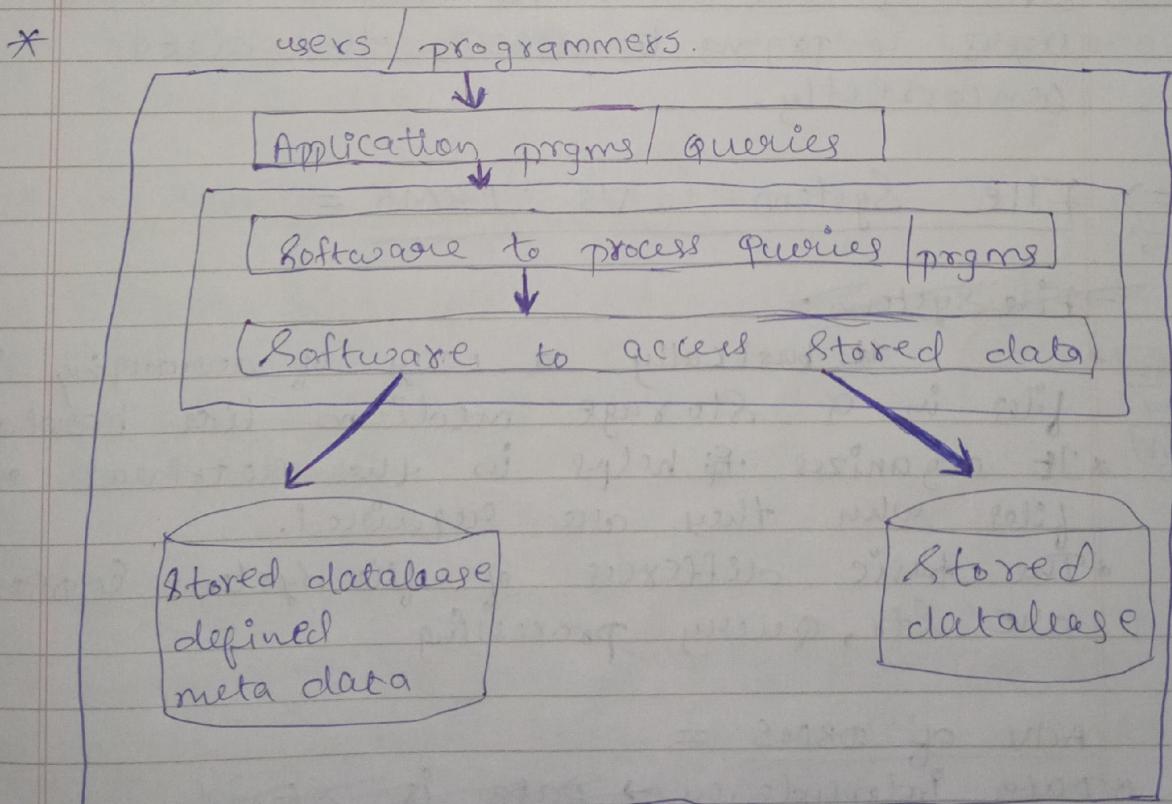
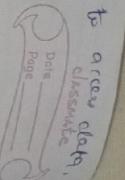


## DBMS

- \* Data = Raw facts / unprocessed facts.
- \* Info... = processed facts / meaningful data.
- \* Database = It is an integrated collection of logically related data into a common pool that provides data for many applications.
- \* Database management system = It is a collection of programs that enables users to create & maintain a database.  
~~and therefore a general purpose software system that facilitates the process defining, constructing, manipulating & sharing data b/w various applications.~~



(D) → databases → easy to access data.



⇒ Functionality of (D) :-

1) Define → defining a (D) involves specifying the datatypes, structures & constraints for the data to be stored in the (D).

2) Construct → (editing data)

Constructing the (D) is the process of storing the data itself on some storage medium (D) controlled by DBMS.

3) Manipulating a (D) → It includes such (D)s as querying the (D), showing the (D) → It allows multiple users to query to access the data correctly.

⇒ File System VS DBMS =

File System

\* It basically a way of arranging the files in a storage medium like hard disk memory.

\* It organizes & helps in the retrieval of files when they are required.

\* The basic differences of file system & DBMS are Ktr, query processing

Adv of DBMS =

\* Data independence → data is stored

\* Efficient file access

\* Data security

⇒ Database User =

1) Naive user

2) Application programmer

3) Sophisticated user (very working people)

⇒ DBA (D) Administrator =

\* A DBA is the info technician responsible for directing & performing all activities related to maintaining (D) environment.

\* A DBA makes sure an organisation (D) & its related applications operate functionally & efficiently.

⇒ (d) Models (Design / str) =

① Hierarchical (m)

tree like structure (one to many)

② Network (n)

graph like str. (many to many)

③ Entity relationship

ER

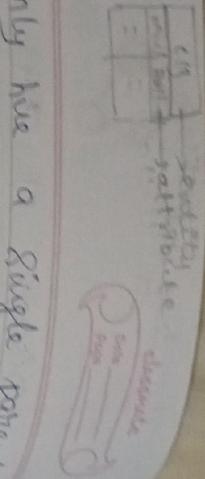
④ Relational

SQL

⑤ Hierarchical (m) =

\* Data are organised in a tree like str with a single root node to which all the other data are linked.

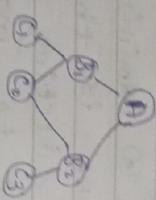
\* Hierarchy starts from root node & expands like a tree adding child nodes to parent nodes.



- \* child node will only have a single parent node.
- \* It is a 1 to many relationship.

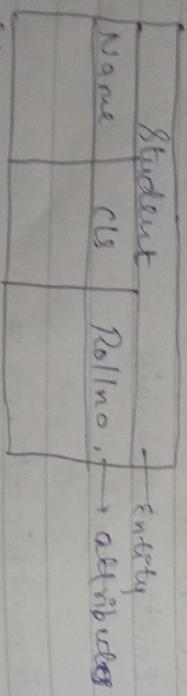
2) network (n) =

- \* It is a many to many relationship.
- \* Data are organized in a graph like structure where nodes are allowed to have more than one parent node.



3) Entity-Relationship (m) =

- \* Relationships are created by dividing objects into entities & its characteristics into attributes.
- \* Different entities are related using relationships.



4) relation (m) =

- \* Data is organised in 2D tables & the relationship is maintained by storing a common field.
- \* Basic structure of data is a relational

model is taken

Schemas & Instances =

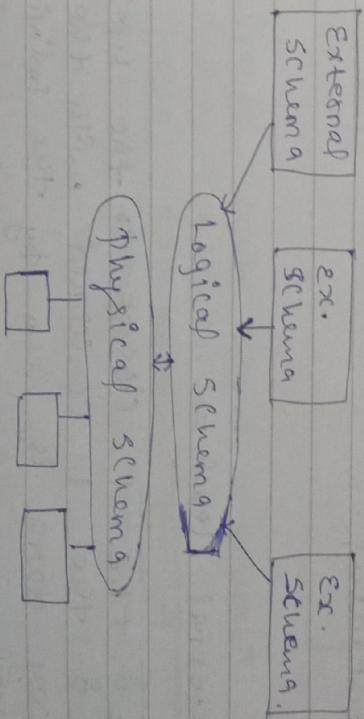
- \* Schemas = Description of a datalage [overall design & (d)] which is specified during (d) design & it is not expected to change frequently.
- \* It specifies the names of the entity & attributes & the relationships existing thus entities.

Instance =

The data in the (d) at a particular moment in time → (d).

It is also → the current set of occurrences instance in (d).

⇒ 3 Scheme Architecture =



### → Physical schema =

- \* Internal level  $\rightarrow$  physical level. It is the closest to physical storage.
- \* It is the concern with the way the data is physically stored.
- \* The internal level has an internal schema which describes the physical storage.
- \* The internal schema uses a physical data model to describe the complete details of data storage & access paths for the DB.

### → Logical schema / conceptual schema =

- \* The conceptual level has a design which is for a community of users.
- \* The conceptual schema hides the details of the physical storage & concentrates on describing either
  - (a) Relation ship, user operation & constraint.

### $\Rightarrow$ Database language =

data def lang	Data manipulation lang	Data control lang	Transmission control lang
CREATE	SELECT	GANT	COMMIT
ALTER	INSERT	REVOKE	ROLL BACK
DROP	UPDATE		
TRUNCATE	DELETE		
RENAME			

Create  $\rightarrow$  Create ~~new~~ obj in ~~existing~~ db.

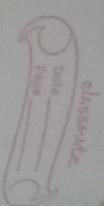
Alter  $\rightarrow$  Alter the str of db.

Drop  $\rightarrow$  delete ~~old~~ obj from db.

Truncate  $\rightarrow$  ~~remove~~ remove all rec from table.

Rename  $\rightarrow$  ~~change~~ rename an obj

Data def lang = (DDL)



### =>

### Data Independence =

- \* DB schema difference  $\rightarrow$  changes in DDL, changes in data ( $\rightarrow$  data  $\rightarrow$  schema)
- \* capacity to change the schema at lower without having the change the schema at the higher level  $\rightarrow$  D.I
- a) Logical D.I  $\rightarrow$  logical  $\rightarrow$  physical
- b) Logical D.I = used to separate external from the logical level

### b) Physical D.I =

used to separate logical levels from physical level.

(E) → table.

- \* DDL → Data Def Lang (d)  
\* used to define pattern  
\* used to create schema, tables, indexes, etc.
- \* used to interact with DBMS.

→ CREATE → used to create objects in (d).

## II Data Manipulation Lang = (DML)

- \* DML → Data Manu Lang.  
\* used for accessing & manipulating data in (d).

- 1) SELECT → retrieve data from (d).
- 2) INSERT → Insert data into a table.
- 3) UPDATE → update existing data within a table.
- 4) DELETE → dlt a record from (d).

## III Data Control Lang = (DCL)

(giving permissions in (d).)

- \* used to control privilege in a (d).

GRANT → gives user access privilege to the (d).

REVOKE → takes back permission from user.

[grant → permission established  
revoke → deny permission]

## IV Transaction Control Lang = (TCL)

- \* used to run the changes made by DML statements.

(change transaction block)  
COMMIT → saves the work done

ROLLBACK → restores the (d) to the original  
(change transaction & transaction, cancel also)

⇒ Interface = (5 types)

- \* menu-based (1) = (menu select mode)

2) menu-based (1) = (menu select mode)

3) Options → present the user with list of options → menus

4) Form-based (1) = (form type)

- \* It displace a form to each other.
- \* users can fillout all forms entries to insert a new data, they can fillout certain entries

## 3) Graphical user (3) = (GUI)

(diagrammatical (3))

- \* It typically displace a schema (list of database) to the user in: diagrammatical form.

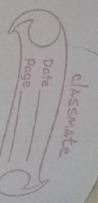
## 4) Natural Lang (1) =

- \* It accepts requests actions in eng [house other lang.]

## 5) Speech input & output =

- \* limited use of speech as an input  
query & speech as an answer to the question / result of request is becoming

## (c) → Entity



classmate  
Date \_\_\_\_\_  
Page \_\_\_\_\_

common place.

⇒ Basic concept of Entity relationship model :-

- Entity :- It is a thing / Sub in the real world that is distinguishable from all other obj.
- It has Set of properties & the values for some set of properties may uniquely identify an entity.

→ Attributes :- (A).

A An (E) is represented by a set properties

→ (A).

- \* (A) are descriptive (prop) processed by each member of an (E) Set.

6 types →

- 1) composite (A) (name → first name, 2nd name)
- 2) simple (A) (A) cannot have (A)
- 3) single valued (A)
- 4) derived (A)
- 5) multi valued (A)
- 6) stored (A)

(4) derived (A)

- (A) that can be derived from other (A)s.

Eg → age of a person.

Person (A) → Age (A)

Person (A) → Name (A)

(5) stored (A).

- (A) form which the value of other (A)s are derived.
- (A),

Eg → DOB of a person.  
Age (A) → Birth (A)

Age (A) → Birth (A)

⇒ Entities =

\* Types →

- 1) Tangible → Physically (common objects)
- 2) Intangible → Logical (information)

- (A) that can be divided into further (A)s.
- eg → Name can be divided into first name, 2nd name.

- (A) that cannot be divided into further (A)s.
- eg → Salary cannot be divided into anything else.

be divided by 1<sup>st</sup> name, 2<sup>nd</sup> name.

be divided to others.

\* Entity type = Representing entity with a name.

RollNo	name	Branch
101	Sam	CS
102	Ram	BCA
103	Sam	BBA

RollNo	name	Branch
101	Sam	CS
103	Sam	BBA

RollNo	name	Branch
101	Sam	CS
103	Sam	BBA

- ① entity set → Student.
- ② entity type → 

101	Sam	CS
103	Sam	BBA

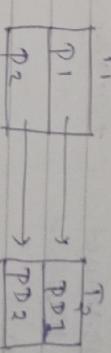
 → Fig 1.1  
(having common entity)

- \* entity type → 2 types.

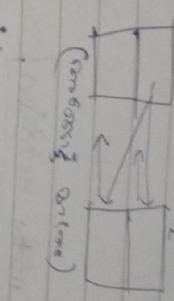
a) Strong entity type = Entity type having key attribute.

b) weak e. type = Entity type not having a key (A).

III one to many =

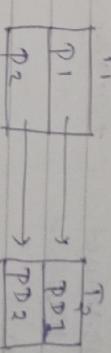


\* If 1 record of  $T_1$  is only related to 1 record of  $T_2$ .



IV many to many =

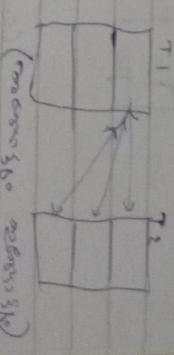
→ many to many =



V many to many =

→ many to many =

\* 1 rec of  $T_1$  is related to multiple recs  $T_2$ .



⇒ Relationship in table =

- ① one to one.

T1	T2
1	2

a) one to one.

b) one to one.

c) one to one.

d) one to one.

e) one to one.

f) one to one.

Fig 1.2

name	branch
Sam	CS
Ram	BCA
Saroj	BBA

→ Fig 1.2 (no key (D))

T1	T2
1	2
2	1

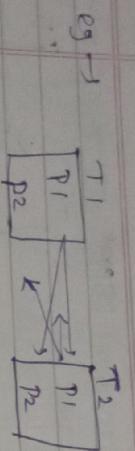
→ Fig 1.2 (no key (D))

g) one to one.

h) one to one.

i) one to one.

j) one to one.



### [Components Entity Relation]

Entity ↓  
Attribute ↓  
Relationship

- \* Strong entity      \* simple
- \* weak entity      \* composite
- \* single-valued      \* 1 to 1
- \* multi-valued      \* 1 to many
- \* stored
- \* derived

→ Notations → Strong entity → [ ]  
→ weak entity → [ ]

→ Attributes notations →

- \* Simple | Single →
- \* multi valued →
- \* composite →
- \* derived →

→ Relationship notations =



- 1 to many →
- many to many →

→ Keys in DBMS =

- 1) candidate
- 2) primary
- 3) Alternate
- 4) Super
- 5) Foreign

eg →

E-id	E-name	E-address
101	John	2103
102	AK	21045
103	BK.	21056

→ \* candidate → E-id , E-address (unique key)  
 → \* primary → (E-key → no other value can be present)  
 → E-id → E-key , ~~E-address (not null)~~  
 (only 1 in a table)  
 (enforcing consistency)

1) candidate key →

- \* It is a subset of a Super key set whose the key which contains no redundant attribute is none other than a candidate key
- \* Role of key is to identify a table row / column uniquely.

2) Primary key →

- \* It is a column / columns containing

value that uniquely identifies each row in a table.

3) Alternate key =

e.g. →  $\text{b} \cdot \text{m} \cdot \text{d} \cdot \text{e} \cdot \text{r} \cdot \text{p} \rightarrow \text{email}$

- \* except p.key all c.keys will be considered as A.keys.

4) Super key =

\* combination of columns that uniquely identifies any row units in a relational database management system.

\* A c.key is closely related concept since the super key is reduced to the min no of columns required to uniquely identify each row.

5) Foreign key =

- "  
 $\begin{array}{|c|c|}\hline \text{id} & \text{name} \\ \hline 1 & \text{John} \\ \hline 2 & \text{Doe} \\ \hline \end{array}$ " \* It is a field in one table that refers to the p.key in another table.
- \* The table with the f.key called as the child table & the table with the p.key → parent table.

(None)

⇒ constraints in DBMS =

- \* constraints are the set of rules that ensures when an authorized user

modifies the data they do not disrupt the data consistency or types of data that can be inserted, updated, deleted from the table.

→ types of constraint = (con)

a) key (con) (unique constraint)

b) Entity integrity (con) (null constraint)

c) Domain (con)

d) Referential integrity (con).

a) key (con) =

\* All values of the p.key must be unique (i.e.) that no 2 tuples can have the same combination for all their attributes

b) Entity integrity (con) =

\* ~~Entity~~ states that the p.key value cannot be null.

\* we use the p.key to identify individual rows in the table

c) Domain (con) = (values in the table domain)

\* ~~Entity~~ it defines the domain / set of values for an attribute.

\* It specifies that the value taken by the attribute must be atomic value from its domain.

id	name	cls.
1	John	10
2	Alice	10
3	Bob	10
4	Cathy	10

→ Table / Specification  
 id, name, cls. → attr / bival  
 each row → tuple / values  
 ↳ values domain

d) referential integrity (con) = (like f-key)

- \* This (con) is enforced using a f-key references the p-key of a relation.
- \* It's specifies that all the values taken by the f-key must be available in the relation at the p-key / be null.