Fundamentals

- **Data:** Raw material or isolated facts about an entity (records) that can be processed further.

  example: text, images, videos etc.

- **Information:** Processed data, meaningfull data. (Data that has been converted into more useful or meaningful data).

- **Database** :- A database is an organized collection of similar/related data.

  e.g: song is an example of data.
  but (collection of songs) softify is an example of database.
  youtube, netflix etc.

- **Database Management System:** It is a collection of programs that enables users to create and maintain a database. The DBMS is a general-purpose software system that facilitates the processes of defining, constructing, manipulating and sharing databases among various users and applications.

  e.g: MySQL, Oracle, MongoDB, PostgreSQL.

Disadvantages of file system :-

1) Data redundancy
2) Data consistency
3) Security problem
4) Difficulty in accessing data
5) Data Isolation
6) Atomicity problem
7) Concurrent access anomalies
8) Integrity problem.

Advantages of using DBMS :

1) Controlling Redundancy: DBMS controls redundancy through various mechanisms and techniques. Redundancy refers to the duplication of data within a database, which can lead to several issues such as data inconsistency, increased storage requirements and decreased performance. Here are some ways DBMS controls redundancy: Normalization Techniques, Data integrity constraints, Concurrency and transaction control.

2) Restricting Unauthorized Access:

A DBMS restricts unauthorized access through various security mechanisms and features such as Database Backup and Recovery, Authentication, Access control. These security measures collectively help DBMS restrict unauthorized access and protect the confidentiality, integrity and availability of data stored within the database system.

3) Provides Persistent Storage for Program Objects.

4) Provides Storage Structures and Search Techniques for Efficient Query processing.

5) Provides Backup and Recovery.

- Difference between Relational Database Management System and DBMS.

→

RDBMS	DBMS
1. Structure: RDBMS is a specific type of DBMS that manages relational databases. It is based on the relational model, where data is organized into tables with predefined relationships.	1. Structure: A DBMS is a software that manages databases, regardless of their organizational structure. It can handle different types of databases, including hierarchical, network and relational databases.

2) Data Model: RDBMS strictly follows the relational data model and relies on tables, rows and columns to represent data.	2) Data Model: DBMS supports various data models such as the hierarchical model, network model, object-oriented model and more.
3) Data Relationships: RDBMS enforces strict relationships between tables using primary keys, foreign keys and referential integrity constraints.	3) Data Relationships: In a DBMS, relationships between data are typically established through navigational pointers or physical pointers. It does not enforce strict relationships like in an RDBMS.

- Data Models : Data Model is a collection of concepts that can be used to describe the structure of a database.

→ By structure of database it means the data types, relationships and constraints that apply to the data.

→ It describes the design of a database at each level of abstraction.

→ It is basically logical structure of a database. It shows us from the design of the data to its proper implementation of data.

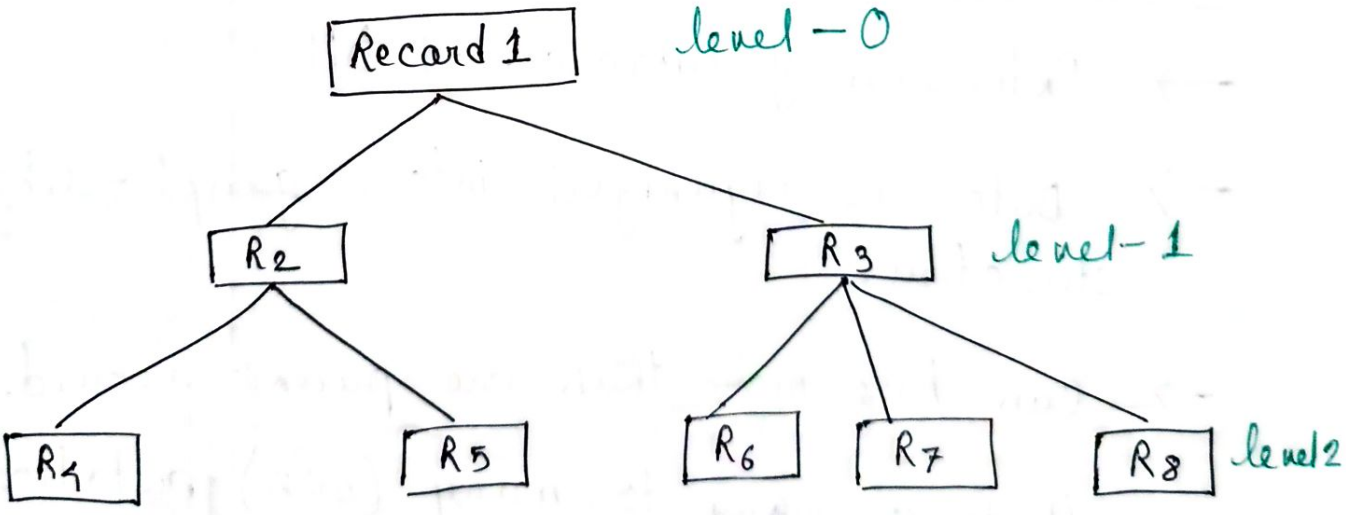
- Types of Data Models in DBMS:

1) Hierarchical Data Model
2) Network - DM
3) Entity - Relationship model/Diagram
4) Relational DM
5) Object based DM.

1) Hierarchical DM :

→ Data is organized into a tree-like structure, where each record consists of one parent and n no. of children.

→ one-to-many (1:N) relationships.



→ Restriction on the number of parents

→ Data stored in n- no. of levels.

e.g:

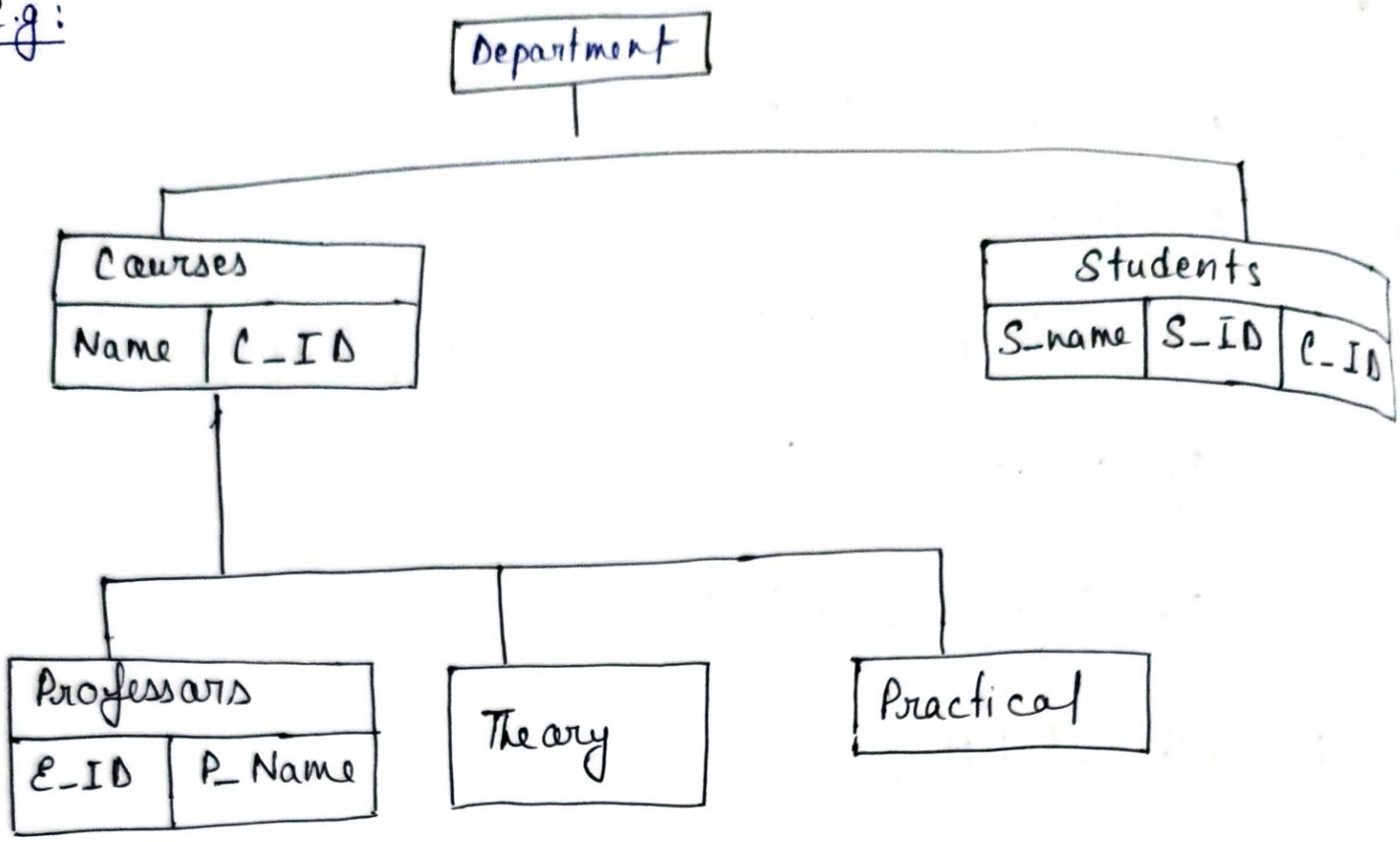


fig: An example of the Hierarchical model representing a university.

2) Network DM :

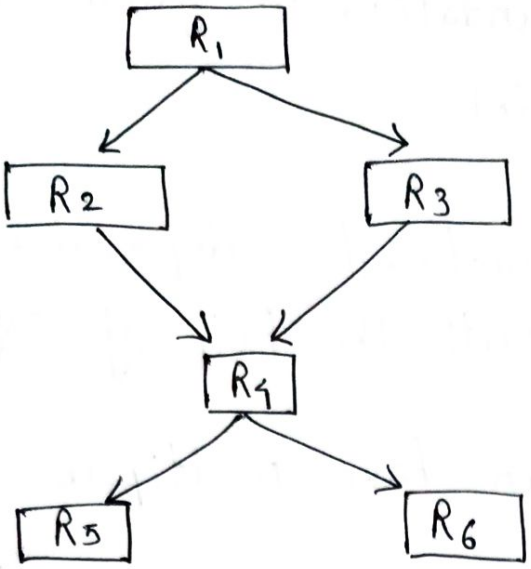
→ Extension of Hierarchical DM

→ Data is organized into a graph-like structure.

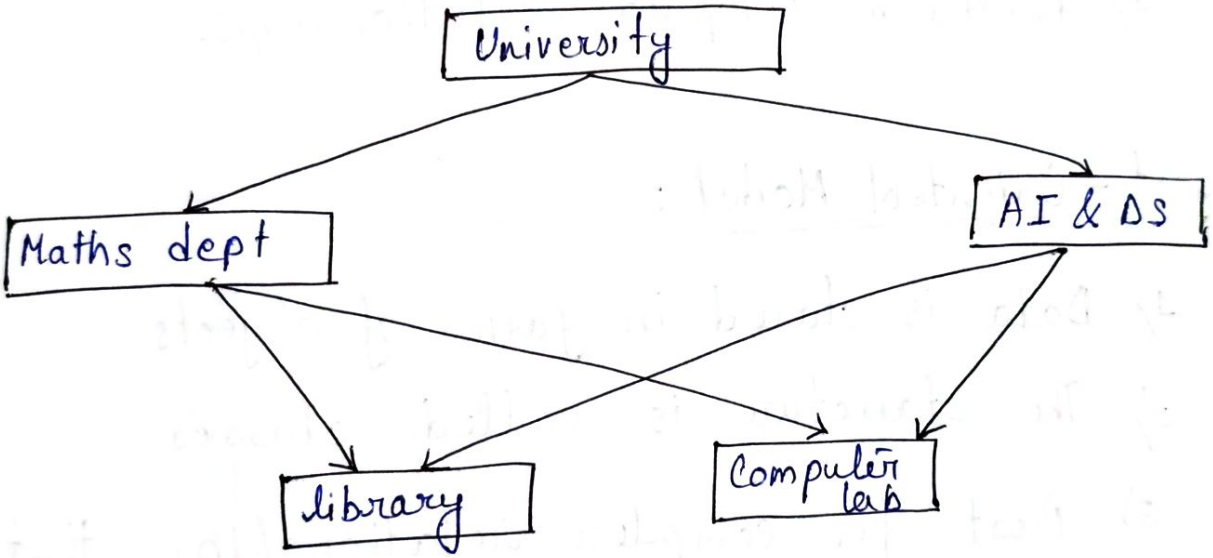
→ Can have more than one parent record.

→ It is a many to many (m:n) relationship.

→ Used for data which are related.



e.g: network model representing an university.



3) Relational Model :

1) Entity set represented by a table/relation.

2) For all attributes there are separate ~~table~~ columns.

3) It is a theoretical concept whose practical implementation is done.

→ It provides a foundation to form the base for the physical model.

→ It enables structured independence and Ad-hoc query with the help of SQL.

→ One database can have multiple tables.

→ Each table is an application entity set.

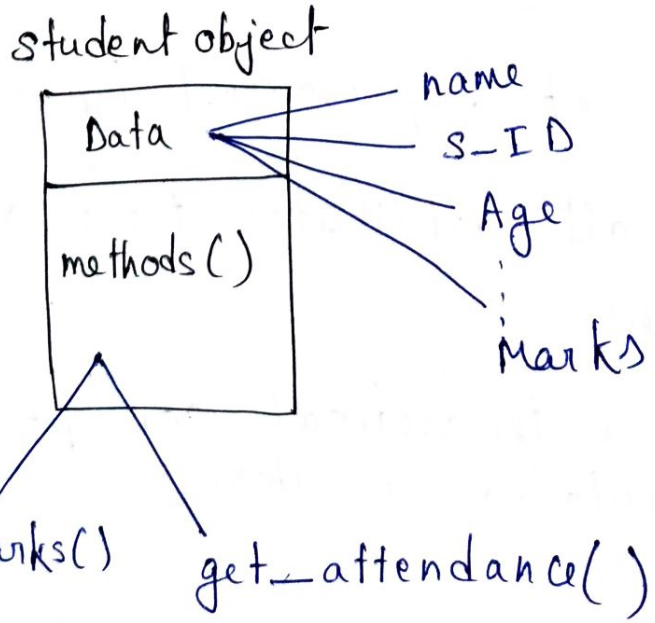
Disadvantage

1) Failed in complex relationships.

4) Object-Oriented Model :

1) Data is stored in forms of objects
2) The structure is called classes
3) Best for complex relationships that is for collection of databases.

e.g.



5) ER Diagram Model:

→ Entity Relationship model was developed to faciliate database design by allowing specification of an enterprise schema that represents the overall logical structure of a database.

→ E-R model is very useful in mapping the meanings and interactions of real-world enterprises onto a conceptual schema.

→ It is free from ambiguities and provides a standard and logical way of visualizing the data.

- Entity - Relationship Model Basics:

- Entity: An entity is a thing or an object in the real world that is distinguishable from other object based on the values of the attributes it passes.

Types of Entity:

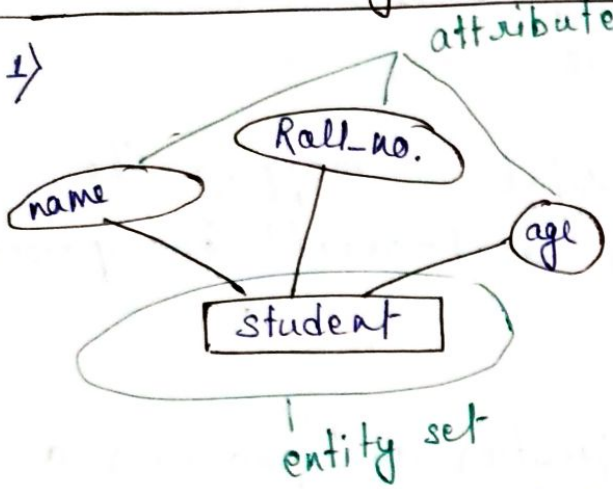
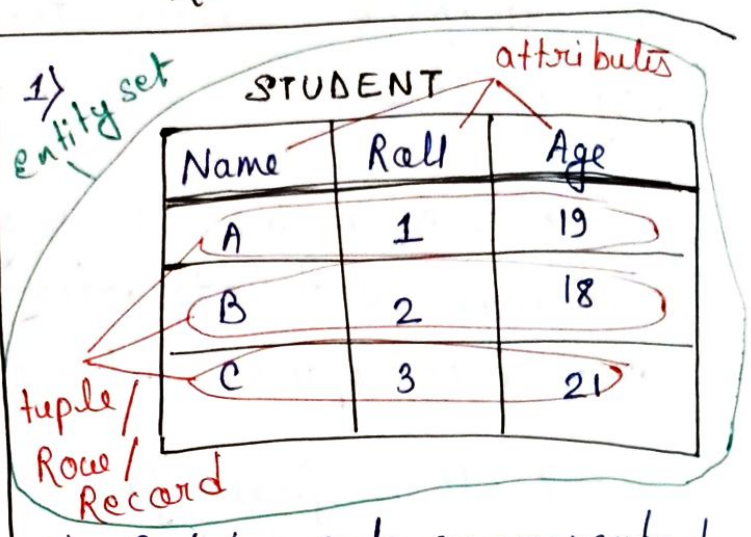
① Tangible: Entities which is physically exist in real world.

e.g: Car, Books, Person etc.

② InTangible: Entities which exist logically.

e.g: Music, Video, Software, Accounts. Email ID etc.

- **Entity Set**: Collection of same type of entities that share the same properties or attributes.

ER Diagram	Relational Model
1) 	1) 
2) Entity set represented by a rectangle.	2) Entity set represented by a relational ~~model~~ table.
3) Attributes are represented in oval or ellipse.	3) For all attributes there are separate columns.
4) No entity can be represen-ted. As in ER Diagram we represent the schema.	4) Entity is required as relational model is for instance.

- In ER diagram we cannot represent an entity, as entity is an instance/value not schema. and ER diagram is designed to understand schema.
- In a relational model entity is represented by a row or a tuple. or record in a table.

- In an ER diagram an entity set is represented by a rectangle while in a relational model it is represented by a separate table.

- **Attributes:**

→ Attributes are the units defines and describe properties and characteristics of entities.

→ Attributes are the descriptive properties passed by each member of an entity set for each attribute there is a set of permitted values called domain.

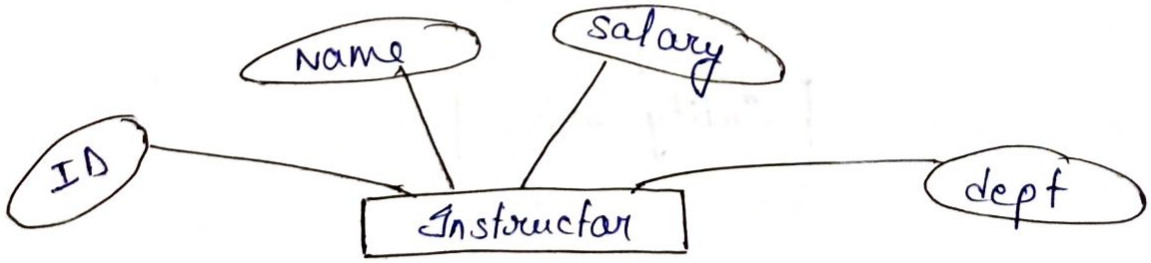
e.g:

Department
↳ only character domain is allowed you cannot write no.s here.

Dept_ID
↳ only integer values allowed.

→ In an ER diagram attributes are represented by an ellipse.

e.g: Instructor (ID, name, salary, dept)



while in relational model they are represented by an independent column.

Instructor

ID	name	salary	dept

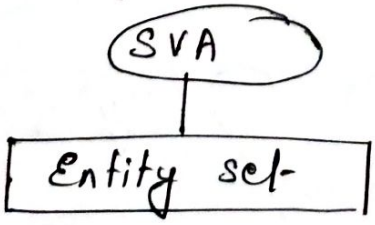
- Types of Attributes:

1) Single valued: Attributes having single value at any instance of time for an entity.
(different values can be there but at a time one value).

e.g: age ← At a time 1 value, but changes with time

designation

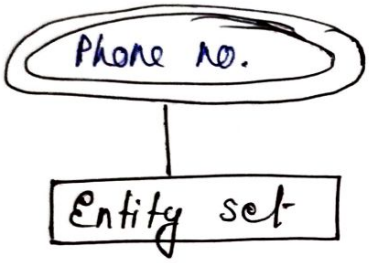
Representation in ER:



2) Multivalued Attribute : Attribute which can have more than one value for an entity at same time.

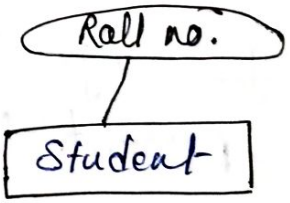
e.g: Phone number, Facebook ID, Email ID

→ A multivalued attribute is represented by a double ellipse in a ER diagram and by an independent / separate table in a relational model.



3) Simple Attribute : Attributes which cannot be further divided into sub parts as simple attributes.

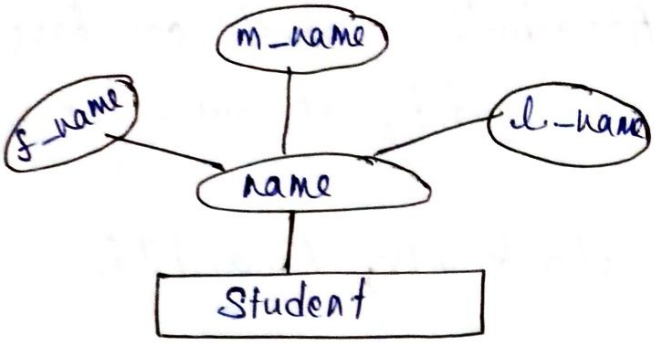
e.g: Age, Roll no.



4) Composite : Attributes which can be further divided into sub parts.

→ A composite attribute is represented by an ellipse connected to an ellipse and in a relation -al model by a separate column.

e.g: address, name, phone number, vehicle number.



Example of composite attribute in an E.R diag.

Student

f_name	M_name	L_name

Exple of composite attribute in R.M

5) Stored: Main attributes whose value is permantly stored in database.

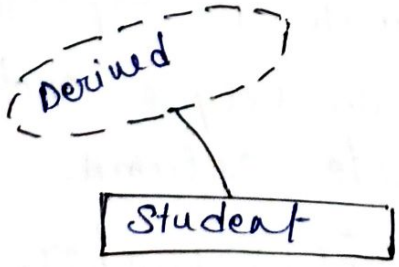
e.g: DOB, DOD

6) Derived: The value of this attribute can be derived from values of other attributes.

e.g: Age can be derived from DOB and present Date.

$$age = f(DOB, Current\ Date)$$

→ It is represented by an dotted ellipse.

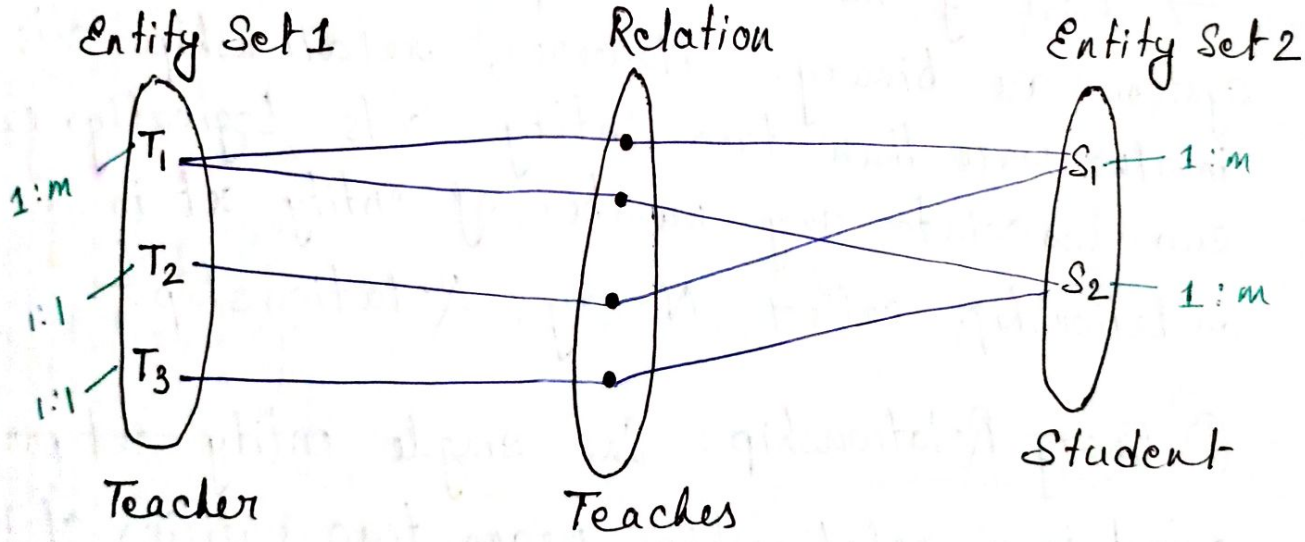


- **Relationship / Association :**

→ Relationship / Association is a relation between two or more entities of same or different entity sets.

→ In ER diagram we can not represent individual relationship as it is used to represent schema not instance.

→ In an ER diagram it is represented by a diamond, while in relational model sometimes through foreign key and other time by a separate table.



Note: When we say relation in ER, we talk about Relationship type/set not individual relation.

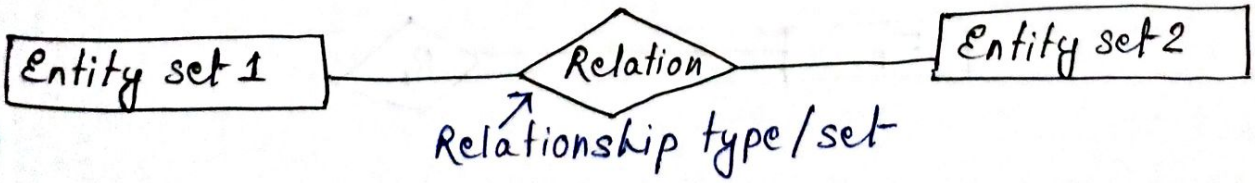


fig: Relationship in ER diagram.

→ Every relationship type has three components:

1) Name
2) Degree
3) Structural Constraints.

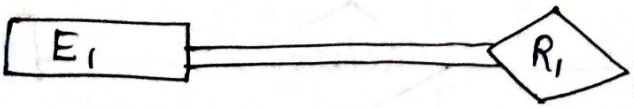
1) **Name**: Every relation must have a unique name.

2) **Degree of a relationship/relationship type/set**:

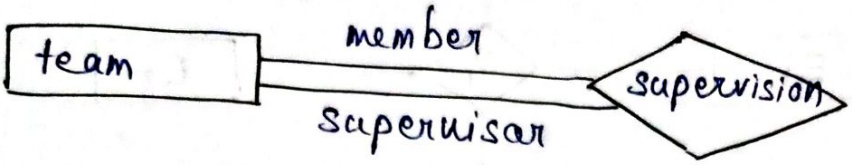
Number of entity set (relations/tables) associated (participated) in the relationship set.

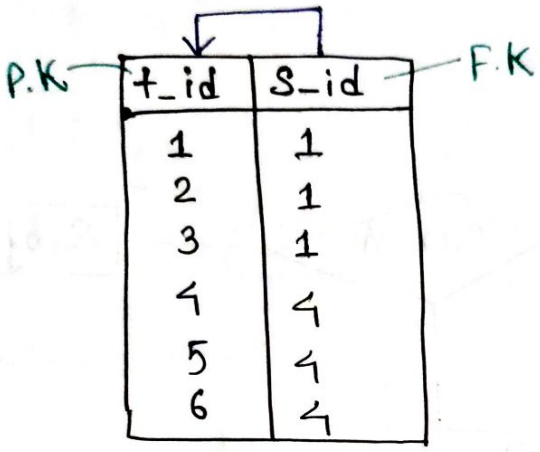
→ Most of the relationship sets in a database system are binary. However, relationship sets involve more than two entity sets. Logically, we can associate any number of entity set in a relationship called N-ary Relationship.

(i) **Unary Relationship**: One single entity set participated in a relationship means two entities of the same entity set are related to each other. These are also called as self-referential relationship set.



e.g: A member in a team may be supervisor of another member in a same teame.





Degree = 1

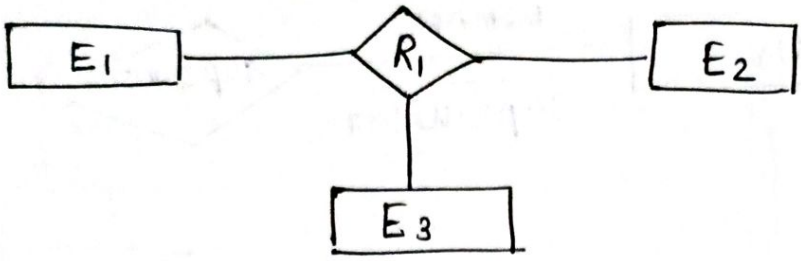
1, 2, 3, 4, 5, 6 are team members among which 1, 4 are supervisor also.

(ii) Binary Relationship: Two entity sets participate in a relationship. most common relationship in DBMS.

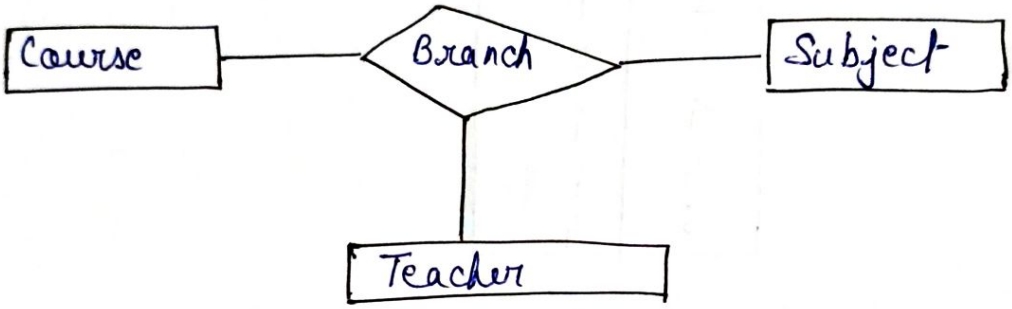
e.g:

Degree = 2

(iii) Ternary Relationship : When three entity sets participate in a relationship.



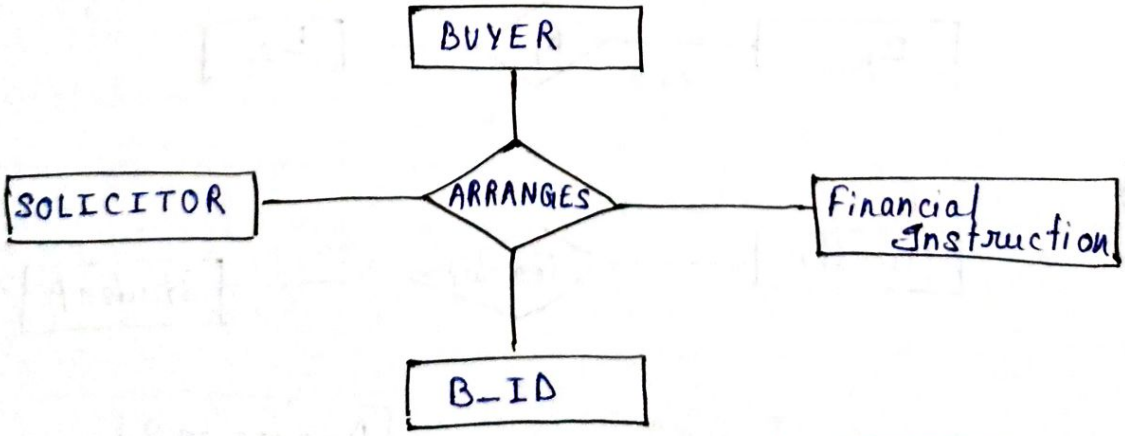
e.g:



Degree = 3

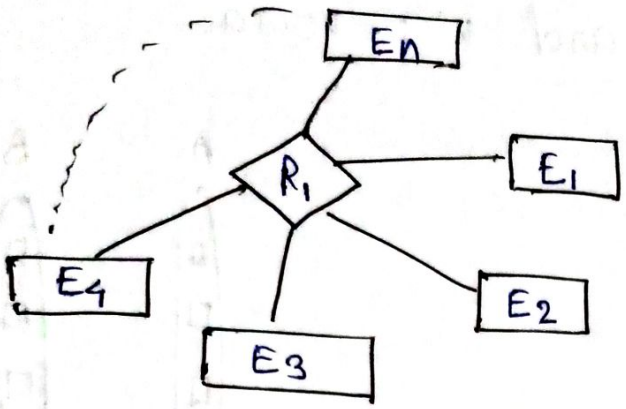
(iv) Quaternary Relationship : When four entities participate in a relationship.

e.g:



Degree = 4

(v) N-ary Relationship: N no. of relationship possible, where n no. of entity set are associated.



Degree = n

3) Structural Schema: An ER enterprise schema may define certain constraints (Rules) to which the contents of a database must conform.

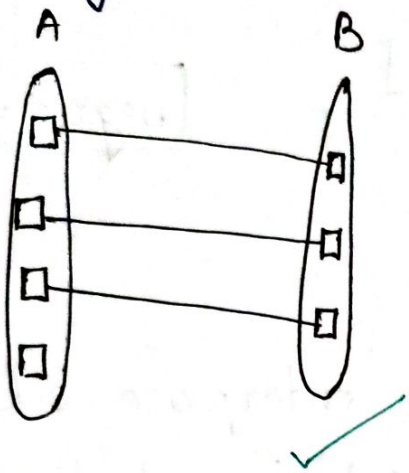
i) Mapping Cardinalities / Cardinality ratio:

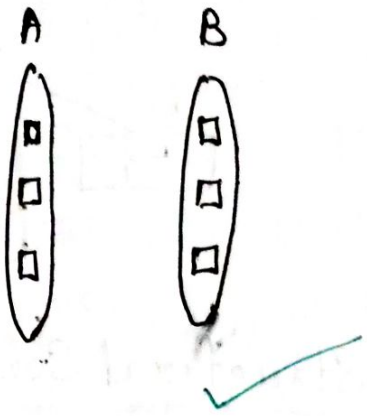
It express the no. of entities to which other entity can be related in a relationship.

There are four possible categories for Cardinality ratio:

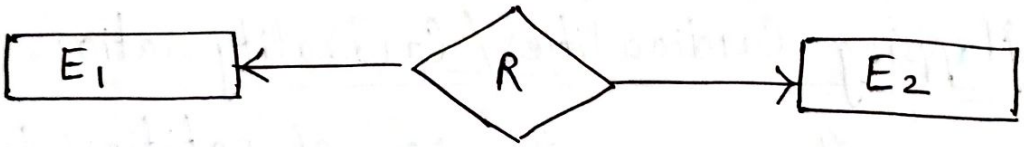
(a) One to One (1:1) Relationship
(b) One to Many (1:M) "
(c) Many to One (M:1) "
(d) Many to Many (M:N) "

(a) **One to One (1:1) Relationship:** An entity in entity set A is associated with at most one entity in entity set B and vice versa.

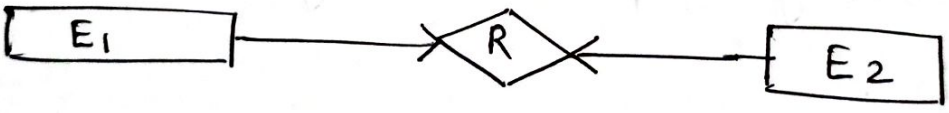




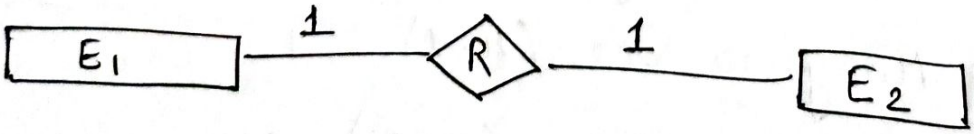
Representation in ER diagram:



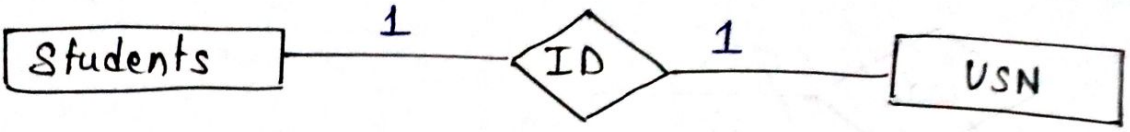
OR



OR

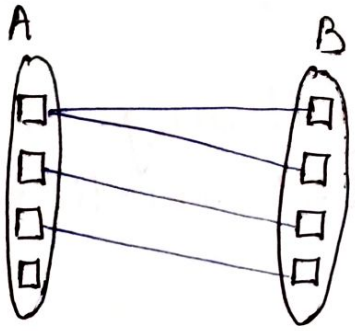


e.g:



Each student will have one USN.

(b) One to Many (1:M) Relationship: An entity in A is associated with any number (zero or more) of entities in B. An entity in B however can be associated with at most one entity in A.

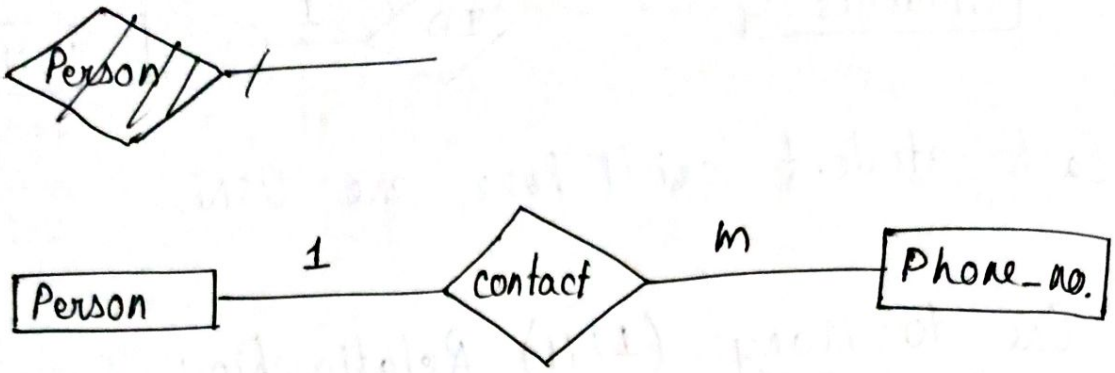


Representation in ER diagram:

OR

OR

e.g:



A person can have ~~no~~ more than one phone number but a phone number is registered to one person.

Q1. Which of the following is true?

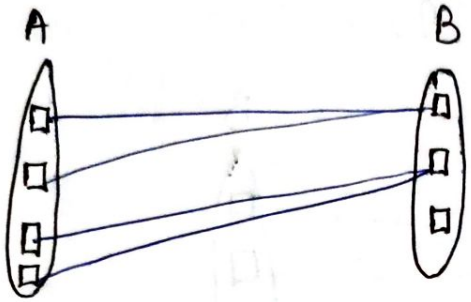
① If it is 1:1 relation then it is 1:M?

② If it is 1:M relation then it is M:1?

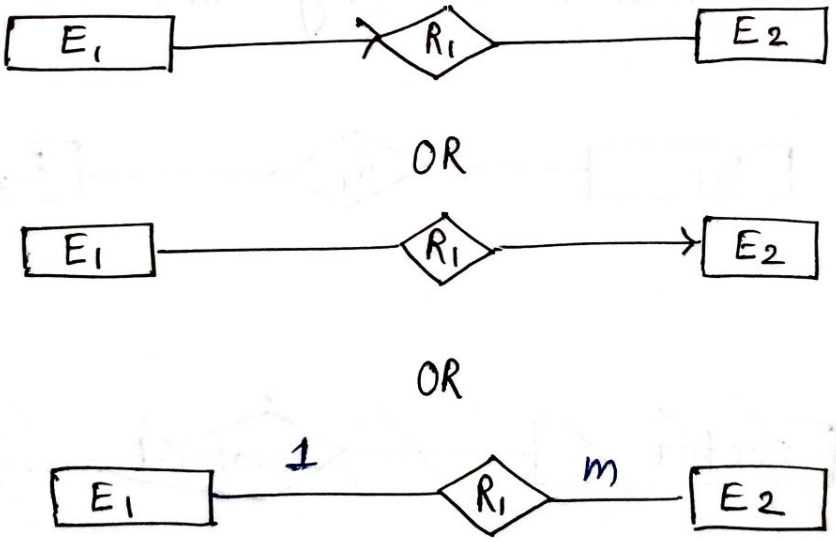
→ ① ✓

© Many to One (M:1) Relationship:

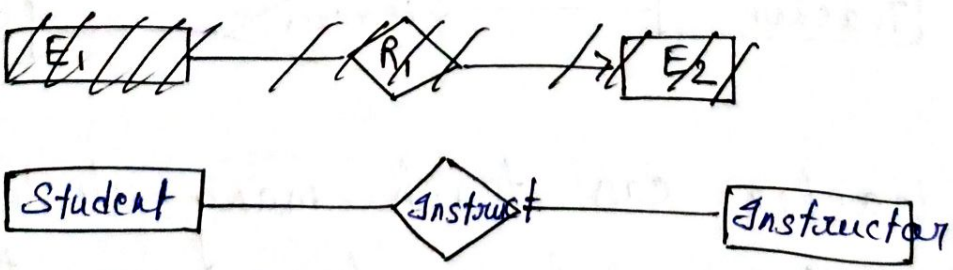
An entity in A is associated with atmost one entity in B, However B can be associated with any number (zero or more) of entities in A.



~~E.g.~~ Representation in ER diagram:



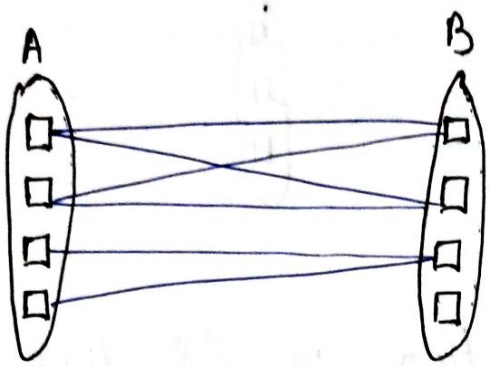
~~e.g.~~:



A student may have 0 or 1 instructor but an instructor can ~~have many~~ instruct many students.

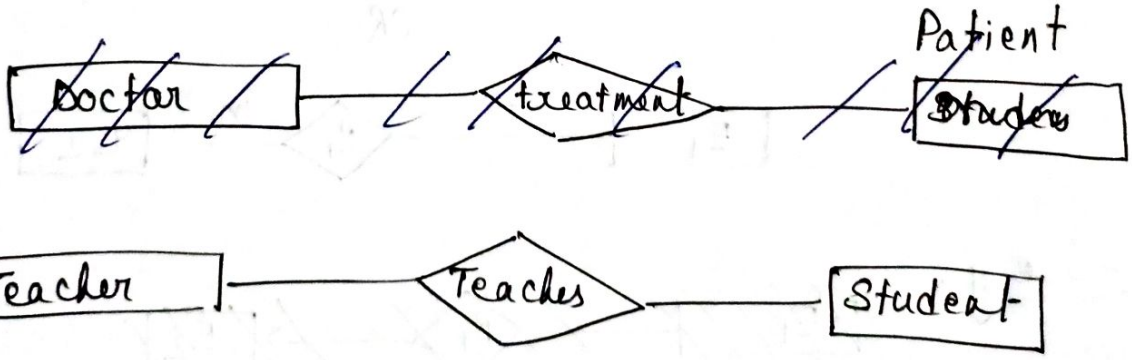
(iv) Many to Many (M:N) Relationship:

An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.



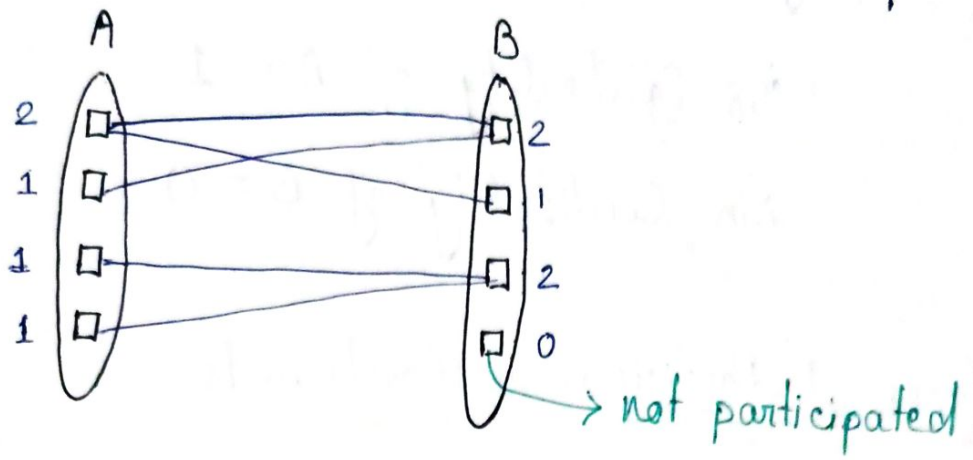
Representation in ER Diagram:

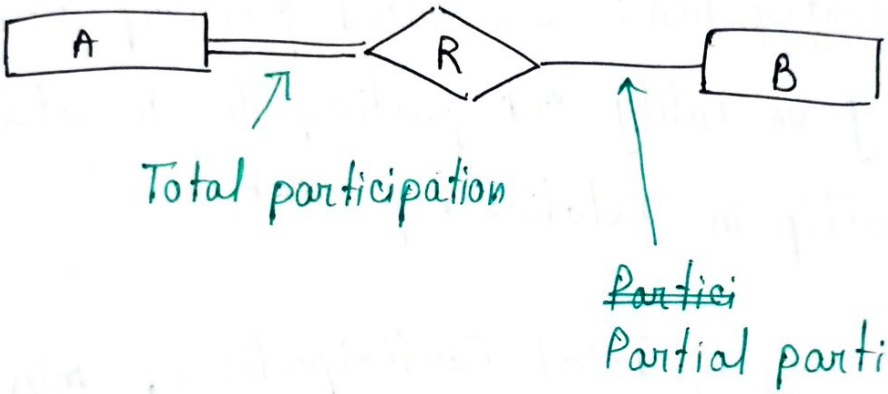
e.g:



A teacher can teach many students and a student can learn from more than one teacher.

ii) Participation Constraints : It defines participations of entities of an entity type/set in a relationship.





→ These constraints specify the minimum and maximum number of relationship instances that each entity must/ can participate in.

• Max Cardinality : It defines the minimum no. of times an entity occurence participating in a relationship.

Max Cardinality of A = 2

Max Cardinality of B = 2

- **Min Cardinality**: It defines the minimum no. of times an entity occurence participating in a relationship.

  Min Cardinality of A = 1

  Min Cardinality of B = 0

- **Types of Participation Constraints**:

(1) **Total Participation**: In total participation every entity of an entity set participate in atleast one relationship in relationship set.

* In case of Total Participation, min cardinality will always be 1.

(2) **Partial Participation**: In partial participation only some entities of entity set participate in Relationship set, that is there exists at least one entity which do not participate in a relation.

* min cardinality = 0.

e.g: 1

Employee

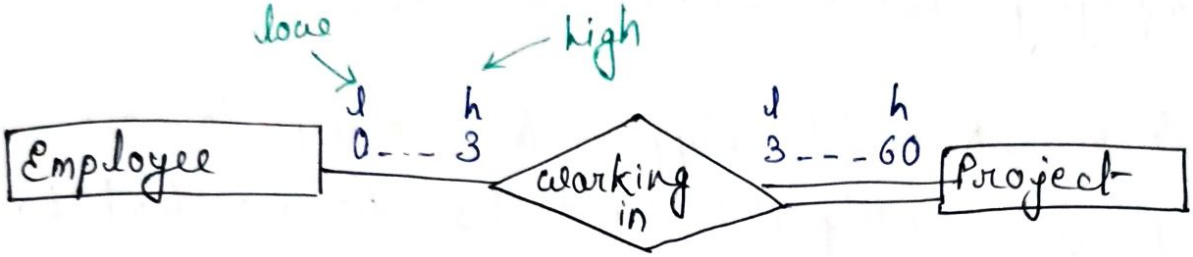
min cardinality = 0

max cardinality = 3

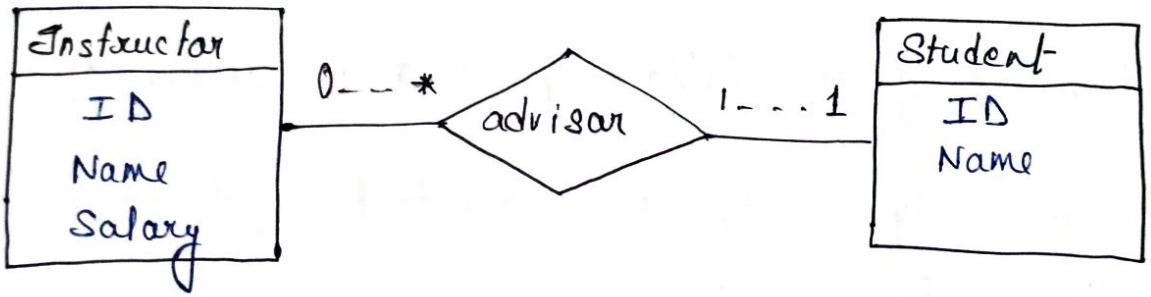
Project

Min Car = 3

Max Car = 60



e.g: 2



→ A line may have an associated minimum and maximum cardinality, shown in the form l---h, where l is the minimum and h is the maximum cardinality.

→ The line between advisor and student has a cardinality constraints of 1---1, meaning the minimum and the maximum cardinality are both 1. that is each student must have exactly one advisor.

→ The limit 0---* on the line between advisor and instructor indicates that an instructor can have zero or more students. Thus the relationship advisor is one-to-many from instructor to students and further participation of student in advisor is total implying that a student must have an advisor.

Q. In an ER model, Suppose R is many-to-one relation-ship from entity set E_1 to E_2. Assume that E_1 and E_2 participate totally in R and that the cardinality of E_1 is greater than the cardinality of E_2.

Which one of the following is true about R?

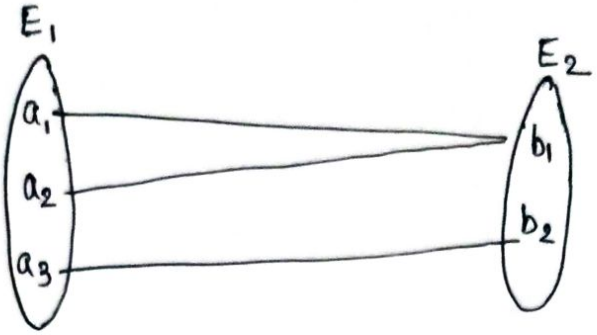
(a) Every entity in E_1 is associated with exactly one entity in E_2.

(b) Some entity in E_1 is associated with more than one entity in E_2

(c) Every entity in E_2 is associated with exactly one entity in E_1

(d) Every entity in E_2 is associated with atmost one entity in E_1.

→



(a) ✓

- Primary key : The set of attribute which allows us to find each tupple uniquely.

- Foreign key : A field/attribute that ~~can uniquely~~ refers to the primary key.

- Strong Entity Set / Identifying / owner entity set :- Has primary key.

- Weak Entity set :- Does not have primary key.

- Conversion of Weak Entity set into a Strong Entity set :

→ A key for an entity is a set of attributes that specifically distinguish entities from each other. keys also help to identify relationships uniquely and thus distinguish relationships from each other. The concepts of super key, candidate key and primary key are applicable to entity sets just as they are applicable to relational schema.

→ An entity set is called strong entity set, if it has a primary key, all the tupples in the set are distinguishable by that key.

→ An entity set that does not passes sufficient attribute to form a primary key is called a weak entity set. It contains discriminator attribute (partial key) which contain partial information about the entity set, but it is not sufficient enough to identify each tuple uniquely. Weak entity set represented by double rectangle.

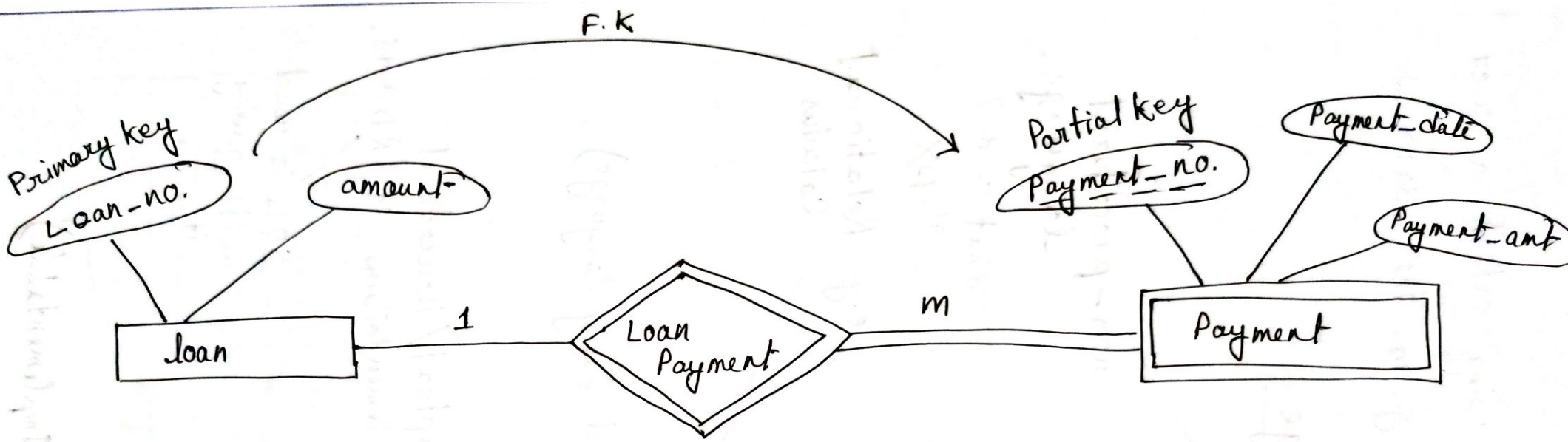
→ For a weak entity set to be meaningful and converted into strong entity set, it must be associated with another entity set called the identifying or owner entity set i.e., weak entity set is said to be existence dependent on the identity set. The identifying entity set is said to own weak entity set that it identifies. The primary key of weak entity set will be the union of primary key (of strong entity set) and discriminator attribute (of weak entity set).

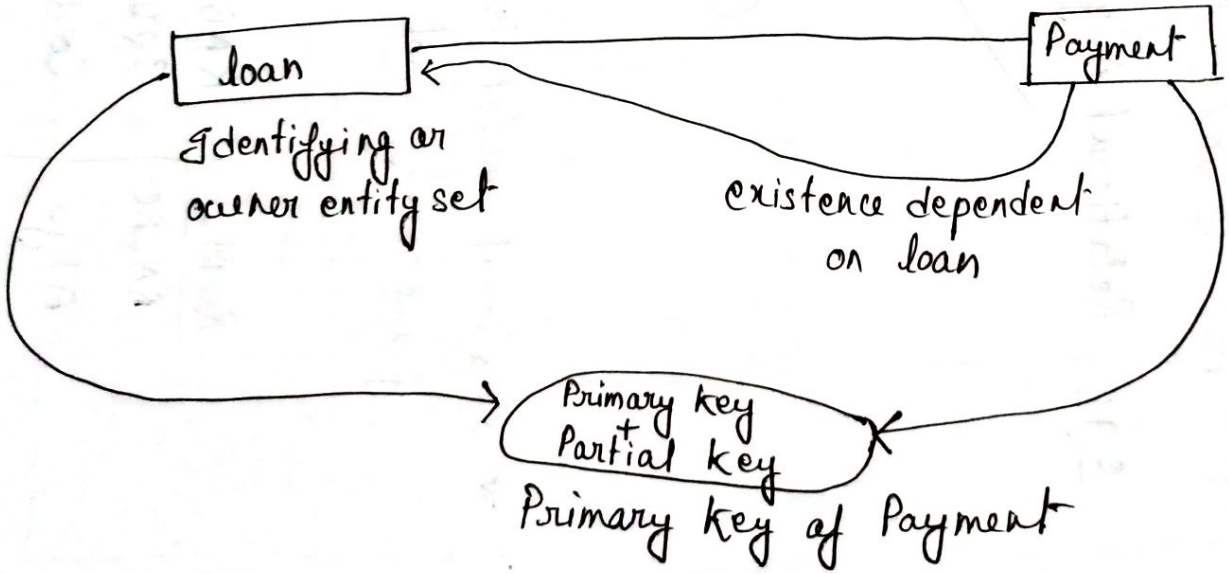
→ The relationship associating the weak entity set with the identifying entity set is called the identifying relationship represented by double diamonds. The identifying relationship is many to one from the weak entity set to identifying entity set and the participation of the weak entity set in relationship is always total.

~~→ The identifying relationship set should not have any d~~

→ The discriminator of a weak entity set is also called the primary key of the weak entity set.

e.g:


F.K
Primary key
Loan-no.
amount
Partial key
Payment_no.
Payment_date
Payment_amt
loan
1
Loan Payment
m
Payment


loan
Payment
Identifying or owner entity set
existence dependent on loan
Primary key + Partial key
Primary key of Payment

P.K

loan

loan_no	amount
1	1,00,000
2	2,00,000
3	3,00,000

Strong Entity set

Partial key

Payment

Pay_no.	P_amt	Date
10	3,000	2nd May
10	3,000	2nd May
11	3,000	3rd May

Weak Entity set
(No P.K no F.K)

To make weak E.S strong/meaningful

primary key + partial key
(of Strong E.S) (of weak E.S)

~~Roadtrans to~~

After conversion we will have two table i.e., loan table (parent table) as default and weak entity set (Payment) will get modified as follows:

loan

loan_no	amount
1	1,00,000
2	2,00,000
3	3,00,000

Strong E.S

P.K of modified table

loan_payment

loan_no	P_no.	P_amt	Date
1	10	3,000	2nd May
2	10	3,000	2nd May
3	11	3,000	3rd May

now it is strong entity set

Q. Construct an ER Diagram for a hospital with a set of patients and set of medical doctors.

→

How to convert E-R Diagram into Relational Model:

1) Convert every strong entity set into an independent table.

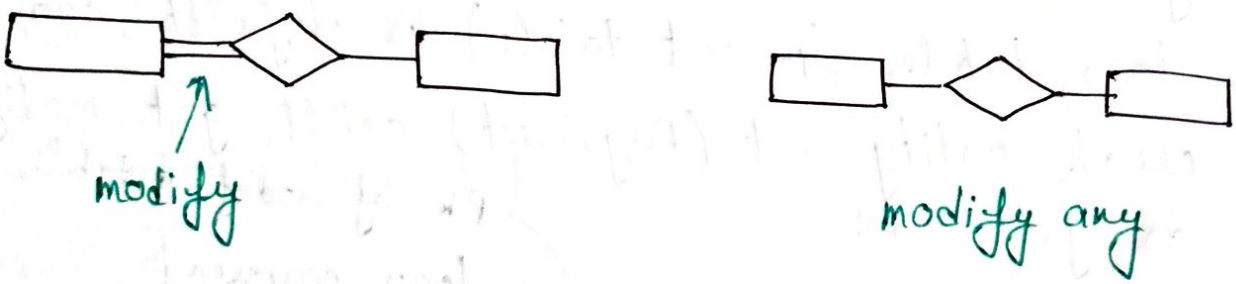
2) Convert weak entity set into a table

3) Separate table for multivalued attribute by taking multivalued attribute and pk of main table as fk.

4) Convert relationship according to following plan:

(a) Conversion of 1-1 relationship (binary)

→ No separate table is required, priority must be given to the side having total participation.



(b) Conversion of 1-n or n-1 relation (binary)

→ No separate table is required, modify n side by taking pk of 1 side as foreign key.

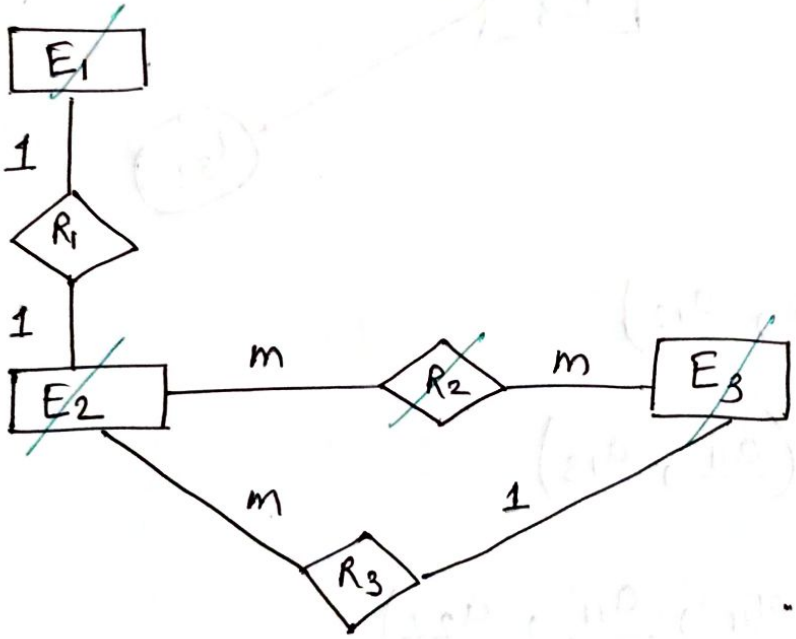
© Conversion of m-n relationship (binary)

→ Separate table is required, take pk of both table and declare it as pk of new table.

ⓓ Conversion of relationship having degree more than 2

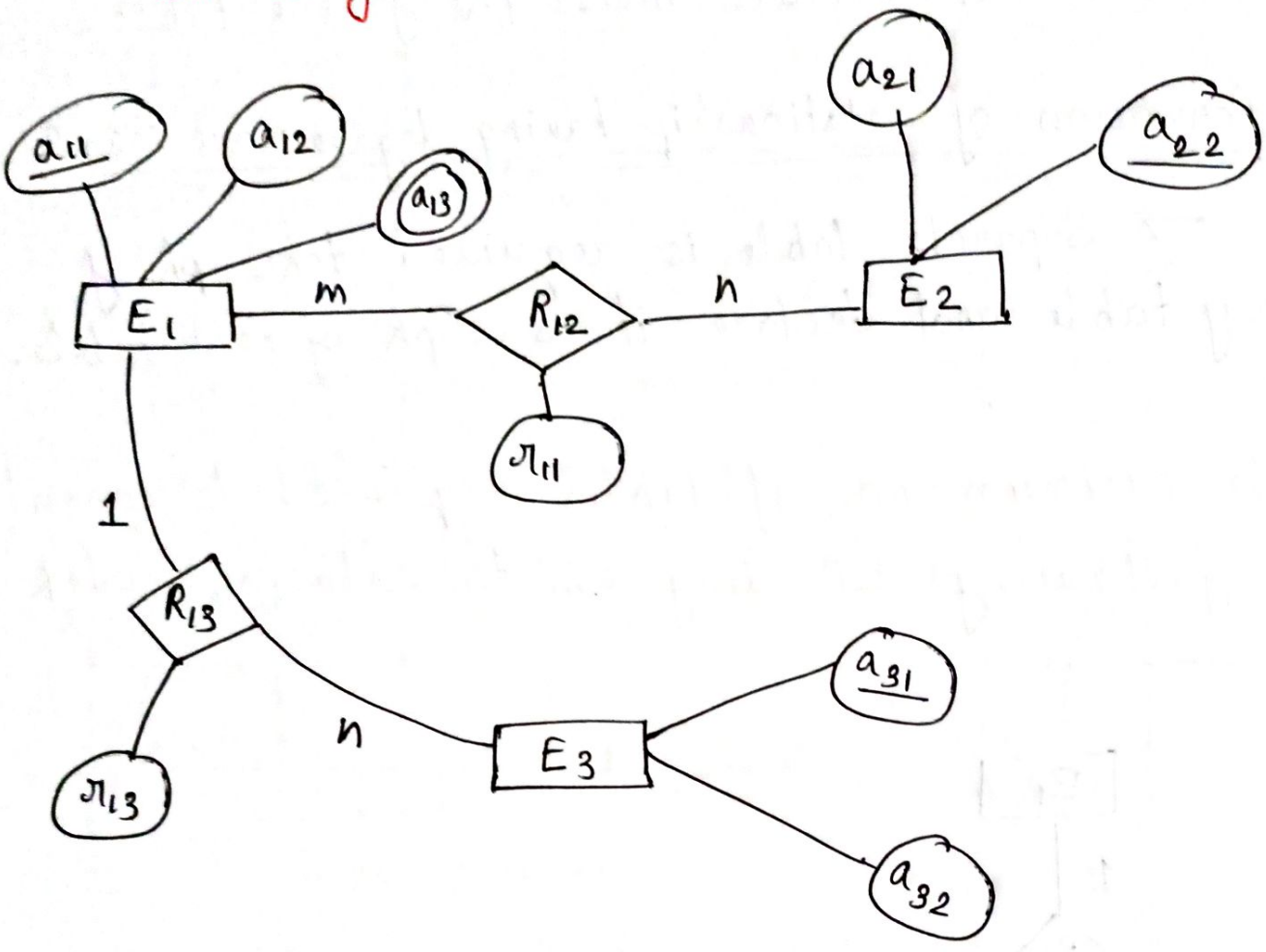
→ Separate table is required take pk of every table and declare it as a pk of new table.

Q. The minimum no. of tables required to convert the following ER diagram to relation model is ——.



→ 4 tables

Q. Determine the relational schema for following ER Diagram.



→ (1) $E_1 (\underline{a_{11}}, a_{12})$

(2) $A_{13} - E_1 (\underline{a_{11}}, a_{13})$

(3) $R_{12} (\pi_{11}, \underline{a_{11}}, \underline{a_{22}})$

(4) $E_2 (a_{21}, \underline{a_{22}})$

(5) $E_3 (\underline{a_{31}}, a_{32})$

2.2.1 The Three-Schema Architecture

The goal of the three-schema architecture, illustrated in Figure 2.2, is to separate the user applications and the physical database. In this architecture, schemas can be defined at the following three levels:

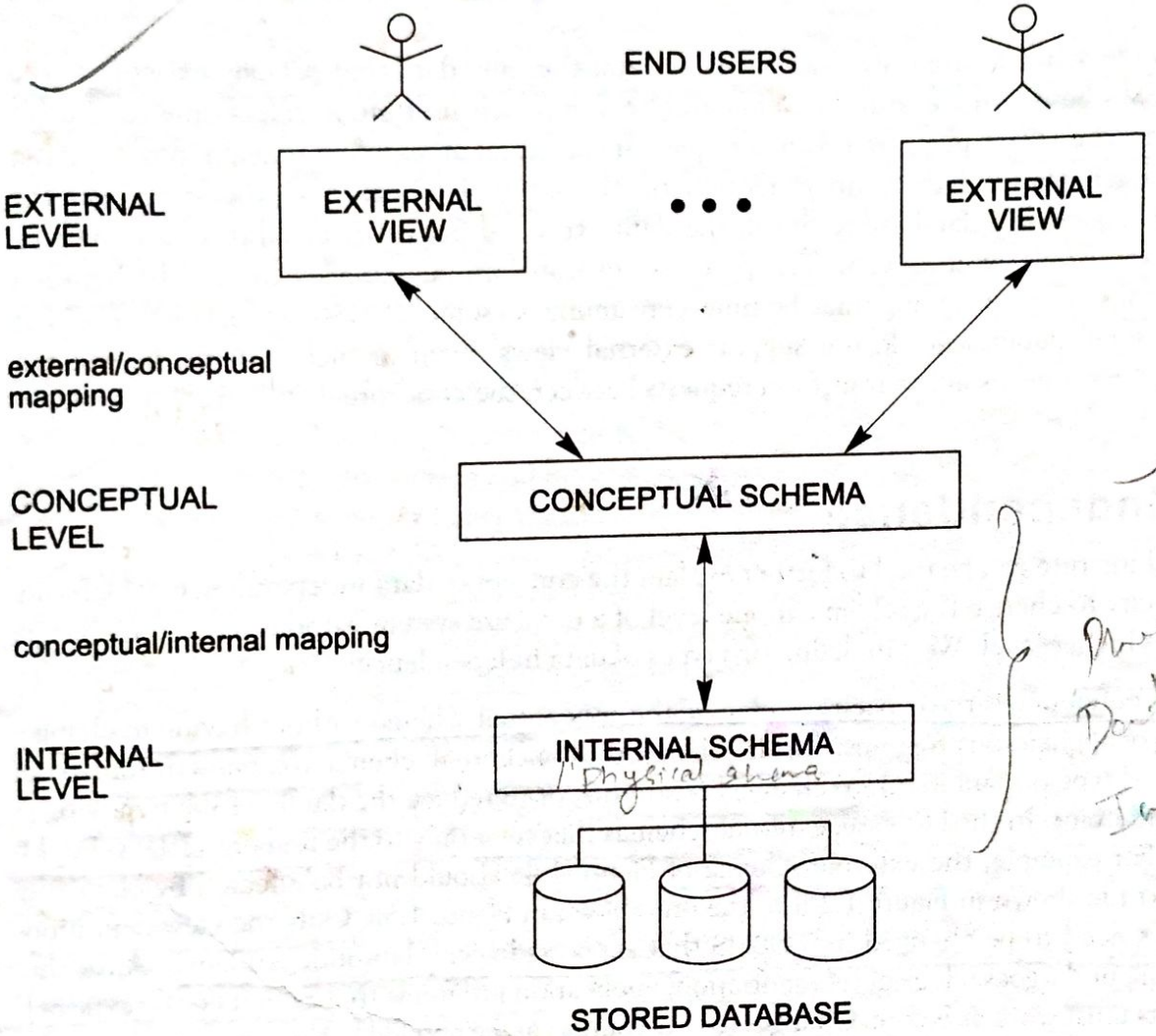


FIGURE 2.2 The three-schema architecture.

1. The **internal level** has an **internal schema**, which describes the physical storage structure of the database. The internal schema uses a physical data model and describes the complete details of data storage and access paths for the database.
2. The **conceptual level** has a **conceptual schema**, which describes the structure of the whole database for a community of users. The conceptual schema hides the details of physical storage structures and concentrates on describing entities, data types, relationships, user operations, and constraints. Usually, a representational data model is used to describe the conceptual schema when a database system is implemented. This *implementation conceptual schema* is often based on a *conceptual schema design* in a high-level data model.
3. The **external** or **view level** includes a number of **external schemas** or **user views**. Each external schema describes the part of the database that a particular user group is interested in and hides the rest of the database from that user group. As in the previous case, each external schema is typically implemented using a representational data model, possibly based on an external schema design in a high-level data model.

The three-schema architecture is a convenient tool with which the user can visualize the schema levels in a database system. Most DBMSs do not separate the three levels completely, but support the three-schema architecture to some extent. Some DBMSs may include physical-level details in the conceptual schema. In most DBMSs that support user views, external schemas are specified in the same data model that describes the conceptual-level information. Some DBMSs allow different data models to be used at the conceptual and external levels.

Notice that the three schemas are only *descriptions* of data; the only data that *actually* exists is at the physical level. In a DBMS based on the three-schema architecture, each user group refers only to its own external schema. Hence, the DBMS must transform a request specified on an external schema into a request against the conceptual schema, and then into a request on the internal schema for processing over the stored database. If the request is a database retrieval, the data extracted from the stored database must be reformatted to match the user's external view. The processes of transforming requests and results between levels are called **mappings**. These mappings may be time-consuming, so some DBMSs—especially those that are meant to support small databases—do not support external views. Even in such systems, however, a certain amount of mapping is necessary to transform requests between the conceptual and internal levels.

2.2.2 Data Independence

The three-schema architecture can be used to further explain the concept of **data independence**, which can be defined as the capacity to change the schema at one level of a database system without having to change the schema at the next higher level. We can define two types of data independence:

1. **Logical data independence** is the capacity to change the conceptual schema without having to change external schemas or application programs. We may change the conceptual schema to expand the database (by adding a record type or data item), to change constraints, or to reduce the database (by removing a record type or data item). In the last case, external schemas that refer only to the remaining data should not be affected. For example, the external schema of Figure 1.4a should not be affected by changing the GRADE_REPORT file shown in Figure 1.2 into the one shown in Figure 1.5a. Only the view definition and the mappings need to be changed in a DBMS that supports logical data independence. After the conceptual schema undergoes a logical reorganization, application programs that reference the external schema constructs must work as before. Changes to constraints can be applied to the conceptual schema without affecting the external schemas or application programs.
2. **Physical data independence** is the capacity to change the internal schema without having to change the conceptual schema. Hence, the external schemas need not be changed as well. Changes to the internal schema may be needed because some physical files had to be reorganized—for example, by creating additional access structures—to improve the performance of retrieval or update. If the same data as before remains in the database, we should not have to change the conceptual schema. For example, providing an access path to improve retrieval speed of SECTION records (Figure 1.2) by Semester and Year should not require a query such as "list all sections offered in First Semester of 1998" to be changed, although the query would be executed more efficiently by the DBMS by utilizing the new access path.

Whenever we have a multiple-level DBMS, its catalog must be expanded to include information on how to map requests and data among the various levels. The DBMS uses additional software to accomplish these mappings by referring to the mapping information in the catalog. Data independence occurs because when the schema is changed at some level, the schema at the next higher level remains unchanged; only the *mapping* between the two levels is changed. Hence, application programs referring to the higher-level schema need not be changed.