INTRODUCTION TO DBMS

DBMS stands Data Base Management System.

DATA :- Data is any known facts or any smallest information that can be recorded and have implicit meaning.

Eg:- Deepak , BTECH , IT, 9999

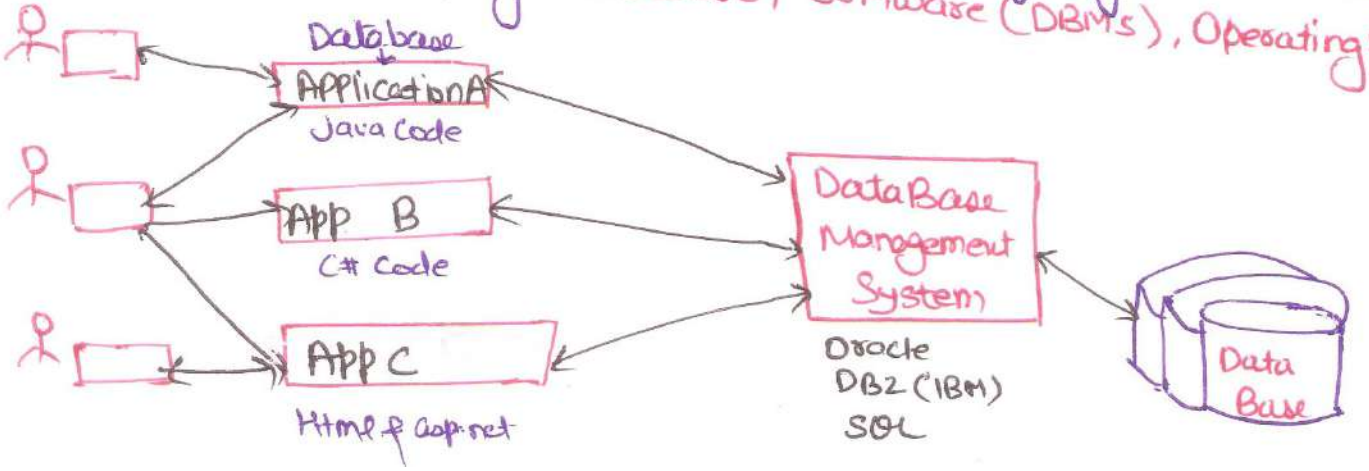
DATA BASE :- It is collection of **Related DATA**

Here Related Data means if are collecting the information of an employee it should be related to employee . And DataBase should have Collection of this employee data

Eg.

	Name	Age	Designation	Salary
Related Data → Collection of Related Data	Deepak	31	Asst. Manager	19000
	Sanjay	29	Software Engg.	14000
	⋮			
	Rahul	29	Data Analyst	16000

DATABASE SYSTEM :- It is a system in which an user uses the Database Technology in order to achieve an organized, store a large no. of dynamic associated Data with the help of Hardware, Software (DBMS), Operating sys.



DataBase System Composed of 5 major parts:
Hardware, People, Data
Software(DBMS), Procedures.

DATABASE MANAGEMENT SYSTEM:- It is a set of software programs that allows users to create, edit and update data in database files, and store and retrieve data from those database files.

Eg:
- → Oracle
- → MS SQL Server
- → MYSQL
- → SQL
- → DB2 (IBM)

PROPERTIES OF DATABASE

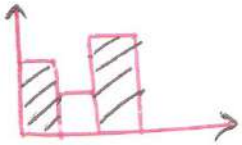
* Question Comes in mind Why we use DATABASE SYSTEMS :-

Ans :- There are 6 important features of DataBases

- → Completeness
- → Integrity
- → Flexibility
- → Efficiency
- → Usability → Redundancy Less

COMPLETENESS : This property ensures that user can access the data they want including ad hoc queries :-

Ad hoc Queries :- They are those Queries which Cannot be determine prior to the Moment the Query is issued.



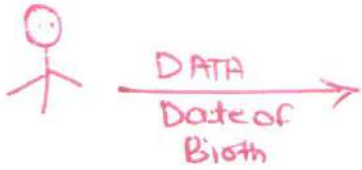
Eg :- 1) Display Bar chart, pie chart for a Vendor and his stocks.

2) Showing plot chart & stack bar chart for an employee

***Integrity :-** Database integrity ensures that Data entered into the DataBase is Accurate, Valid, and Consistent

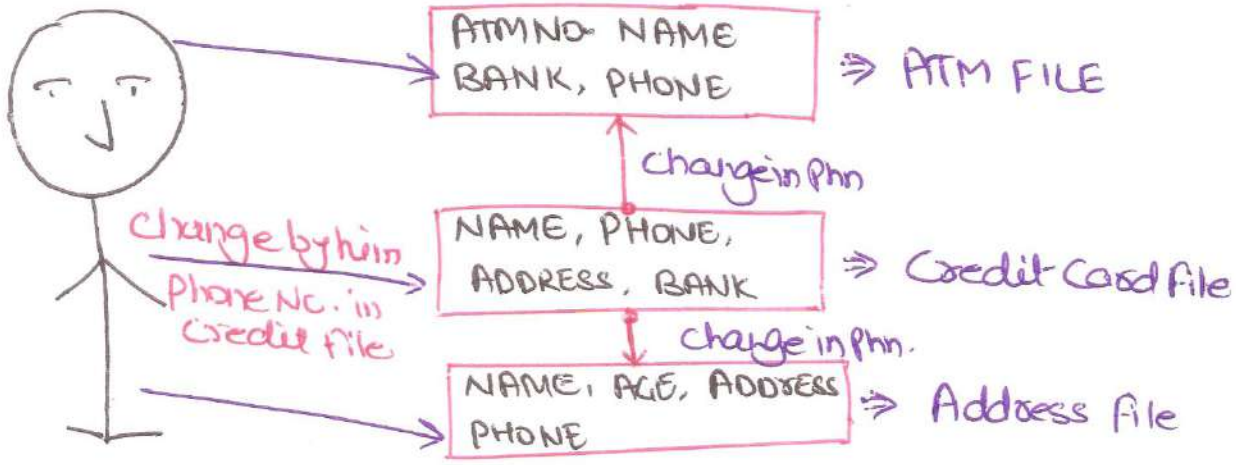
→ **Accurate :-** Data Accuracy refers that data values must be right value and must be represented in a consistent manner.

Eg

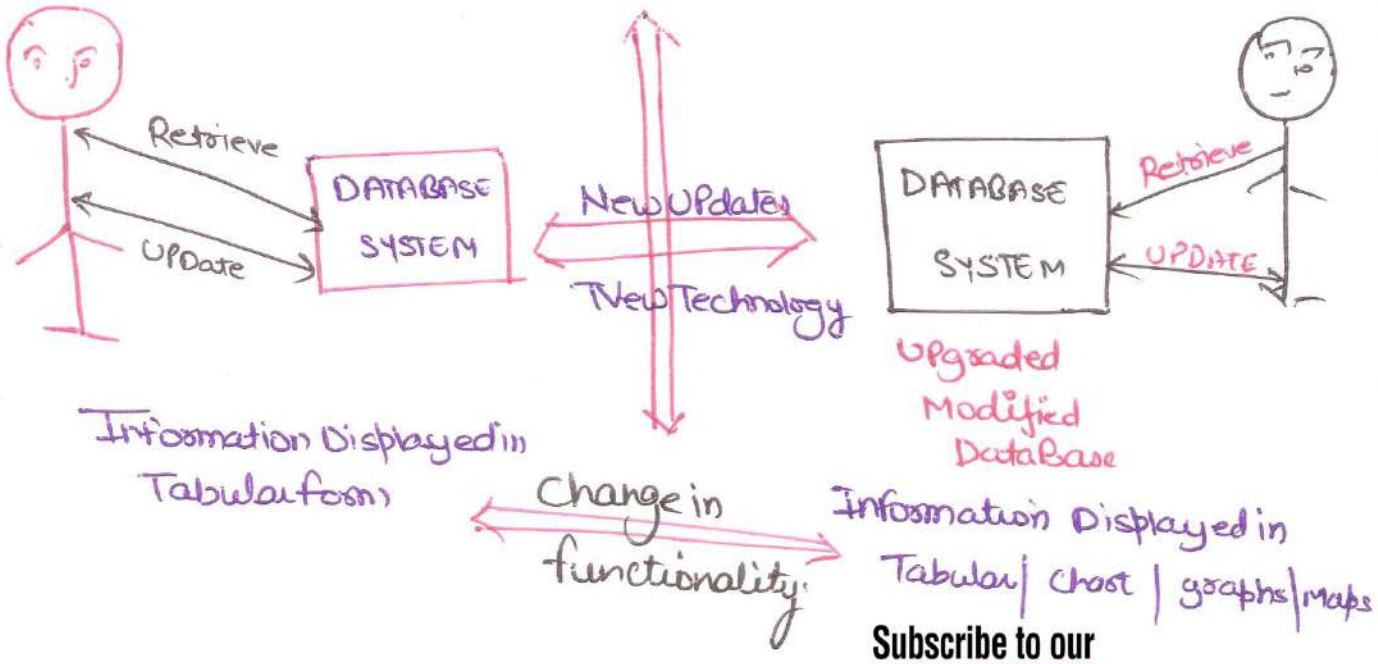


VALID :- It refers Data should be correct like Age of person 200 → Not valid

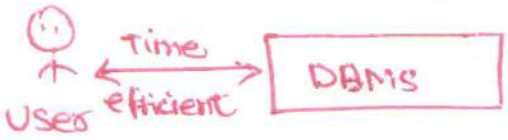
→ CONSISTENT :- Consistent of Data or Consistency of DATA refers that if a person changes his phone no. so it should be changed every where in Database where his Telephone data is saved.



FLEXIBILITY : It is ability of upgrading or changing the functionality of database.

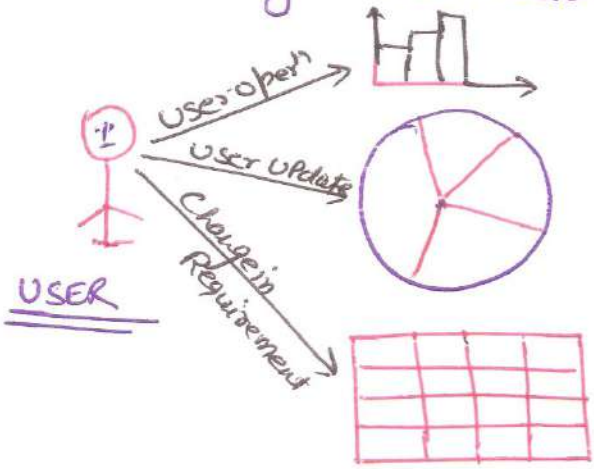


EFFICIENCY:- It ensures that should not wait for long time when accessing data.



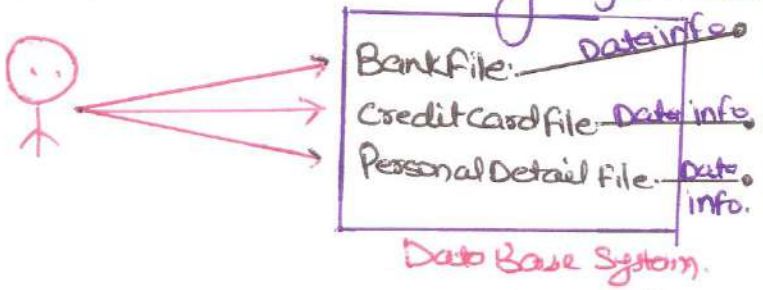
Database should be able to perform effectively

USABILITY:- It ensures that data can be accessed and manipulated in ways which matches the user requirements.

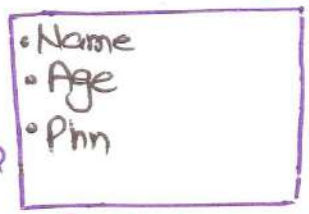


Using Data according to requirements like showing it in Tabular form, Chart form,
Operations can be performed like update, delete, insert

REDUNDANCY LESS:- Redundancy refers duplicate Data



Here Bank file
Credit Card
Personal detail



Redundant Data
(Same info)

So good DataBases are Redundantless:-

PROPERTIES OF DATABASE

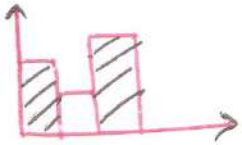
* Question Comes in mind Why we use DATABASE SYSTEMS :-

Ans :- There are 6 important features of DataBases

- → Completeness
- → Integrity
- → Flexibility
- → Efficiency
- → Usability → Redundancy Less

COMPLETENESS : This property ensures that user can access the data they want including ad hoc queries :-

Ad hoc Queries :- They are those Queries which Cannot be determine prior to the Moment the Query is issued.



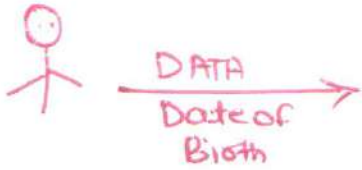
Eg :- 1) Display Bar chart, pie chart for a Vendor and his stocks.

2) Showing plot chart & stack bar chart for an employee

***Integrity :-** Database integrity ensures that Data entered into the DataBase is Accurate, Valid, and Consistent

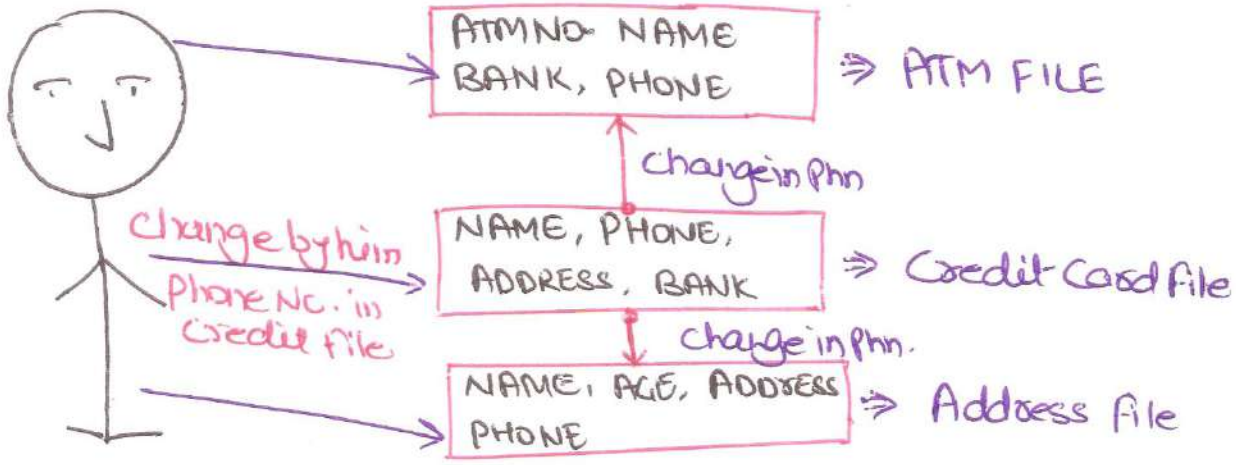
→ **Accurate :-** Data Accuracy refers that data values must be right value and must be represented in a consistent manner.

Eg

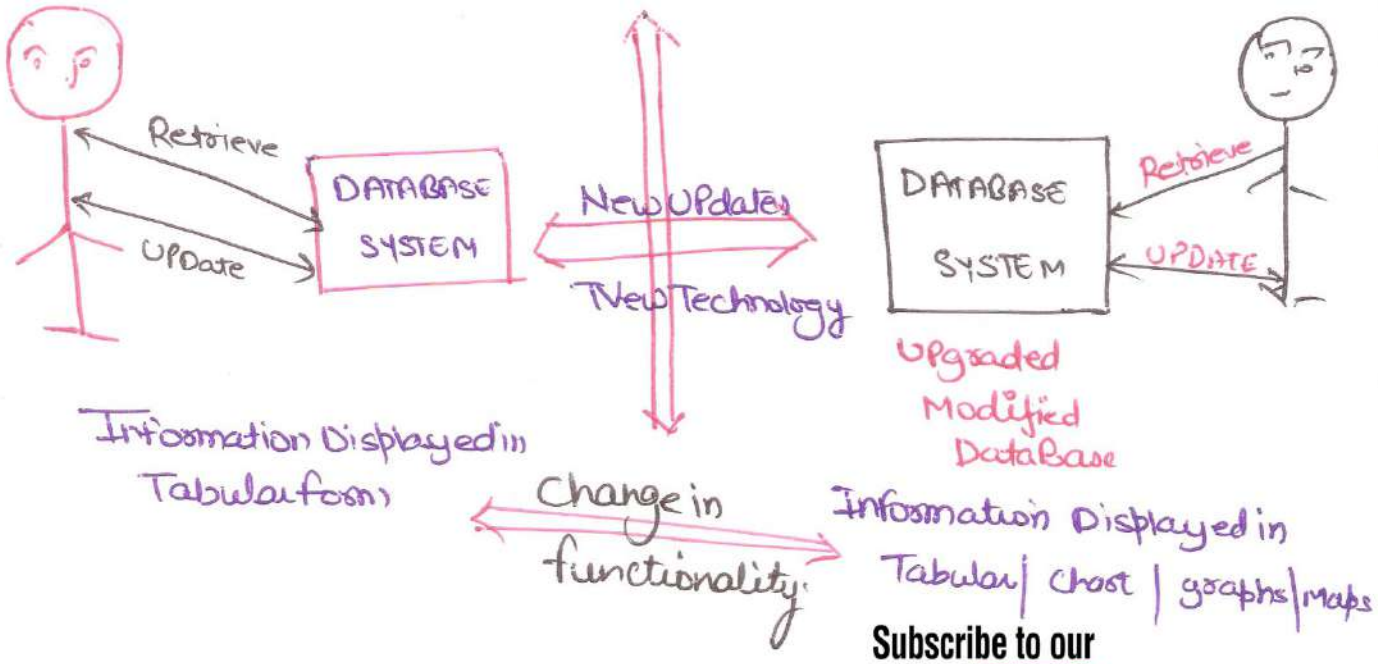


VALID :- It refers Data should be correct like Age of person 200 → Not valid

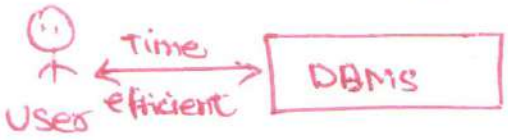
→ CONSISTENT :- Consistent of Data or Consistency of DATA refers that if a person changes his phone no. so it should be changed every where in Database where his Telephone data is saved.



FLEXIBILITY : It is ability of upgrading or changing the functionality of database.

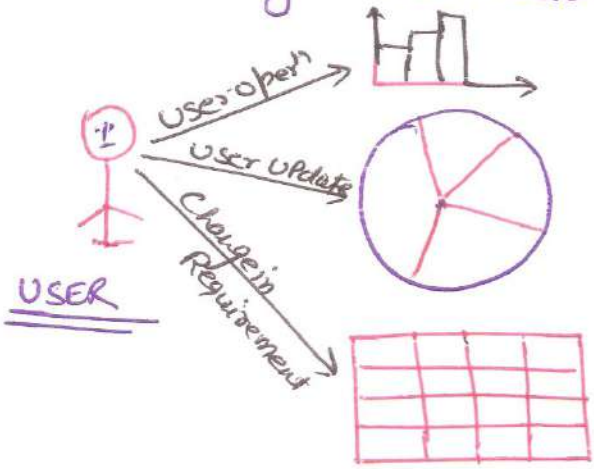


EFFICIENCY:- It ensures that should not wait for long time when accessing data.



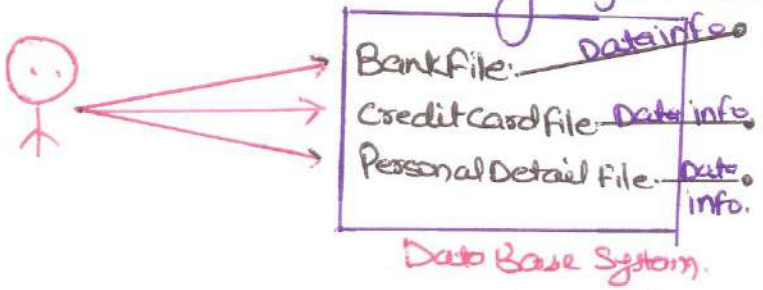
Database should be able to perform effectively

USABILITY:- It ensures that data can be accessed and manipulated in ways which matches the user requirements.

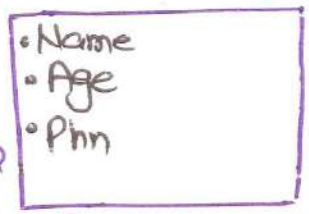


Using Data according to requirements like showing it in Tabular form, Chart form,
Operations can be performed like update, delete, insert

REDUNDANCY LESS:- Redundancy refers duplicate Data



Here Bank file
Credit Card
Personal detail



Redundant Data
(Same info)

So good DataBases are Redundantless:-

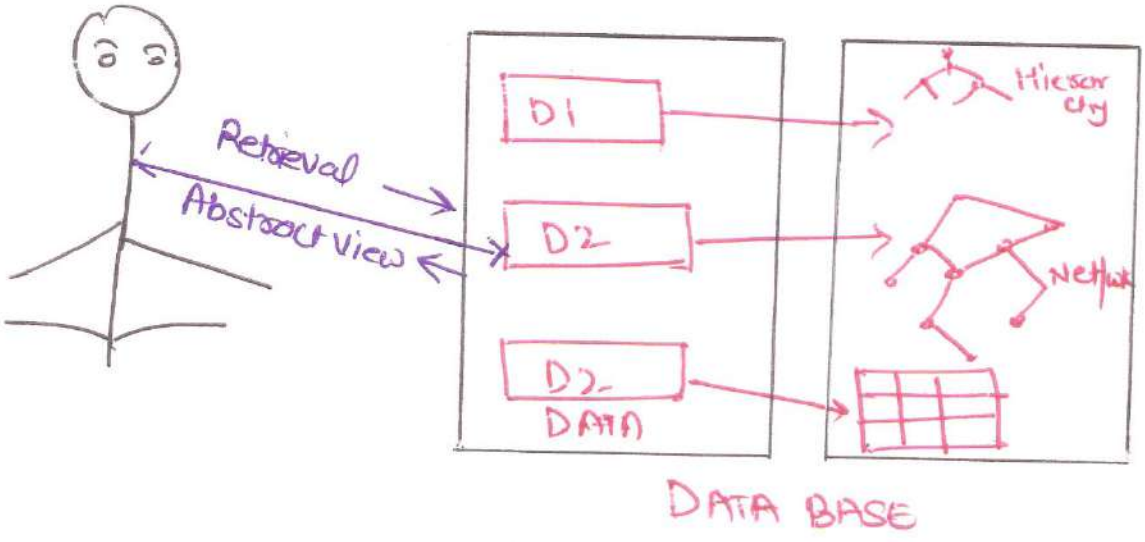
VIEW OF DATA

It regress that how Data is actually stored in database, what data and structure of data used by database for Data. So describe all these database provides user with Views and these are

→ Data Abstraction

→ Instances and schemas

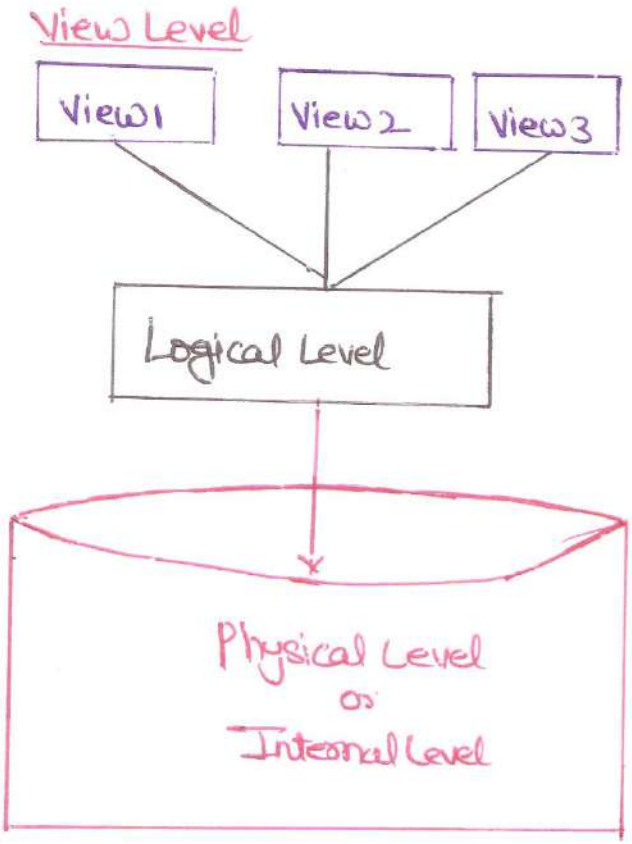
Data Abstraction:- As Data in Database are stored with very Complex Data Structure So when user comes and want to access any Data, he won't be able to access data if he has go through this Datastructure. So to simplify the interaction of user and database, DBMS hides some informations which is not of user interest, this is called Data Abstraction:- So Developer hides Complexity from users and show Abstract View of DATA:-



Data Abstraction has three level of abstraction

- → Physical Level | Internal Level
- → Logical Level / Conceptual Level
- → View Level / External Level

Physical Level --> This is the lowest level of data abstraction which describe how data is actually stored in the database. This level basically describe the Data Structure and Access path | Indexing use for accessing file



Account Holder

View1
AC_Name
Amount

Manager

View2
AC_NO
AC_Name
Type

Conceptual	Logical view
AC_NO	numeric(15)
AC-Name	character(20)
Amount	numeric (15)

Internal view	Physical View
Stored-acc	Length=60
Account #	type = bytes(15)
Type	Offset(10)

Logical Level : The next Level of abstraction describes what data are stored in the database and what relationship exist among those data.

View Level :- In this level user only interact with database The complexity remain unview. Users see data Data and there may be many views of one Data like, Chart, Graph.

→ Let suppose we have Customer information so at Physical Level these Records [Customer information] can be describe as block of storage.

→ At the Logical Level these records can be described as Fields and Attributes along with their data-type and relation Ship among each other.

→ At View Level user just interact with System with the help of GUI and enters the detail at the screen.

** L→ User Not aware of what and how data is stored

• INSTANCE and Schema in DBMS :-

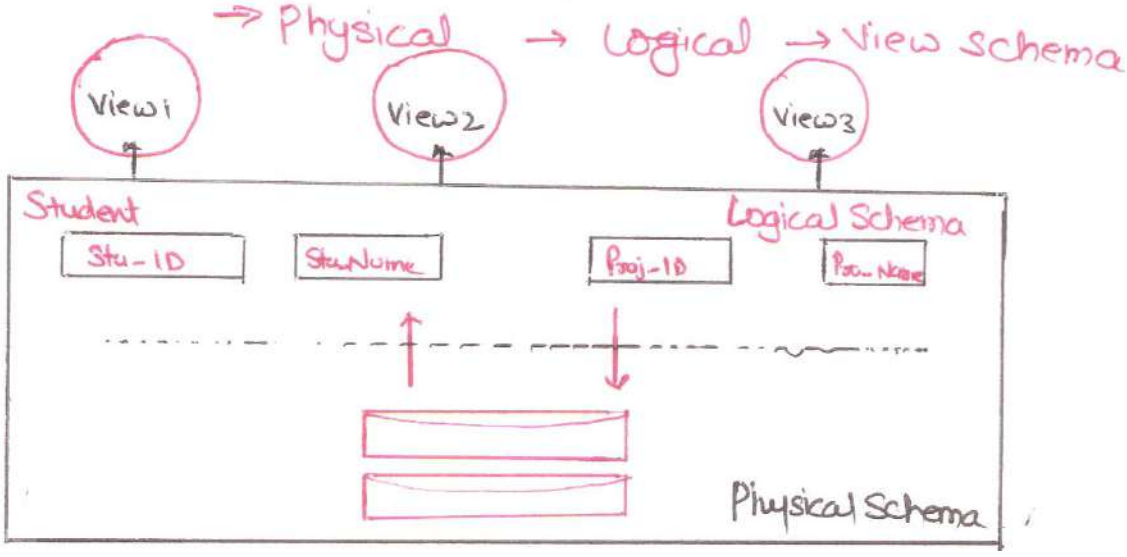
What is Schema ☞ Design of database is called the Schema.

It is a basically skeleton structure that represents the logical view of the entire database.

It defines how data is organized and how the relationship among them are associated. It formulates all the Constraints that are to be applied on the data.

Database System has various schema

→ Physical → Logical → View Schema



→ **Physical Database Schema →**

→ It describes how data will be stored in hard disk | s/c storage

→ It describe the database design at physical level.

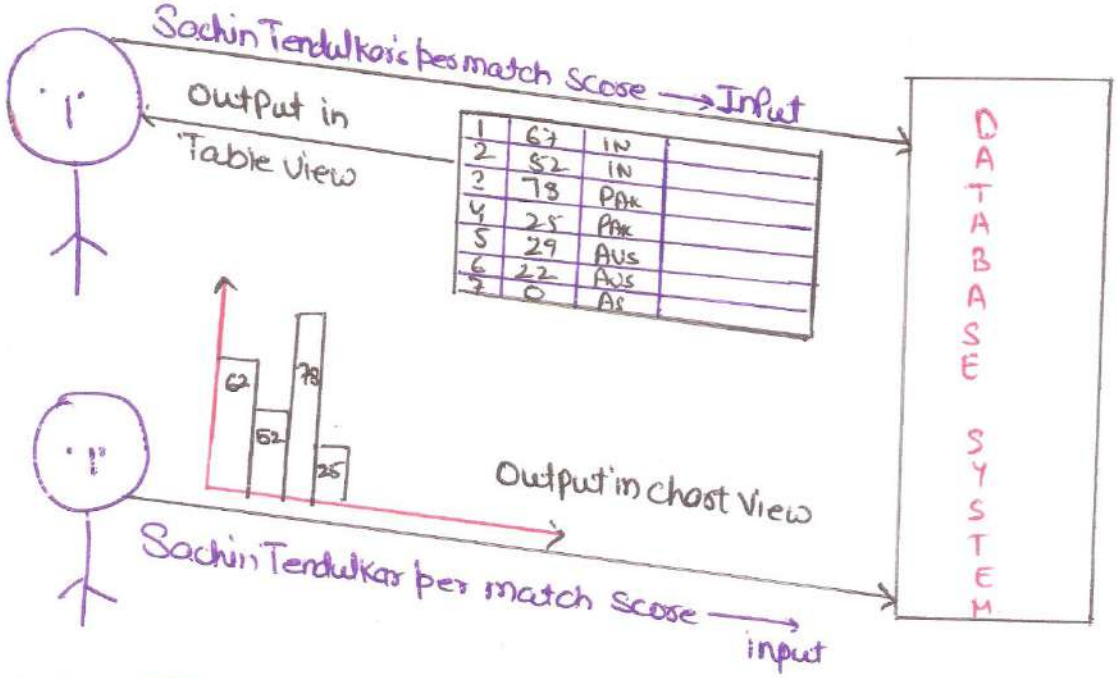
→ This Schema related to the actual storage of data and its form of storage like files, indices.

→ **Logical Database Schema →** This schema defines all the logical constraints that need to be applied on the data stored.

→ It defines Tables, Views, and integrity Constraint.

→ Defines Relation b/w Tables and Keys applied.

→ **View Schema:-** It describe different views of database and some time also called Sub Schema



INSTANCE :- The collection of information stored in the database at a particular moment is called an Instance

Table Student	St_Name	St-ID	St_Gender
Instance →	Deepak	819	M

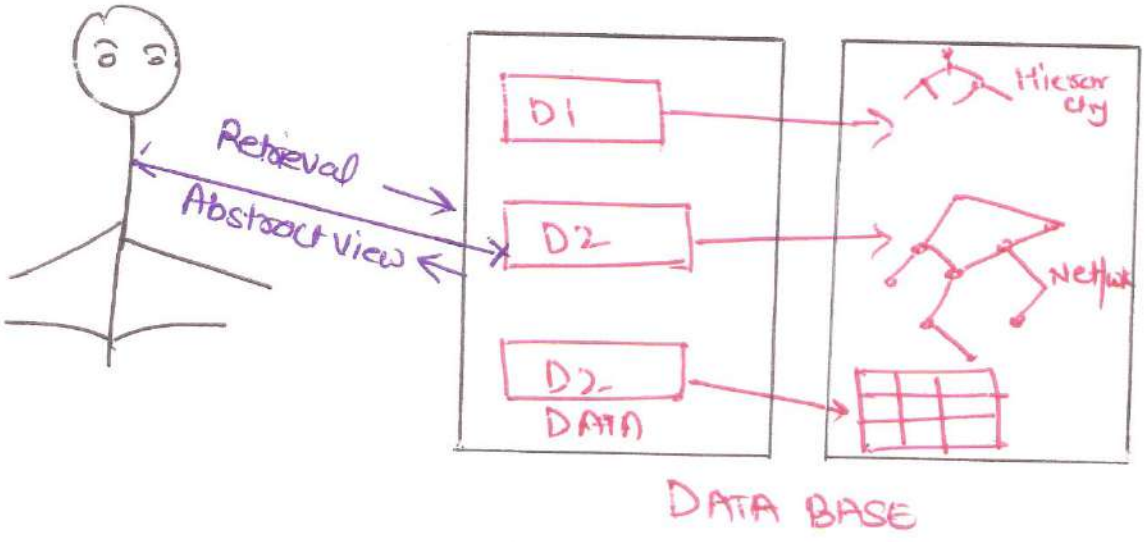
VIEW OF DATA

It regress that how Data is actually stored in database, what data and structure of data used by database for Data. So describe all these database provides user with Views and these are

→ Data Abstraction

→ Instances and schemas

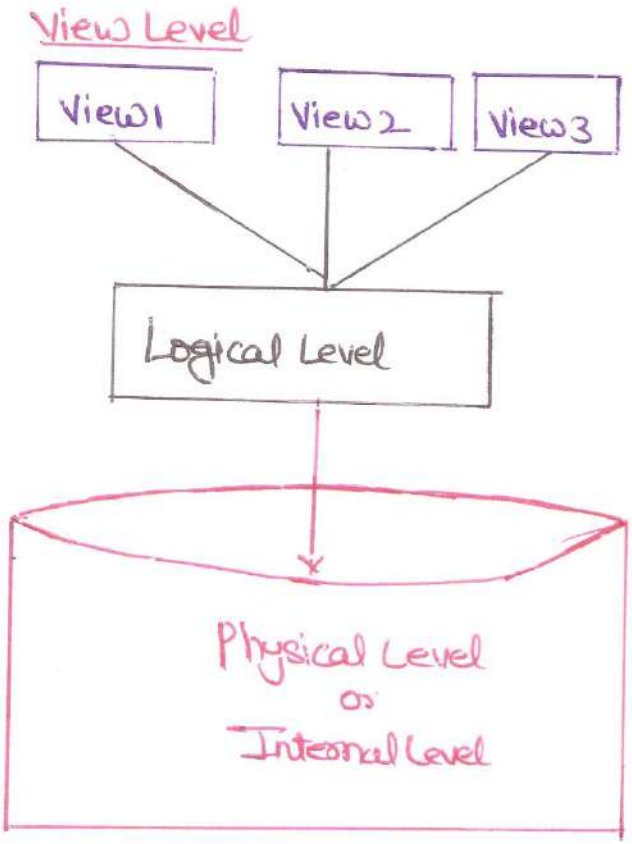
Data Abstraction:- As Data in Database are stored with very Complex Data Structure So when user comes and want to access any Data, he won't be able to access data if he has go through this Datastructure. So to simplify the interaction of user and database, DBMS hides some informations which is not of user interest, this is called Data Abstraction:- So Developer hides Complexity from users and show Abstract View of DATA:-



Data Abstraction has three level of abstraction

- → Physical Level | Internal Level
- → Logical Level / Conceptual Level
- → View Level / External Level

Physical Level --> This is the lowest level of data abstraction which describe how data is actually stored in the database. This level basically describe the Data Structure and Access path | Indexing use for accessing file



Account Holder

View1
AC_Name
Amount

Manager

View2
AC_NO
AC_Name
Type

Conceptual	Logical view
AC_NO	numeric(15)
AC-Name	character(20)
Amount	numeric (15)

Internal view	Physical View
Stored-acc	Length=60
Account #	type = bytes(15)
Type	Offset(10)

Logical Level : The next Level of abstraction describes what data are stored in the database and what relationship exist among those data.

View Level :- In this level user only interact with database The complexity remain unview. Users see data Data and there may be many views of one Data like, Chart, Graph.

→ Let suppose we have Customer information so at Physical Level these Records [Customer information] can be describe as block of storage.

→ At the Logical Level these records can be described as Fields and Attributes along with their data-type and relation Ship among each other.

→ At View Level user just interact with System with the help of GUI and enters the detail at the screen.

** L→ User Not aware of what and how data is stored

• INSTANCE and Schema in DBMS :-

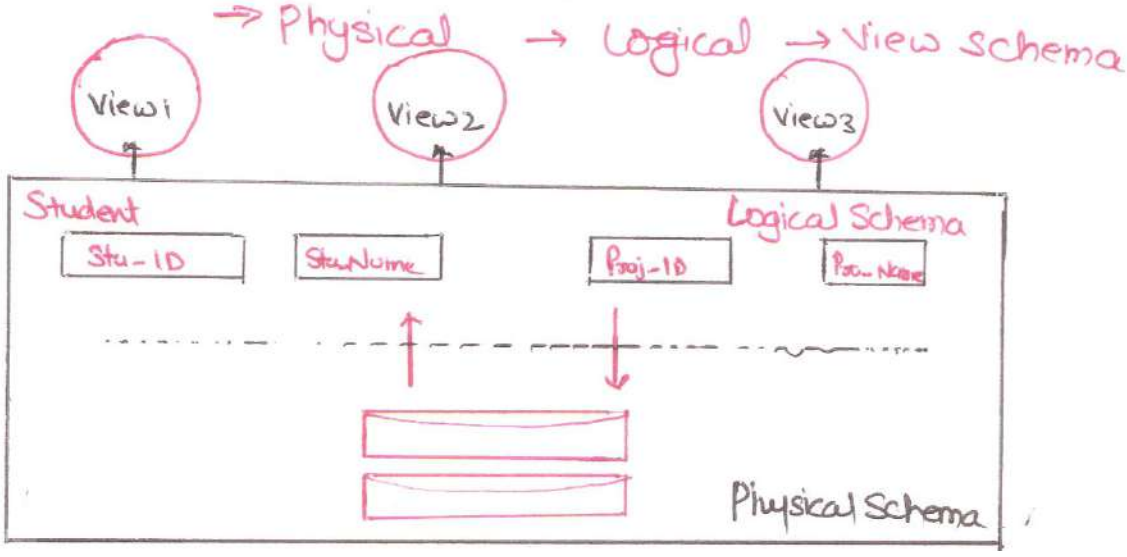
What is Schema ☞ Design of database is called the Schema.

It is a basically skeleton structure that represents the logical view of the entire database.

It defines how data is organized and how the relationship among them are associated. It formulates all the Constraints that are to be applied on the data.

Database System has various schema

→ Physical → Logical → View Schema



→ **Physical Database Schema →**

→ It describes how data will be stored in hard disk | s/c storage

→ It describe the database design at physical level.

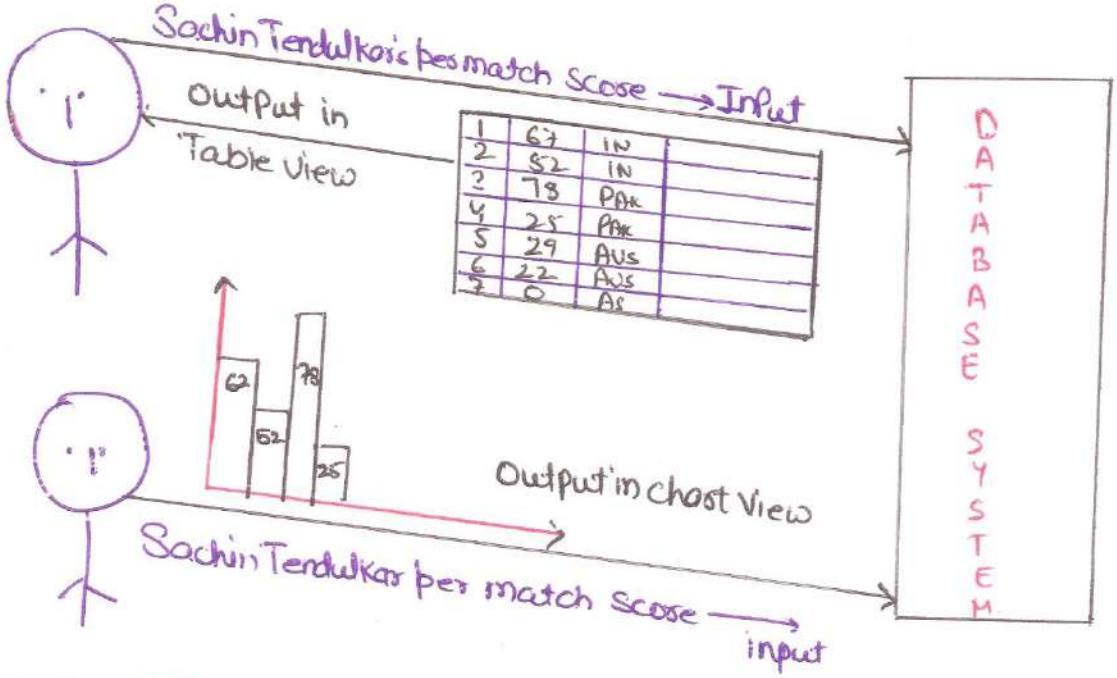
→ This Schema related to the actual storage of data and its form of storage like files, indices.

→ **Logical Database Schema →** This schema defines all the logical constraints that need to be applied on the data stored.

→ It defines Tables, Views, and integrity Constraint.

→ Defines Relation b/w Tables and Keys applied.

→ **View Schema:-** It describe different views of database and some time also called Sub Schema



INSTANCE :- The collection of information stored in the database at a particular moment is called an Instance

Table Student	St_Name	St-ID	St_Gender
Instance →	Deepak	819	M

Overview of Data Models :-

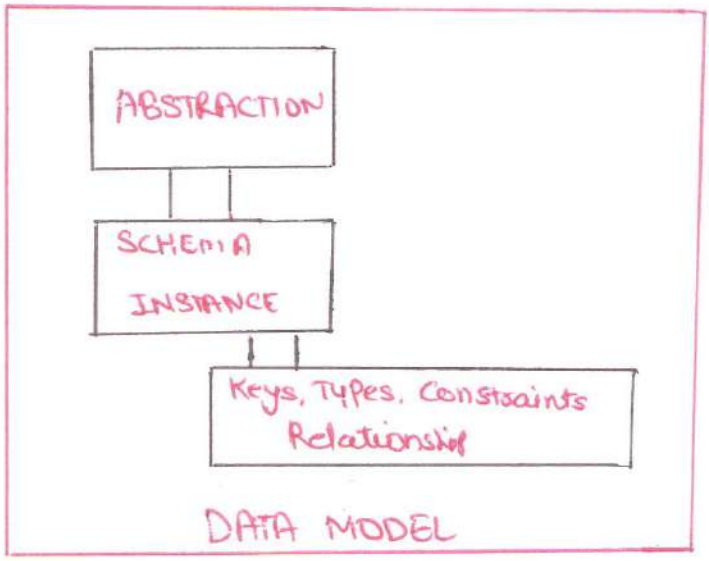
What are Data Models :-?

Definition.1 :- Data Models defines how the logical structure of database is modeled.

Database are the fundamental entity to introduce Abstraction in a DBMS

: Data Models defines the datatypes, the constraint and the relationships for the description or storage of data respectively

Definition :- It is a collection of 'Concepts and Rules' for 'implementing" Abstraction, Instance & Schemas."



Various Data Models are

→ E-R Model :- Entity-Relationship Model

→ Relational Model

→ Object-oriented Data Model.

ENTITY RELATIONSHIP MODEL

The entity-relationship (E-R) data model is based on concept of Real world that consists of a collection of basic objects called 'Entities', and relationship among these Objects.

Entity An entity in an ER-Model is a real-world object having properties called Attributes.

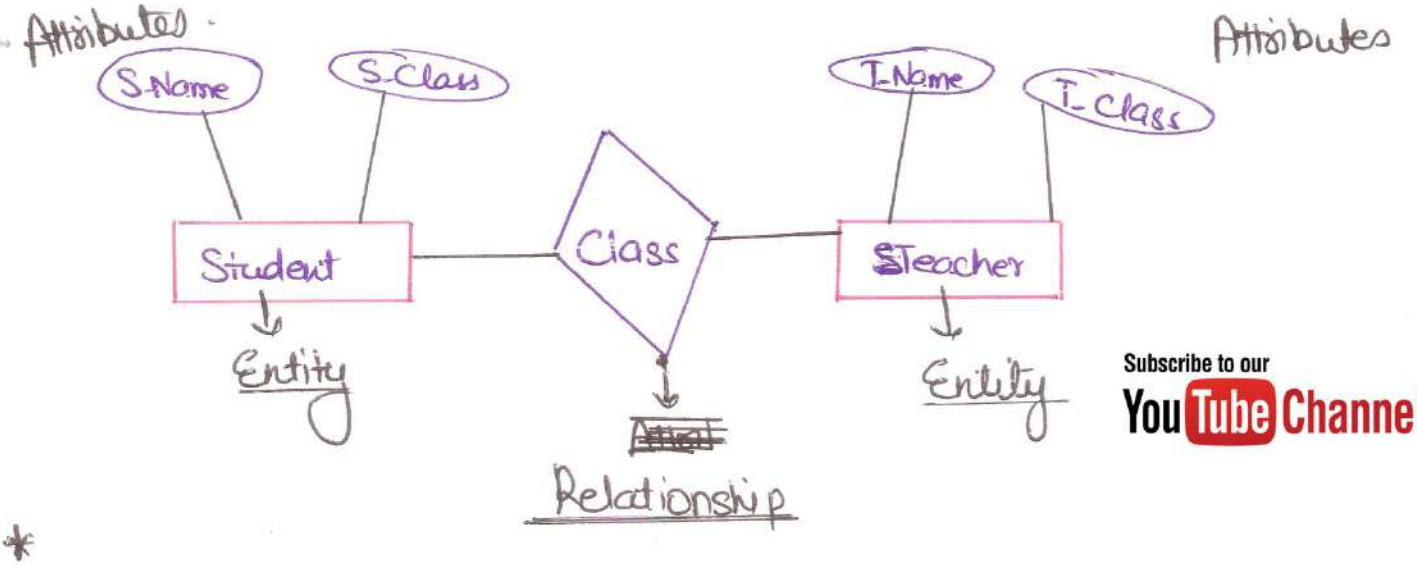
Attributes:- Every Attribute is defined by its set of values called Domain.

Eg In a school database, a student is considered as an Entity.

**→ Student has various Attributes like S.Name, S.Age, S.Class etc.

Relationship A logical association among several entities

Eg:- In the above Eg we add 'Teacher' Entity with Attribute T.Name, T.class. So Student and Teacher Entity Can be Relationshiped with Relation class.

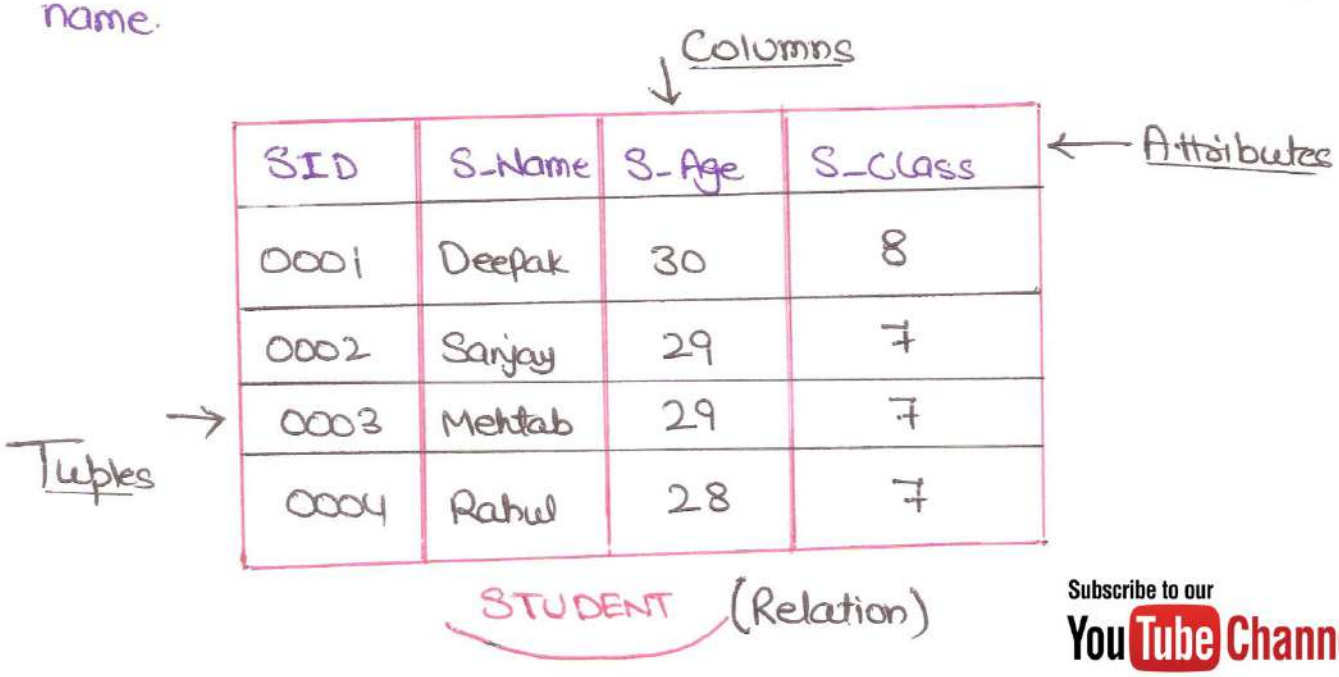


* One Entity Can have 2, 3 or more Relationships with Other Entity

RELATIONAL DATA MODEL

Relation Data Model uses a collection of 'Tables' to represent both Data and the Relationships among those data.

→ Each Table has multiple Columns and each column has unique name.



→ The Data is arranged in a Relation which is visually represented in a Two Dimensional Table.

→ The Data is inserted into the Table in the form of Tuples (Rows). A Tuple is formed by one or more than one Attributes. Tuple in this eg is a row (complete row)

0003 Mehtab 29 7

& Attributes are SID, S_Name, S_Age, S_Class.

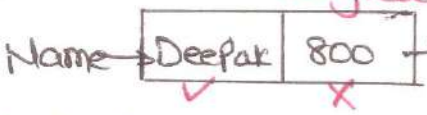
→ Attributes are used as basic building blocks in the formation of various expression that are used to derive a meaningful information

→ There can be number of Tuples in the table (Relation), but all the tuple contains fixed and same Attributes with varying values.

→ A Relation is represented by a Table,
A Tuple is represented by a Row
An Attribute is represented by a Column of the Table.
Attribute name is the Name of the Column
Eg S_ID, S_Name, S_Age.
Attribute values contains the value for the column in the Row.

→ Constraint (Set of Rules and limitations)

*** Constraints are applied to the table and form the Logical Schema



As we know the age of a person cannot be 800 years so we will apply some constraint

→ To select a particular Row/Tuple from Table/Relation we use Attributes/Columns name with the help of unique value or field of an Attributes

→ This field which are unique from other fields are used as indexes which helps in searching fast.

→ EID → Attribute with every unique fields.

E_ID	E_Name	E_AGE	E_Gen	E_Sal
0006	Deepak	29	M	18000
0010	Ajay	28	M	15000
0020	Deepak	31	M	18000
0310	Kapil	42	M	17000
0289	Rahul	52	M	14,000
0628	Deepak	29	M	16000

→ All the Relational Algebra Operations,
Like Select, Intersection, Product
Union : JOIN, DIVISION, MERGE
Can also be performed on the Relation Data Model.

→ Operations on RDM (Relational Data Model) are facilitated with the help of different Conditional Expressions, Various Key Attributes & pre-defined Constraints etc.

→ Data Integrity is maintained by process like Normalization

→ Description of Data interms of this Model is called a Schema

→ Schema for Relation specifies, its name, name of each field.

Student (S_id : Integer,
S_Name : String,
S_login : String etc)

Object - Oriented - Data - Model

Object Oriented Data Model (OODM)

In this both Data and their Relationship are Organized or contained in a Single Structure known as an Object.

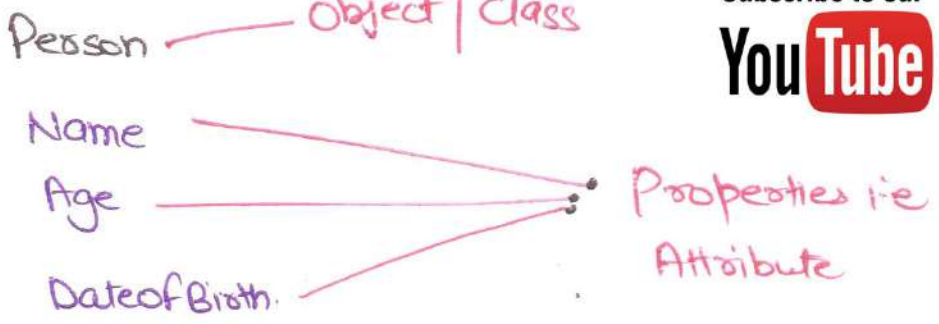
Object includes information about Relationship between the facts within the object, as well as information about its Relationship with other objects.

It is also said to be Semantic Data Model.

Components of OODM :-

→ An Object is the abstraction of the Real World Entity. and an Object represents only one occurance of entity.

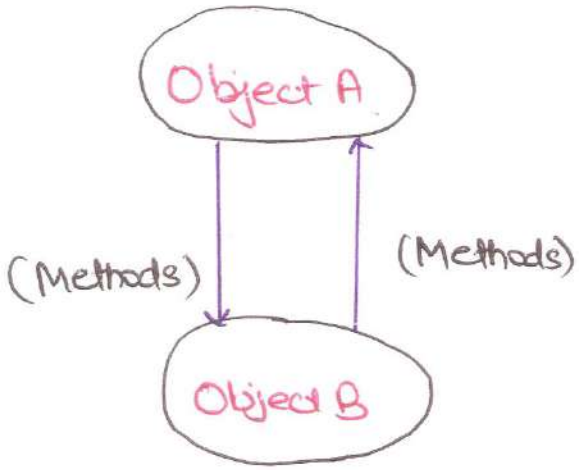
→ Attributes :- It describes the property of an object.



→ Class : Objects that are Similar in Characteristics are grouped in class.

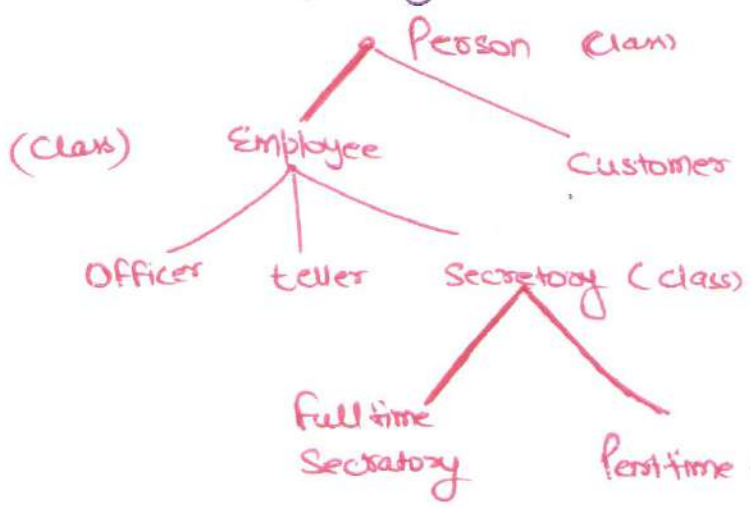
So Class is a Collection of Similar Objects with Shared Structure (Attributes) and Behavior (Method)

→ Method:- Methods Represents a real World action such as finding a selected persons Name, Changing persons Name or printing a persons Address.



Classes are Organized in a class hierarchy & it resembles an upside-Down tree in which each class has only one Parent (Head).

Inheritance is an object's ability to inherit the Attributes and Methods (Messages) of the class above it.



UML & XML are mainly used to Represent the Structure of class, object, attributes & Methods.

In this full time Secretory & Part time Secretory inherits the Attributes and Methods of class Person.

Hierarchical DataBase Model

→ This Model is based on Parent | child Relationship just like a tree Structure.

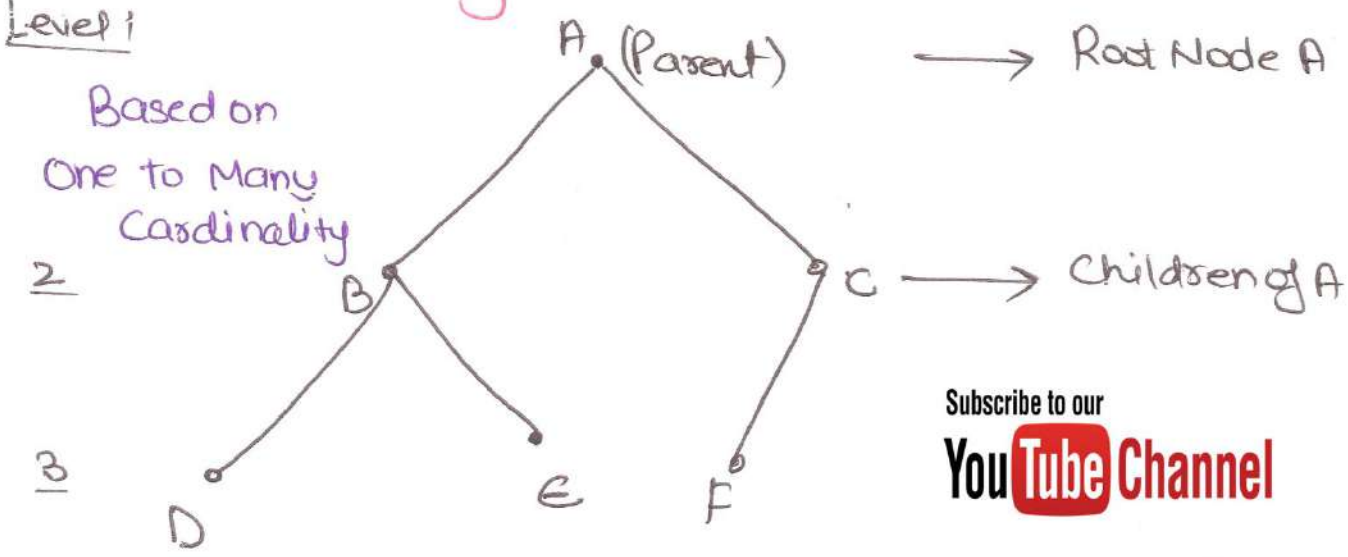
→ Data is stored as Records which are Connected to each Other through links.

→ A record in this model Corresponds to a Row | tuple in Relational DataBase Model.

→ And an entity i.e Parent | child Correspond to a Table (Relation)

*** Each child Can have only One parent, where as each parent record can have more child or no child at bottom level.

→ To Retrieve any record the Whole tree needs to be traversed Starting from Root Node.



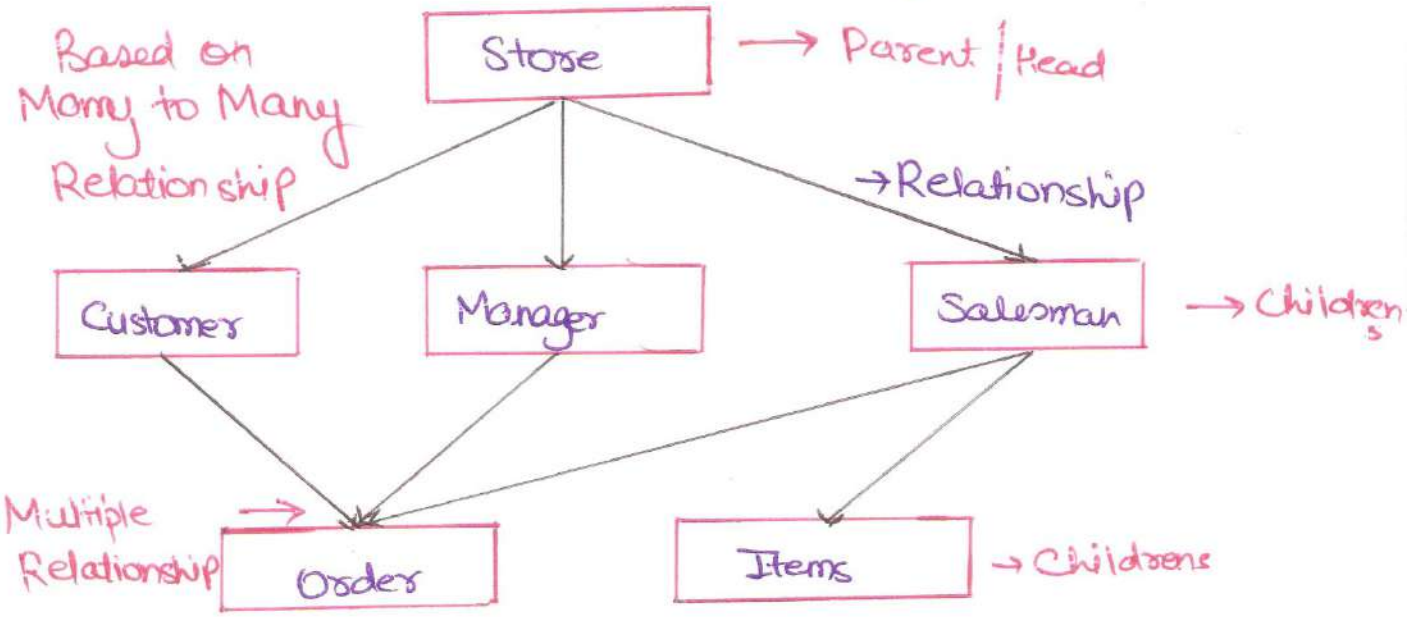
* Records Can be accessed and Manipulated rapidly as Relationship between records are defined in Advance.

Network Data Base Model.

This Network Database Model is designed with the Concept of Parents and childrens with their features and Relationship.

The linkage or Relationship b/w Parent & Child are describe with the help of Tree. An Upside-Down Stree Structure is used.

Branches are used to show relationship

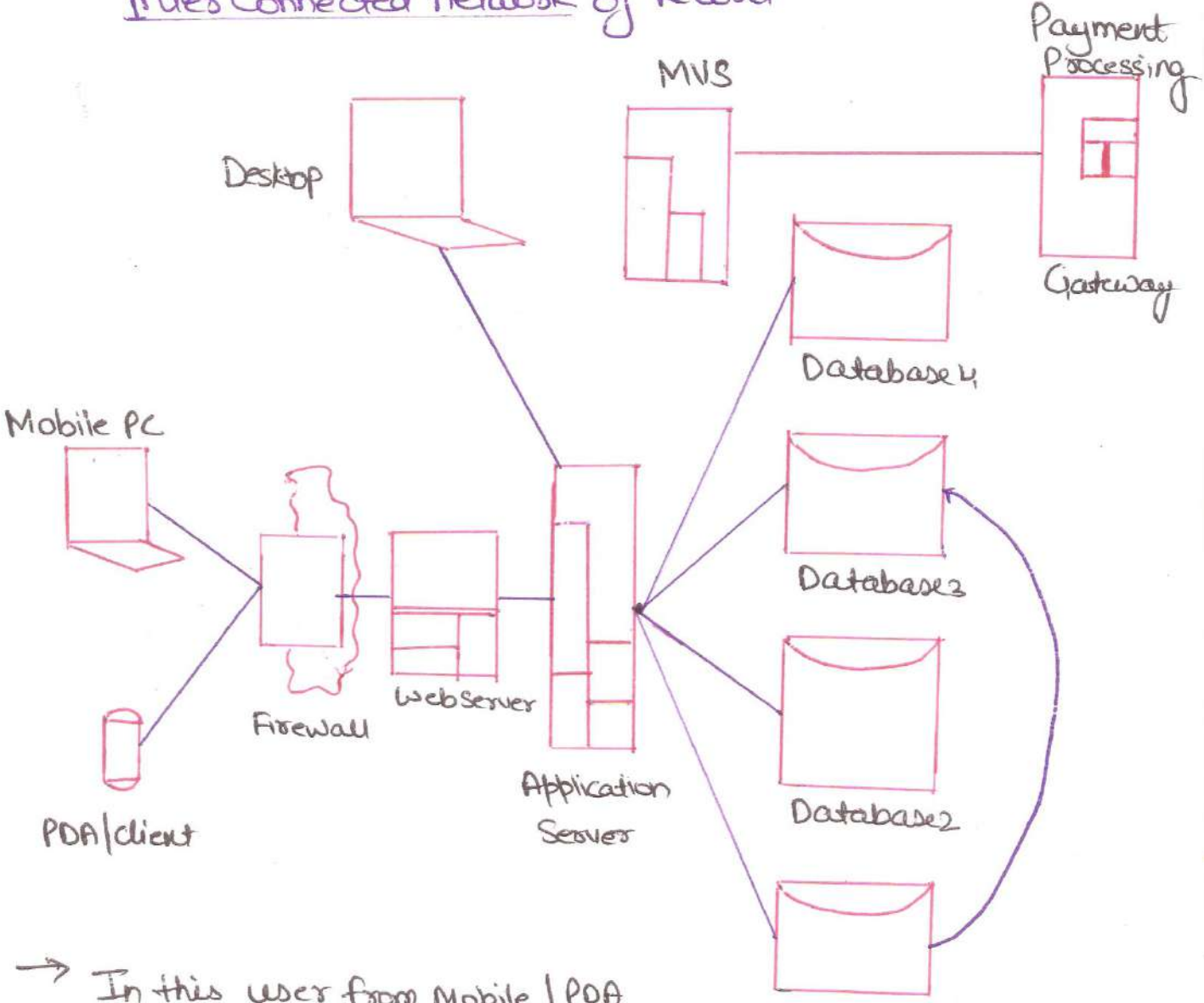


→ In Network we can have Multiple Relationship with other. This feature makes the Network Data model More flexible.

→ N:N Many to Many Relation is followed in this

→ This Network Models are mainly used on a large Digital Computers.

→ Network Data Model look like Cobweb or 'Inter Connected network' of Record



→ In this user from Mobile / PDA devices insert a request to payment of electricity bill.

→ This Request goes to firewall which for any kind viruses or false Request.

→ Then this request goes to webser to check the validation or todo check regarding Requestion.

→ Then this Request goes to Application Server where it Converts the Request according to the device where it has to forwarded.

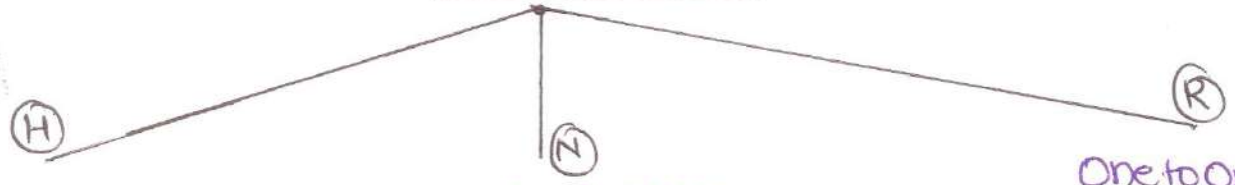
→ And at the upper Level this help, ie Request is processed, with Desktop, MVs and Database.

→ After the process of Request it finally goes to the Payment processing Gateway where it Request is accepted and finally processed.

→ After final processing Gateway send a message ie output regarding Payment which goes to user.

Comparison Among Hierarchical (H) Model, Network (N) Model And Relational (R) Model :-

Data Structure



a) One to One or One to Many Relationship

Many to Many Relationship

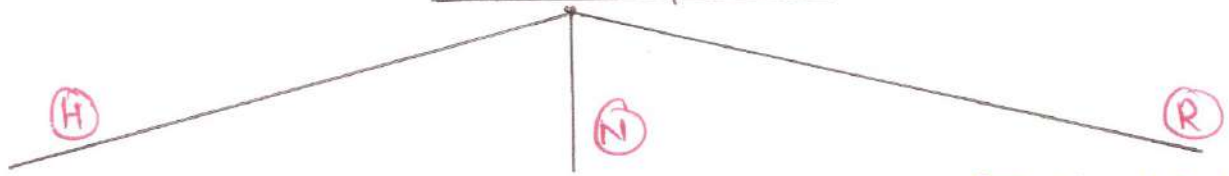
One to One, One to Many, Many to Many Relationship

b) Based on parent Child Relationship

A Record can many have parent as well as many children

Based on Relation Data Structure.

Data Manipulation



a) Not any independent Stand alone query Interface

CODASYL (Conference on Data System Language)

Relational Data Model are what brings many sources into common query (Such as SQL)

b) Retrieve Algorithms are Complex and Asymmetric

Retrieve Algorithms are Complex and Symmetric

Retrieve Algorithm are Simple & Symmetric.

Data Integrity

a) Cannot insert any information of a child who does not have any parent.	Does not Suffer any insertion Anomaly	Does not Suffer any insert Anomaly
b) Multiple Occurance of Child records which lead to problems of inconsistency during the update operation	Free from update Anomalies.	Free from Update Anomalies. b/c of Normalization process.
c) Deletion of parent results in deletion of Child records	Free from delete Anomalies	Free from delete Anomalies.

SQL SERVER 2008

DATABASE LANGUAGES

Now it is time to do some practical, before it each & every thing was conceptual & logical.

Now Database provides two Languages for describing Schema and to perform queries & operation

→ Data Definition Language : DDL
→ Data Manipulation Language : DML

[Combining them become SQL Structured Query Language]

DATA DEFINITION LANGUAGE :-

* Now Suppose you have a Software for DBMS Let MS SQL Server 2008 & you have installed that Software.

* Now it is time to do practical.

Now you have to Create Database then Create a table and define their Schema

So we use DDL to do this all.

There are set of Queries and settings through which we do this.

→ To create a Database we use this Query or Statement

→ CREATE DATABASE AIRTELLTD

This Query will Create the Database Named AIRTELLTD.

→ Now its time to Create Table in it.

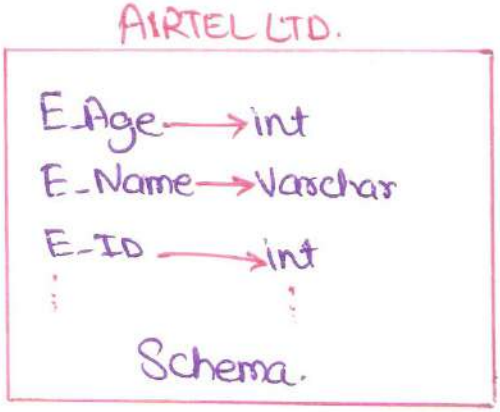
```
CREATE TABLE Employee ( EID int , E-Name varchar , E-Age int);
```

This Query will create table Named 'Employee' with attributes E-Name, E-AGE- EID etc with the type of value they can store only.

* As soon as we execute this Statement it updates a Special set of tables Called Data Dictionary or Data Directories.

Data Dictionary Contains Meta Data. i.e Data about the Data.

* Now Here the Schema of the Database with table has created with those information which we have provided.



Data Dictionary

→ Storage Structure and access Methods [Physical Schema] used by the Database System by a set of statements in a special type of DDL, called a Data Storage and Definition Language. It defines the Implementation Details of Database Schema, which are usually hidden from the user.

→ Now its time to enter or insert some data to the table's Attributes. But before entering data values we have to apply some Constrainst | Rules and limitations On this values.

Eg:- Value in a bank Account takes integer value but Money in an Account Never be -ve Value

Account Name	Amount	
Deepak	-2000.00	X
Sanjana	25000.00	✓

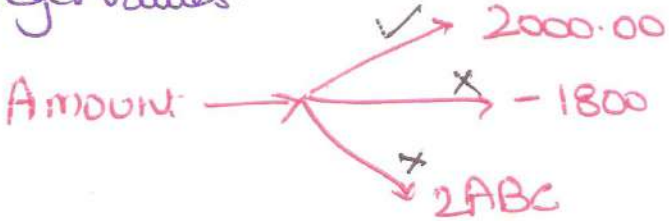
So Here we have to apply constraint to remove or stop this kind of data errors.

Different Types of Constraints are:-

→ Domain Constraints
→ Referential Integrity
→ Assertion
→ Authorization

DOMAIN CONSTRAINT : It Specifies the set of possible values that may be associated with an attribute

Eg Like Amount Attribute it will only take positive and integer values.



Referential Integrity :- Value that appears in one relation for a given set of attributes also appears in a other set of attributes in another Relation.

Eg There are two tables called Emp Detail and Emp Salary

Table Emp Detail has Attributes like

Emp ID Emp Name

and Emp Salary :- This table has attribute Emp ID and Salary, Month

and we have made Emp ID referential to the table Emp Salary's Attribute Emp ID, then values in Emp ID of table Emp Detail Should match to the Emp ID of Emp Salary

* Emp ID of table Emp Salary can only have Values matches from Emp ID of table Emp Detail

Emp_ID	Emp_Name
002	Deepak
009	Ajay
008	Sanjana
006	Raman
007	Sumita
092	Ashoka
085	Rana

Emp Details

Emp ID	Salary	Month
008	6000	Jan
007	5000	Jan
092	13000	March
008	6000	Feb
092	15000	April
085	16000	Jan
008	6500	March
002	18000	Aug

Emp Salary

→ Referential Integrity

So Emp ID of Emp Salary can only have value from Emp ID of Emp Details

Referential Integrity is implemented with the foreign key

→ **Assertions:-** It is any condition that the database must always satisfy.

for Eg:- A table of student has attributes Marks, Name as the Maximum Marks is 100 so there will be an assertion on Marks is that Value in it Cannot be -ve & more than 100

Subject	Name	Marks	
Hindi	Deepak	200	✗
Maths	Ajay	75	✓
S.S.T	Sanjay	-70	✗
Science	Rahul	80	✓

WRONG VALUES

Value Cannot be >100 & -ve so we have to apply Assertion on it

* Domain Constraint & Referential Integrity are Special form of Assertion.

→ **Authorization :-** Authorization Means Permission

In DBMS ; different Authorizations are applied according to the users.

Different Authorization are

→ Read Authorization :- Only Reading No Modification

→ Insert Authorization :- Only insert of data which should be new.

→ Update Authorization :- Only update/modification No Deletion

→ Delete Authorization :- Only Deletion

SQL SERVER 2008

DATABASE LANGUAGES

Now it is time to do some practical, before it each & every thing was conceptual & logical.

Now Database provides two Languages for describing Schema and to perform queries & operation

→ Data Definition Language : DDL
→ Data Manipulation Language : DML

[Combining them become SQL Structured Query Language]

DATA DEFINITION LANGUAGE :-

* Now Suppose you have a Software for DBMS Let MS SQL Server 2008 & you have installed that Software.

* Now it is time to do practical.

Now you have to Create Database then Create a table and define their Schema

So we use DDL to do this all.

There are set of Queries and settings through which we do this.

→ To create a Database we use this Query or Statement

→ CREATE DATABASE AIRTELLTD

This Query will Create the Database Named AIRTELLTD.

→ Now its time to Create Table in it.

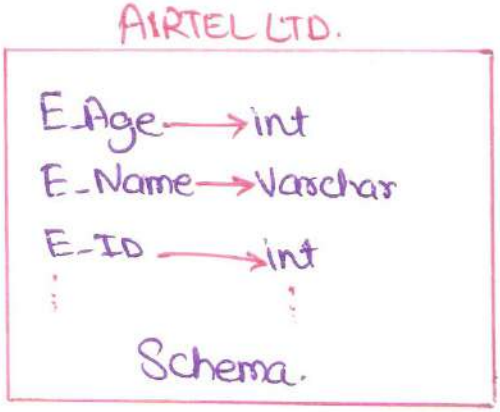
```
CREATE TABLE Employee ( EID int , E-Name varchar , E-Age int);
```

This Query will create table Named 'Employee' with attributes E-Name, E-AGE- EID etc with the type of value they can store only.

* As soon as we execute this Statement it updates a Special set of tables Called Data Dictionary or Data Directories.

Data Dictionary Contains Meta Data. i.e Data about the Data.

* Now Here the Schema of the Database with table has created with those information which we have provided.



Data Dictionary

→ Storage Structure and access Methods [Physical Schema] used by the Database System by a set of statements in a special type of DDL, called a Data Storage and Definition Language. It defines the Implementation Details of Database Schema, which are usually hidden from the user.

→ Now its time to enter or insert some data to the table's Attributes. But before entering data values we have to apply some Constrainst | Rules and limitations On this values.

Eg:- Value in a bank Account takes integer value but Money in an Account Never be -ve Value

Account Name	Amount	
Deepak	-2000.00	X
Sanjana	25000.00	✓

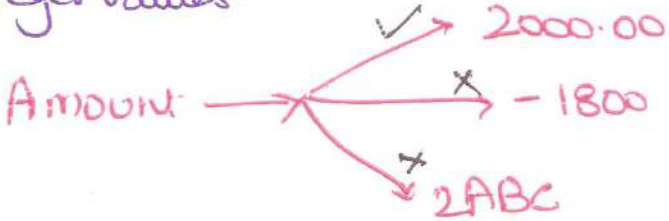
So Here we have to apply constraint to remove or stop this kind of data errors.

Different Types of Constraints are:-

→ Domain Constraints
→ Referential Integrity
→ Assertion
→ Authorization

DOMAIN CONSTRAINT : It Specifies the set of possible values that may be associated with an attribute

Eg Like Amount Attribute it will only take positive and integer values.



Referential Integrity :- Value that appears in one relation for a given set of attributes also appears in a other set of attributes in another Relation.

Eg There are two tables called Emp Detail and Emp Salary

Table Emp Detail has Attributes like

Emp ID Emp Name

and Emp Salary :- This table has attribute Emp ID and Salary, Month

and we have made Emp ID referential to the table Emp Salary's Attribute Emp ID, then values in Emp ID of table Emp Detail Should match to the Emp ID of Emp Salary

* Emp ID of table Emp Salary can only have Values matches from Emp ID of table Emp Detail

Emp_ID	Emp_Name
002	Deepak
009	Ajay
008	Sanjana
006	Raman
007	Sumita
092	Ashoka
085	Rana

Emp Details

Emp ID	Salary	Month
008	6000	Jan
007	5000	Jan
092	13000	March
008	6000	Feb
092	15000	April
085	16000	Jan
008	6500	March
002	18000	Aug

Emp Salary

→ Referential Integrity

So Emp ID of Emp Salary can only have value from Emp ID of Emp Details

Referential Integrity is implemented with the foreign key

→ **Assertions:-** It is any condition that the database must always satisfy.

for Eg:- A table of student has attributes Marks, Name as the Maximum Marks is 100 so there will be an assertion on Marks is that Value in it Cannot be -ve & more than 100

Subject	Name	Marks	
Hindi	Deepak	200	✗
Maths	Ajay	75	✓
S.S.T	Sanjay	-70	✗
Science	Rahul	80	✓

WRONG VALUES

Value Cannot be >100 & -ve so we have to apply Assertion on it

* Domain Constraint & Referential Integrity are Special form of Assertion.

→ **Authorization :-** Authorization Means Permission

In DBMS ; different Authorizations are applied according to the users.

Different Authorization are

→ Read Authorization :- Only Reading No Modification

→ Insert Authorization :- Only insert of data which should be new.

→ Update Authorization :- Only update/modification No Deletion

→ Delete Authorization :- Only Deletion

Data-Manipulation Language

Data Manipulation Language (DML) is one of the Database Language which enables users to Access or Manipulate data of an organized Data Model.

Through DML we can

- Retrieve
- Insert
- Delete
- Modification

] Information stored in DataBase

DML is of 2 types

- Procedural DML
- Declarative DML

Procedural DML :- It requires a user to specify 'what' Data are needed and how to get those Data.

Declarative DML :- OR Non Procedural DML requires a user to specify what Data are needed without specifying how to get those Data.

DML Component of SQL Language is Non-Procedural.

A Query is a statement requesting to perform any of the DML operation like Retrieval, Insertion etc. and portion of a DML that involves information Retrieval is called Query Language.

Eg:-

```
Select  Customer . Customer-name
From    Customer
Where   Customer . Customer-id = 192-83-7465
```

This Query in the SQL Language finds the name of the Customer whose Customer-id is 192-83-7465

** This Query Language depends upon the which Data Base Management System i.e Software we are using.

Every DBMS has only little change in their SQL Language

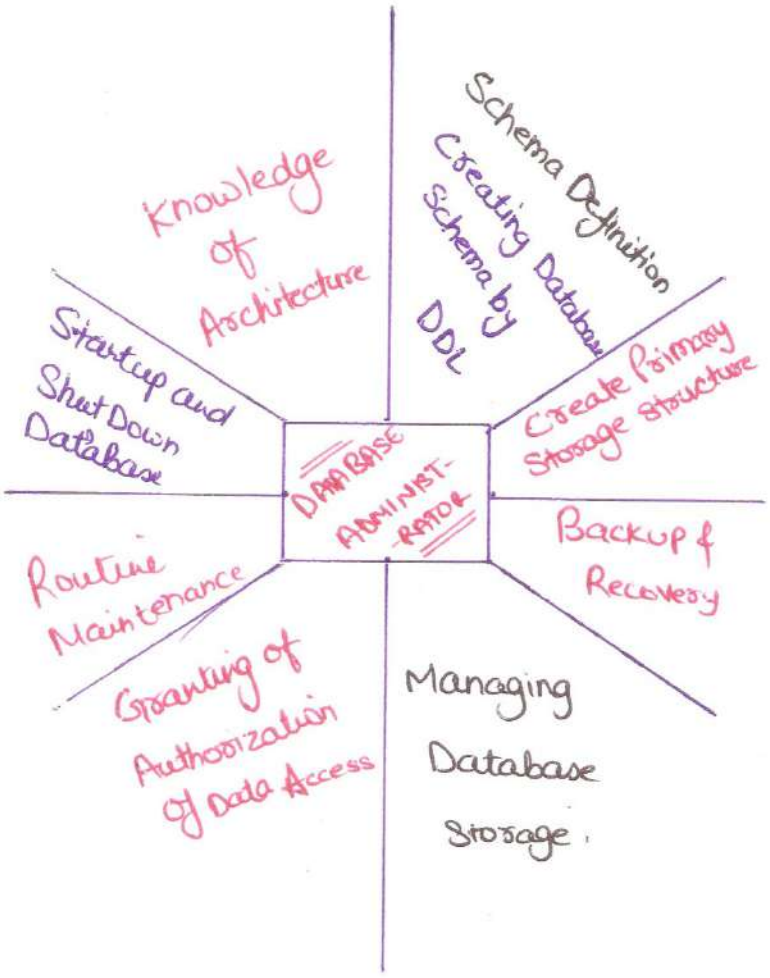
DATABASE ADMINISTRATOR

TutorialsSpace.com
A SIMPLE LEARNING

RESPONSIBILITY

Database Administrator is a person who has Central Control of both the Data and the programs that access those Data.

Responsibilities :-



Three-Tier Architecture

Three-Tier or Three-Level Architectures defines that how user access the at different level so we can say How data comes to the user in different level in DBMS.

In this DBMS we follow the Three Tier Architecture.

In this we go through three Tiers ie at Three Levels

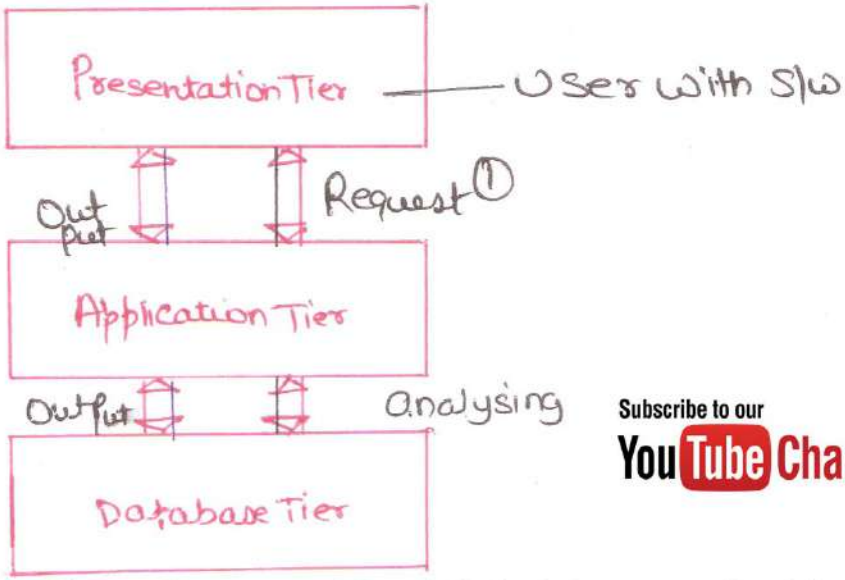
→ Presentation Tier

→ Application Tier | Bussiness Layer

→ Database Tier

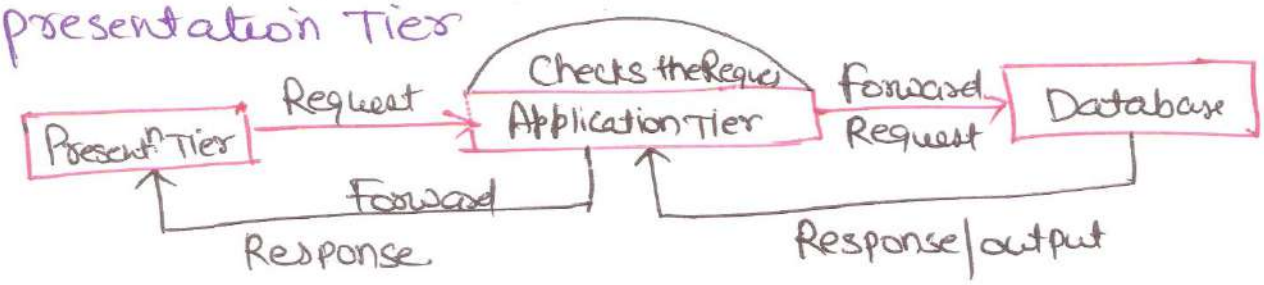
Presentation Tier :- An user operates/work on this tier with knowing the existence of the database beyond this layer

→ In this layer user sends a request to see or to do some task by using an application or software.



Application Tier :- This Tier is most important Tier.

This is kind of middle man which take Request from presentation Tier and check this request if request is ok then forward to Database Tier. Then this tier takes the output from Database Tier and forward to presentation Tier



Database (Data) Tier :- In this Tier we have a database System which process the Request of Application Tier take a Response and forward Response/output to the Application Tier.

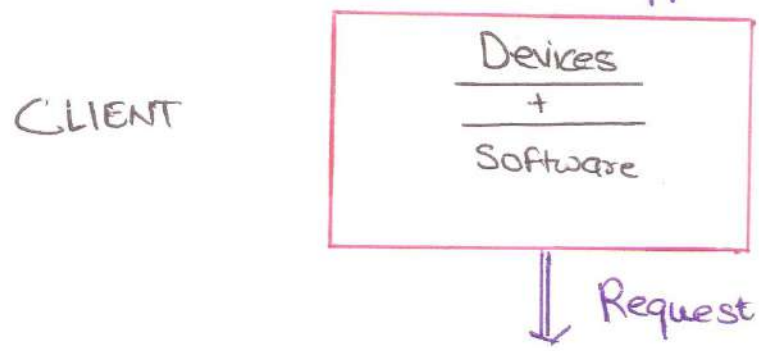
Introduction To Client/Server Architecture [In 3-Tier Architecture]

This is Same as of 3-Tier Architecture:-

In this data or Response to the user goes to 3 levels and these three levels are

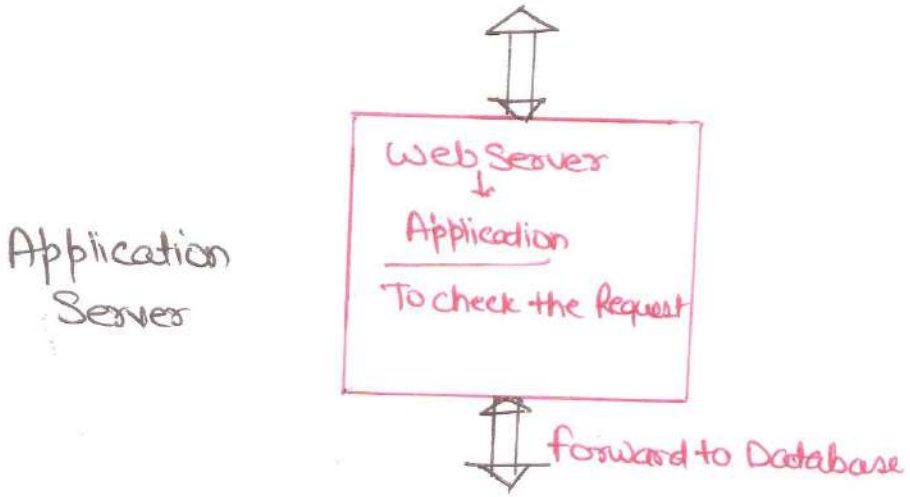
→ Client
→ Application Server or Web Server
→ Database Server.

1) **Client** :- A client here stands an end user here uses an application/Device it may be Computer - Mobile etc with Softwares or application.



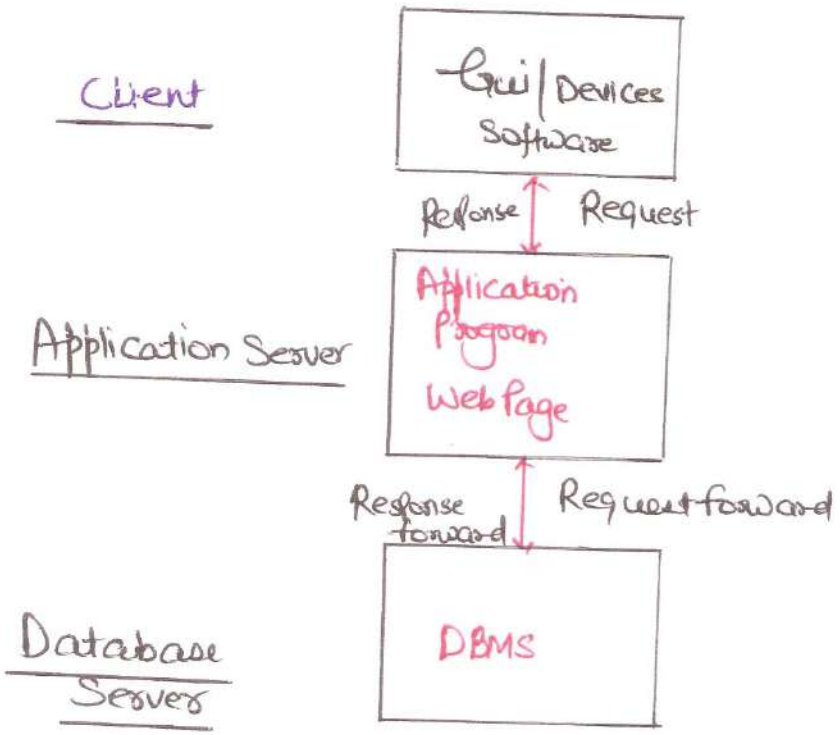
2) **Server Application or Application Server Layer** : In this Layer Data ie Request from the client is Validated/check and then transfer to the Database Layer

→ This Layer Transfer data b/w Client to Database server
→ This Layer uses web Server/application to Check Request from Client
→ It some where also convert the view of data



Database Server :- This layer has all the data or we can say It is our main device or server which has all informations.

It takes input | Request from Application Layer → Process the Request and generate the response and forward it to Application Server



Client Server Architecture :-

ENTITY RELATIONSHIP MODEL

The entity-relationship (E-R) data model is based on concept of Real world that consists of a collection of basic objects called 'Entities', and relationship among these Objects.

Entity An entity in an ER-Model is a real-world object having properties called Attributes.

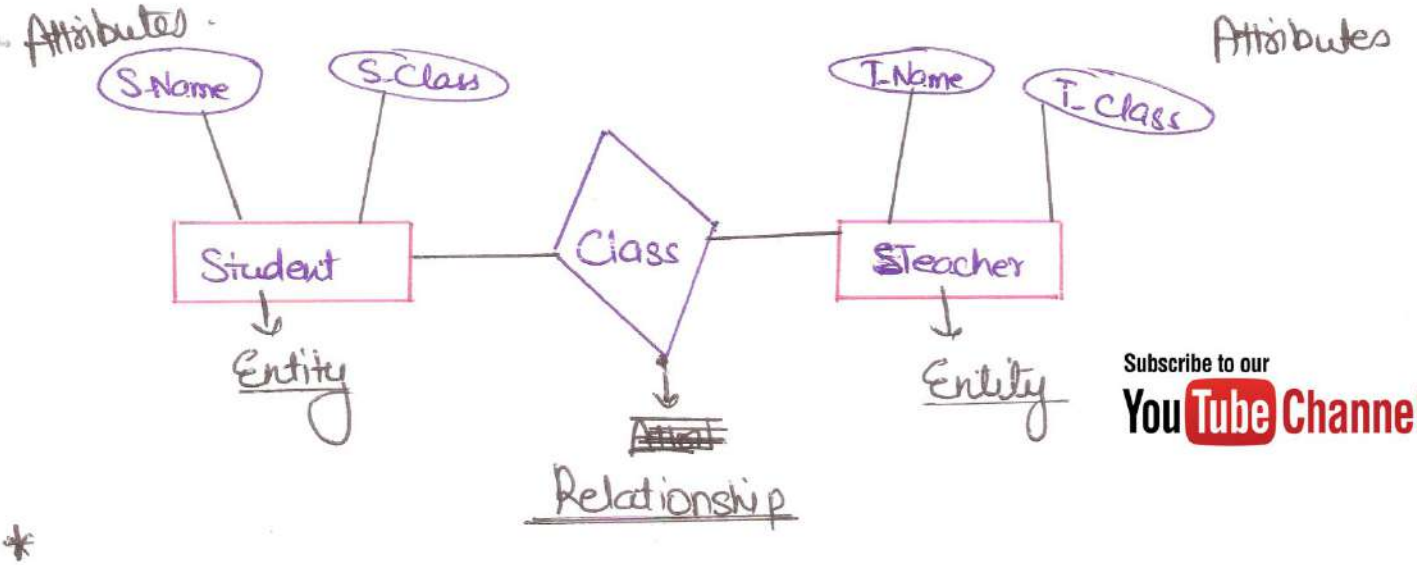
Attributes:- Every Attribute is defined by its set of values called Domain.

Eg In a school database, a student is considered as an Entity.

**→ Student has various Attributes like S.Name, S.Age, S.Class etc.

Relationship A logical association among several entities

Eg:- In the above Eg we add 'Teacher' Entity with Attribute T.Name, T.class. So Student and Teacher Entity Can be Relationshiped with Relation class.



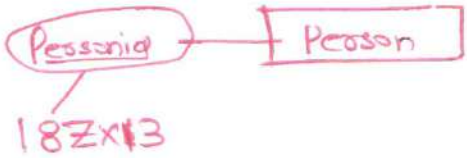
* One Entity Can have 2, 3 or more Relationships with Other Entity

E-R MODEL

ENTITY SETS

Entity Sets :- An Entity is an object of Real world. So set of entities of the same Type

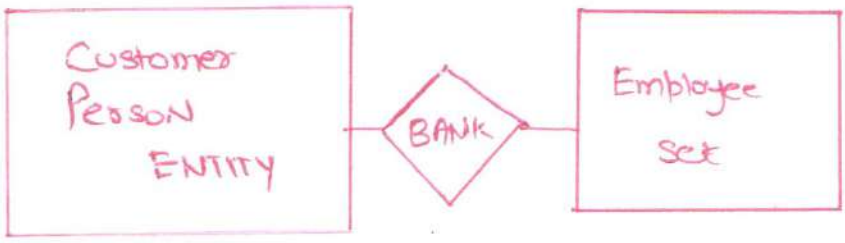
Eg: A person in an Enterprise is an entity. and it has one attribute which uniquely identified it i.e P_id (Personid) which is 188613



→ Similiarly a loan entity with Loan_No as its attribute and Let L_No is L573.

So here we can say 'Set of all persons who are Customers at a given bank Can be defined as 'entity set Customer' Similarly an entity set of Loan might represent set of all Loans

** → Entity set do not need to be disjoint



*** In this an employee in the bank may be also Customer in that bank.

Attribute :- Entities are described by a set of attributes.

Eg: an employee or Customer set → have attributes like Customerid, Customer-name, Customer-street and Customer-city.

Similarly a Loan entity set have attributes like
→ Loan_Number
→ Loan_Amount

Customer_id attribute in Customer entity set is our primary key which means through value of it we can uniquely access particular Row. b/c

CustomerName :- It Can be Same Name of other person
Customer_City :- Same City for other Customer.

Domain or Value Set :- for Each Attribute, there is a set of permitted values which is called Domain or Value set

→ The Domain of attribute 'Customer-name' might be set of all text Strings of Certain Length.

→ Similarly domain of attribute Loan_Number might be set of string of form "L-n" where n is a positive integer.

→ Entity can be described by a set of (attribute, data value)

Eg (Customer id-, 677-89-9011) for Customer entity set
(Customer_Name, Deepak) ..... This is done for all entities for their all attributes.

677-89-9011	Deepak	Delhi	New-Delhi
688-62-6999	Rahul	Delhi	Punjabi bagh
:	:	:	:
:	:	:	:
:	:	:	:

Customer

L-17	10000
L-19	20100
L-28	18100
:	:
:	:

LOAN

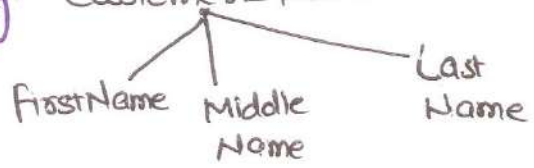
Attributes Can be Categorized:-

1) **Simple and Composite Attribute :-**

Simple Attributes are those which cannot be Subdivided

Composite Attributes are those which can be subdivided. ie(Others attributes)

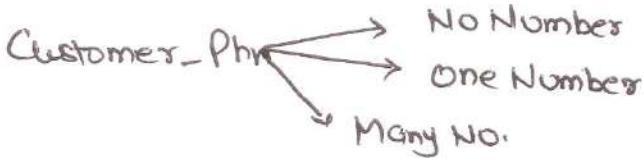
Eg. Customer_Name



2) **Single Valued & Multivalued Attributes** : Single Valued Attribute are those attribute which can have only Single Value

Eg: Loan_Number attribute for a specific loan Entity refers to only one Loan No.

MultiValued Attributes are those which has more than One Value



3) **Derived Attribute** :- In this one attribute is derived from other attribute

Eg:- let Loan_held is an attribute of Entity set Customer so Loan_held can be derived to No_of Loans by a Customers in Orders to Check how many Loans a Customer have.

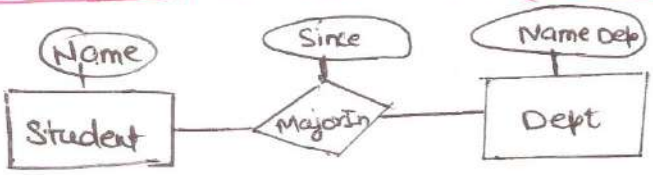
Relationship Sets - Keys As primary key of an entity set allows us to distinguish among the various entities of the set. So we also need a similar mechanism to distinguish among the various relationships of a relationship set.

Let R be a Relationship set involving Entities sets $E_1, E_2, E_3 \ldots E_n$. Let primary key (E_i) denotes the set of attributes that forms the set of attributes that forms the primary key for entity set(E_i).

** → Assume the attributes names of all primary keys are unique, and each entity set participates only once in the relationship

→ Then the composition of the primary key for a Relationship set depends on the set of attributes associated with the relationship set R.

As a Relationship can also have (Zero - More) attribute.



Here Since is an attribute of Relationship MajorIn.

like 'John majors in "CS" since 2000

→ If Relationship set R has no attributes associated with it, then the set of attributes

Primary-key(E_1) ∪ primary-key(E_2) ∪ ..... ∪ primary-key(E_n)

describes an individual Relationship in set R.

CONSTRAINTS IN E-R MODEL

There are 2 Constraints in E-R Model

→ Mapping Cardinalities

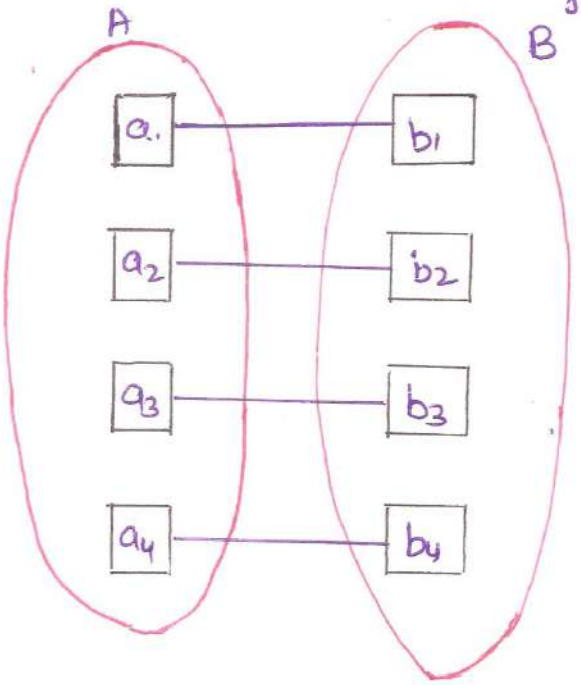
→ Participation Constraints.

Mapping Cardinalities :- Or Cardinality ratios, It tells the No. of entities to which another entity can be associated through a Relationship set.

Mapping Cardinality is used Binary Relationship sets.

So for a Binary Relationship Set R between entity set A and B, there will be one of this Mapping Cardinality :-

- **One to One** :- An entity in A is associated with "at most" one entity in B and an entity in B is associated with atmost one entity in A

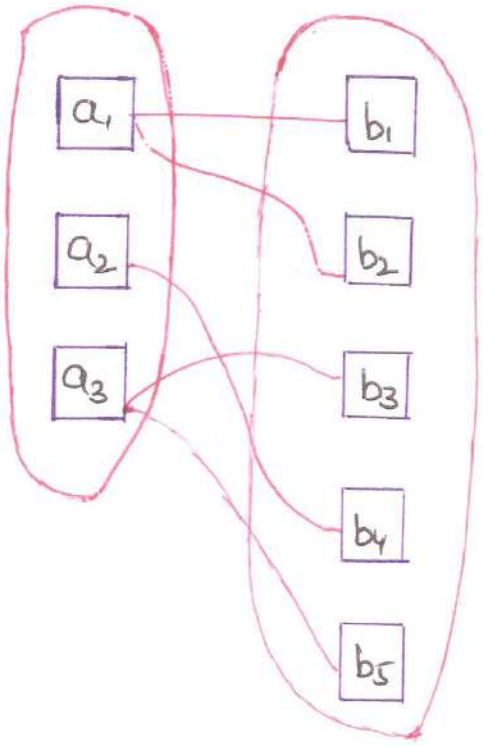


Automobile —has— Engine

Department —Headed by— Head of dept

Real world Example

- **one to many** :- 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 atmost one entity in A

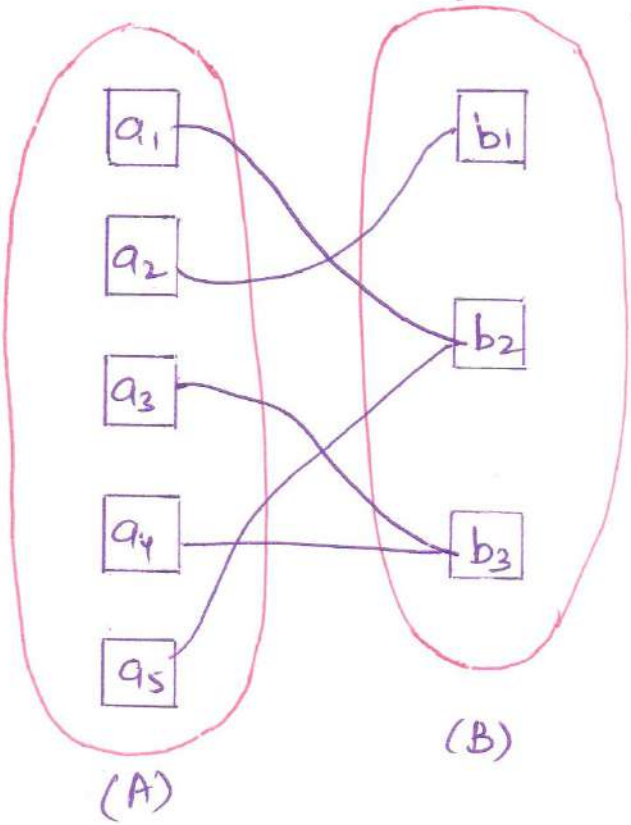


Father $\xrightarrow[\text{Many}]{\text{has}}$ Children

Teacher $\xrightarrow[\text{Many}]{\text{teaches}}$ Courses

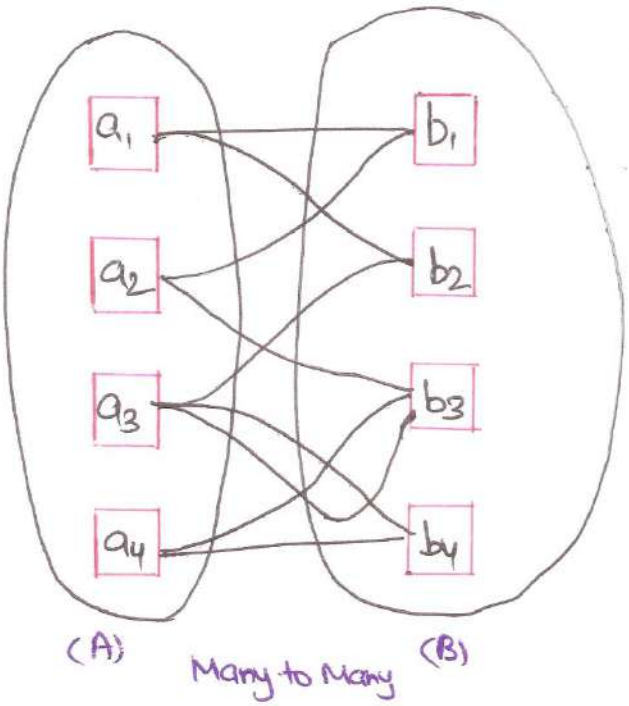
One to many

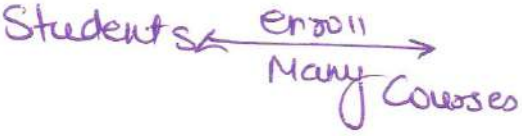
- **Many to One** : An entity in A is associated with atmost one entity in B. An entity in B, however can be associated with any number (Zero or More) of entities in A.



** Students $\xrightarrow[\text{for a single}]{\text{Many option}}$ College

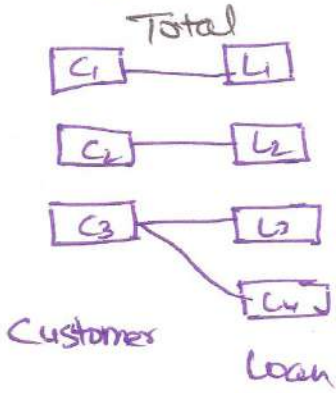
- Many to Many → An entity in A is associated with any number (zero or more) of entities in B, and an entity in B is associated with any number (zero or more) of entities in A.

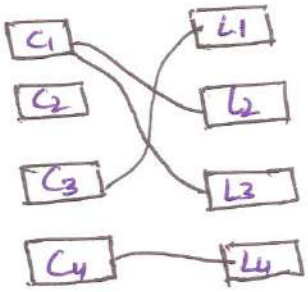




2) Participation Constraints :→ The participation of an Entity set E in a Relationship set R is said to be 'total' if every entity in E participates in at least one relationship in R.

→ if Only Some entities in E participate in Relationships in R, then the of entity set E in relationship R is said to be partial.





KEYS
In E-R-Model

No two entities in an entity set are allowed to have exactly the same value for all Attribute.

So, a Key allows us to identify a set of attributes that suffice to distinguish entities from each other.

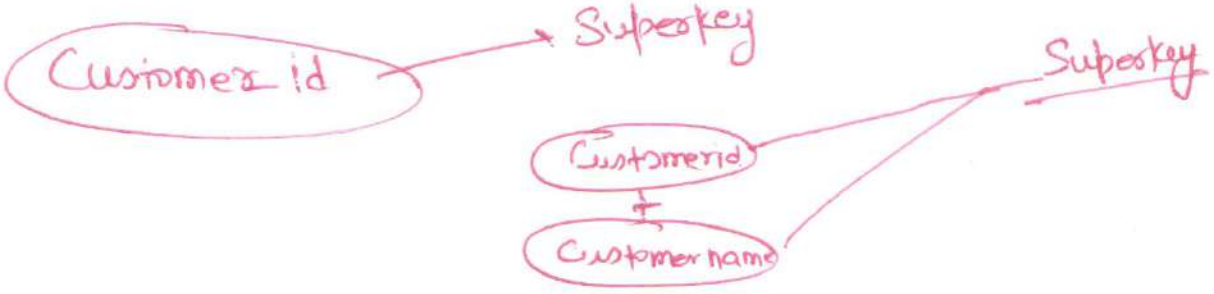
Eg Data in Entity attributes Customer_id of entity Customer should be unique and should be not be same for other entity set of same attribute Customer_id.

Keys in Entity Sets :- There are many keys through which we can uniquely identify an entity and Row.

Superkey → A Superkey is a set of one or more attributes that taken collectively, allow us to identify uniquely an entity in the entity set.

Ex:-
Customer_id attribute of the entity set Customer is sufficient to distinguish one Customer entity from another. Thus Customer_id is our Superkey.

Combination of Customer_name & Customer_id is Super key for the Entity set Customer. But Customer_Name attribute is not Superkey as persons has same name



2) Candidate key :- Minimal Super keys are called Candidate
If we consider K is Super key then Candidate key will be any Super set of K.

3) Primary key :- is a Candidate key attribute/Column that is most suited to maintain uniqueness in a table at the Row/Level.

Ex:- Lets take Entity Employee with Attributes

① Employee ID, ② Employee Name ③ DOB (Date of birth)
④ DOJ (Date of joining), ⑤ SSN (Social Security No.),
⑥ Dep ID (Department ID) ⑦ MgrID (Manages ID)

→ In this Employee entity, attribute Employee ID & SSN individually can maintain uniqueness in a table. So they are eligible for Candidate key. Attributes Employee_Name + DOB Combined can also make up a candidate key. but there is narrow chance that 2 Employees with same name can be born in same day.

→ In this Employee ID & SSN (Not combindally) only be used here as primary key because every value in this attributes are unique.

→ In this Employee ID + Employee Name is a Superkey because Superkey is a Superset of Candidate key. If we add any attribute in Candidate Key then it becomes a Super key.
As Employee Name can be name of Other employee so Combining them gives unique value..

Relationship Sets :-

A Relationship is an association among Several Entities

for Eg:- a Relationship that associates Customer Deepak with loan L-17. This relationship specifies that 'Deepak' is a Customer with Loan Number 'L-17.'

A Relationship Set : It is a set of Relationships of the Same type.

Mathematical Relation on

$n \geq 2$ (Possibly non distinct) entity sets.

If $E_1, E_2, \ldots E_n$ are entity sets the a Relationship Set R is a Subset of

$$\{(e_1, e_2, \ldots e_n) \mid e_1 \in E_1, e_2 \in E_2 \ldots e_n \in E_n\}$$

where $(e_1, e_2, \ldots e_n)$ is a Relation ship

Let 'Customer' and 'Loan' entity sets defines the Relation ship 'set Borrow' to denote the association b/w Customer and bank Loans

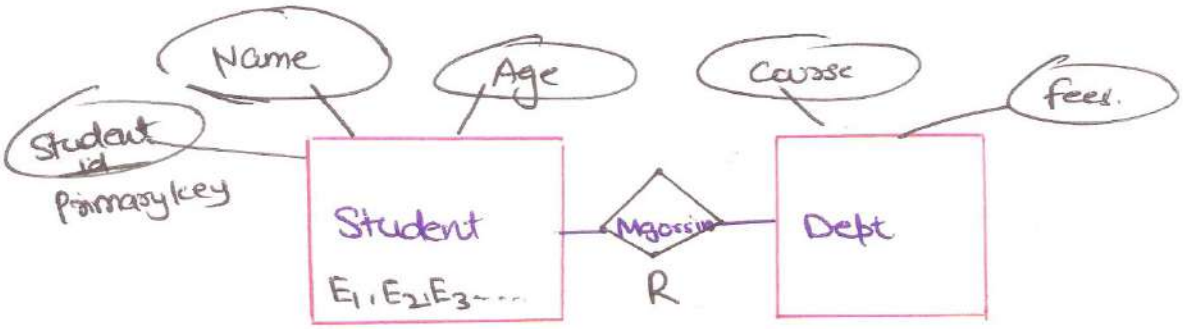
677-89-9011	Deepak	Delhi	New-Delhi
688-62-6999	Rahul	Delhi	Punjabi-bagh
⋮	⋮	⋮	⋮
688-62-6111	Sanjana	Punjabi	Rht.

Customer

L-17	10000
L-19	20000
L-28	18100
⋮	⋮
L-18	20000
⋮	⋮

Loan

Relationship Set Borrow

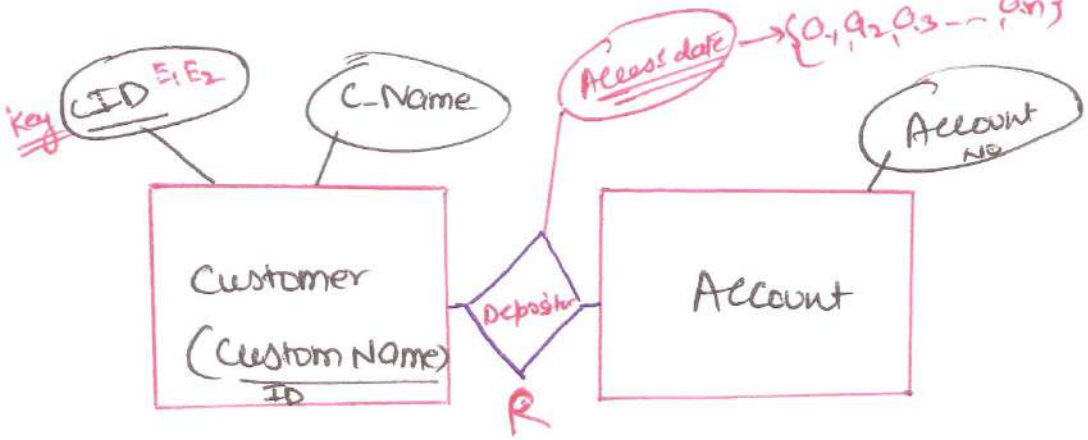


Relationship set R has no attribute

→ If the Relation set R has attributes $(a_1, a_2, \ldots, a_n)$ associated with it. then the set of attributes

primary-key (E_1) ∪ primary-key (E_2) ∪ ..... ∪ (Primary-key (E_n) ∪ $\{a_1, a_2, \ldots a_m\}$

describes an individual relationship in set R.



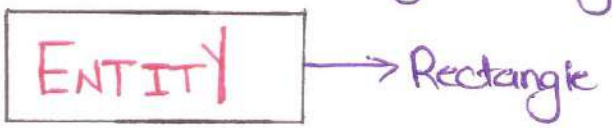
ENTITY RELATIONSHIP DIAGRAM

An Entity Relationship Diagram describes the how entities Such as people, Objects or concepts relate to each Other with in a System.

These ERD are Composed of Entities, Relationships & Attributes.

Now we will see How Entities and attributes are Relationship ed with help of ERD Components.

→ Rectangle: Entities are represented by Rectangles.



→ Double Rectangle: A weak Entity is represented in the Double Rectangle. and it is implemented by Foreign key Relationship with another Entity.

→ Diamond Shape: Relationship | Action are represented by Diamond.

→ Ellipses | Oval: → Attributes are represented by Ellipses | Ovals.

→ **Double Ellipse:-** An Attribute can have more than 1 field/value like phn NO. So it is Represented in Double ellipse

→ **Dashed ellipse**

↓

Derived Attributes

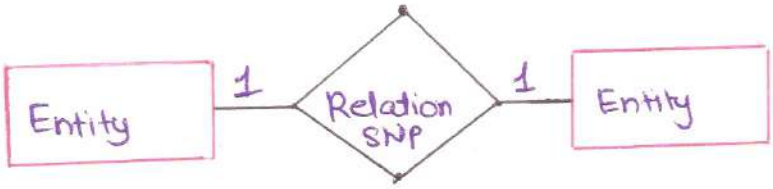
→ Attribute which is based on Another attribute are called Derived attributes and they are represented by Dashed Ellipse

like employ's Monthly Salary based on his annual Salary.

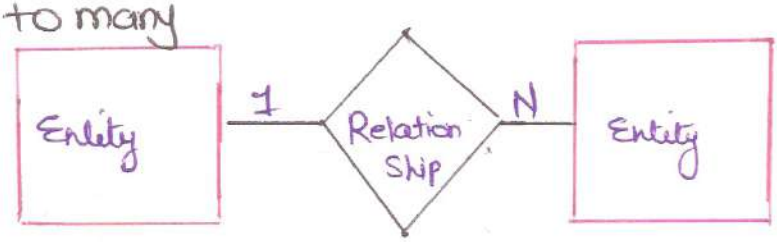
→ **Lines:-** They are ways of showing Relationships & Cardinality.

Binary Relationship and Cardinality

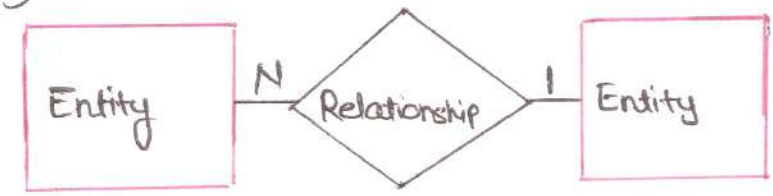
→ One to-one



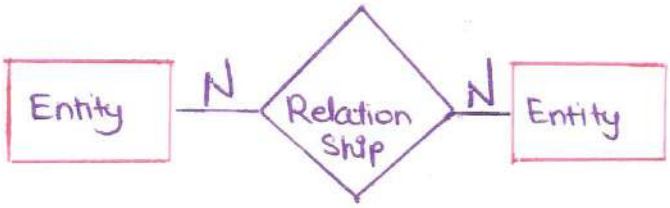
→ One to many



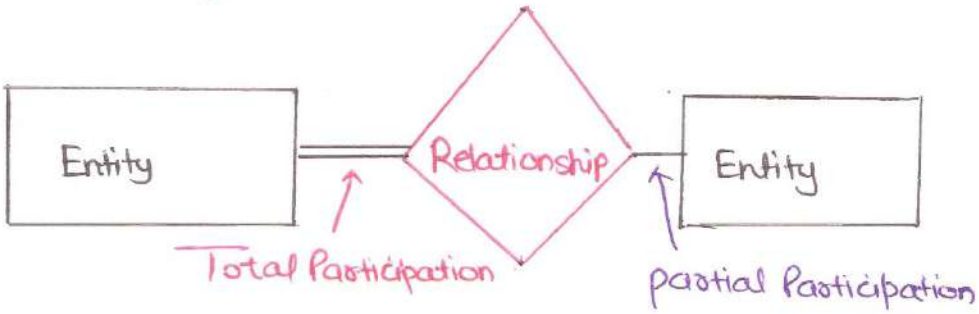
→ Many to One



→ Many to Many



Representation of participation Constraints



Reduction of E-R Diagram To Tables

For converting Strong Entity Types these are some points that should be keep in Mind.

→ Each Entity Type becomes a Table.

→ Each Single-Valued attribute become a column

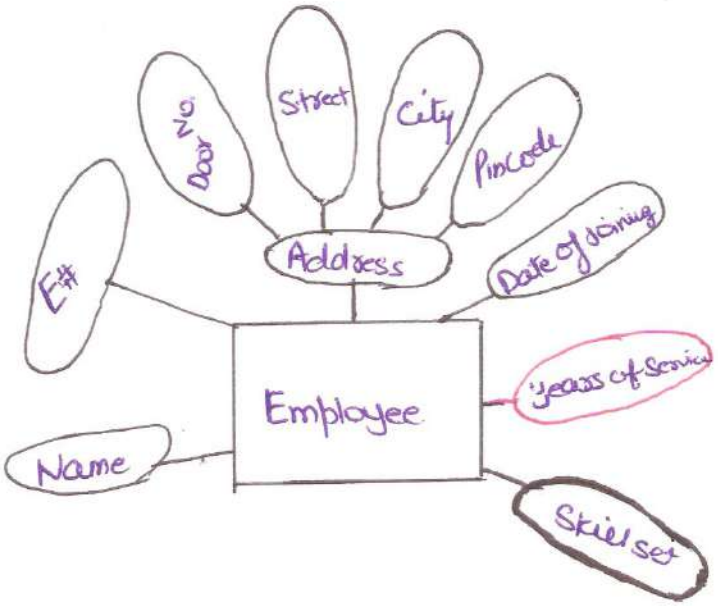
→ Derived attributes are ignored.

Composite attributes are represented by Components

→ Multivalued attributes are represented by a Separate Table

→ Key Attribute of the entity type becomes the primary key of the Table

Example



- Here 'Address' is 'Composite 'attribute'
- Years of Service is Derived attribute
- Skill set is a multi-Valued Attribute → We have to Make Seprate Table but Remember there Should be relation blw an Employee Table & Skill set Table

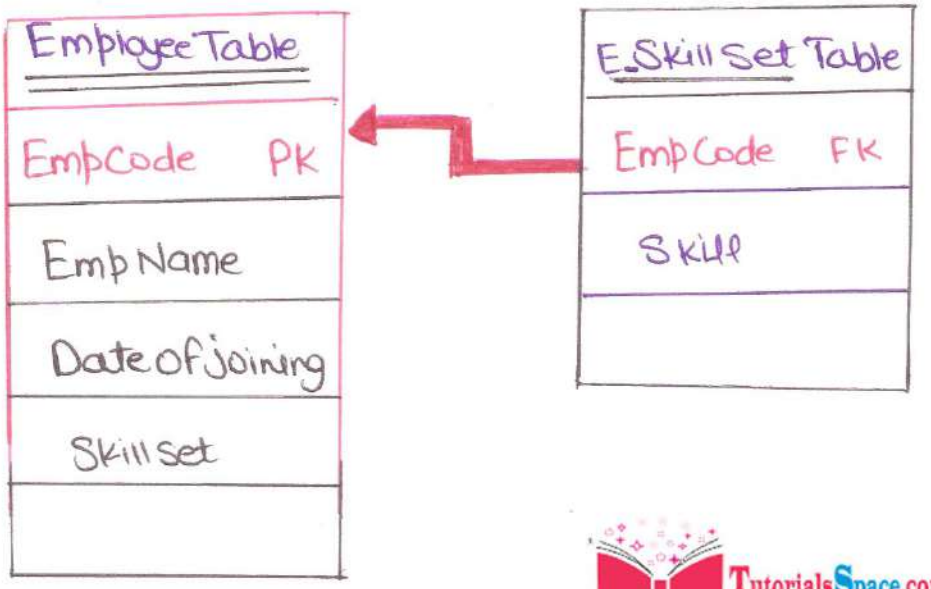
So the Relational Schema & Employee Skill set Table is

Employee (E# , Name , Door_NO , Street, City , Pincode, Date_of_Joining)

Employee Skill set (E# , Skill)

As we saw E# has to be primary key for Employee Table to uniquely identify an entity & Employee skill set has skill which can come from skill set (Multivalued)

So to create a Relationship b/w these two table we use E# i.e Employee code in Employee Skill set table as a foreign key which implements Referential Integrity.



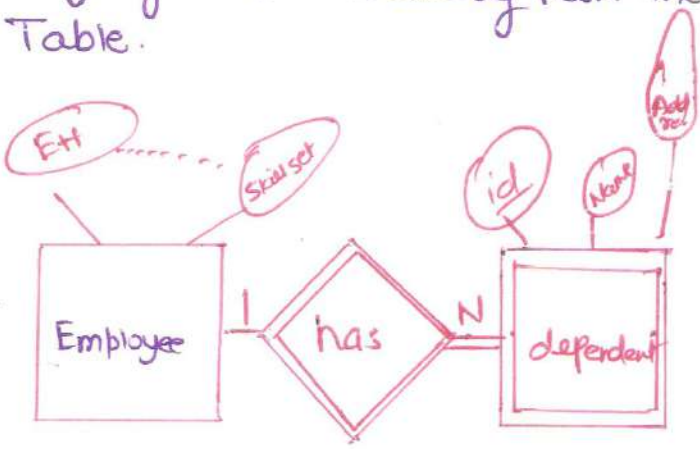
Converting Weak Entity Types

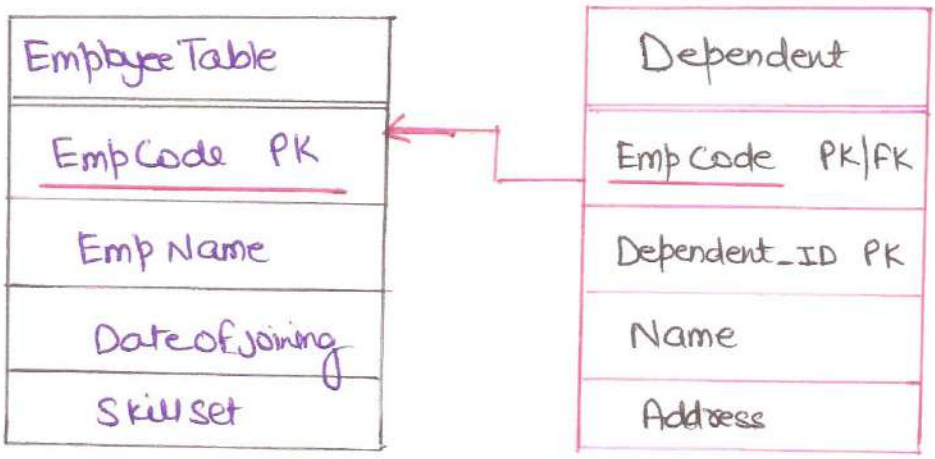
- Weak Entity types are converted into table of their own, with the primary key of the strong entity acting as a foreign key in the Table.

- This foreign key along with the key of the weak entity form the composite primary key of this Table.
- The Relational Schema

Employee (E# .....)

Dependent (E#, Dependent_ID, Name, Address)

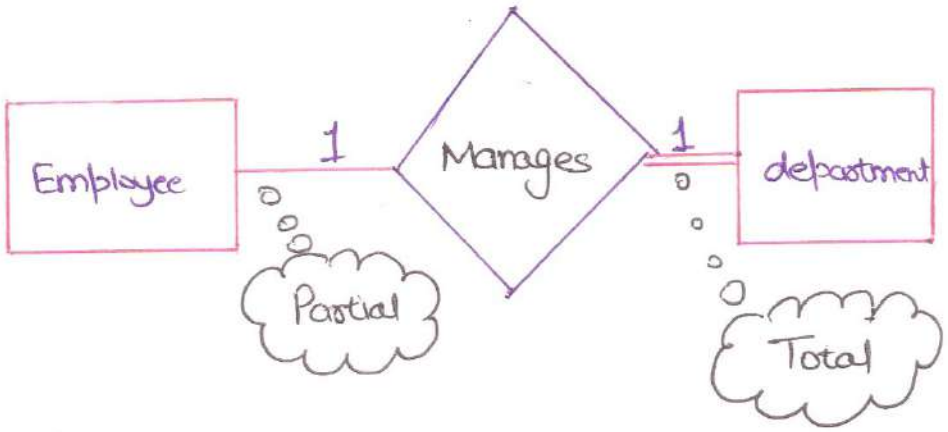




CONVERTING RELATIONSHIP

→ The way relationships are represented depends on the Cardinality and the degree of the Relationship.

→ Possible Cardinalities are :- 1:1, 1:M, N:M



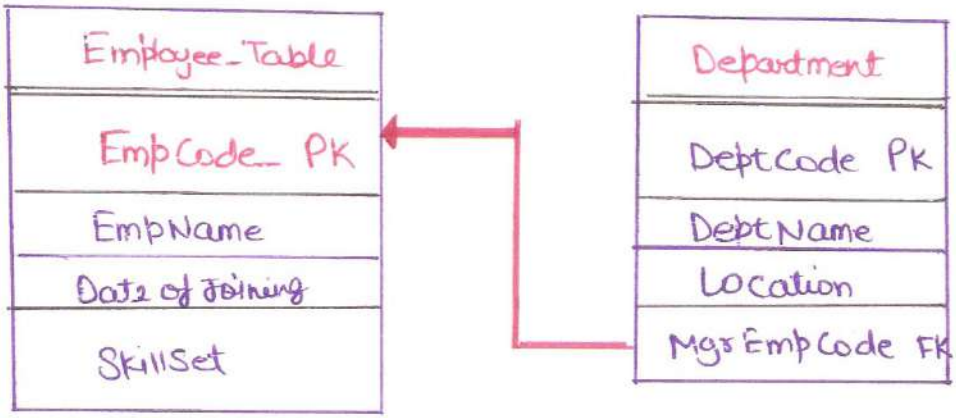
Case1: Combination of Participation Types.

The primary key of the partial participant will become the foreign key of the total participant

Relationship Schema is

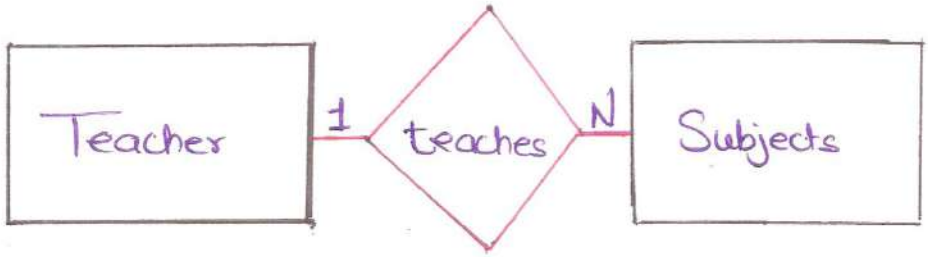
Employee (E#, Name, ...)

Department (Dept#, Name, ... Mgr#)



Converting Relationship of Cardinality 1 : N (one to Many)

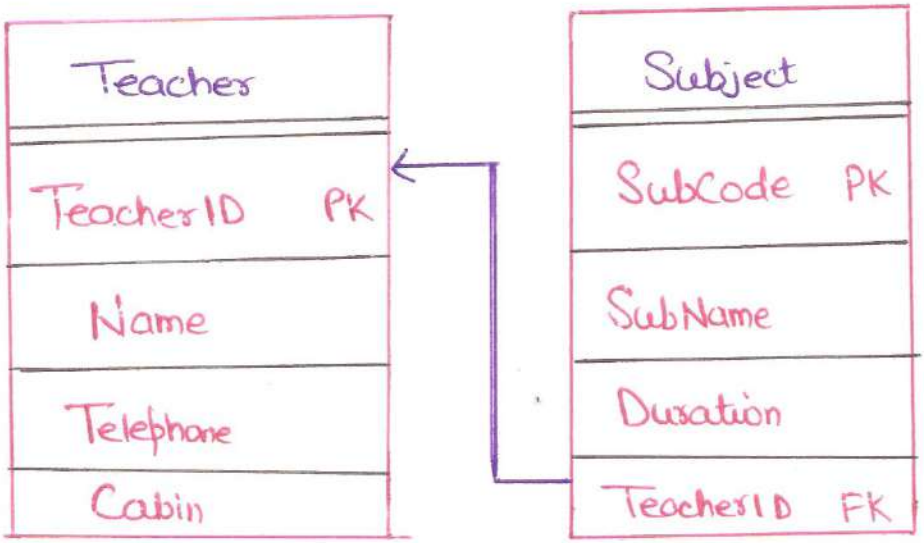
1 : N



In this Primary Key of the Relationship on th '1' side of the Relationship becomes a foreign key in the Relation on the "N" side.

Teacher (ID , Name, Telephone, ...)

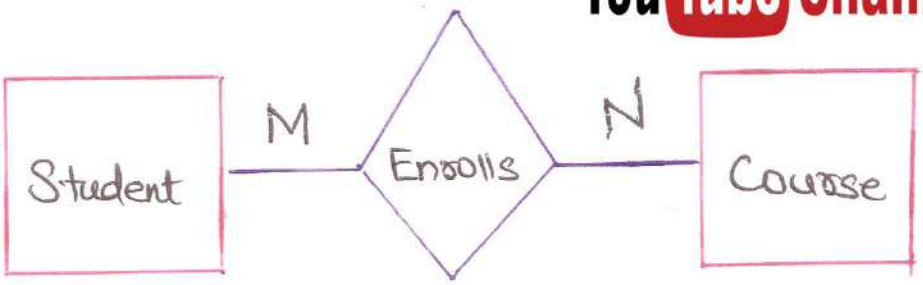
Subject (Code , Name ..... Teacher)



Converting Relationship of Cardinality (M:N)

→ A new table is created in this type of cardinality to represent the relationship.

M:N



→ It has to contain two Foreign Keys - One from each of the participants in the Relationship.

→ The primary key of the new table is the Combination of the two foreign keys

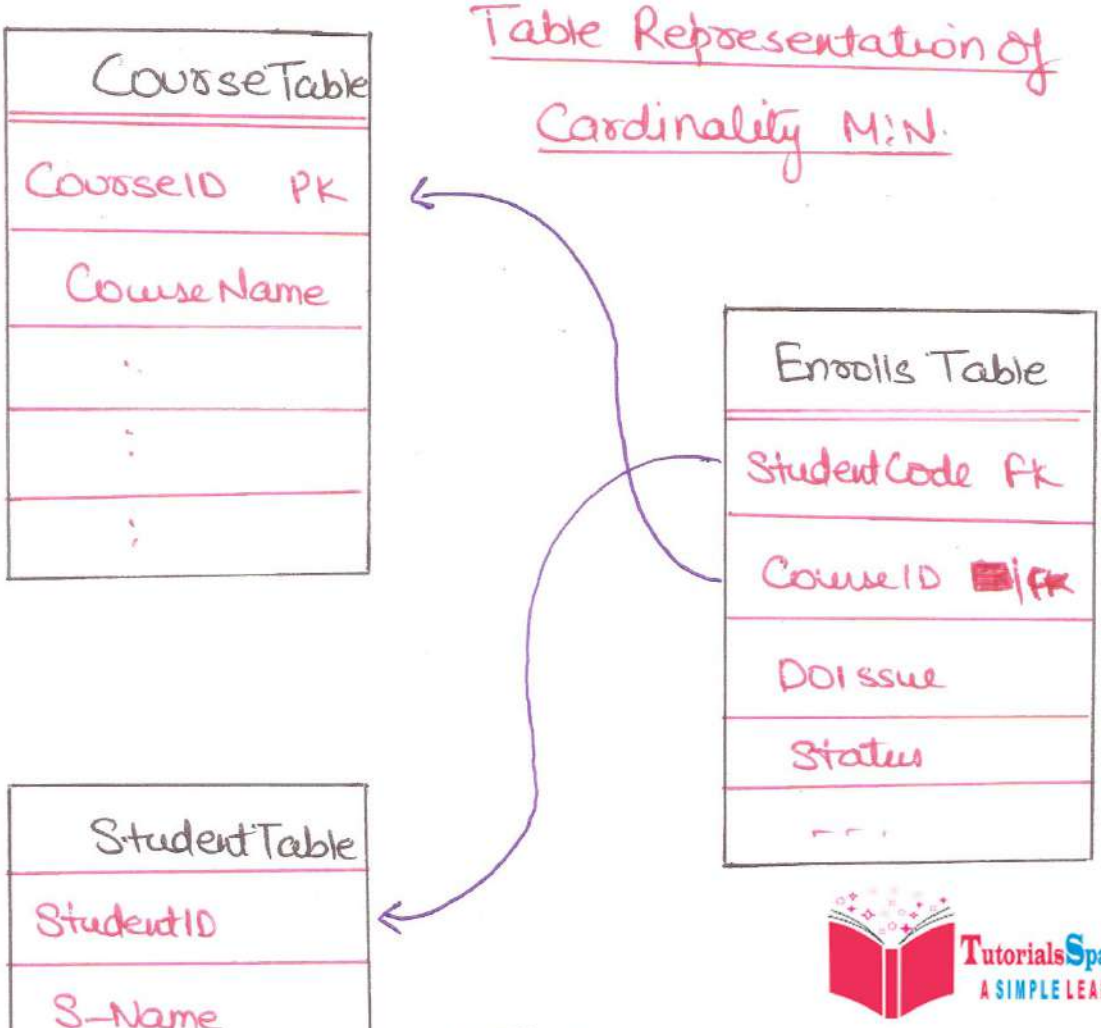
Student (S_id#, Title.....)

Course(C_id#, C_Name)

Enrolls (S_id#, C#)

In this table of Course we have two foreign keys i.e (Student Code, Course id)

Table Representation of Cardinality M:N.



In this 3 tables which are under Relationship with primary keys | foreign keys.

→ CourseTable attribute CourseID (PK) is a foreign key for Enroll Tables of attribute CourseID

→ StudentID of table Student is a foreign Key for table Enroll for attribute Student Code