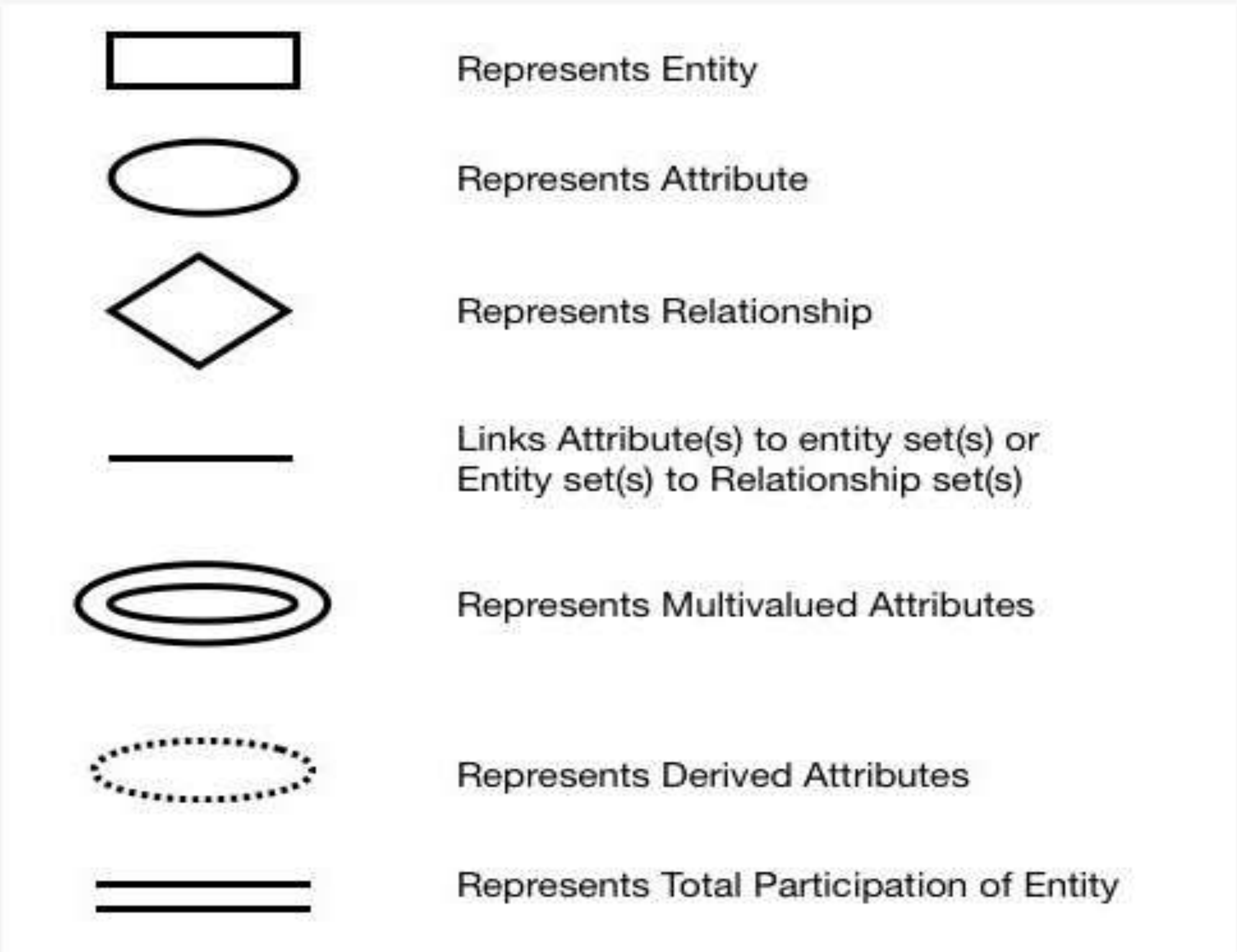
EID	Name	DNO
01	Divine	12
02	Dino	22
04	Vivian	14

DNO	Place
12	Jaipur
13	Mumbai
14	Delhi

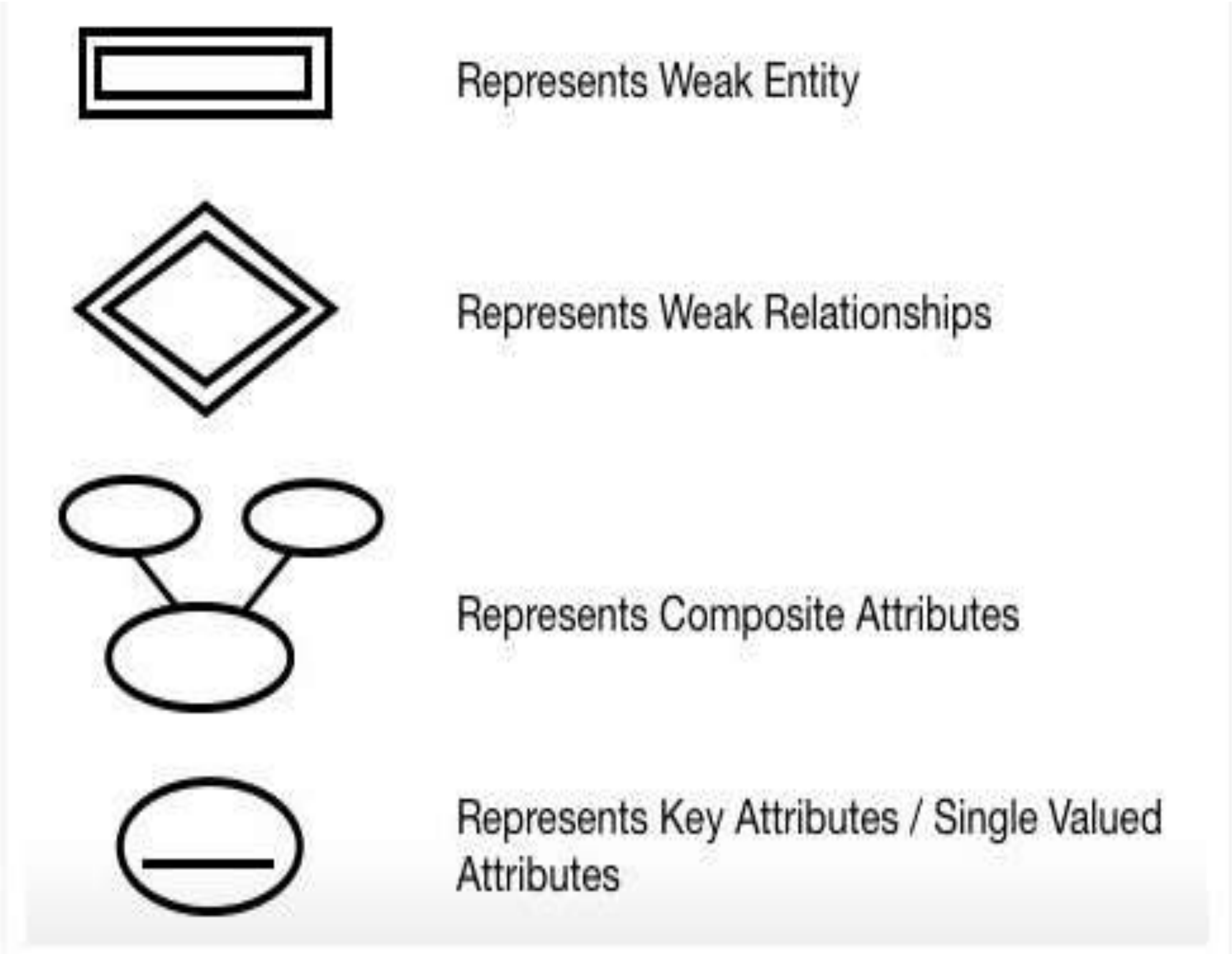
Entity - Relationship Model -

ER Diagram –

Example of various symbols used in ER Diagram



▭	Represents Entity
⬭	Represents Attribute
◇	Represents Relationship
—	Links Attribute(s) to entity set(s) or Entity set(s) to Relationship set(s)
◎	Represents Multivalued Attributes
⬯	Represents Derived Attributes
═	Represents Total Participation of Entity



Represents Weak Entity

Represents Weak Relationships

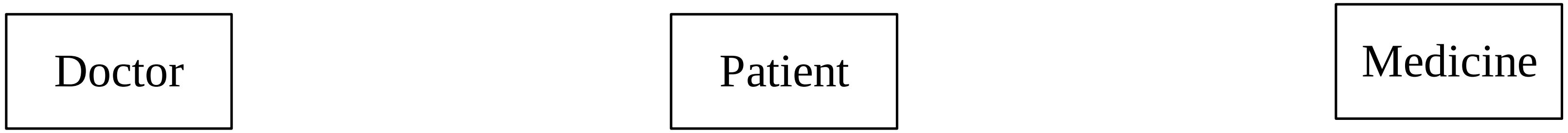
Represents Composite Attributes

Represents Key Attributes / Single Valued Attributes

ER Diagram -

Draw the ER diagram based on the Hospital Management System.

Step – 1] Entities -



Doctor

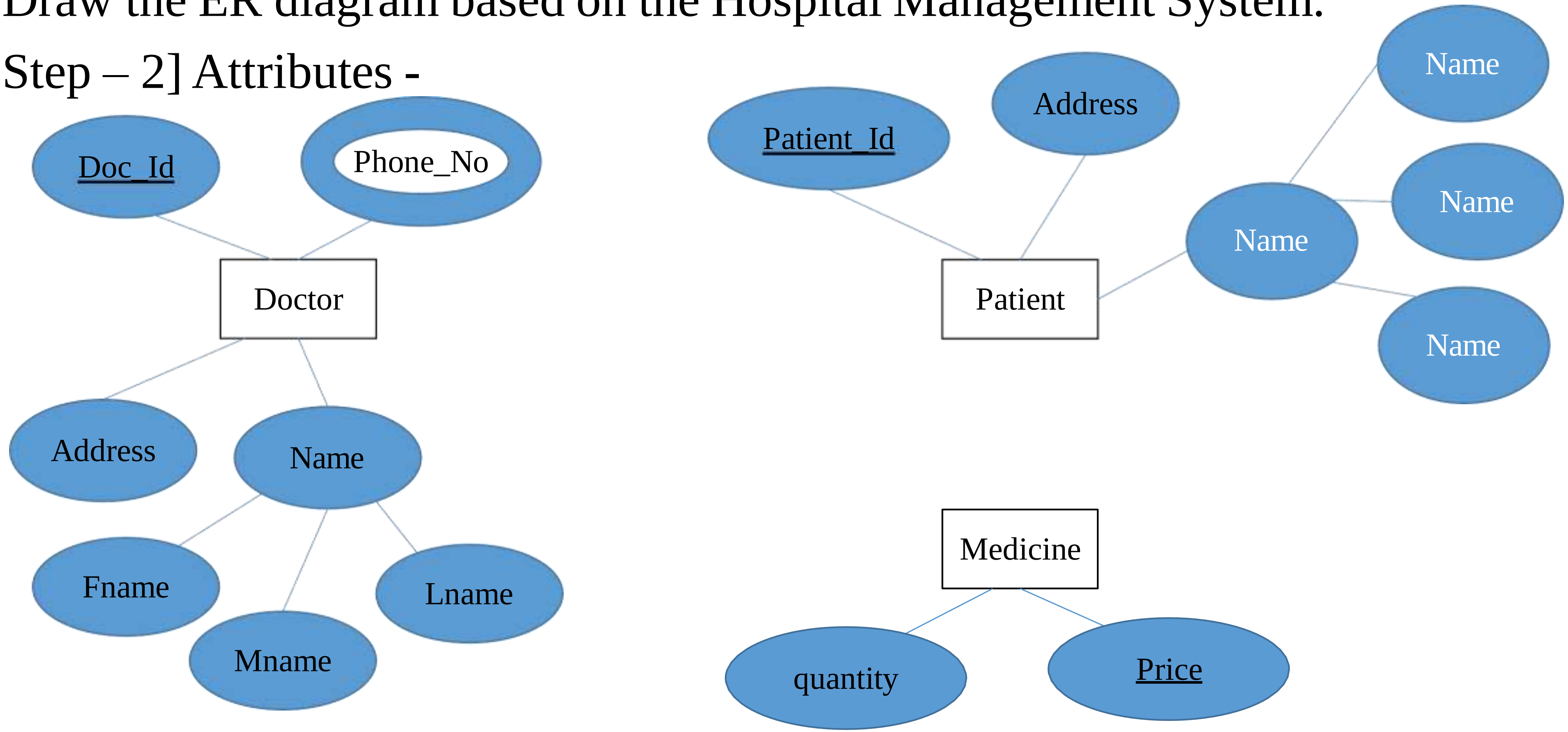
Patient

Medicine

ER Diagram -

Draw the ER diagram based on the Hospital Management System.

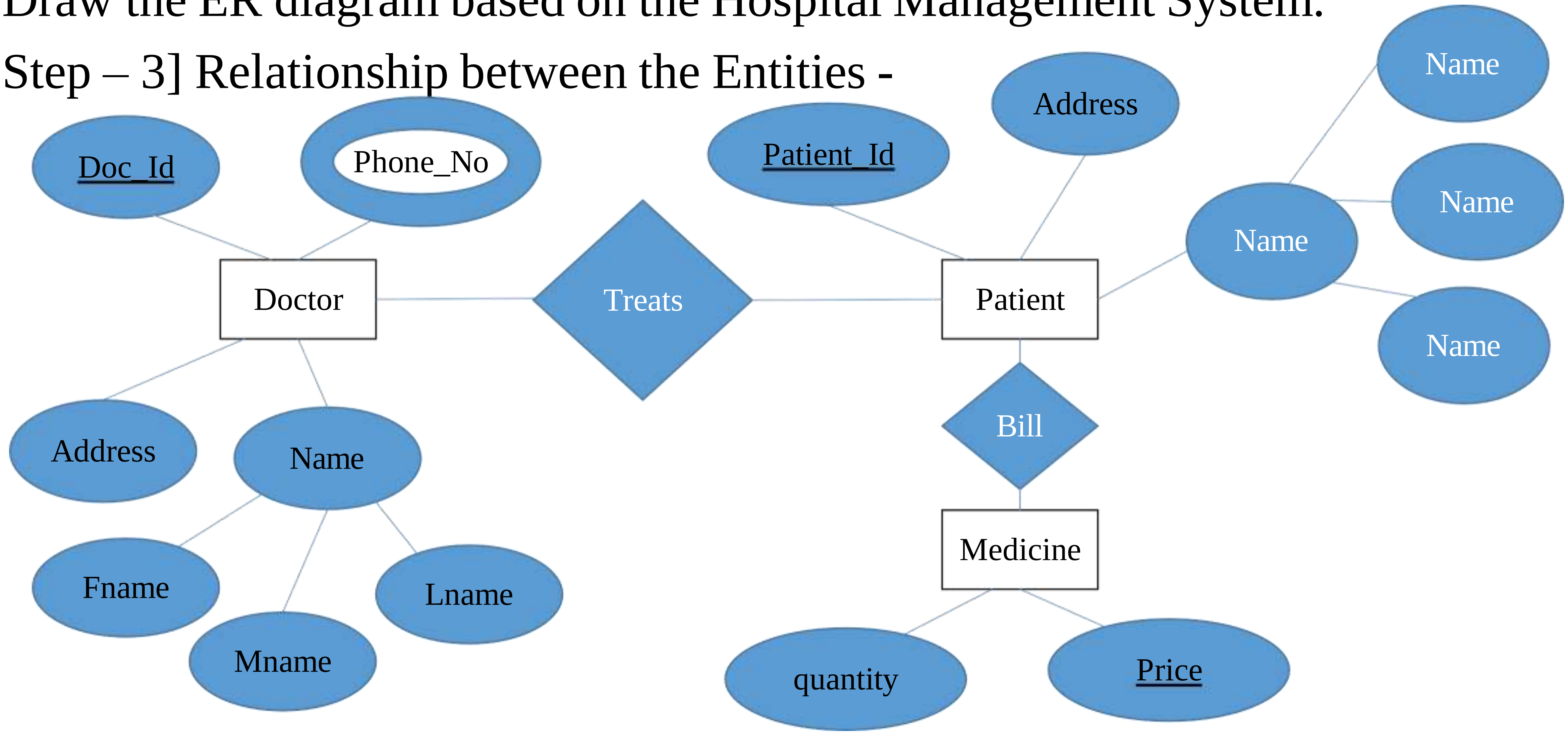
Step – 2] Attributes -



ER Diagram -

Draw the ER diagram based on the Hospital Management System.

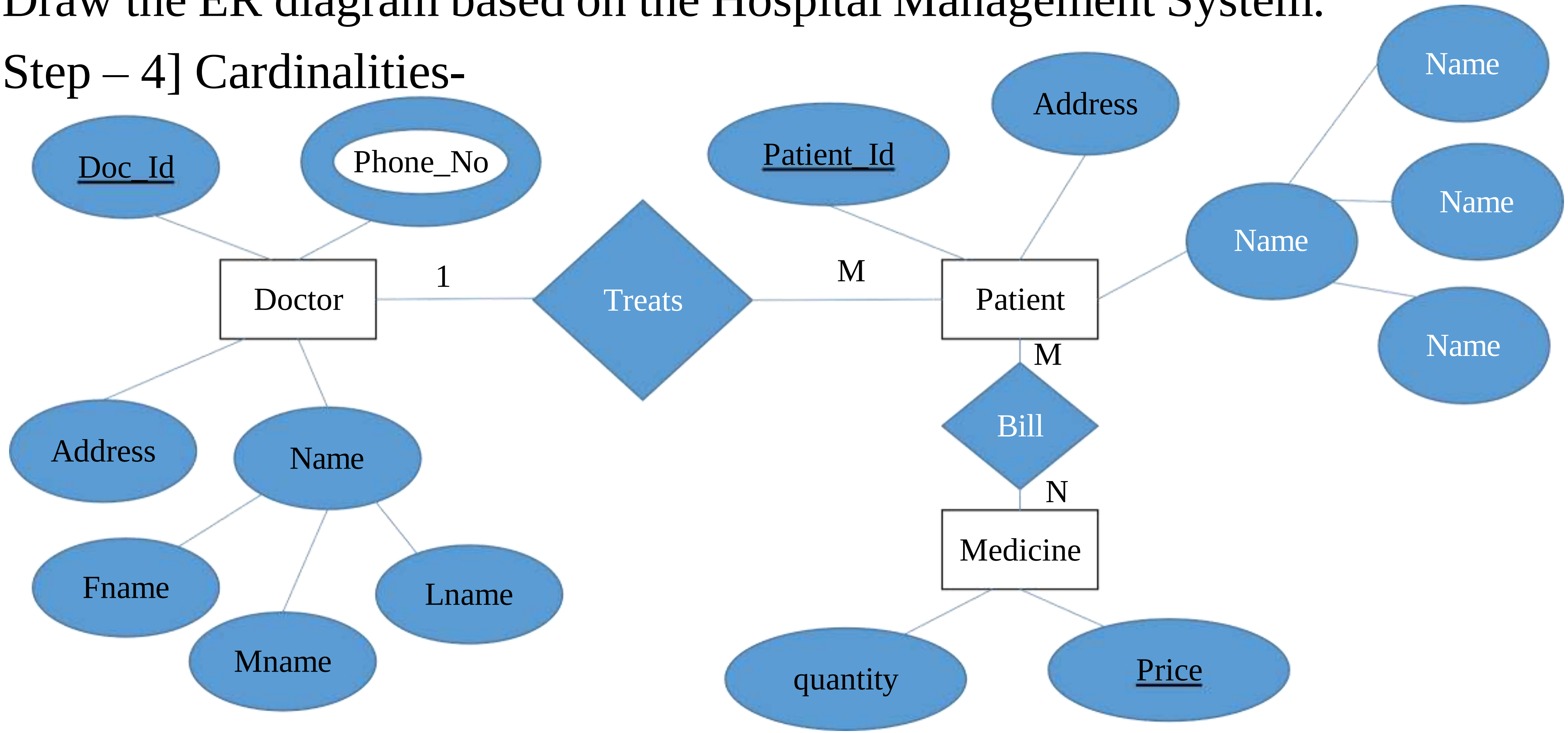
Step – 3] Relationship between the Entities -



ER Diagram -

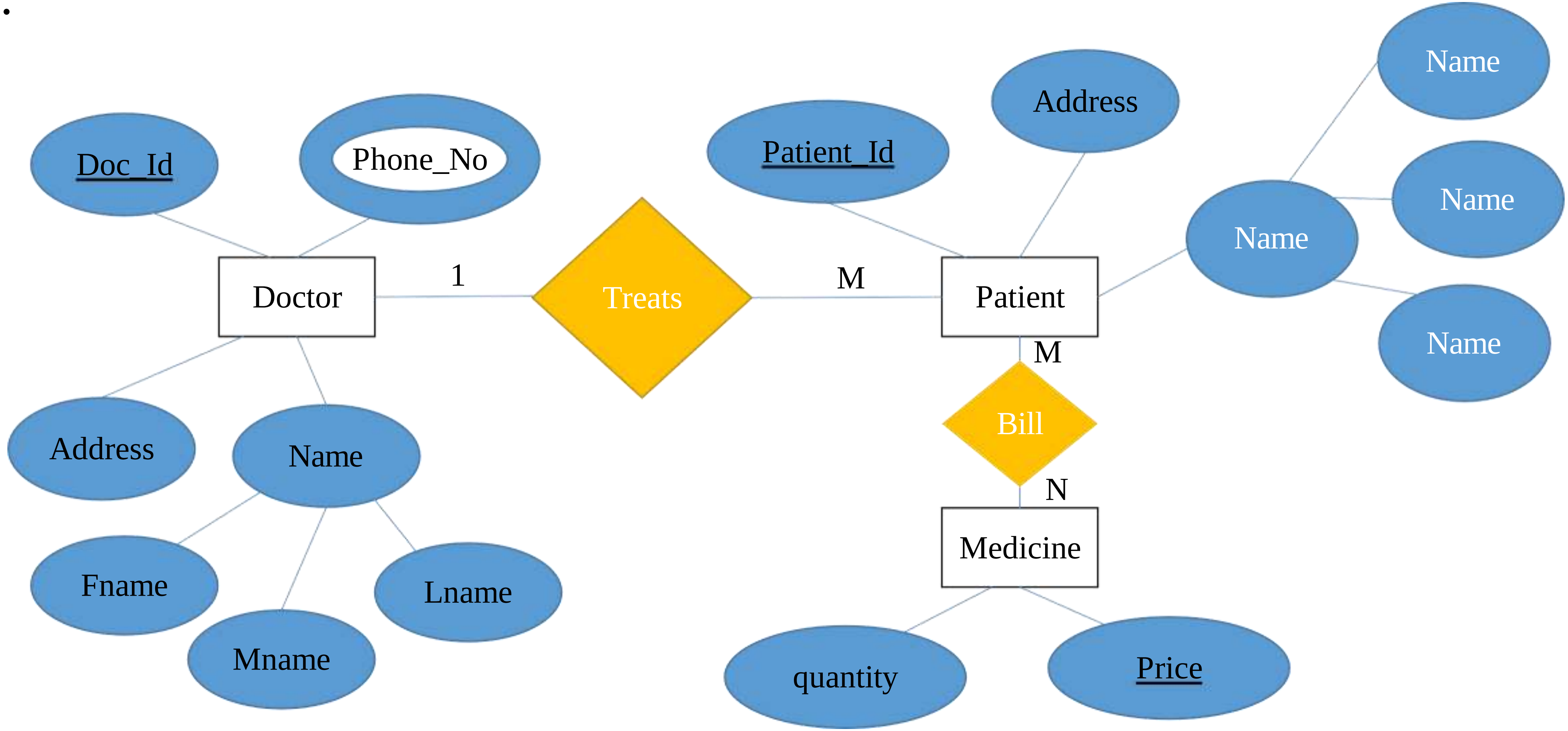
Draw the ER diagram based on the Hospital Management System.

Step – 4] Cardinalities-

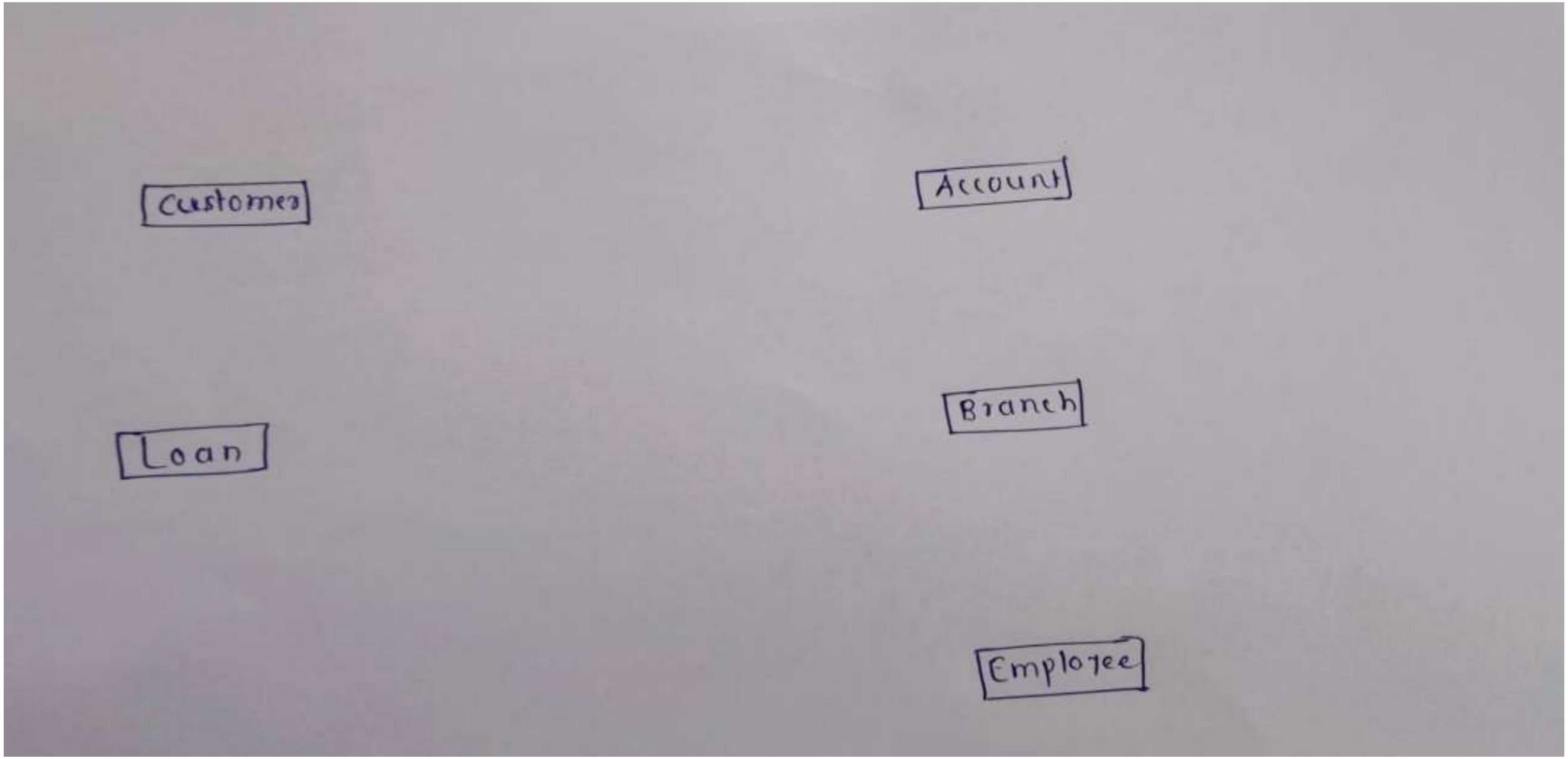


ER Diagram for Hospital Management System -

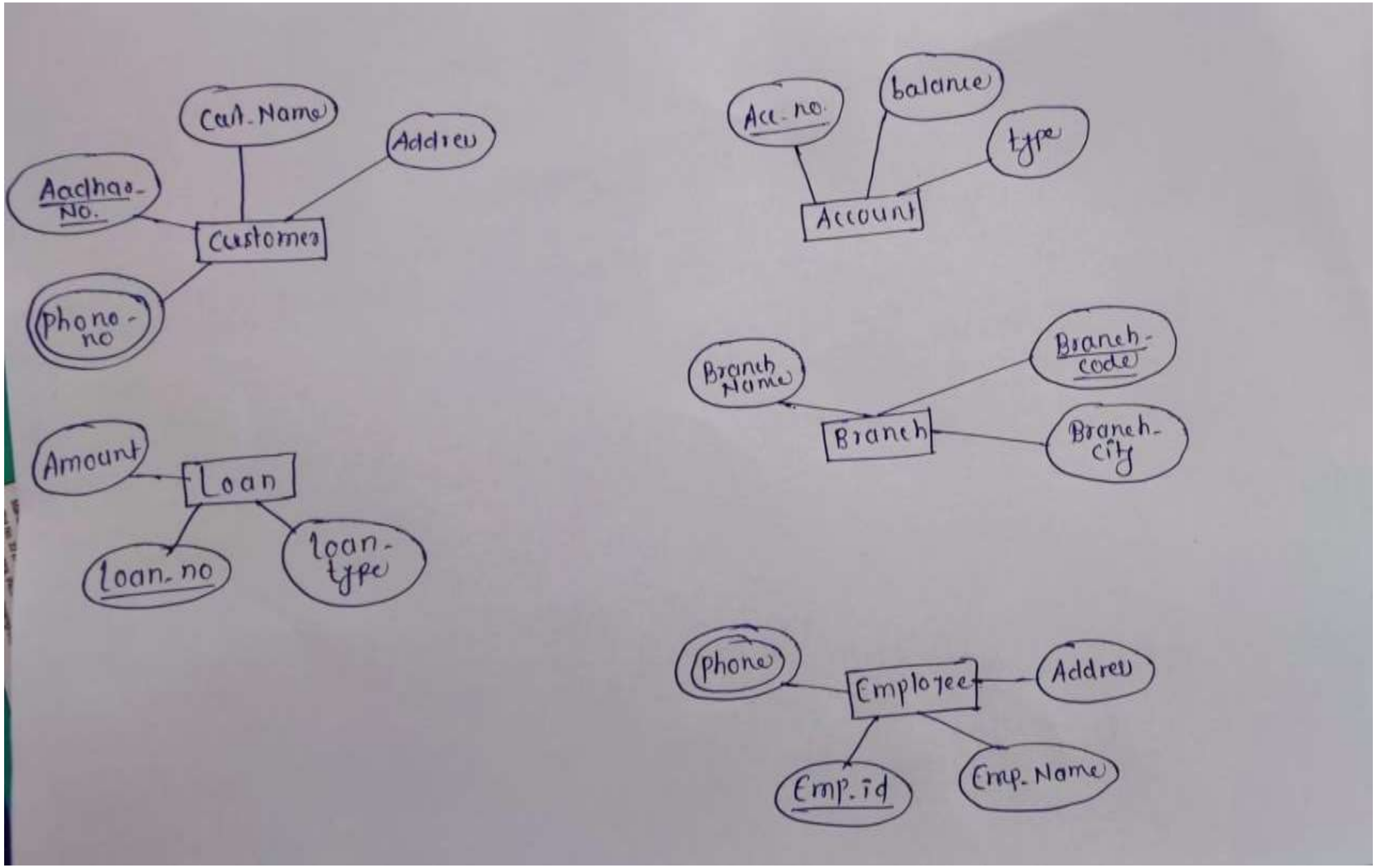
-



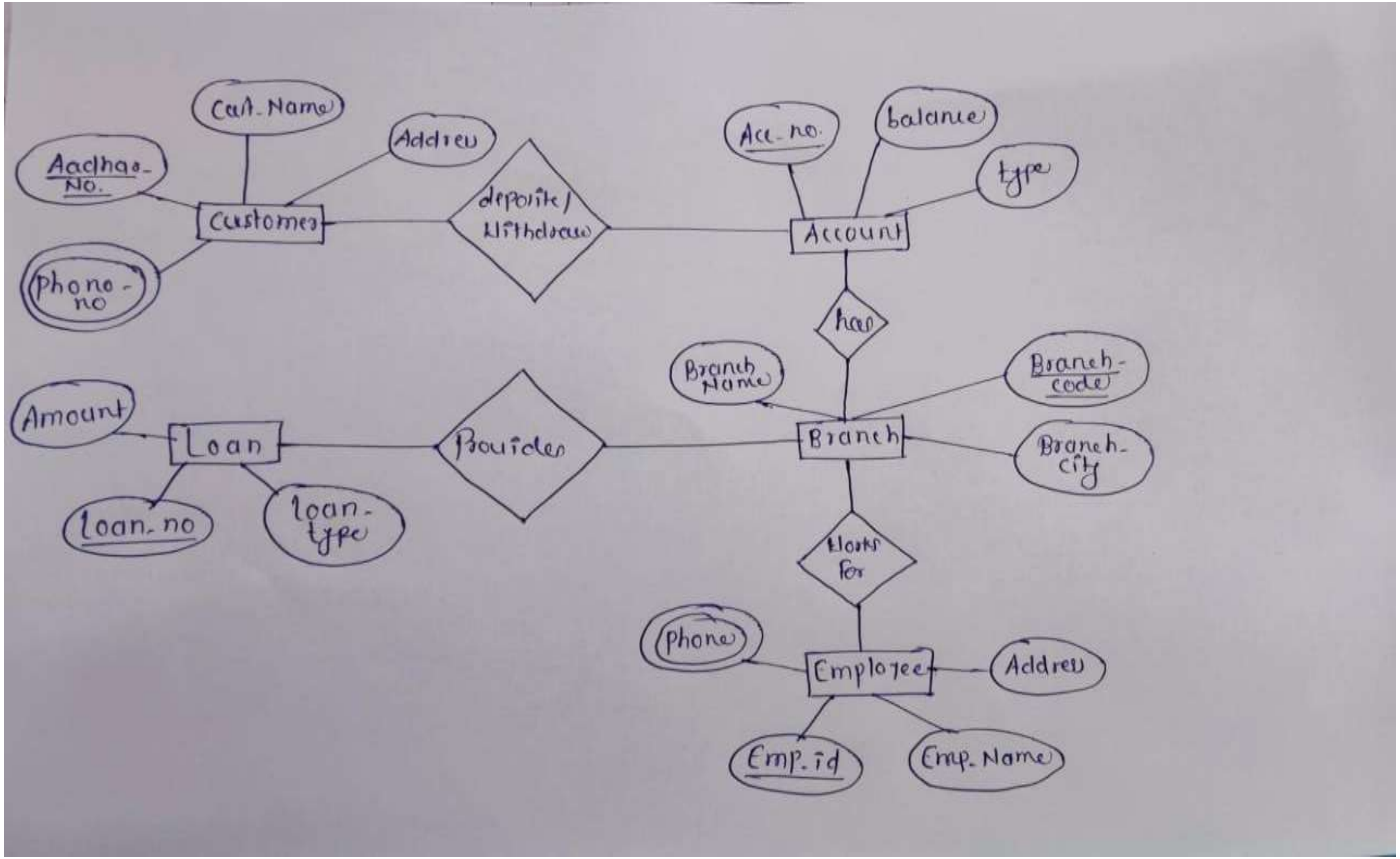
ER diagram on Banking System -



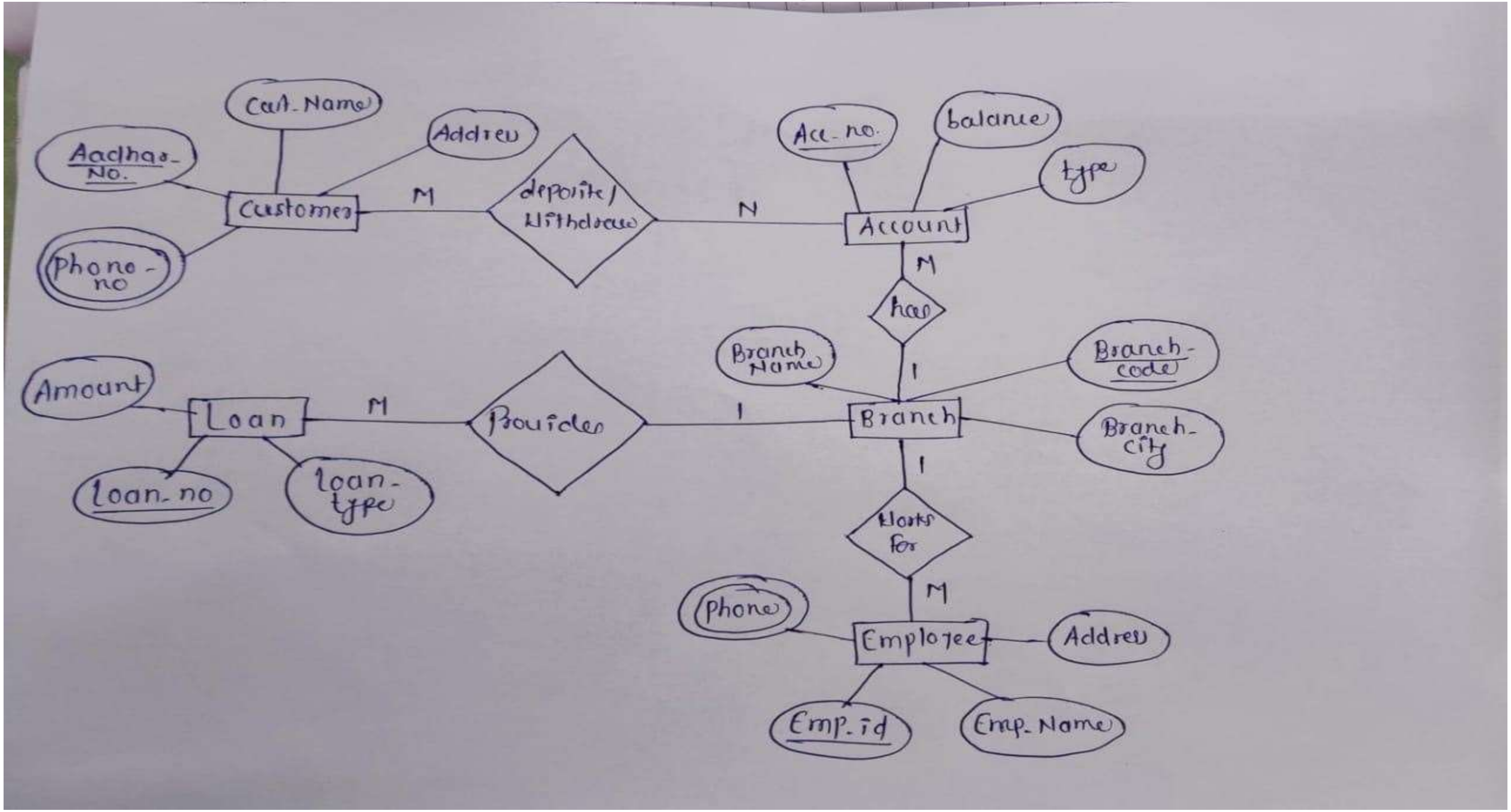
ER diagram on Banking System -



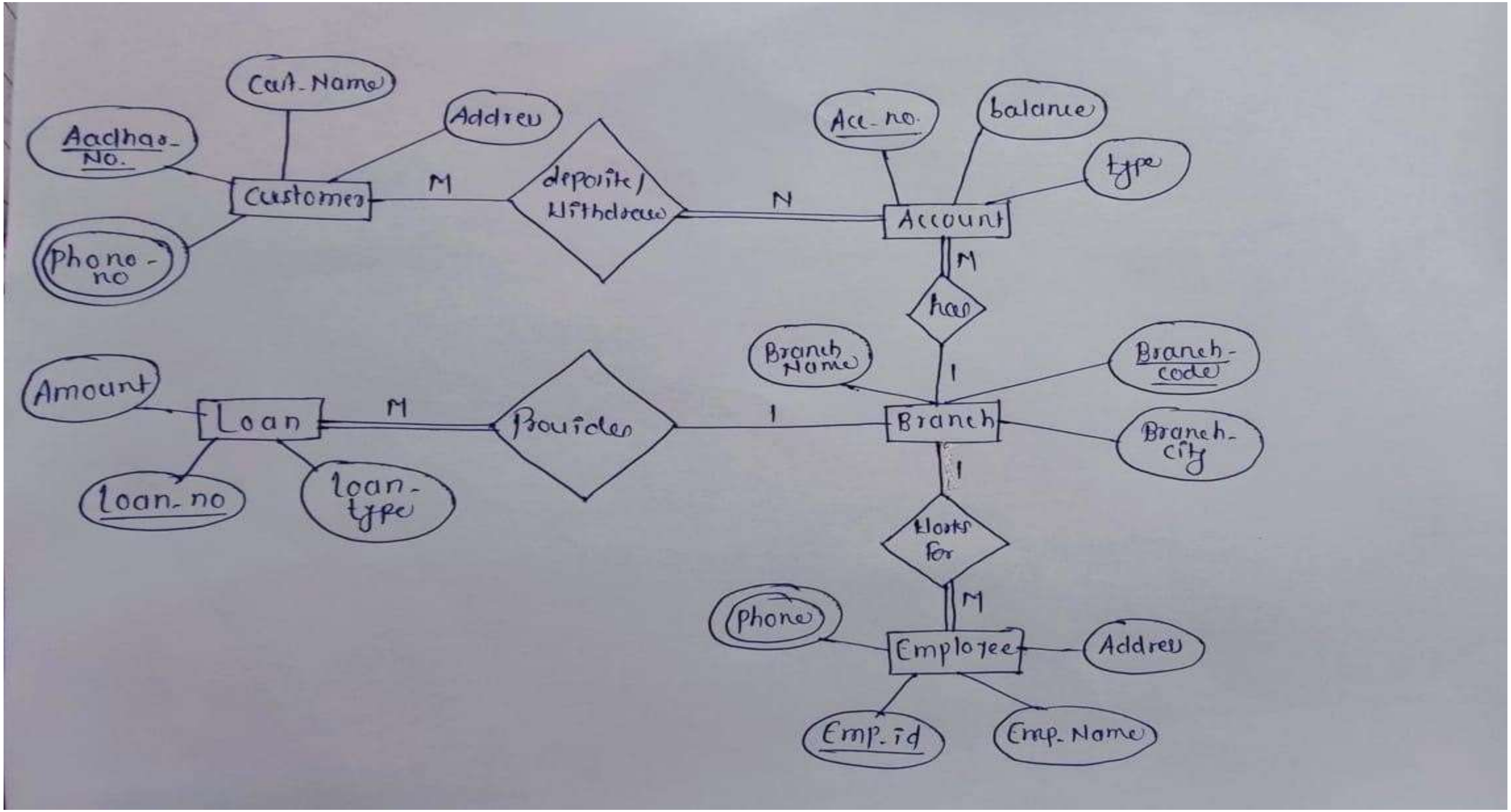
ER diagram on Banking System -



ER diagram on Banking System -



ER diagram on Banking System -



Exercise -

1] Draw ER diagram on the College Management System.

2] SALESMAN (Salesman_id, Name, City, Commission)
CUSTOMER (Customer_id, Cust_Name, City, Grade)
ORDERS (Ord_No, Purchase_Amt, Ord_Date, Customer_id, Salesman_id)

Design Issues -

- Here are some of the issues that can occur while ER diagram design process:

1. Use of entity sets vs attributes – While designing ER-Diagram the question arises as whether a value is represented as a separate entity-set or an attribute?

How choosing an entity set vs an attribute can change the whole ER design semantics. Example, we have an entity set Student with attributes such as student-name and student-id. Now we can say that the student-id itself can be an entity with the attributes like student-class and student-section.

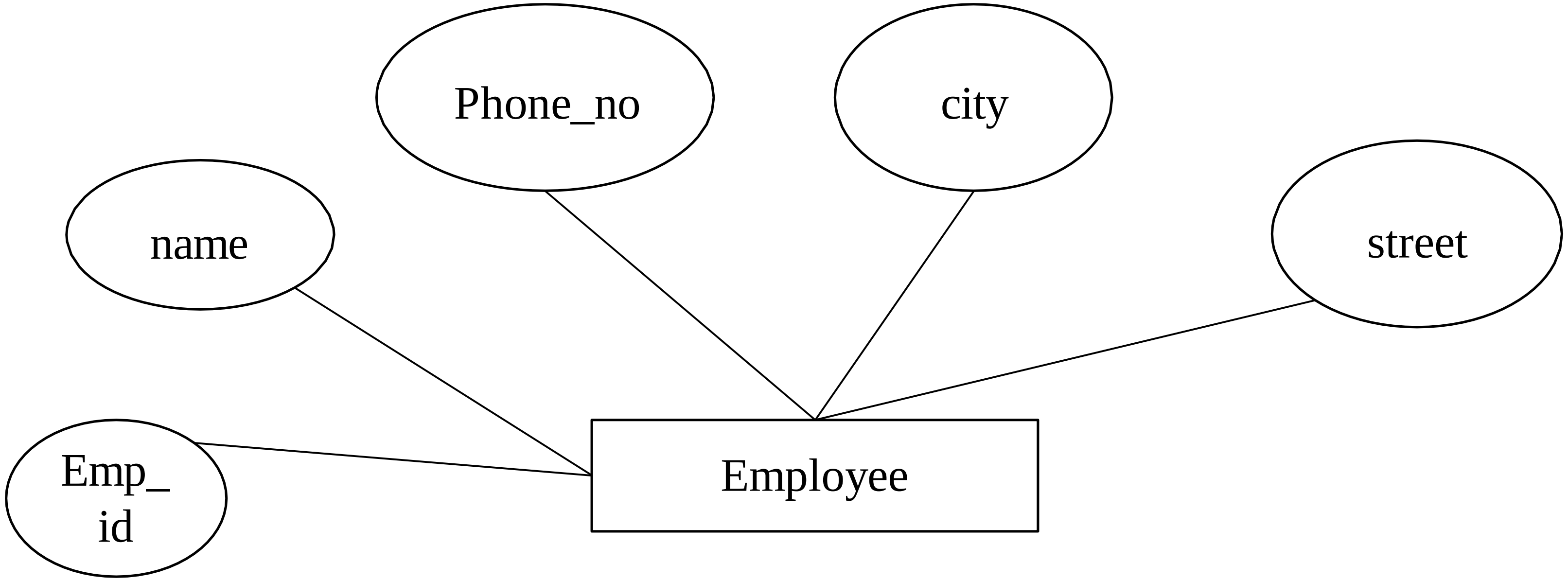
Now if we compare the two cases we discussed above, in the first case we can say that the student can have only one student id, however in the second case when we chose student id as an entity it implied that a student can have more than one student id.

Design Issues -

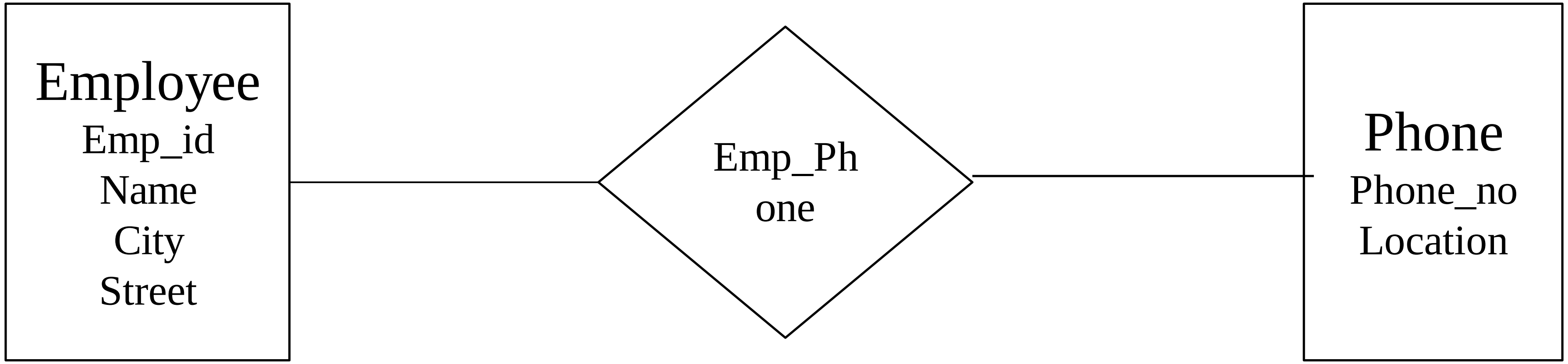
1. Use of entity sets vs attributes –

Employee entity has 5 attributes – emp_id, name, phone, city, street.

An employee has single name but he/she can have multiple phone numbers so it is good to represent phone as a separate entity with two attributes phone number and location; rather than an attribute. The location stand for office or home. If the phone numbers with different locations are important when we cannot add the phone as attribute, rather we will add it as separate entity having its own attributes.



The relationship emp_phone can be created between entities employee and phone as follows:

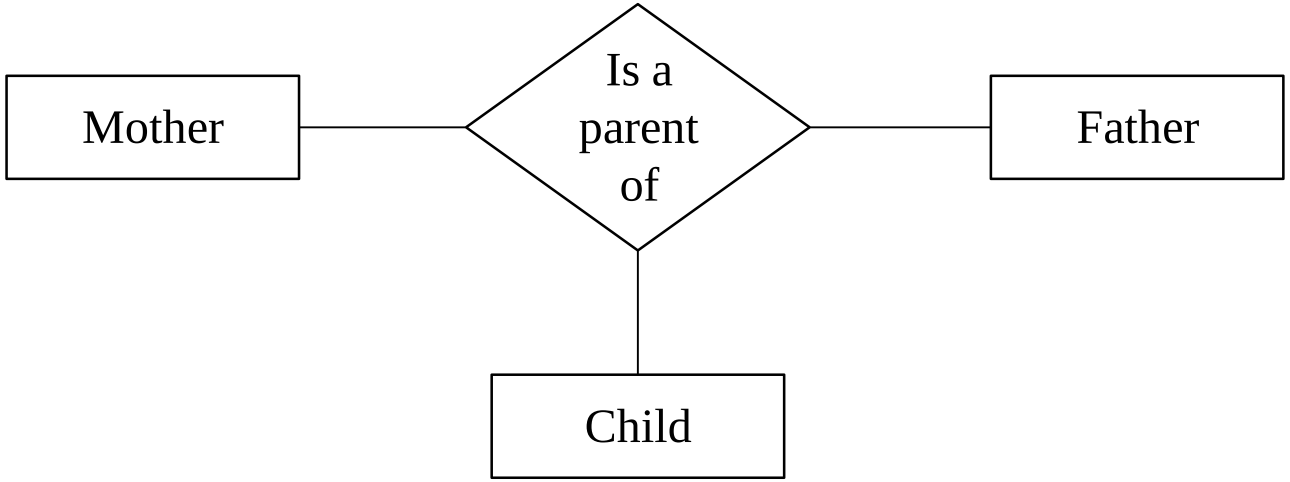


2) Use of entity set vs relationship sets - It is hard to decide that an object can be best represented by an entity set or relationship set. To comprehend and decide the perfect choice between these two (entity vs relationship), the user needs to understand whether the entity would need a new relationship if a requirement arise in future, if this is the case then it is better to choose entity set rather than relationship set.

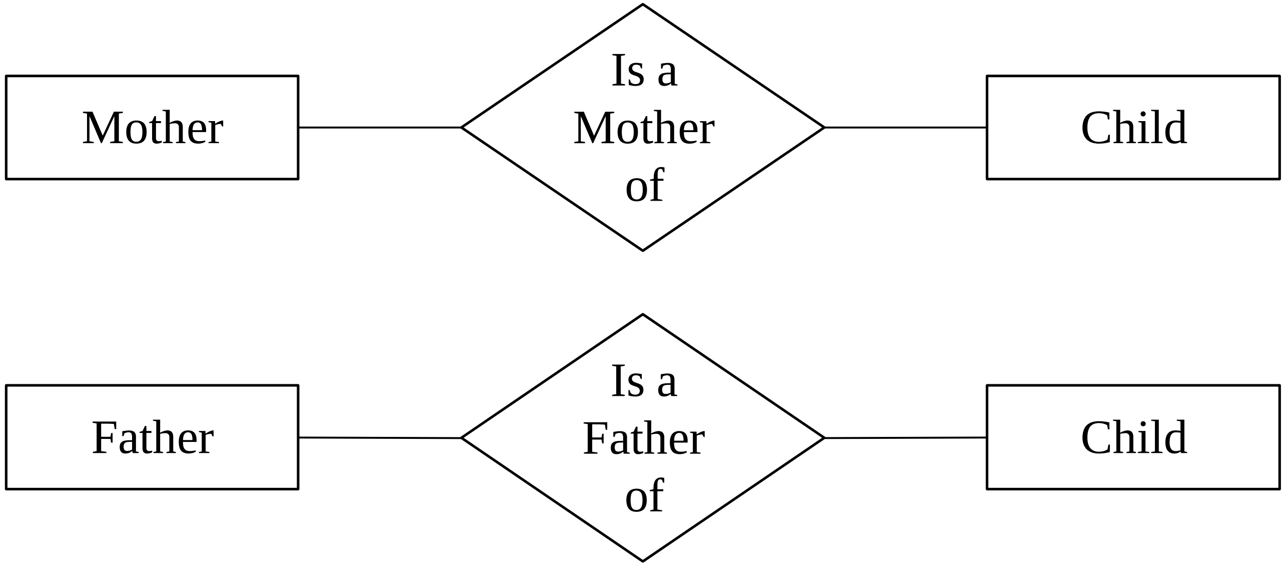
- Example : A person takes a loan from a bank, here we have two entities person and bank and their relationship is loan. This is fine until there is a need to disburse a joint loan, in such case a new relationship needs to be created to define the relationship between the two individuals who have taken joint loan. In this scenario, it is better to choose loan as an entity set rather than a relationship set.

3) Binary vs n-ary relationship set - In most cases, the relationships described in an **ER diagrams** are binary. The **n-ary** relationships are those where entity sets are more than two, if the entity sets are only two, their relationship can be termed as binary relationship.

- The n-ary relationships can make ER design complex, however the good news is that we can convert and represent any n-ary relationship using multiple binary relationships.



Example : how we can convert an n-ary relationship to multiple binary relationships. we have to describe a relationship between Three family members: father, mother and child. This can easily be represented in forms of multiple binary relationships, father-mother, relate a child to his/her Mother and Father separately:

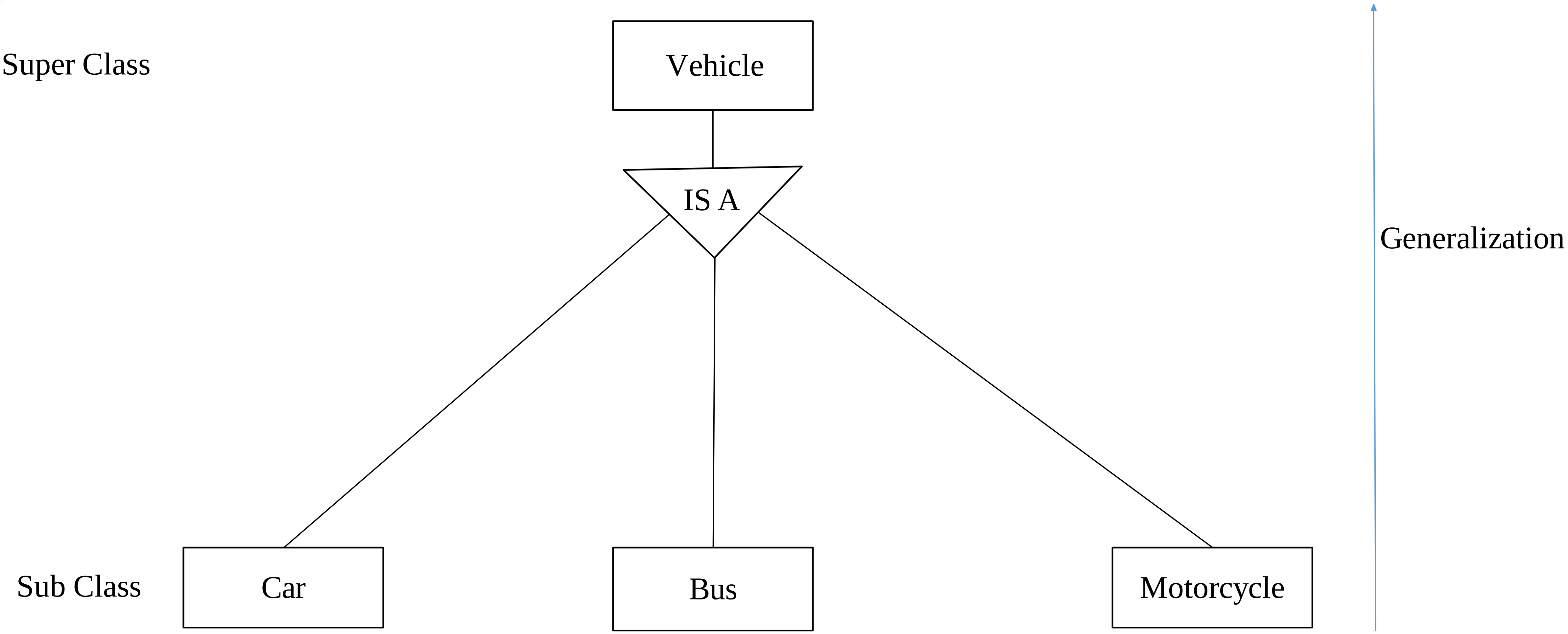


Extended ER Features -

- Extended ER is a high-level data model that incorporates the extensions to the original ER model. Enhanced ER models are high level models that represent the requirements and complexities of complex databases.
- The extended Entity Relationship (ER) models are three types as given below –
  - Aggregation
  - Specialization
  - Generalization

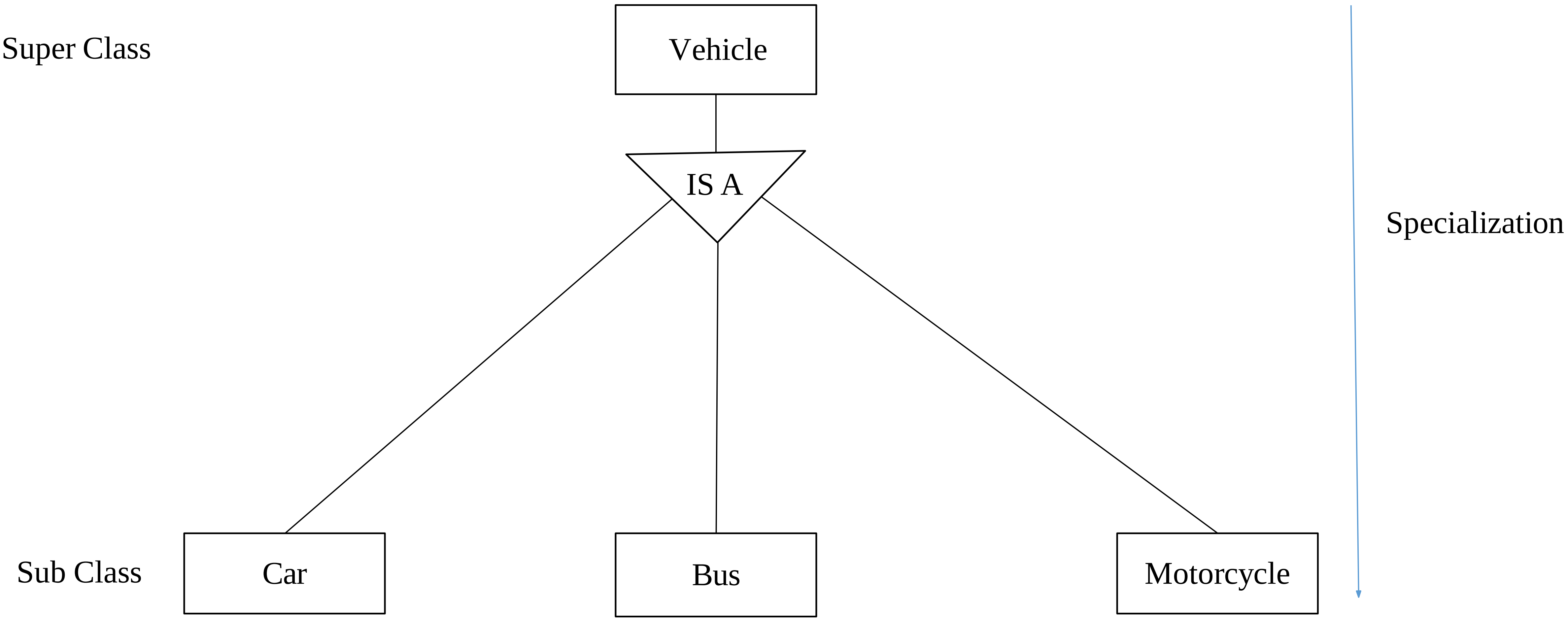
Generalization -

- Generalization is the process of extracting common properties from a set of entities and create a generalized entity from it.
- Generalization is a “bottom-up approach” in which two or more entities can be combined to form a higher level entity if they have some attributes in common.
- It converts subclasses to superclasses.
- Generalization is used to emphasize the similarities among lower-level entity set and to hide differences in the schema.


Super Class
Vehicle
IS A
Generalization
Sub Class
Car
Bus
Motorcycle

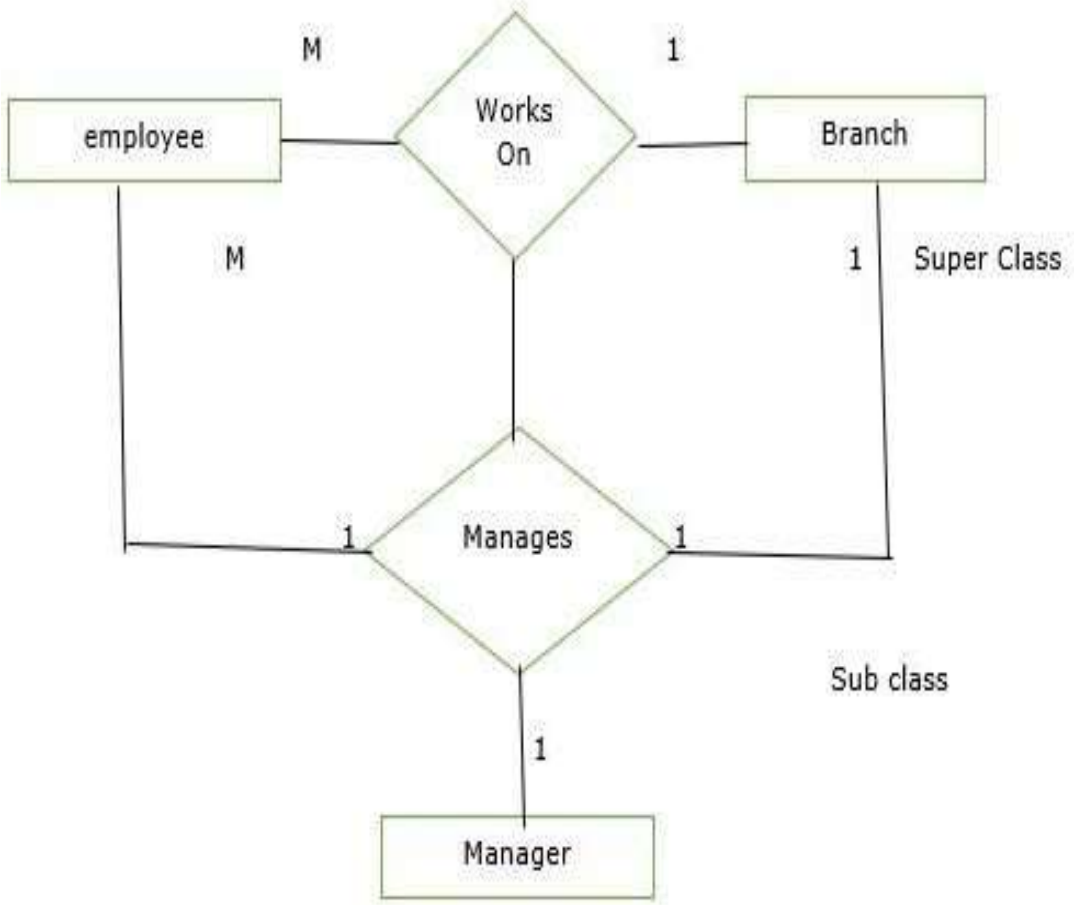
Specialization

- Specialization is opposite to Generalization.
- In Specialization, an entity is broken down into sub-entities based on their characteristics.
- Specialization is a “Top-down approach” where higher level entity is specialized into two or more lower level entities.
- Specialization is used to identify the subset of an entity set that shares some distinguishing characteristics.
- Specialization can be repeatedly applied to refine the design of schema depicted by triangle component labeled ISA.


Super Class
Vehicle
IS A
Specialization
Sub Class
Car
Bus
Motorcycle

Aggregation -

- It is an abstraction in which relationship sets are treated as higher level entity sets and can participate in relationships. Aggregation allows us to indicate that a relationship set participates in another relationship set.
- Aggregation is used to simplify the details of a given database where ternary relationships will be changed into binary relationships. Ternary relation is only one type of relationship which is working between three entities.
- Aggregation is shown in the image below –


M
1
employee
Works On
Branch
M
1
Super Class
1
Manages
1
Sub class
1
Manager

Converting ER and EER into Table -

Rule 1 – For strong entity set with only simple attribute.

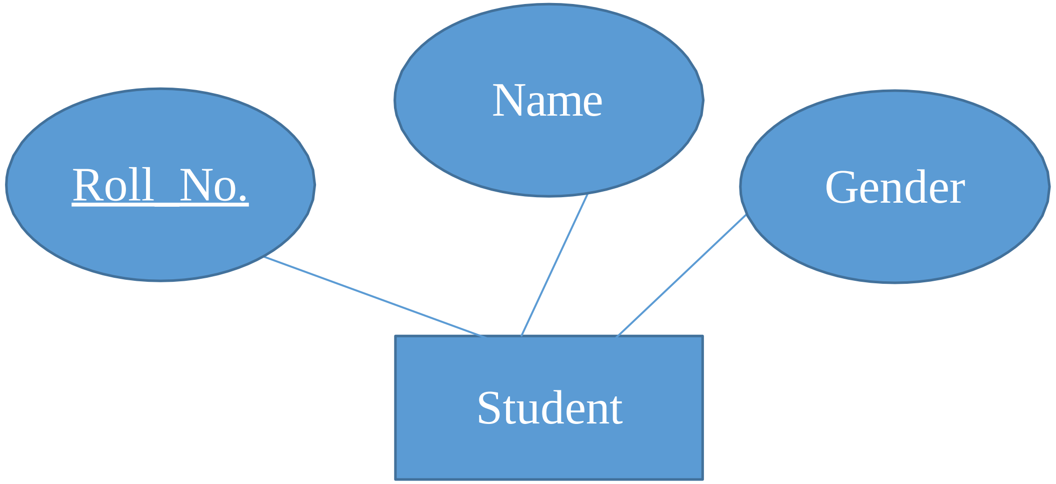


Table Name – Student

Roll_No.	**Name**	**Gender**

Schema = Student(Roll_No. , Name, Gender)

Converting ER and EER into Table -

Rule 2 – For strong Entity set with only composite Attribute -

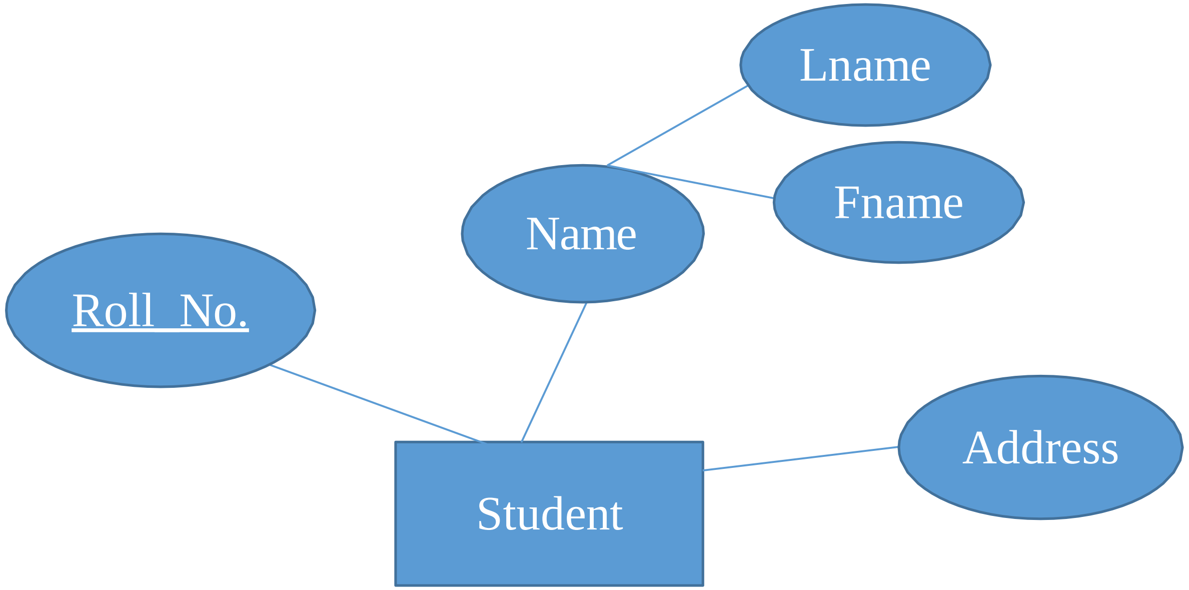


Table Name – Student

Roll_No.	Fname	Lname	Address

Schema = Student(Roll_No. , Fname, Lname, Address)

Converting ER and EER into Table -

Rule 3 – For strong Entity set with only multivalued Attribute -

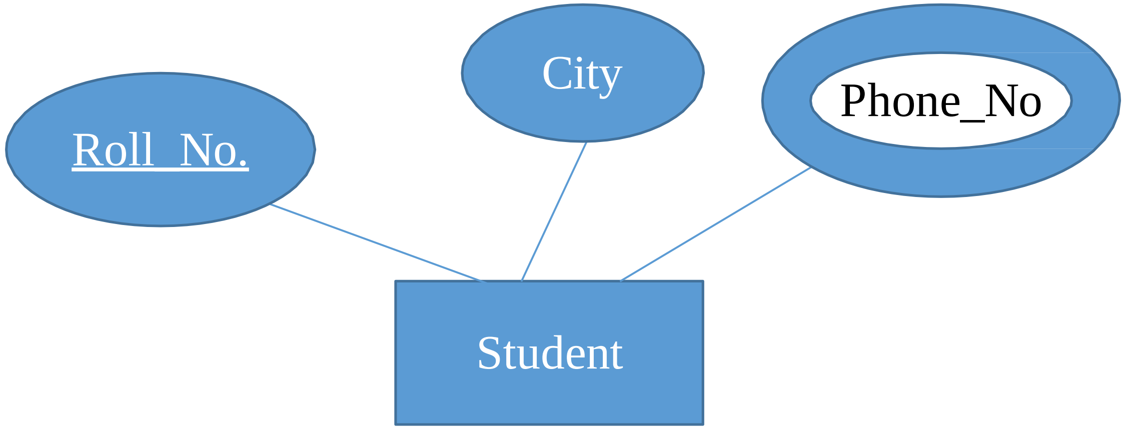


Table Name – Student

Roll_No.	City

Student_Phone

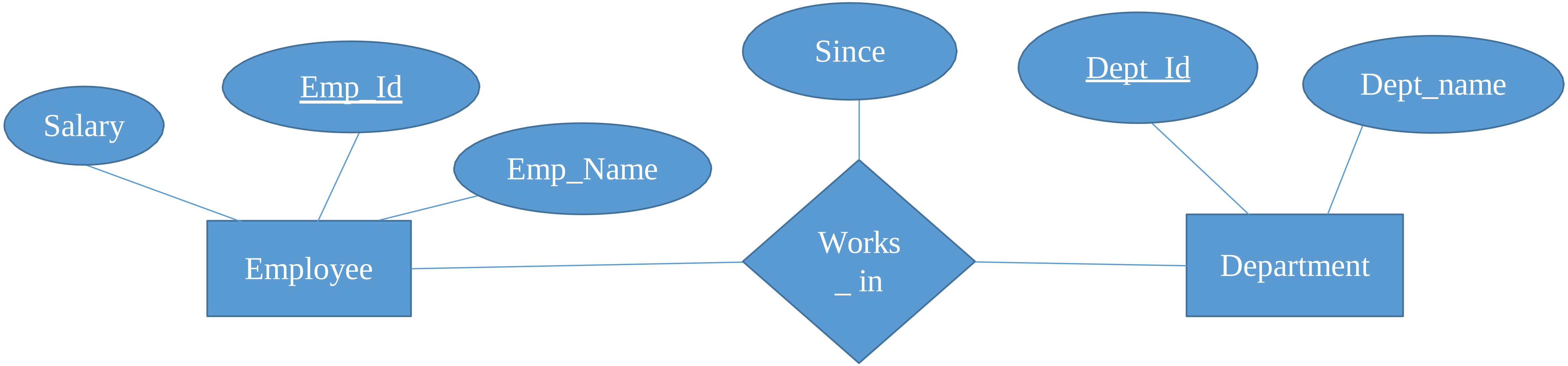
Roll_No.	Phone_No

Schema = Student(Roll_No. , City)

Schema = Student(Roll_No. , Phone_No)

Converting ER and EER into Table -

Rule 4 – Translating Relationship set into a table -



Works_in

<u>**Emp_Id**</u>	**Since**	<u>**Dept_Id**</u>

Schemas = Works_in(Emp_Id, Since, Dept_Id)

Converting ER and EER into Table -

Rule 5 – For Binary Relationship with Cardinality Ratio -

Cases are – 1. One – to – One

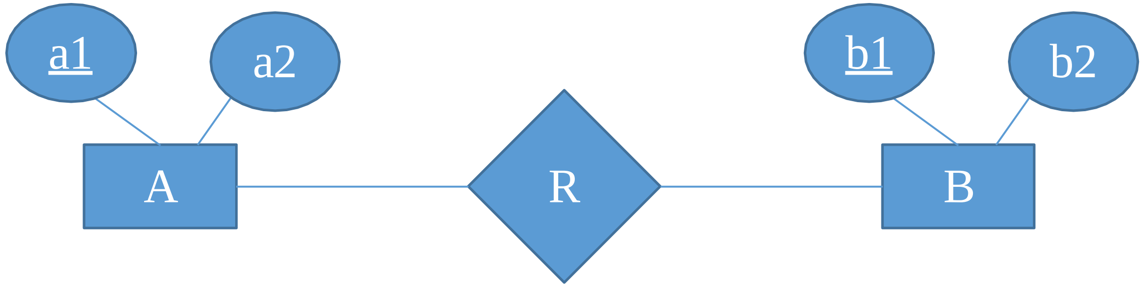
2. One – to – Many

3. Many – to – One

4. Many – to – Many

Converting ER and EER into Table -

Case 1 – For Binary Relationship with Cardinality Ratio m:n -



A

a1	**a2**

R

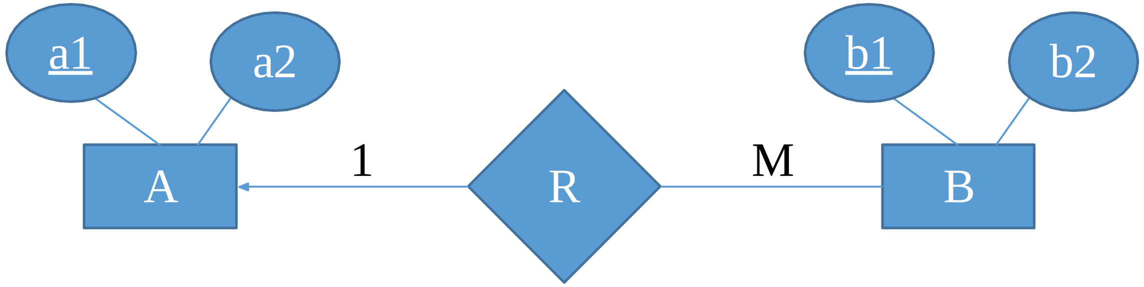
a1	**b1**

B

b1	**b2**

Converting ER and EER into Table -

Case 2 – For Binary Relationship with Cardinality Ratio 1:m -



A

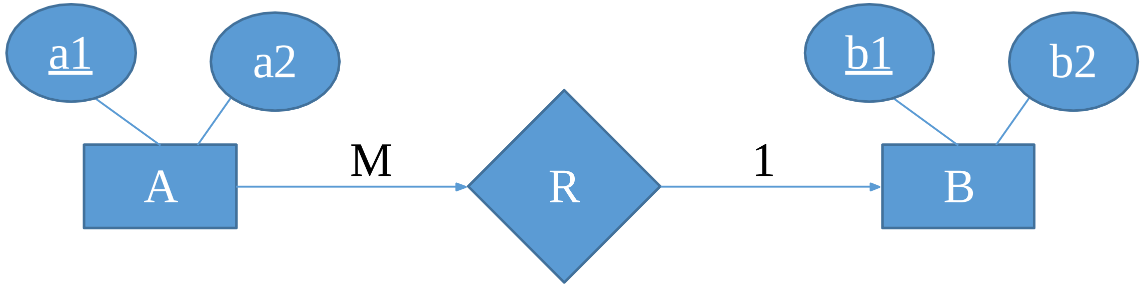
a1	**a2**

BR

a1	**b1**	**b2**

Converting ER and EER into Table -

Case 3 – For Binary Relationship with Cardinality Ratio m:1 -



AR

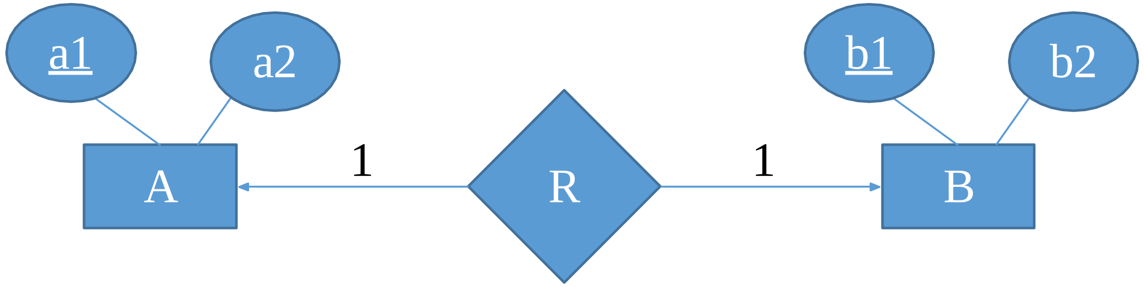
<u>**a1**</u>	**a2**	<u>**b1**</u>

B

<u>**b1**</u>	**b2**

Converting ER and EER into Table -

Case 4 – For Binary Relationship with Cardinality Ratio 1:1 -



AR

a1	a2	b1

B

b1	b2

OR

BR

b1	b2	a1

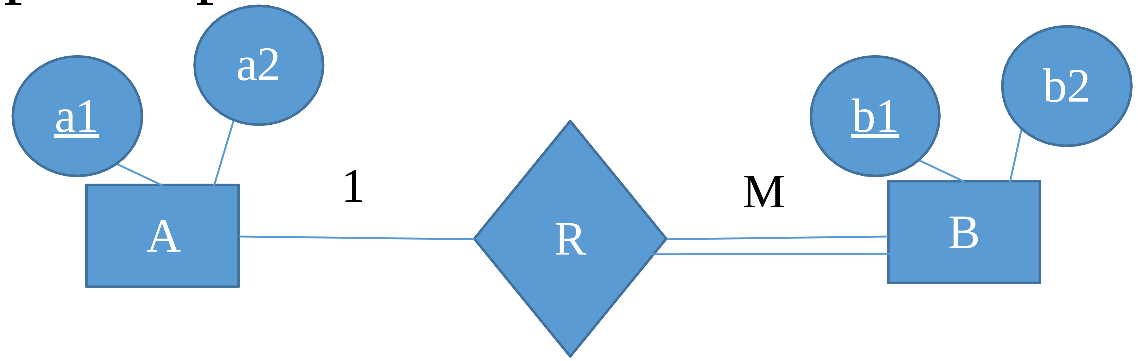
A

a1	a2

Converting ER and EER into Table -

Rule 6 – For Binary Relationship with Both cardinality constraints and participation constraints.

Case 1 – For Binary relationship with Cardinality constrain & total participation constraint from one side.



A

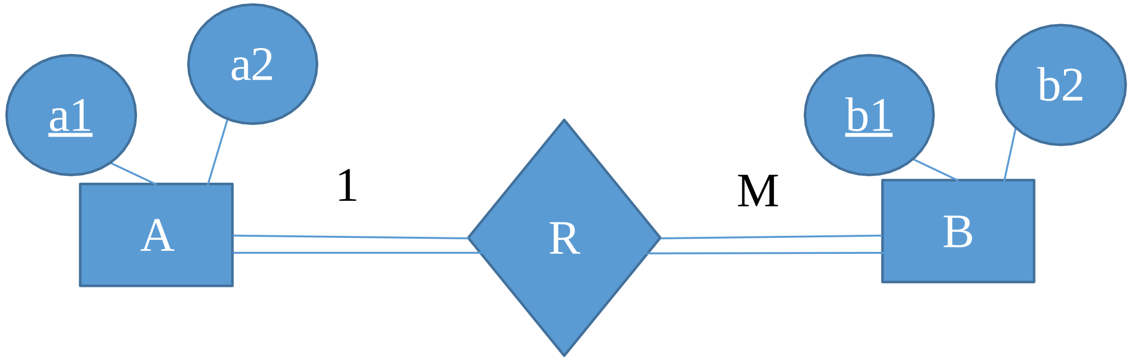
a1	a2

BR

a1	b1	b2

Converting ER and EER into Table -

Case 2 – For Binary relationship with Cardinality constrain & total participation constraint from both side.

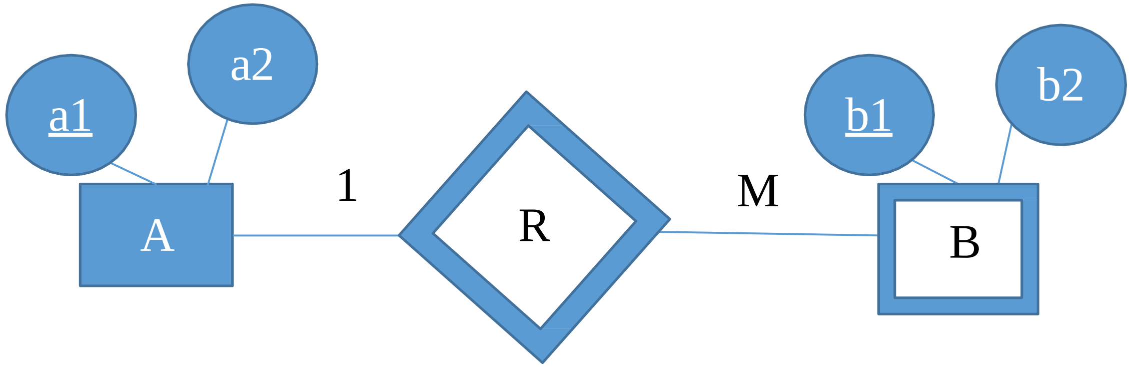


ARB

a1	**a2**	**b1**	**b2**

Converting ER and EER into Table -

Rule 7) For Binary relationship with weak entity set.



A

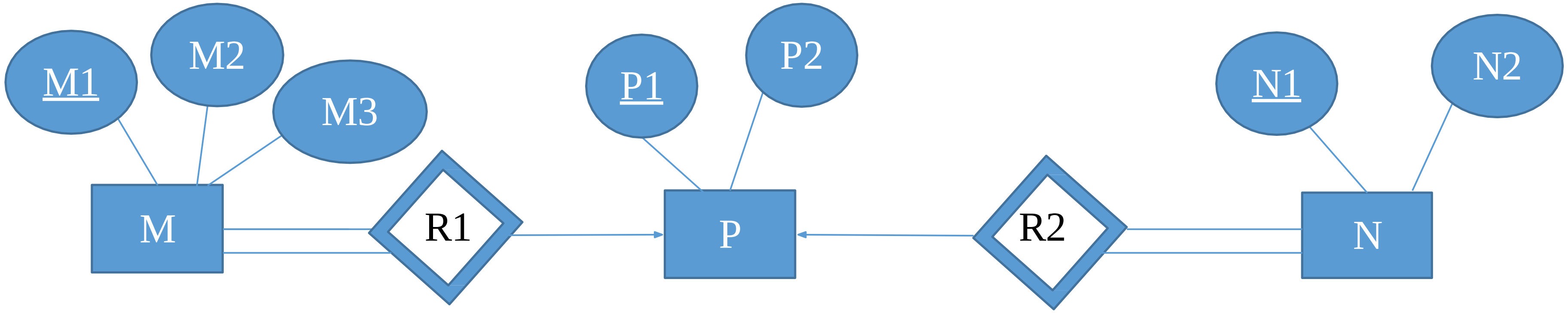
a1	a2

BR

a1	b1	b2

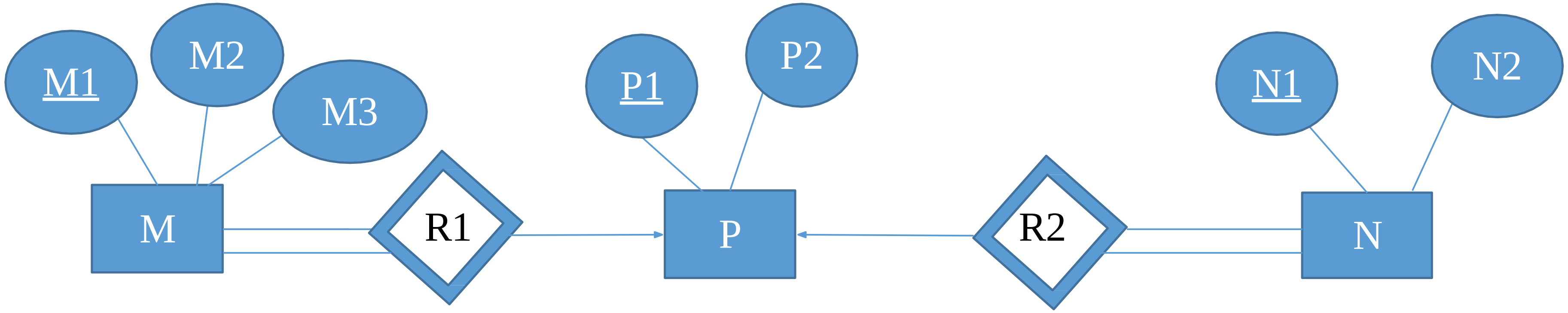
Converting ER and EER into Table -

- Find the minimum no. of table required for the following ER diagram.



Converting ER and EER into Table -

- Find the minimum no. of table required for the following ER diagram.



MR1

M1	M2	M3	P1

P

P1	P2

NR2

N1	N2	P1