# DBMS

# Introduction
# Database System Architecture
# Data Models
# ER- Models
# Keys
# Relational Models
# Relational Algebra
# Function Dependencies
# Normalization
# SQL in Detail
# PL/SQL [ie Triggers & Cursors]

# Fundamental of Database System

## Unit - 1



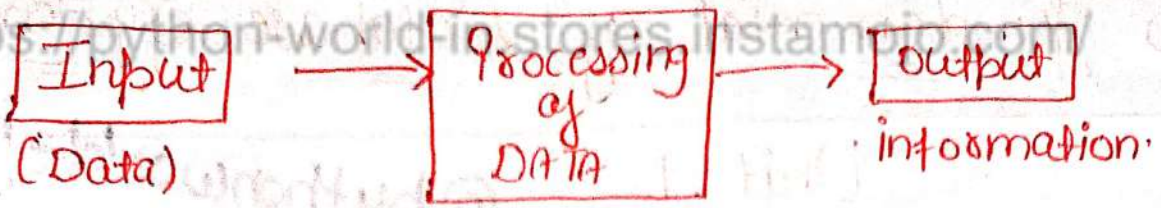
# Data :→ Data is collection of raw facts and figures is called Data. Data is collected from different sources, it is collected for different purposes.

- Data may consist of numbers, characters, symbol etc.
- Data is unorganized and unstructured.
- Data is independent
- We can represent the data in the form of a graph, data tree, tabular data etc.
- Data is a low-level knowledge.
- Data never depends on knowledge.
- Data which is gathered.

- Two types of Data
  - → Qualitative
  - → Quantitative.

# Information :→ The processed data is called information. Information is an organized and processed form of Data. It is more meaningful than data and is used for decision making

- Information is organized and Structured.
- We can represent the information in the form of languages, thoughts on the basis of the given data.
- Information is the second level of knowledge.
- Information depends on Data.
- Information which is processed.




# Database

A database is an organized collection of data, so that it can be easily accessed and mangged.

- The Main purpose of the database is to operate a large amount of information by storing, retrieving and managing data.
- There are many databases avaible like MySQL, Oracle, informix etc.
- Modern databases are managed by the database management system. (DBMS)
- SQL or Structured Query Language is used to operate on the data stored in a database. SQL depends on relational algebra and tuple relational calculus.
- A cylindrical structure is used to display the image of a database.

## → Need of Database:

1) Manages Large amount of data :→ A database stores and manages a large amount of data on a daily basis

2) Accurate :-> It means that the information is free from errors. and it clearly and accurately reflects the meaning of data on which is based.

3) Easy to update data :-> In a databases, it is easy to update data using various Data manipulation languages [DML] available. One of these languages is SQL.

4) Security of Data :-> Databases have various methods to ensure security of data. There are user logins required before accessing a database and various specifiers.

5) Data integrity :-> This is ensured in databases by using various constaints for data.

6) Easy to research data :-> It is very easy to access and research data in a database. This is done using data Query languages (DQL) which allow searching of any data in the database.

---

# Elements of Database System:

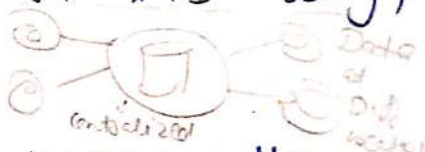
- Character [alphabet or number]
- fields [Store one type of details]
- Records [Collection of related fields]
- files [Collection of related records]
- Database [collection files]

# Objectives of Database

a) Data integration
b) Data integrity
c) Data independence

a) Data Integration:→ The data in file System is Stored in separate files. It is very difficult to access data stored in separate and independent files

- An important objectives of database is to solve this problem.
- The data in database may be located at different computers physically but it is connected through data communication links. In this way, data appears centralized logically.

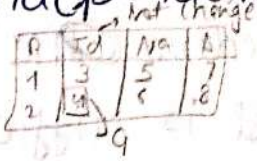


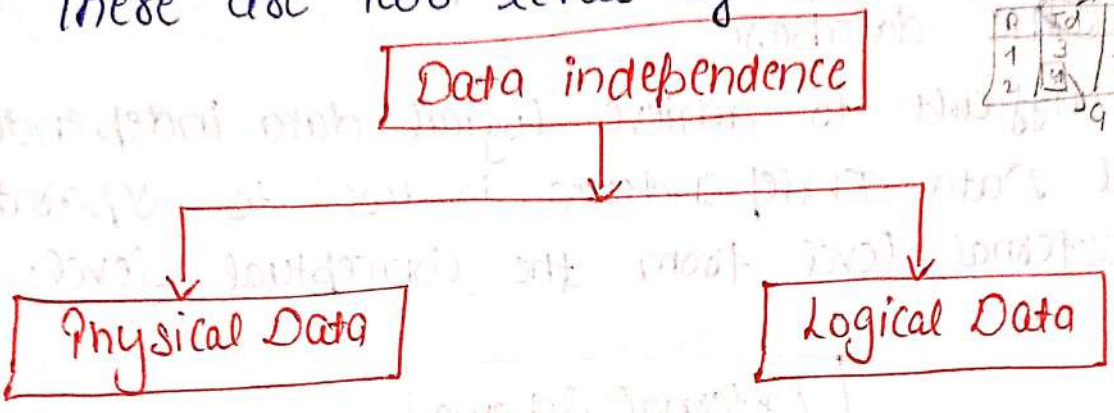
b) Data Integrity :→ Data integrity means the reliability and accuracy of data.

- Integrity rules are designed to keep the data consistent and correct. These rule act like a checks on the incoming data.
- Enforcing data integrity ensures the quality of data in the database.
- for ex:→ If an employee ID are entered as "123" this value should not be entered again. The same ID should not be assigned to two or more employes.

c) Data independence: Database approach provides the facility of data independence.

- It is a capacity of changing the schema at one level without affecting the other level is called Data independence.
- Data independence is one of the main advantages of DBMS
- There are two levels of data independence.



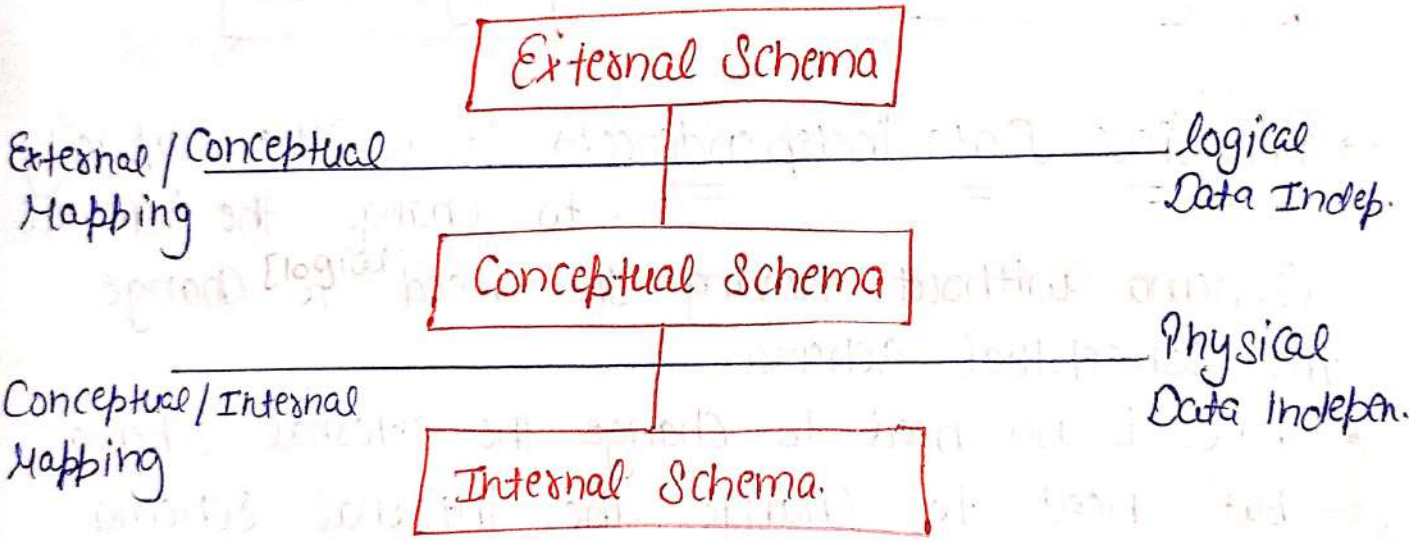



→ Physical Data independence :→ It is the capacity to change the internal schema without having the need to change the conceptual schema.

- there is no need to change the external schema but need to change the internal schema because some physical file were reorganized -by to improve the performance and creating additional structures
- It is easier to achieve logical data independance
- Physical data independence is used to separate conceptual levels and from the internal levels.

2) Logical Data Independence :- It is the capacity to change the conceptual schema without having to change external schema or application program.

- 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
- It is difficult to achieve logical data independence.
- Logical Data Independence is used to separate the external level from the Conceptual level.




# Records :- A Collection of related fields used as a single unit is called a record.

- also called row
- It contains Multiple field / set of fields.

Example :-

Std. Table

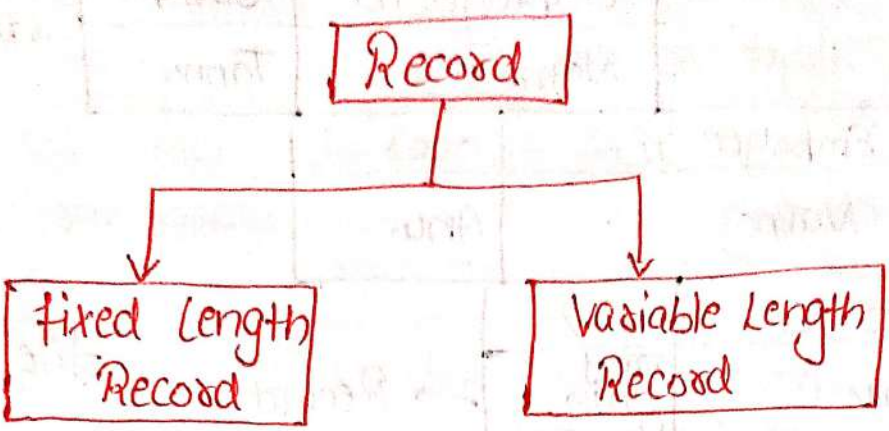
| Std. Name | Std. Roll | Address |
|---|---|---|
| Ali | 1 | Lahore |
| Ahmad | 2 | fsd |
| Hamza | 3 | Islam |

field → Combination of one or more characters → Column → Unique name

Record.

| Std. Nam | Roll | Addd |
|---|---|---|
| Ahmad | 2 | Fsd. |

- Record are of two types:




- Fixed Length Record :→ All records of file are of same size.

- Variable Length Record :→ Records of file can be different size.

---

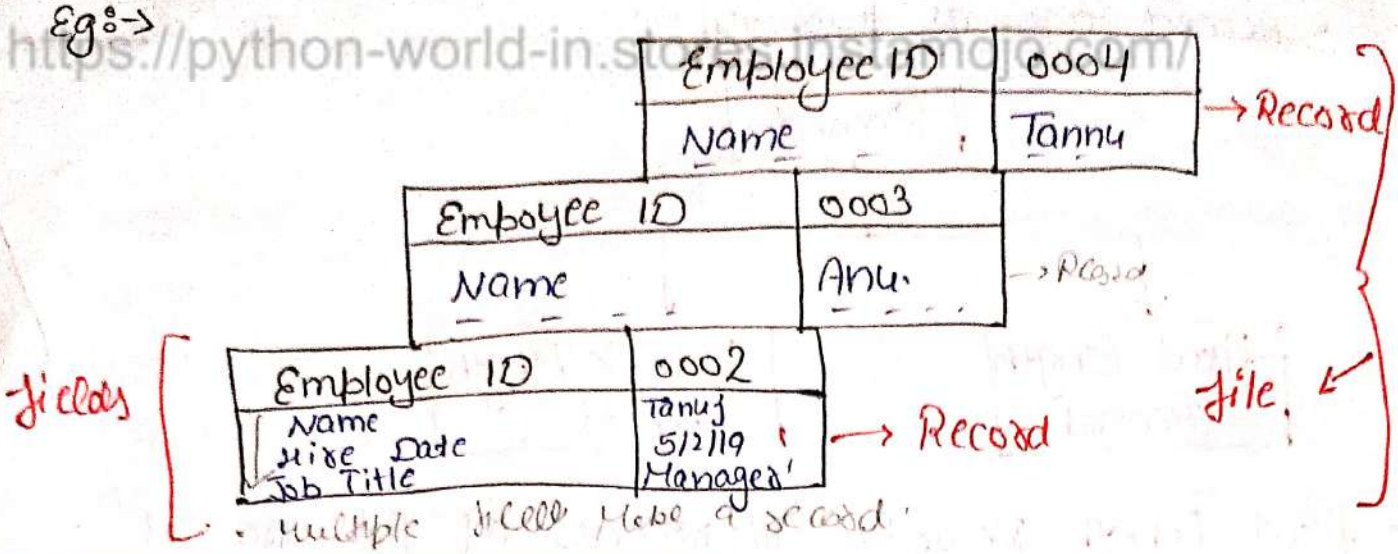
# File: A collection of related records used as a single unit is called file.

- file is also known as data set.
- files are stored on different storage media like hard disk, USB, optical disk etc.

Eg:→ "Employee" file may contain the records of hundred employees".

- Each employee's record consists of same fields, but each field contains different data.

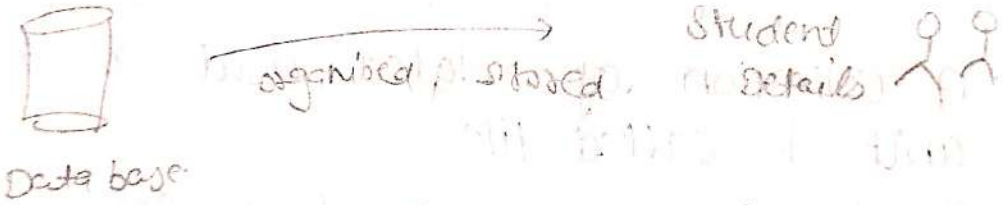
Eg:->



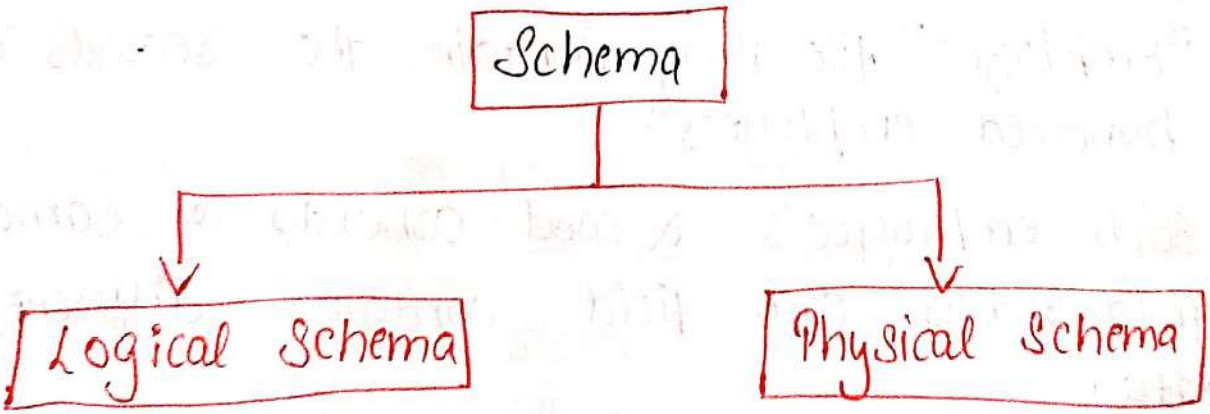

# Schema :-> A Schema can be defined as the design of a database. The overall description of the database is called Schema.

- It defines how the data is organized and how the relations among them are associated.

Ex:-> User Store Student Details in Database




- It can be divided into two parts:




# Logical Schema :-> - Programmer Works on that Level.
- At this level, data can be described as a certain types of data records which can be sotored in the form of data Structures.

# Physical Schema:- Data are storage (in bytes, gigabytes, terabytes or higher) in memory and these elements often remain hidden from the programmers



---

# Instance: It is the collection of information stored in the database at a particular moment is called as an instance.

- Data in instances can be changed using addition, deletion, updation.
- It changes frequently
- It is the set of information stored at a particular time.

---

# Traditional file System:-

A file system is a method of storing and organizing the computer files and the data they contain to make it easy to find.

- Data is stored in files
- Each file has specific format
- Programs that use these files depend on knowledge about that format.

# Advantage of Traditional file System:

1) Cost effective :-> • file management system is extremly cost effective since it is a digital filling system
   • Whatever the type of document that needs to be stored as a paper can be in the form of digital.

- Therefore, there is no cost involved in building rents, purchasing cabinets and physical papers.

2) Security: The traditional method to storing files cannot match the level of security provided by the file management system.
- In fact, security is one of the reason why many organization prefer to use file management system
- The document stored in the file management system is protected using authentication method like username and password.

3) Reliability:
- The data that is stored in a file management system is far more reliable than physically storing it using papers and files.
- All the data of the users are stored inside the servers.

4) Data retrieval:- Using file management system means that is will be very easy to retrieve data.
- Users don't need to search copies of documents manually here.
- there is very less amount of time spent for data retrieval.

5) Data Backup:- In case of a failure, file management system provides a seamless way for backing up data.
- However, if needed there can be also third party application program to be used.

6) Environment friendly: • Due to the fact that file management system follows a digital system and there is no paper works involved. it can be said that this technique is more environment friendly.

• As a result, it can provide tax benefits and other advertising opportunities as well.

---

# Disadvantage of file System / Limitation / Drawbacks

---

1) Data redundancy: [Duplicating data]

• It is possible that the same information may be duplicated in different files. this leads to data redundancy results in memory wastage.

---

2) Data Inconsistency: [Duplicate data की value Same नहीं होना]

• Due to data redundancy, Data Inconsistency problem created.

• Multiple version of data means that many files can have different data of some person or thing.

• Data updated in one file may not be updated in another file which lead to data inconsistency.

---

3) Data Isolation:- Different programmers - Different files - Different Structures

• Data are stored in different files and it becomes difficult to develop an application based on data of Different files

4) Integrity Problem :- Data in file processing system does not follow metadata hence integrity problem.

5) High Storage Cost :- Too much version of same data can result in additional storage cost

6) Program Data Dependency :- As the programs are developed according to the data hence any change in format of data requires to change the application program accordingly.

7) Atomicity problem :- Any operation on database must be atomic. this means, it must happen in its entirely or not to all.

8) Security Problems :- No Mechanism for security of Data

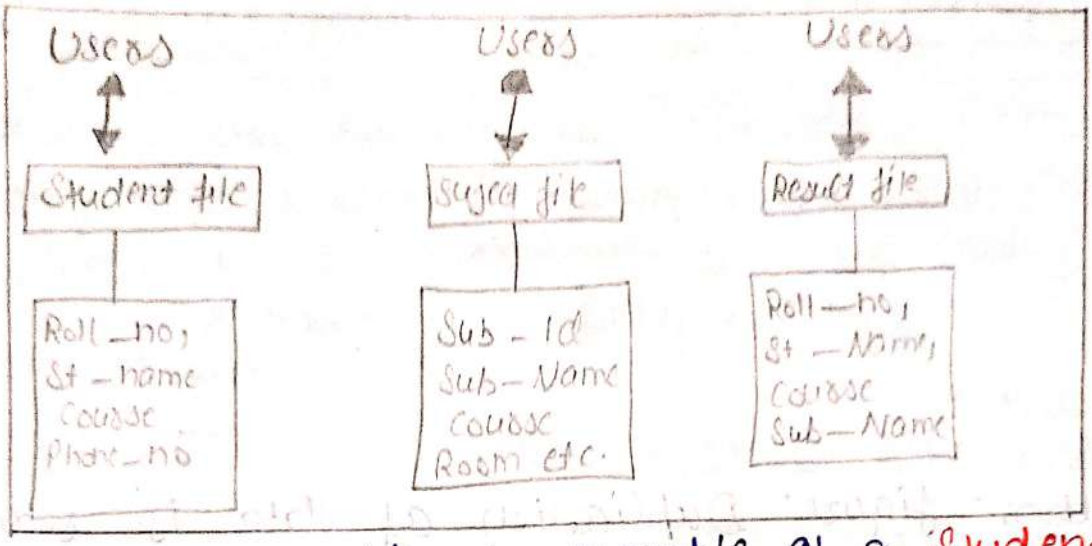
9) Program Maintenance :- Program developed are not only difficult to maintain but a crucial budget is spent on it.

10) Decentralised :- Same Multiple fields Different Meaning.

## Question: Different b/w DBMS and file System?

File System :→ It is also called traditional based approach.

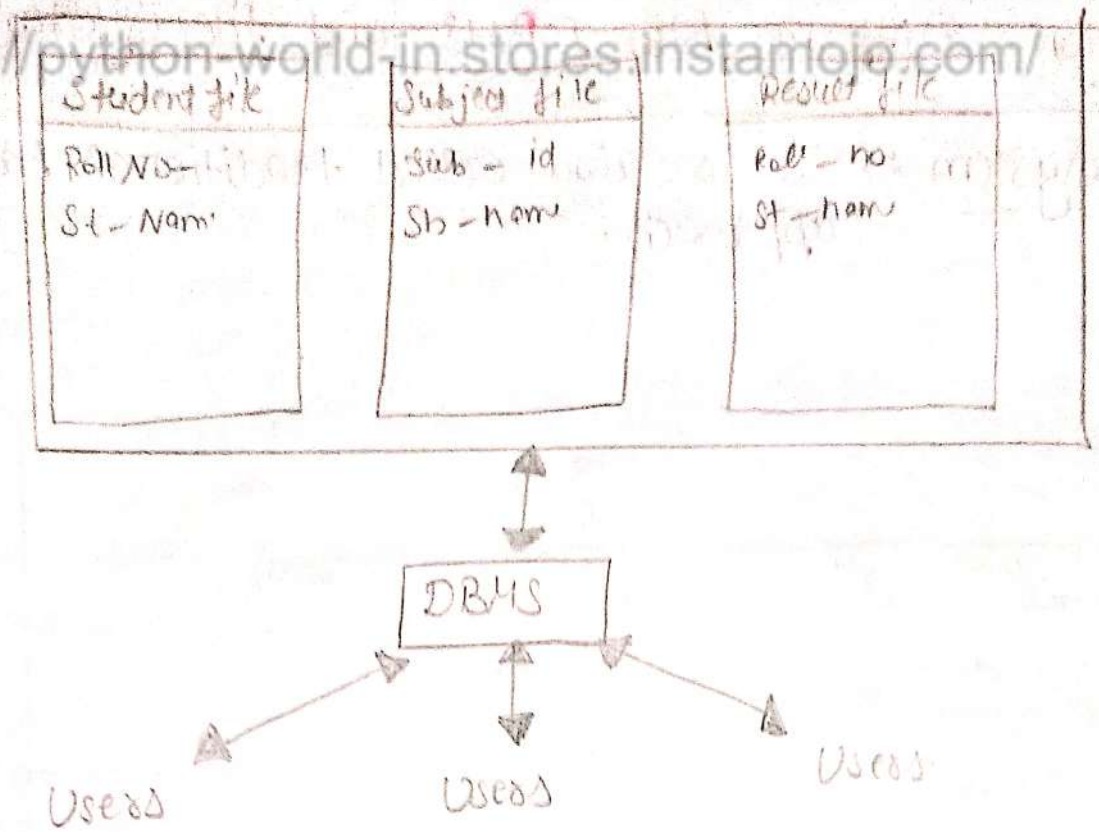
Eg :→




In above figure: Consider an example of a Student's file System. The Student file will contain information regarding the Student (i.e roll no, subject, name etc). Illy, we have a Subject file that contains information about the Subject and the result file which contains the information regarding the result.

- Some fields are duplicated in more than one file, which leads to data redundacy so to overcome this problem, we need to create a centralized System, i.e DBMS Approach.

DBMS :→ A database approach is a well-organized collection of data that are related in a meaningful way which can be accessed by different users but stored only once in a System.




In above figure: Duplication of data is reduced due to centralization of data.

| Basis | DBMS Approach | File System Approach |
|---|---|---|
| Meaning | • DBMS is a collection of Data. In DBMS, the user is not requir to write the procedur | • The file system is a collection of data. In this system, the user has to write the procedures for managing the database. |
| Sharing of Data. | Due to centralized approach, data sharing is easy. | Data is distributed in many files, and it may be of Different formats so it isn't easy to share data. |
| Data abstraction | DBMS gives an abstract view of data that hides the details. | The file system provide the detail of the data representation and storage of data. |

| | | |
|---|---|---|
| Data abstraction | DBMS gives an abstract view of data that hides the details | The file system provide the detail of the data representation and storage of data. |
| Security and Protection | DBMS provides a good protection mechanism. | It isn't easy to protect a file under the file system |
| Recovery Mechanism | DBMS provides a crash recovery mechanism. i.e DBMS protects the users from system failure | The file system doesn't have a crash mechanism then file content will be lost |
| Where to use. | It is used in large system. | It is for small system. |
| Structure. | They have very compler structure | They have very simple structure. |
| Cost | It is expensive | It is cheap |
| Data redundancy and Inconsistency | Due to the centralization of the database the problems of data redueny are controlled | In this, the file and application program are created by different programmers so that there exist a lot of duplication of data which causes inconsistency |
| Integrity constraints | are easy to apply. | are difficult to design. |
| Example: | Oracle, SQL Server etc. | Cobal, C++ etc. |

# Characteristics of Database Approach

## 1) Self-describing Nature of a database System

→ It Means that the Database system not only contains the database but also the information about the structure and constraints of the database.

→ This information about database structure and constraints is known as Meta Data.

→ The Meta data is stored in the Catalogue of the database

## 2) Insulation b/w programs and data and data abstraction:

(change are not effect on programs)

→ In a file system if some changes are made in the file structure, then to handle these changes, more changes have to be made in all the programs that access this file.

## 3) Support of Multiple views of the data:

→ A database has multiple users, each user may need different view of the database.

→ Database "view" can be the information that is stored in the database or is derived from the database

→ A database system allow multiple such "views" from the same database without letting the user know wheather the information is stored just derived.

## 4) Sharing of Data and Multiuser.

A Multiuser DBMS as the name suggest means has Multiple user.

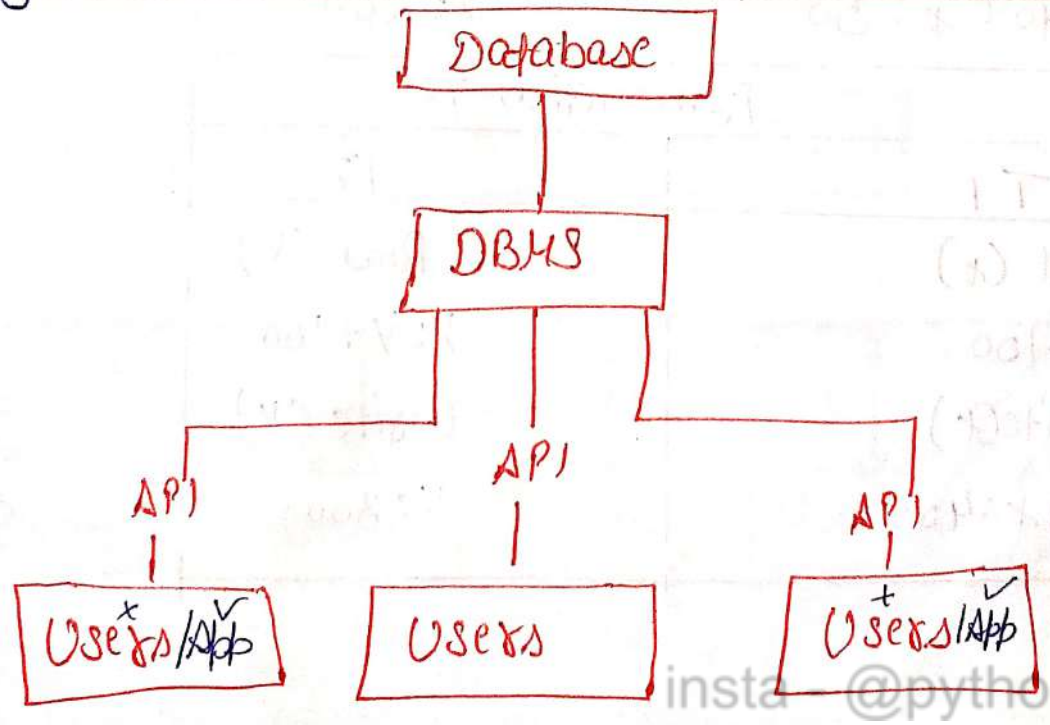
- It must allows Multiple users to access and use the database at the same time.




Database Management System.

Collection of data — Set of program to Store & retrieve of data.

- DBMS is a collection of data & Set of program to access & Store those data in an easy & efficient manner.
- DBMS is a Software that is used to manage the database. eg:→ MySQL, Oracle.
- It is also used to organize the data in the form of a table, Schema, views etc.
- The various operations performed by the DBMS System are: Insertion, deletion, Selection, Sorting etc.



# Properties of DBMS [using read and write operations]

- A transaction is a unit of program execution that accesses and possibly updates various data items.

→ in order to ensure accuracy completeness & data integrity.

A C I D

Atomicity Consistency Isolation Durability

[In order to maintain consistency in DBMS, before & after the execution of a transaction, data should not change]

→ **Atomicity**:→ Transaction must be atomic [all transaction is completed or none]
- All or nothing rule

operations:

Abort → Changes made to database are not visible

Commit → Changes made to be visible.

**Example**: Consider the following transaction T consisting of T1 and T2 : Transfer of 100 from account X to account Y.

| Before X : 500 | Y : 200 |
|---|---|
| Transaction T | |
| T1 | T2 |
| Read (X) | Read (Y) |
| X : X − 100 | Y : Y + 100 |
| Write(X) | Write (Y) |
| After X : 400 | Y : 300 |

A : 500 B : 200
T1 T2
Read(A) R(B)
A : A-100 B: B+
write(A) write(B)
400 300
500 after
before 100

2) Consistency: It means all the operations performed by user is logically correct & executed completely, then Database Transferred from one consistent state to another.

(No loss of data)

- Application programmer Responsibility for Maintaining properties

3) Isolation: If the Transactions are executing concurrently then Ti Transaction should not interfere in Ji Transaction.

Not depend on each other

- Concurrency control Manager

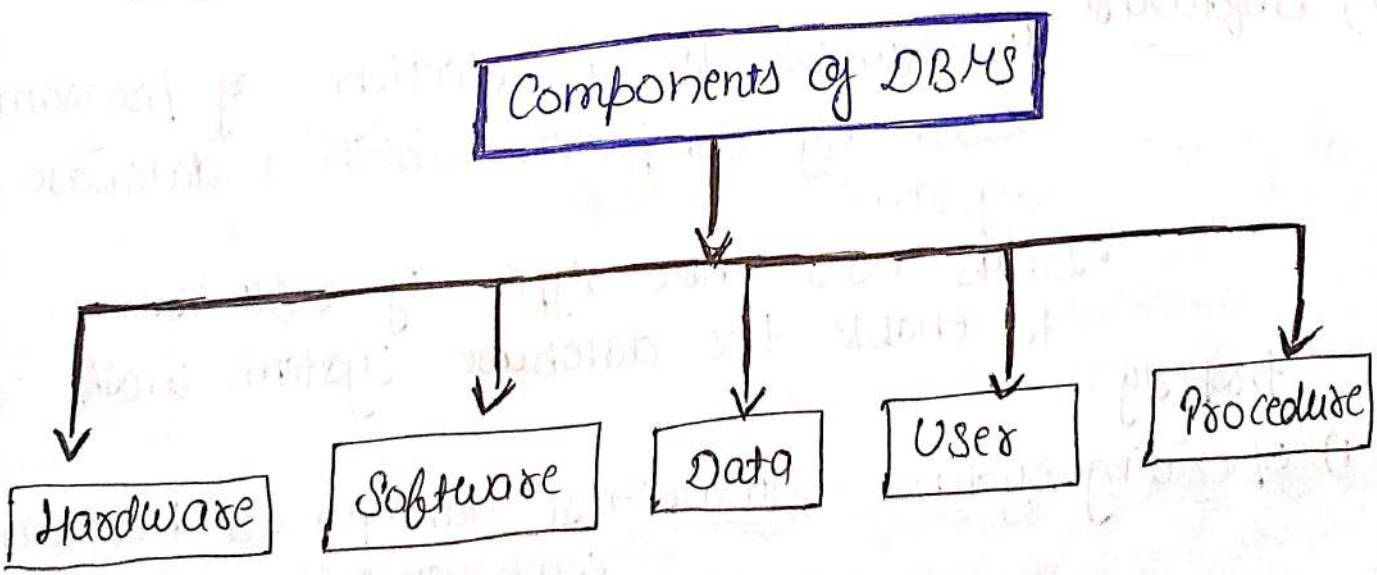
4) Durability: The commited Transactions should persist in DB. Changes not be Lost even of failure.

written data will not be lost

- Recovery Manager

# Components of DBMS

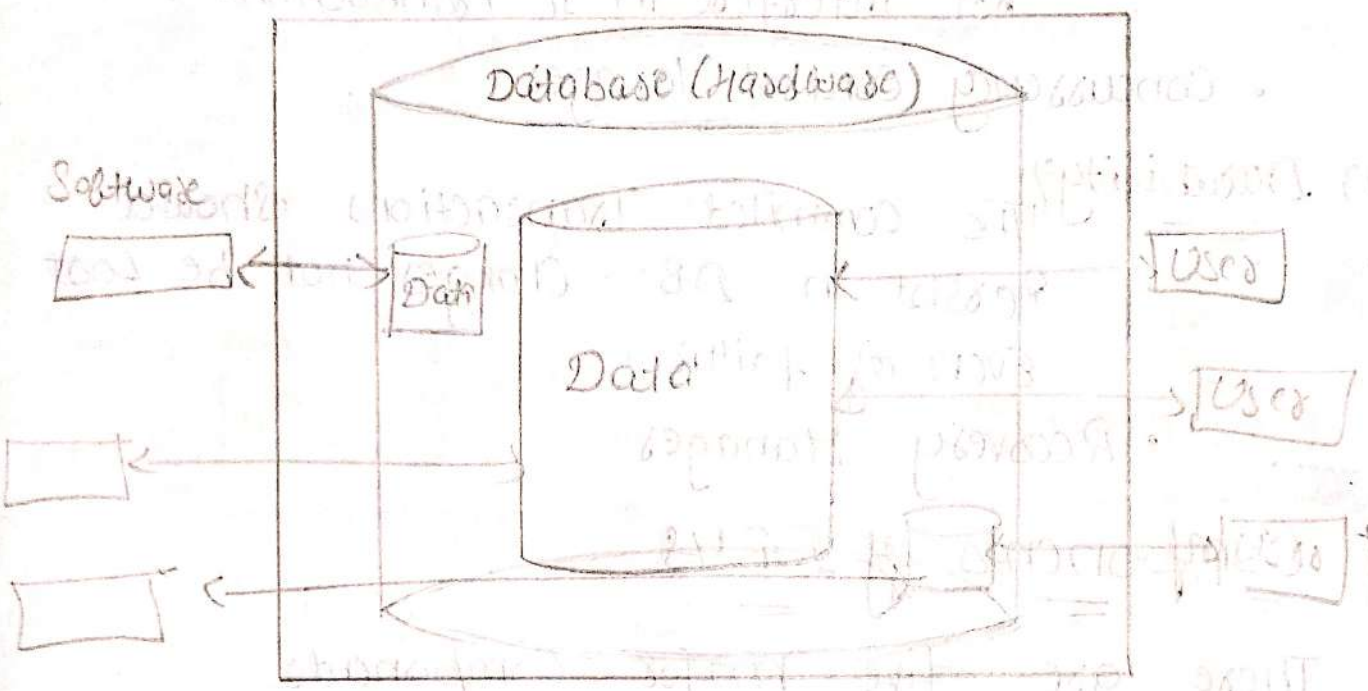
There are five major components



1) Hardware: → The physical components of a computer system are called Hardware.

- The hardware is used to perform different tasks such as input, storage and processing.
- Some important hardware components are as follows:
  - → Secondary Storage
  - → I/o devices
  - → Processors
  - → Main Memory






2) Software: → Software is a collection of program used by computer within a database system.

- DBMS uses three types of software to enable the database system work properly.

1) Operating System Software:→ It manages all hardware components.
- It also enables all other software to run on the computer.

2) DBMS Software: ↳ to manages the database in the database system.

3) Application programs: ↳ These are used to access & process the data stored in the database.

3) Data: ↳ Data is the most important components of database system.
- Data is a collection of facts stored in the database.
- The basic purpose of a database system is to store, maintain and process data for the user

→ Data can be divided into two types:
1) User Data
2) Meta Data.

4) User: ↳ The users are the people who control and manage the databases and perform different type of operation on the databases in the database management system.

⇒ five types of Users:-
1) Database Administrator:- They are responsible of the whole database system.
2) Application Programmers:- They write application programs to access data from database.
3) End Users:- These people use application programs to perform different tasks on database.
4) System Administrative

5) Procedures :- Procedures are the instructions and rules that govern the design and use of the database System. Procedures play an important role in a company, because they enforce the standards by which business is conducted within the organization.

---

# Function of DBMS

1) Data Manipulation :- Manipulation of refer to the process of adjusting data make it organized and easier to read.
- The DBMS must include a DML processor component.

2) Data Security :- Data Security refer to the process of protection data from unauthorized access and data corruption throughout its lifecycle. includes data encryption, hashing.

3) Data Dictionary :- Data Dictionary is the collection of names, definition and attribute about the data element.
- The contains data about the data rather than just raw data.

4) Data Storage management :- These process is help to store data on existing hardware.

5) Multi users access Control :- It is the process that allow the Multiple users to a computer.
- Time - Sharing Systems are multi-users systems.

# Advantages of DBMS

1. Control Database Redundancy :→ It can control data redundancy because it store all the data in one single database file and that recorded data is placed in the database.

2. Data Sharing :→ In DBMS, the authorized users of an organization can share the data among multiple users.

3. Easily Maintenance :→ It can be easily maintainable due to the centralized nature of the database system.

4. Reduce time :→ It reduces development time and maintenace need.

5. Backup :→ It provides backup and recovery subsystems which create automatic backup of data from hardware and software failure and restores the data if required.

6. Multiple user interface :→ It provides Different types of user interface like graphical user interfaces, applicat program interfaces.

7) Privacy and security

# Disadvantages of DBMS

1. Cost of Hardware and Software :→ It requires a high speed of data processor and large memory size to run DBMS Software.

2. Size : It occupies a Large Space of disks and large Memory to run them efficiently.

3. Complexity :- Database System creates additional complexity and requirements

4. Higher impact of failure :- failure is highly impacted the database because in most of the organization, all the data stored in a single database and if the database is damaged due to electric failure or database corruption then the data may be lost forever.

5) Performance :- Performance may not run as fast as desired.

---

# Action in Database

- Four main types of action in DBMS:-

1) Defining a database :- It includes the datatype, structures and constraints of the data have to store in the database. The database descriptive information is also stored by DBMS in the form of a database catalog or dictionary it is called Meta data.

2) Constructing the database :- It is the process of data storing on some storage medium that maintained by DBMS.

3) Manipulating a database :- It includes functions such as retrieve the database by using query, updating the database to reflect changes in the system, and generate reports from the data.

4) Sharing a database :- It allows Multiple Users and program to access the database simultaneously.

# Roles in the Database Environment

There are four distinct type of people that participate in the DBMS environment.

1) Data and database adminsistrators
2) Database Designers
3) Application Developers
4) End - users

## 1) Data and Database administrators:

- **Data administrators:** It is responsible for defining data elements, data names and their relationship with the database. They are also known as Data Analyst.

- **Database Administrators:** A database adminstrator (who define scheme) is a person who have complete control over database of enterprise. Database administrators is responsible for overall performance of database.
- He is free to take decision for database and provide technical support.
- Some of the main responsibilities
  1) Deciding the information contents
  2) Deciding hardware device to be used.
  3) Deciding the Back-up and Recovery Method
  4) Defines integrity constraints
  5) Ensuring availability of data
  6) Security

• Roles of DBA

1) Database Backup :→ A database responsibility to back up every data in the database, recurrently. This is necessary, so that operations can be restored in time of disaster.

2) Database Availability :→ A database administrator has the responsibility of ensuring database accessibility to users from time to time.

3) Database restore :→ A database administrator has the responsibility of restoring a file from a backup state, when there is a need for it.

4) Database design :→ A database administrator has the responsibility of designing a database that meets the demands of users.

5) Database upgrade :→ A database administrator has the responsibility of upgrading database software file when there is a new update for them, as this protect software from security breaches.

6) Database Security :→ Security is the major concern in the database. Database administrator takes various steps to make data more a secure against various disasters and unauthorized access of data.

7) Database Monitoring :→ Database administrator is responsible for overall performance of database. Database A regularly Monitors the database to maintain its performance and try to improve it.

8) Capacity Planning: A database administrator has the responsibility of planning for increased capacity, in case of sudden growth in database need.

- If the company is growing quickly and adding many new users, the DBA will have to create the capacity to handle the extra workload.

9) Troubleshooting: DBA are on call for trouble-shooting in case of any problems.

Wheather they need to quickly restore lost data or correct an issue to minimise damage, a DBA needs to quickly understand and respond to problems when they occur.

---

# Database Designers

The responsibility of database designe is to identify.

- Data to be Stored
- Structure of the data
- Store the data.
- Database desigers interact with all the users of database to identify the requirements of database.
- They Classified into two types:
  1) Logical Database Designers
  2) Physical Database Designers.

1) Logical Database :- The one who identify the data and the relationship among the data.

- The Logical Database designer must have a through and complete understanding of the organization data and any constraints on this data

2) Physical database: It decides as to how to represent the logical database design physically.
- Mapping the logical database design into a set of tables and integrity constraints.
- Selecting specific storage structure and access methods for the data to achieve good performance

---

# Application Developers

These are the software developers who develop the application programs or user interfaces the native and online users.
- These programmer must have the knowledge of Programming language such assembly C, C++, Java or SQL etc.
- Since the application progammer are written in these language.

---

# End-Users

End users are the clients of database who access the database for various reasons.

1) Native users:→ Native users do not know anything about the database system. They simply navigate through all the provided user interface to get the data.

2) Sophisticated users:→ The users who understand the capabilities of DBMS and users certain queries to retrieve the data.

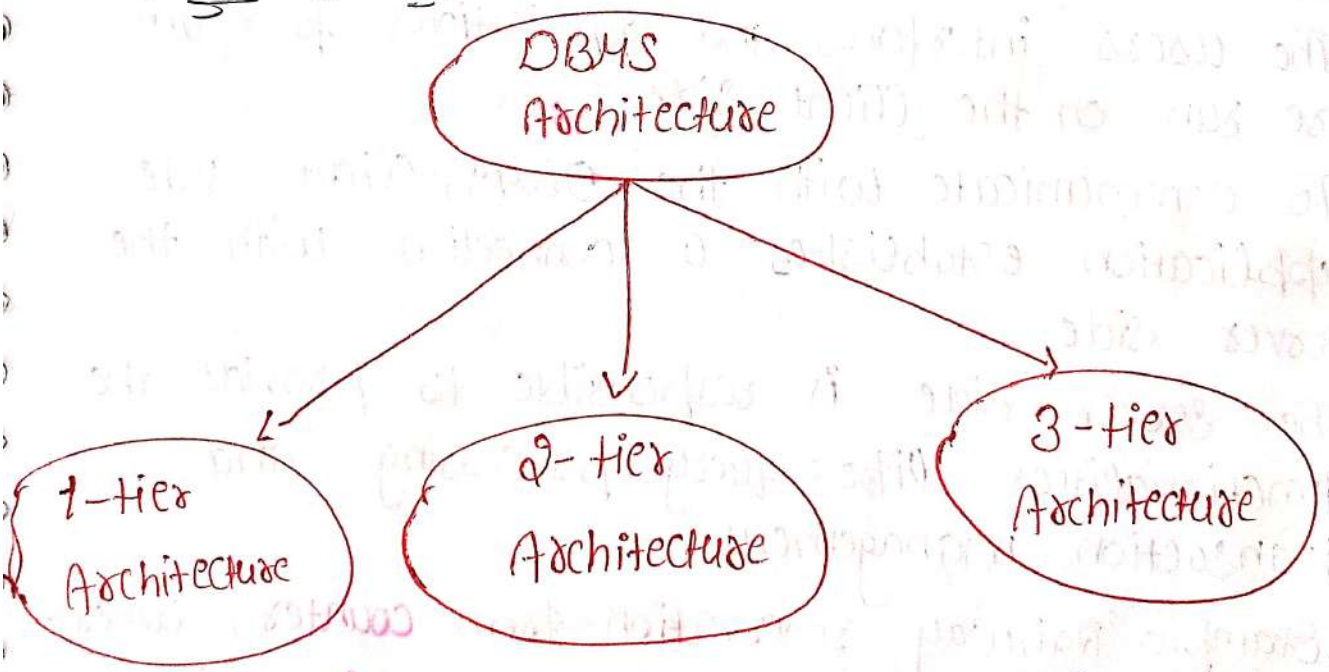
3) Specialized users:→ It write specialized application program for queries which can not be answered directly.

Unit-2

# Database System Architecture

- The design of a DBMS depends on its Architecture
- DBMS architecture helps in design, development, implementation, and maintenance of a database.
- A database stores critical information for a business. Selecting the correct Database Architecture helps in quick and secure access to this data.
- DBMS can be seen as either single tier or multi-tier
- An n tier architecture divides the whole system into related but independent Modules which can be independently, modified altered.
- DBMS Architecture depends upon how users are connected to the database to get their request done.

→ Types of DBMS Architecture




- But logically, database architecture is of two types like 2-tier and 3-tier Architecture

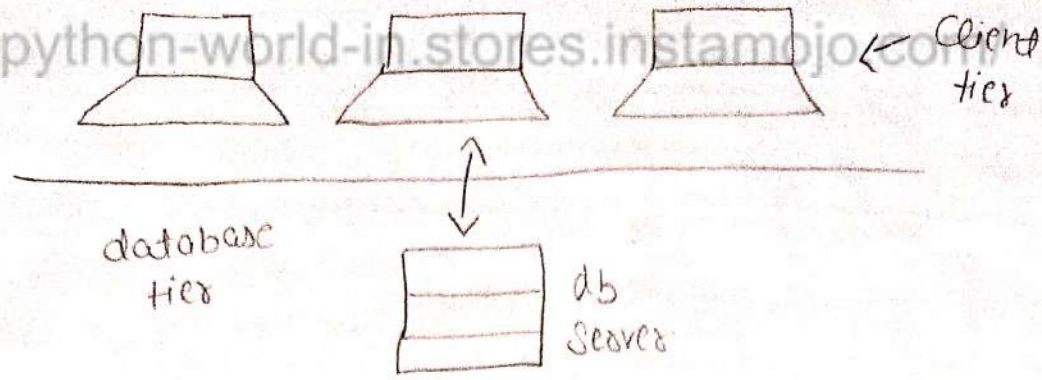
# 1-Tier Architecture

- In this architecture, the database is directly available to the user. It means the cusers can directly sit on the DBMS and uses it.
- Any changes done here will directly be done on the database itself. It doesn't provide a handly tool for end users
- 1-Tier Architecture is used for development of the local application where programmers can directly communicate with the database for the quick response.



# 2-Tier Architecture

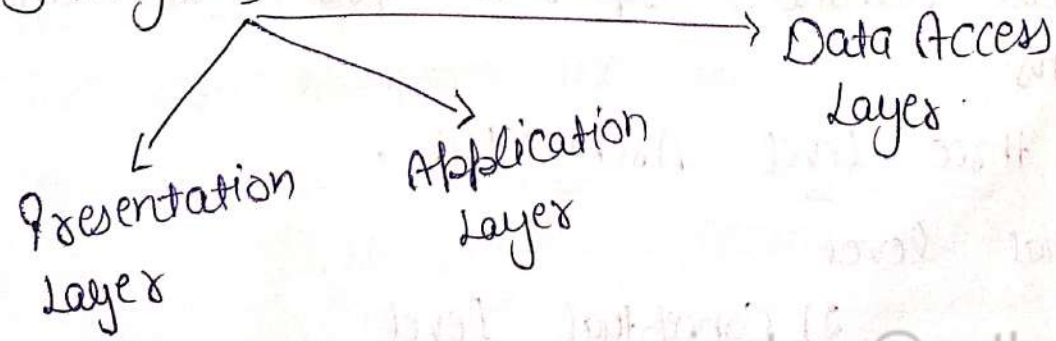
- It is same a Client-server.
- Direct communication with the database at the server side
- An application programming interfaces (APIs) like ODBC & JDBC are used by client side program to call the DBMS.
- The users interfaces and applications programs are run on the Client-side
- To communicate with the DBMS, Client-side application establishes a connection with the server side
- The server side is responsible to provide the functionalities like: query processing and transaction management.
- Example. Railway reservation from counter, where Clerk as a Client accesses the railway server directly.




- Advantages → Direct and faster communication.
  - Maintenance and understanding is easier.
- Disadvantages: Scalability
  - Less secure as client can access the server directly

# 3-tier architecture

- The 3-tier architecture contains another layer b/w the client and server. In this architecture, client can't directly communicate with the server.
- The application on the client-end interacts with an application server which further communicates with the database system.
- It is used in large web applications.
- Advantages: Enhanced Scalability.
  Data integrity.
  Security
- Disadvantage: Increased Completity.
- 3 Layers



# Three levels of Architecture

- The three schema architecture is also called ANSI architecture or three-level Architecture.
- This framework is used to describe the structure of a specific database system.
- The three schema architecture is also used to separate the user applications and physical database.
- The three schema architecture contains three-levels. It breaks the database down into three different categories.

Diag

## Why is 3 level architecture created?

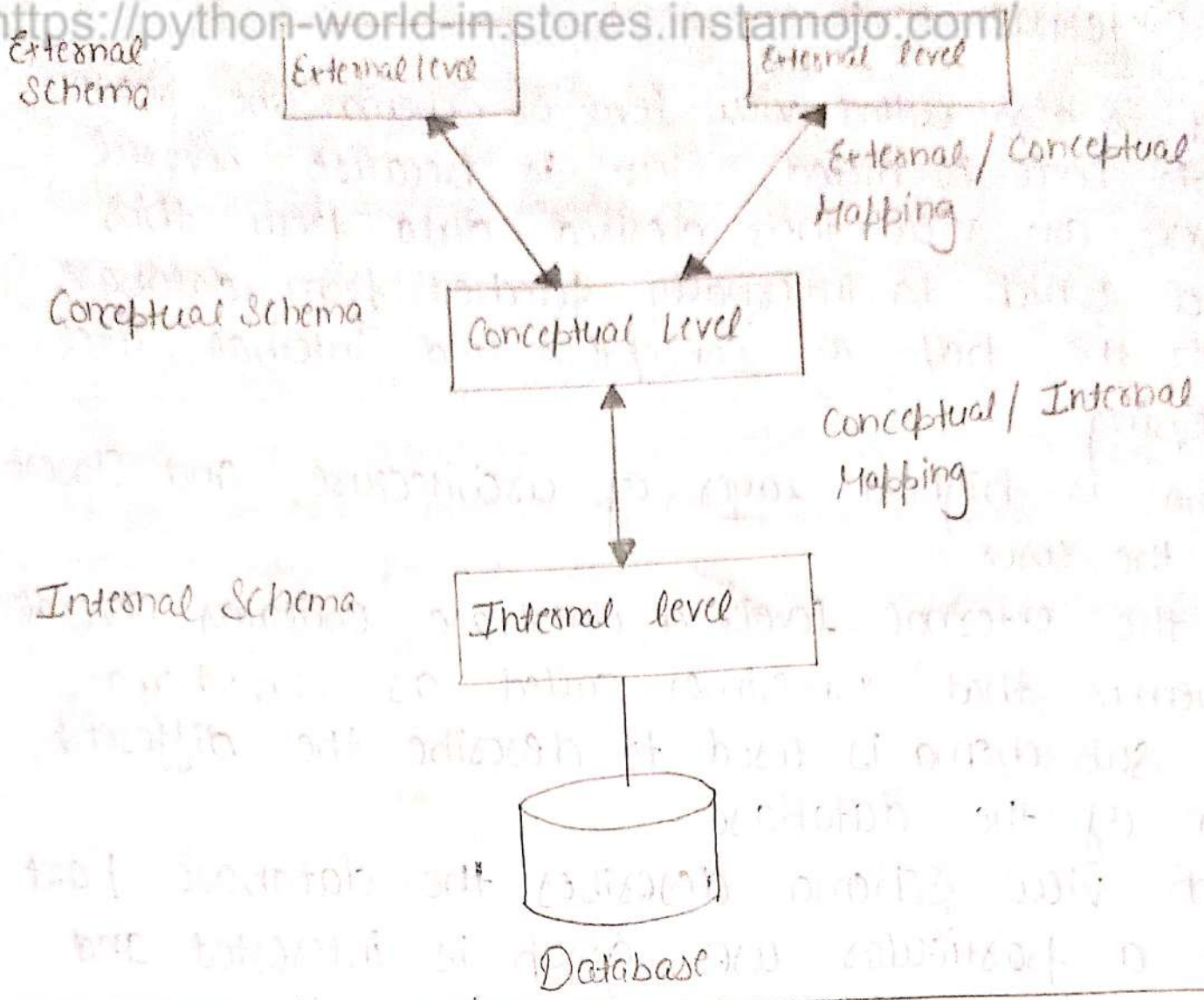
Three level architecture was created to keep the physical database separate from the user applications.

## Draw three-level Architecture.

1) External level

2) Conceptual level

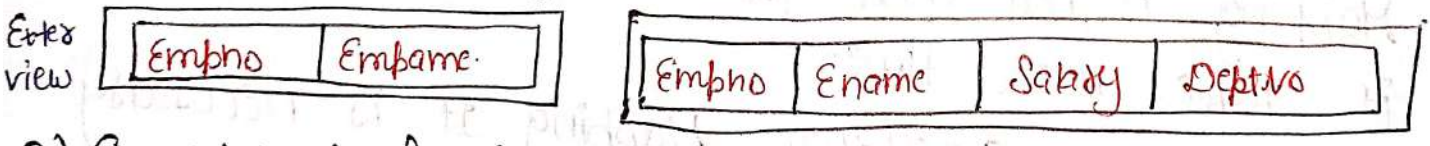
3) Internal level.




- It shows the DBMS Architecture
- Mapping is used to transform the request and response b/w various database levels of architecture
- Mapping is not good for small DBMS because it takes more time.
- In external / conceptual Mapping. It is necessary to transform the request from external level to conceptual Schema.
- In conceptual / Internal Mapping, DBMS transform the request from the conceptual to internal level.

## 1) External Level

- It is also called view level or Schema. The reason this level is called "view" is because several users can view their desired data from this level which is internally fetched from database with the help of conceptual and internal level. Mapping.
- This is highest layer of architecture and closest to the user
- At the external level, a database contains several schemas that sometimes called as subschema. The subschema is used to describe the different view of the database.
- Each view schema describes the database part that a particular user group is interested and hides the remaining database from that user group
- The view schema describes the end user interaction with database systems.




## 2) Conceptual level

- It is also known as Logical level.
- The conceptual schema describes the structure of the whole database.
- It is the middle level b/w external or internal levels
- The conceptual level is a higher level than the physical level.

- This level is Maintained by DBA.
- The conceptual level describes what data # to be stored in the database and also describes what relationship exists among those data.

## B. Internal level

- It is also known as physical level
- This level describes how the data actually stored in the storage device.
- This is the lowest level of the architecture.
- This level is also responsible for allocating space of the data

Internal view

| STORED_EMPLOYEE record. |
| --- |
| Empno: 4 decimal<br>Ename: String length.<br>Salary: 8.2<br>Deptno: 4 decimal. |

→ how the data will be stored in a block] uses.

---

## # Advantages of Three-tier Architecture

1) The Main objective of it is to provide data abstraction.
2) Same data can be accessed by different users with different customized views.
3) Physical storage structure can be changed without requiring changes in internal structure of the database as well as users view.
4) Conceptual structure of the database can be changed without affecting end users

## 1) Mapping b/w views

Mapping is the process of converting one level to another level. In this process, the data at one level is related to the data at another level. There are two levels of mapping

- The three levels of DBMS architecture don't exist independently of each other. There must be correspondence b/w the three levels. i.e how they actually correspond with each other.
- DBMS is responsible for correspondence b/w the three types of schema. This correspondence is called Mapping.

→ There are basically two types of Mapping

1) Conceptual / Internal Mapping
2) External / Conceptual Mapping

### 1) Conceptual / Internal Mapping

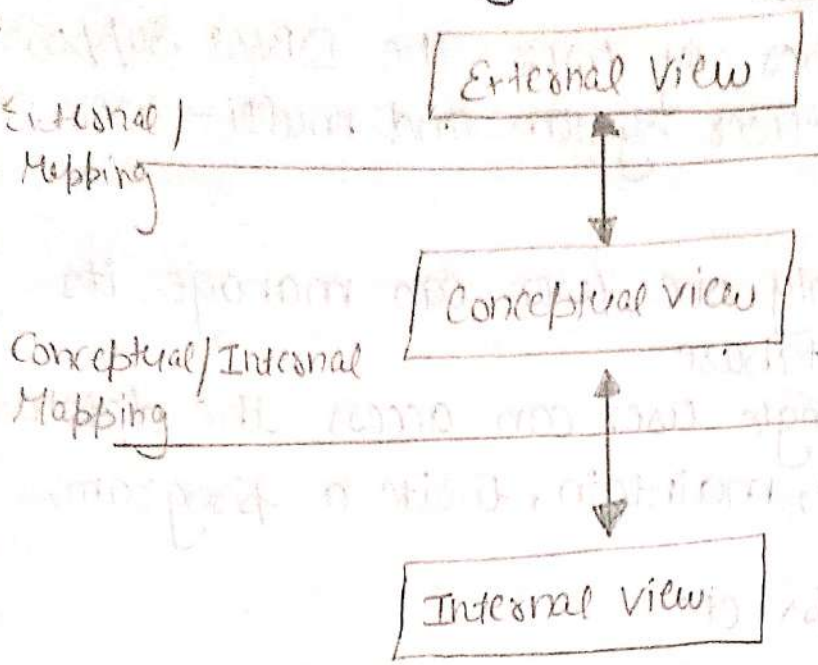
- The Conceptual / Internal Mapping lies b/w the conceptual level and the internal level.
- It specifies how conceptual record and fields are represented at the internal level.
- There could be one Mapping b/w conceptual and internal levels

### 2) External / Conceptual Mapping

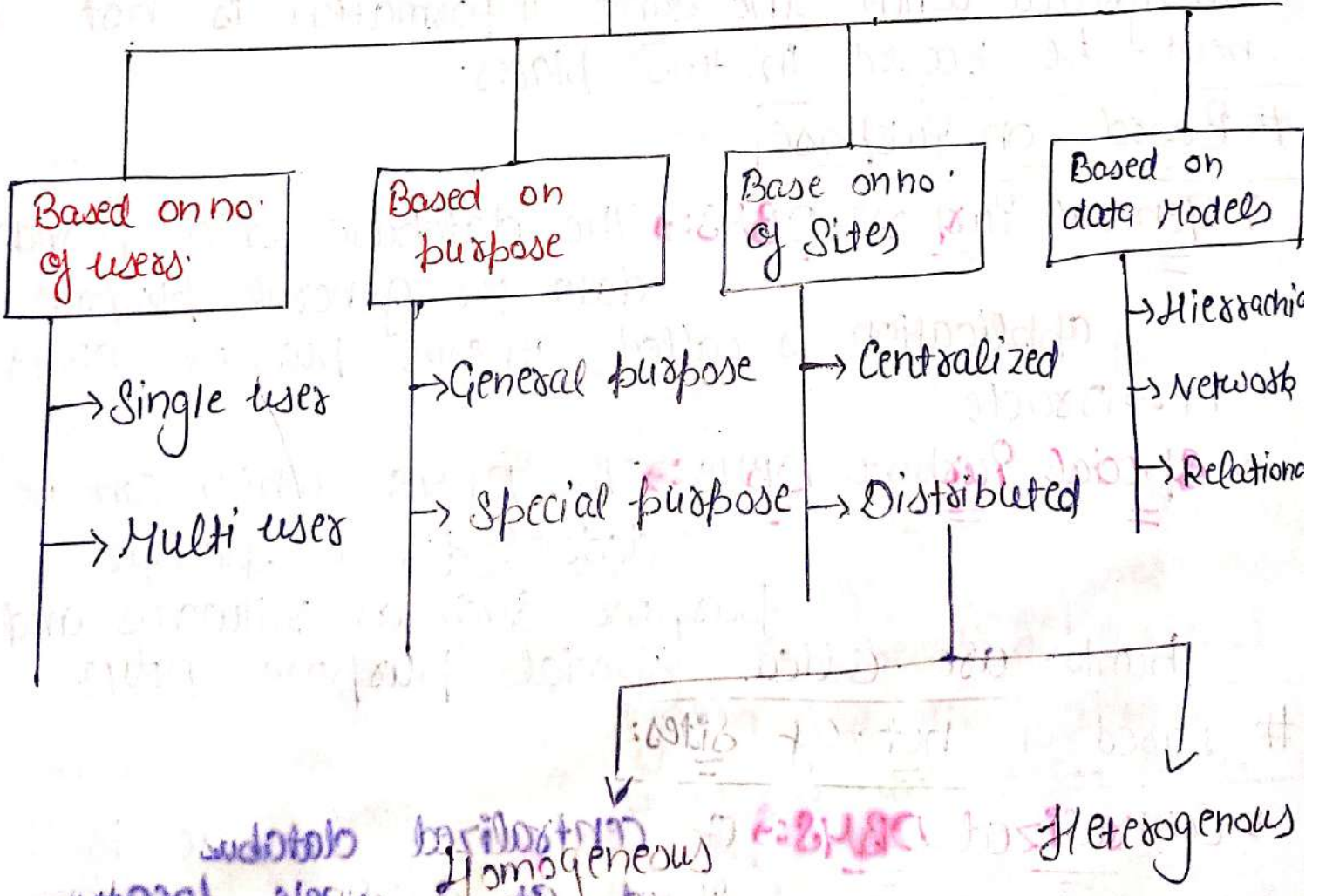
- The external / conceptual Mapping lies b/w the external level and the conceptual level.
- It specifies how the data is retrieved from conceptual level and shown at external level because at external level because at external level Some part of database is hidden from

fields are changed etc.






# Classification of DBMS



## # Based on no of users

Depending on the number of user the DBMS support, two categories, Single-user System and multi-user System.

→ Single-user System:- only one user can manage its database
- The Single user can access the database design, maintain, write a program, access

Ex:→ Laptops etc.

→ Multi-User System: It supports multiple users concurrently. Data can be both integrated and shared, a database should be integrated when the same information is not need be record in two places.

## # Based on Purpose

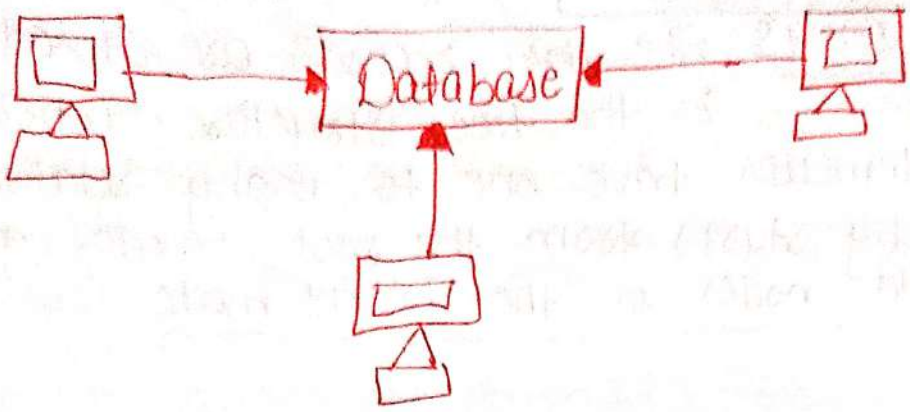
→ General Purpose DBMS:→ The database which provides data for general purpose application is called general purpose DBMS

Ex:→ Oracle

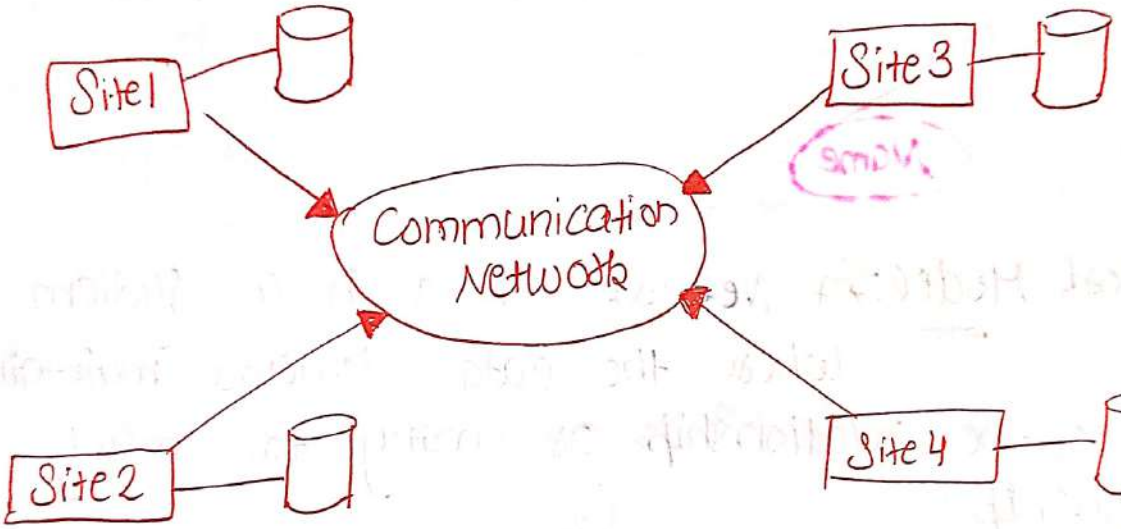
→ Special Purpose DBMS:→ The DBMS which can be decided for a specific purpose such as railways and banks are called special purpose DBMS

## # Based on not of Sites:

→ Centralized DBMS:→ centralized database is stored at a single Location and it is maintained and Modified from that Location only and usually accessed using an internet connection such as LAN and MAN




2) Distributed DBMS :→ A distributed database is basically database that is not limited to one system, it is spread over different sites.




- database that is not limited to one system it is spread over different sites

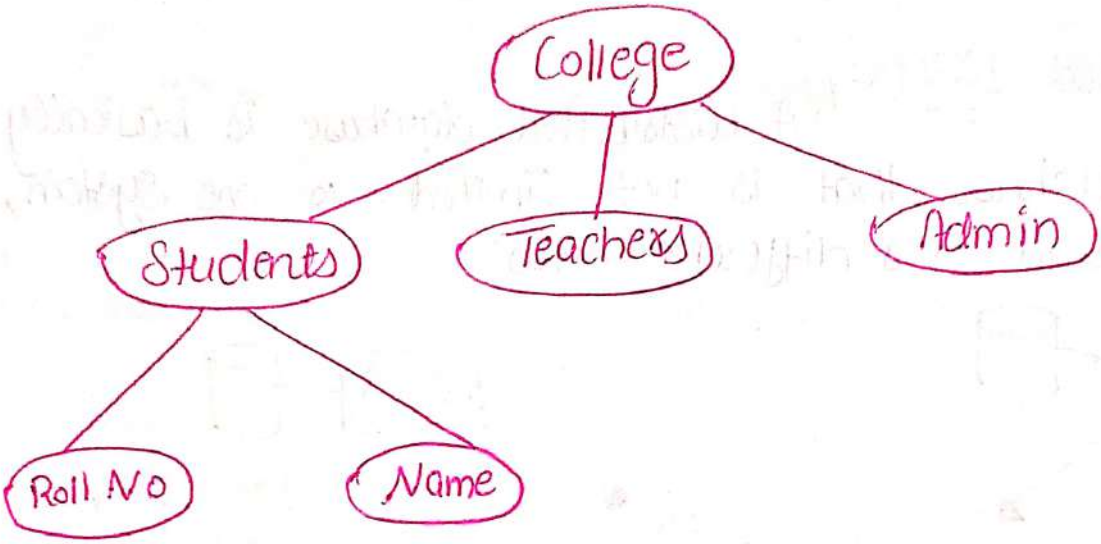
→ Homogeneous DBMS :→ In a homogeneous DBMS, all the different sites store database identically. The operating system, database management system and the data structures used - all same at all sites. Hence, they're easy to manage.

→ Heterogeneous DBMS :→ A heterogenous distributed database uses different operating system, DBMS and different data Models. Hence they're difficult to manage.

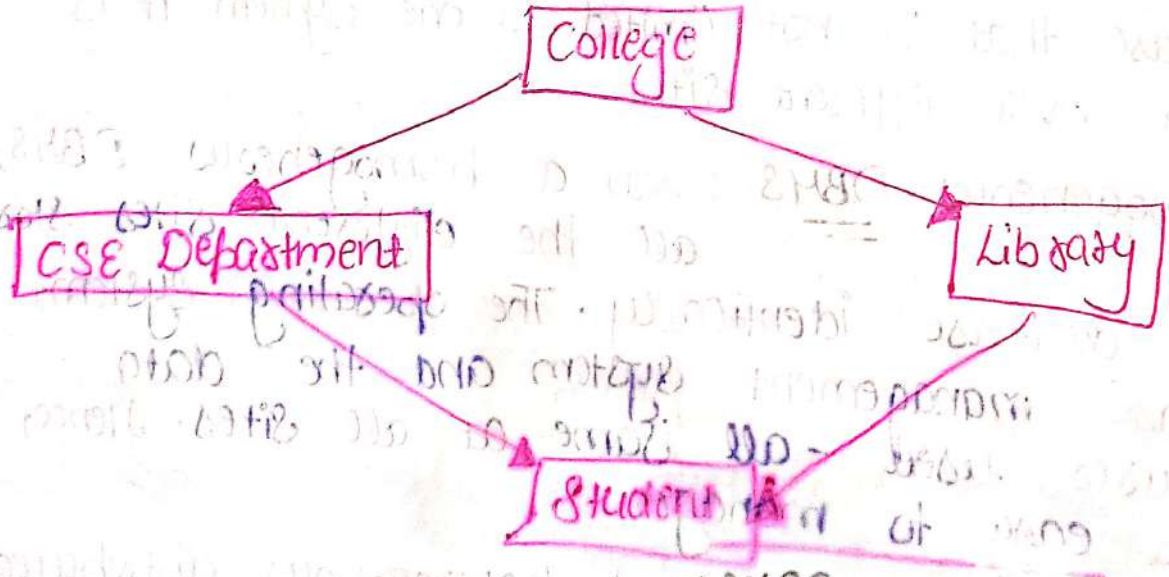
# Based on no. of Models:

→ Hierarchical DBMS :→ The data records are stored in the tree structure. Where the data elements have one to many relationship.

- The Hierarchy starts from the root node, connect all the child nodes to the parent node.




→ Network Model :→ Network DBMS is a system where the data elements maintain one to one relationship or many to many relationship.




→ Relational Model :→ Data stored in the form of rows and columns and together forms a table. A relational DBMS uses SQL for storing Manipulating as well as

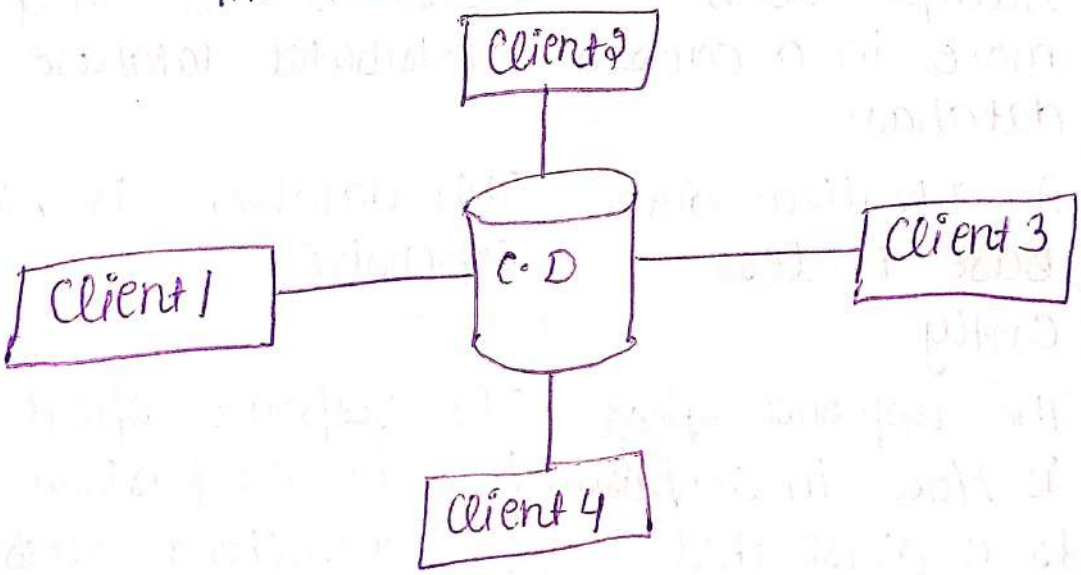
Maintaining the data. @python-world-in

Eg:→ MYSQL, oracle

# Centralized and Client Server Architecture

- Centralized DBMS

→ In a centralized database, there is a single database file at one location in the network.
→ Multiple users can access this single database
→ As there is an only single database file, it is easier to get a complete view of the data.
→ It is easier to Manage, update and takes backups of data
→ On the other hand, as there are multiple users accessing the same database file, it has a higher usage
- This will minimize the ~~productivity~~




→ Advantages of Centralized Database

1) It has decreased the risk of data management
2) It provides better data quality
3) It is less costly because

→ Disadvantages

1) The size of the centralized database is large, which increases the response time for fetching the data.
2) It is not easy to update such an extensive database system.
3) If any server failure occurs, entire data will be lost, which could be a huge loss.

| Comparison | Centralized Database | Distributed Database |
|---|---|---|
| Defination | It is a database that is stored, locate as well as maintained at a single location only. | It is a database that consists of multiple databases which are connected with each other |
| Access time | The data access time in the case of multiple users is more in a centraliz database. | The data access time in the case of multiple users is less in a distributed database. |
| Cost | A centralized data-base is less costly. | This database is very expensive. |
| Response Speed | The response speed is More in comparison to a distributed database. | The response speed is less in comparison to a centralized database |
| Ex: | A Desktop or Server CPU | Cluster point |

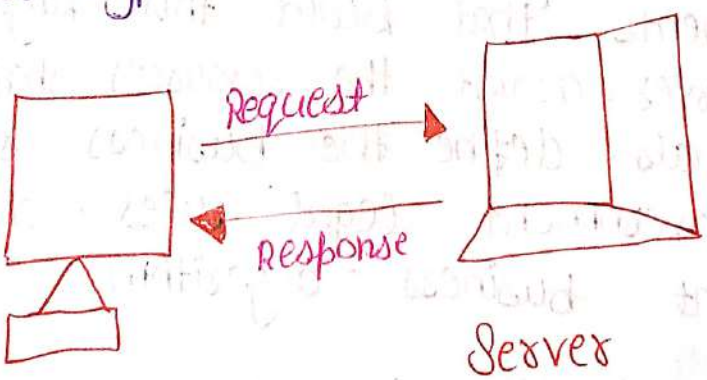
## ⇒ Client Server Architecture

- In Client-server architecture many clients connected with one server. The server is centeralines. It provides services to all clients. All clients request to the server for difficult service. The server displays the results according to the client's request
- Client/Server architecture is a computing model in which the server hosts, send and manages most of the resources and work to be required by the client
- In this type of architecture has one or more client computers attached to a central server over a network

→ By using this architecture structure this software is divided into three different tiers

1) Presentation tier
2) Logic tier
3) Data tier

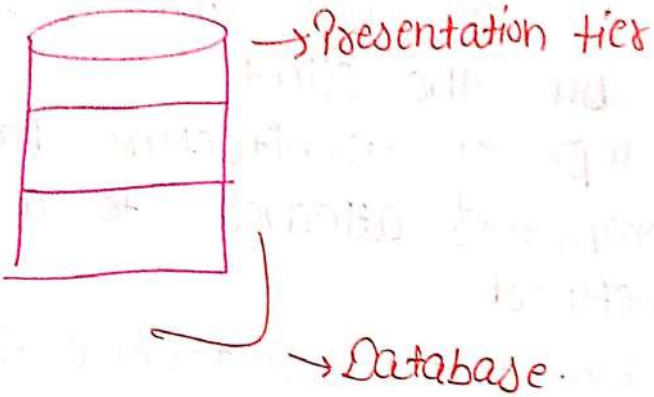
→ Each tier type builds and maintains independently




"The client-server architecture is one approach to communication b/w two software application that usually reside on physically distinct machine"

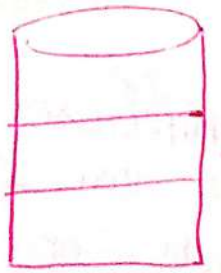
## 1) Presentation tier

- This is the first and topmost level of the applicatio n. The basic work of this layer provides user interface. the interface is a graphical uses interface.
- The graphical user interface is an interface that consists of menu, buttons and icons etc.
- The presentation tier presents information related to such events as browsing, sales purchasing and shopping cart contents.




## 2) Logic tier:

- The logical tier is also known as data access tier and middle tier. It lies b/w the presentation tier and the data tier
- It basically controls the application's function by performing processing.
- The components that build this layer exist on the server; assist the resources sharing these components also define the business rule like different government legal rules, data rules and different business algorithm.



3) 3-Data tier

- This is basically the DBMS layer. This layer consist of database it can be used through the business services layer in this layer, data is stored and retrieved.
- The responsibility of this layer to keep data consistent and independent.

"Client/Server architecture is also called as a networking computing Model and client-server network because all the requests and demand are sent over a network".

# Advantages of Client - Server

1) Management is easy
2) Easily accssible
3) Servers are Scalable
4) Centralized control
5) Security.

# Disadvantages

1) Less Robust
2) Requires Regular Maintenance.
3) Requires Cost [Hardware & Software very expensive.
4) Network congestion
5) Traffic is a big problem in this network

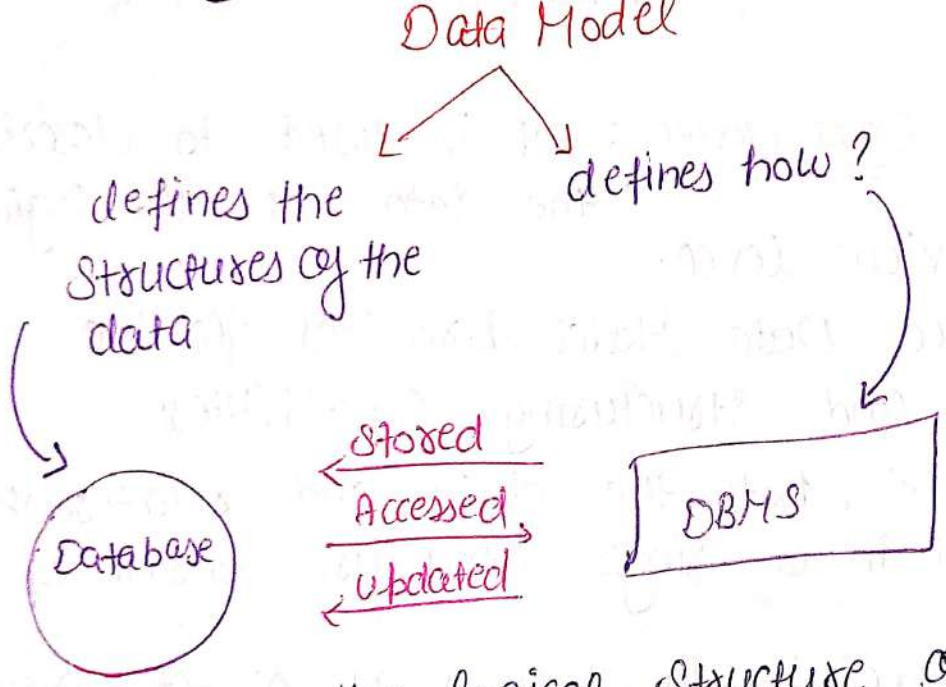
# Unit - 3



## # Data Models:

→ Data model provides the conceptual tools for describing the design of a database at each level of data abstraction.

- A Data Model can also be define as the collection of high-level data discription constructs that hide many low level storage Details

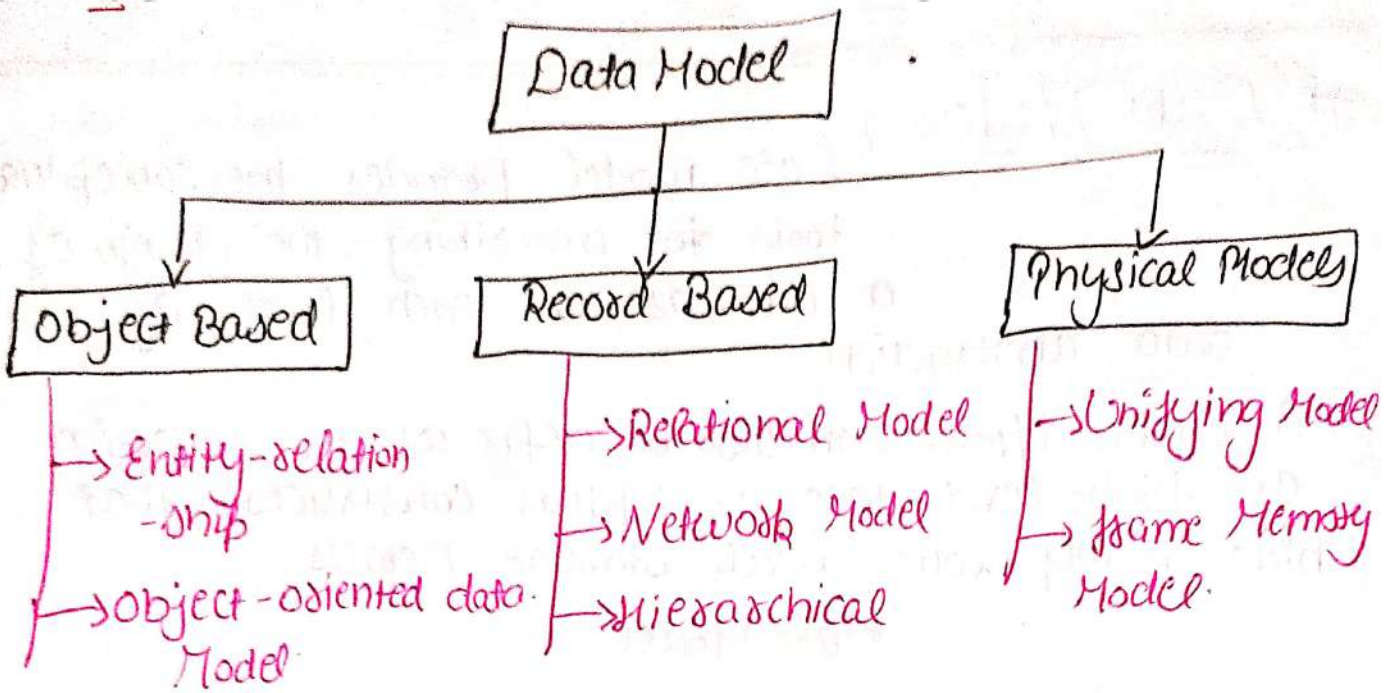



- Defines how the logical structure of a database is modelled.
- Data Models defines how data is connected to each other.

- Major components of DBMS

1) Structural part. [Set of rules]
2) Manipulative part [Defining types of operation]
3) Set of integrity rules. [Data is accurate]

# Three Data Models

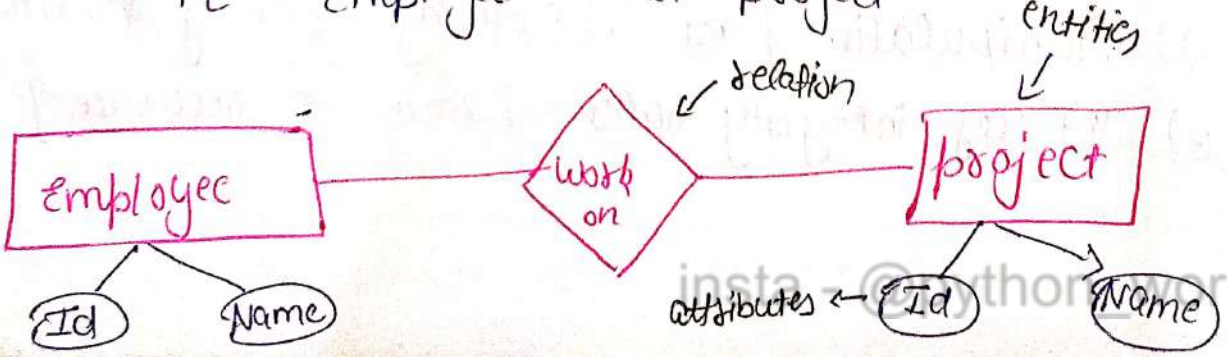



1) Object Based Data Model :- It is used to describe the data at the logical and view level.

- Object Based Data Model provides flexible structuring and structuring capabilities.
- In this Model, both the data and relationship are present in a single structure known as an object.
- There are mainly two types of object oriented

1) Entity relationship Model :-

- Collection of basic objects called entities and relationship.
- ER Model is a high-level data Model Diagram.
- Ex: Let say we have two entities i.e Employee and project.

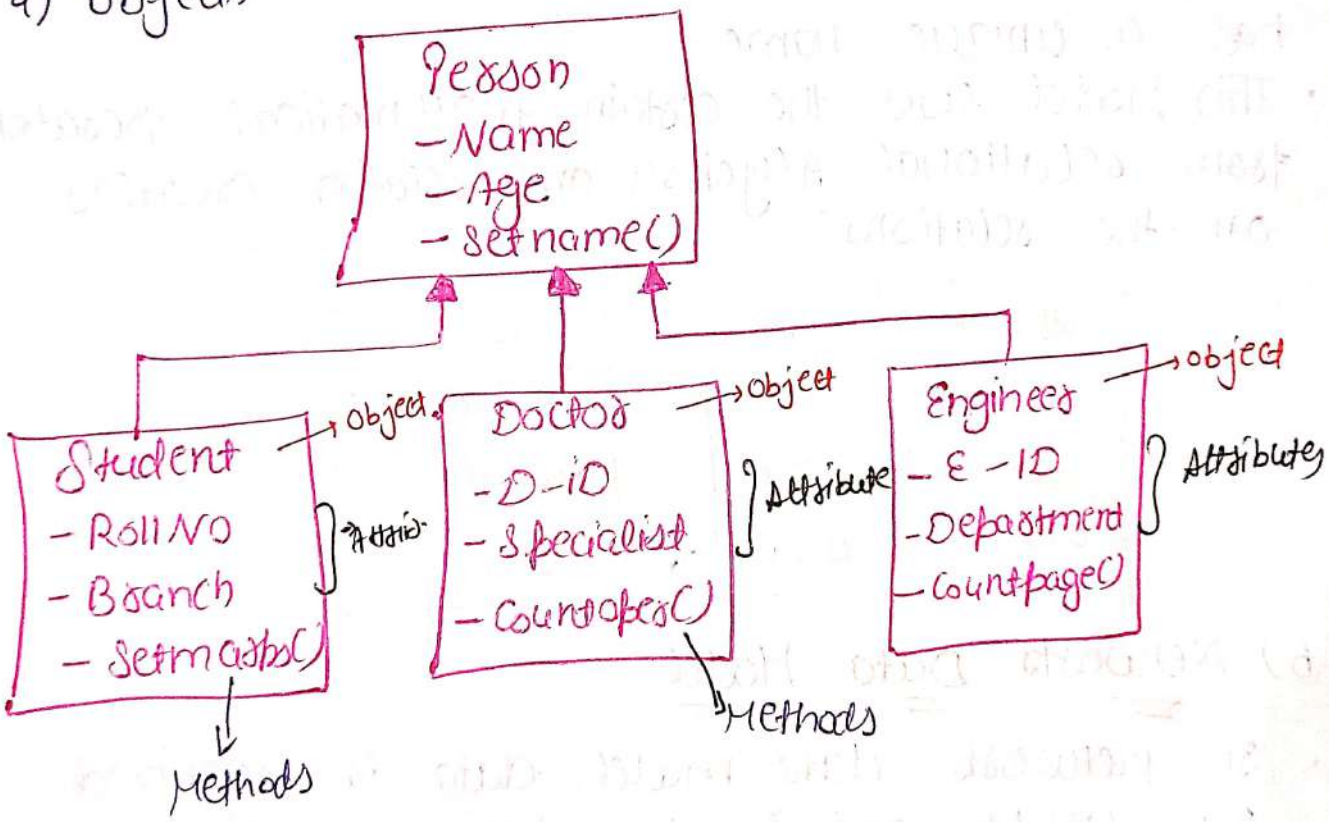


## 2) Object - oriented Data Model

- In an object oriented Model, information or data is displayed as an object and these objects store the value in the instance variable.
- This Model is used to represent real-word entities
- or data is store in the form of objects

→ Components of object oriented Data Model

1) Attributes and Method.
2) Class
3) Inheritance
4) objects




- Codes can be reused due to inheritance.
- Easily understanable.
- This Model works with object - oriented programming Language like - python, Java, VB.net and Perl etc.

Adv.
- Extensibility
- Improved performance
- Single - level Memory
- Easy to recover whole object

Dis.
- Lack of experience
- Lack of support of views
- Lack of support for security
- Lack of standards

## 2) Record Based Data Model

- It is used to discribe data at logical and view level.
- This data Model is used to specify the overall Logical structure and to specify the highes level structure and provide highes level description.
- These are described into 3 parts

a) Relational Data Model

- This types of Model designs the data in the form of rows and columns within a table.
- Each table has Multiple columns and each column has a unique name.
- This Model uses the certain mathmatical operations from relational Algebra and relation Calculas on the relations.

| Roll No | Name | Address |
|---|---|---|
| 01 | Tanu | Panipat |
| 02 | Anu | Karnal |
| 03 | Golu | Delhi |

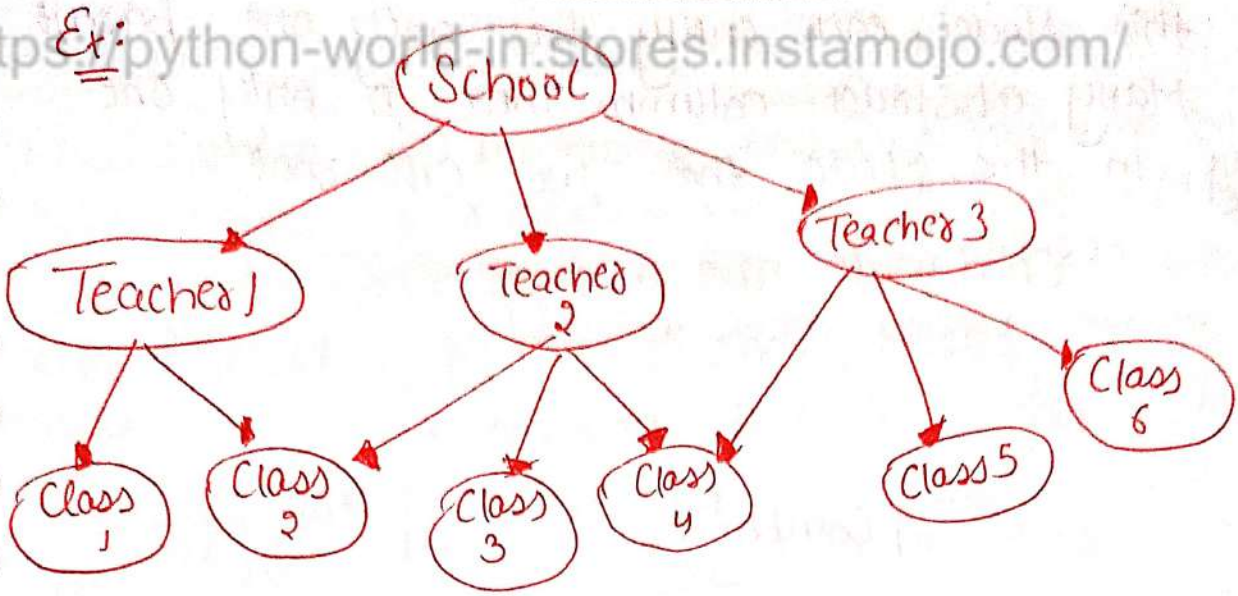
b) Network Data Model

In network data model, data is organized into graph and it can have more than one parent node.

- It permits the Modeling of ~~Many~~ One to many relationships in data.

"A Child node have multiple parent node" → Main difference

Ex:




→ features of Network Model

1) Ability to Merge relationship
2) Many paths
3) Circular linked list



→ Advantages

1) Simple concept
2) Ability to Manage More relationship Types.
3) Easy access of data
4) Data integrity
5) Data independence
6) Data access is flexible

→ Disadvantages

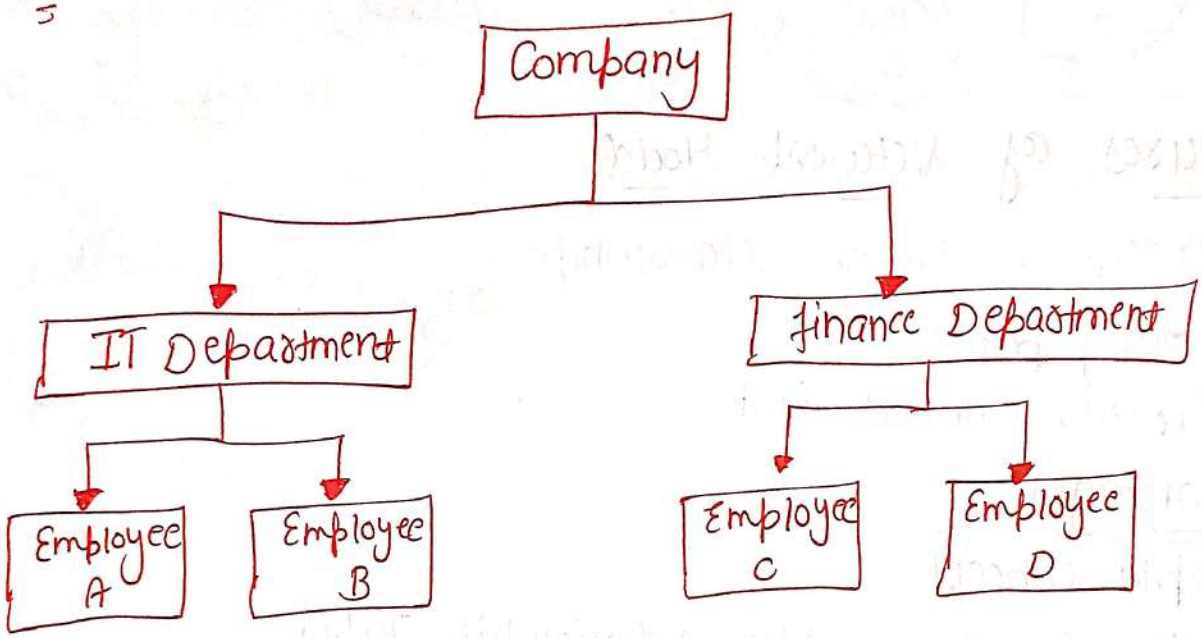
1) System Complexity
2) Operational Anomalies
3) Absence of structural independence

b) Hierarchical Data Model: → The Hierarchical Data Model organizes Data in a tree Structure.

- In this Model, each entity has only one parent and Many abstract column. These 0 only one. Enity in this Model that we call root

"Child node have a single parent Node".

Ex:-




→ Advantages

1) easy to understand
2) More efficient than the ER Model
3) Data integrity.

→ Disadvantages

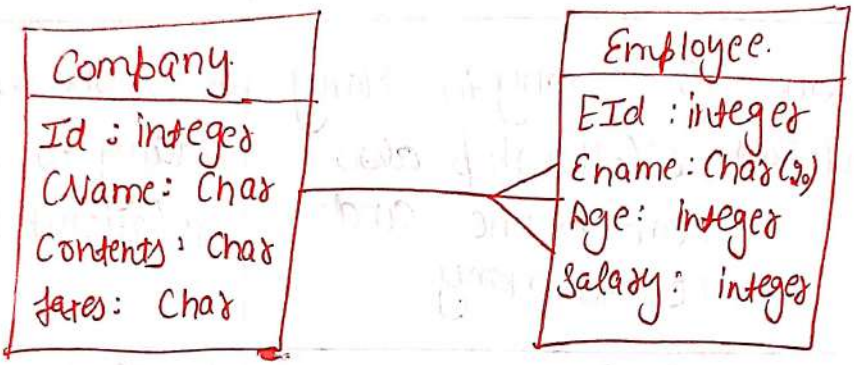
1) Complex implementation
2) Difficult to image
3) Lack of Standard
4) Poor flexibility

| # Hierarchical | Network | Relational. |
|---|---|---|
| 1. Relationship b/w records is of the parent child type. | 1) Relationship b/w records is express in the form of pointer or links. | 1) Relationship b/w records is represented by table. |
| 2. Arranges data in a tree structure. | 2. organizes data in a graph structure. | 2) Arranges data in tables. |
| 3. Represents "one to many relationsh | 3. "Many to Many relationship also one to one and one to many. | 3) " one to Many and Many to Many relationships. |
| 4. Difficult to access data | 4. Easier to access data | 4. Easier to access data. |
| 5. Less flexible | 5. flexible. | 5. flexible |
| 6 Delation anomaly exists in this Model. | 6. There is no delation & insertion anamaly | 6 There is no insertion & delation anomaly |
| 7. Pointer are used to establish relat among records | 7. A linked list is used to establish a relationship among record physically | 7. The logical representati is used with rows a columns to depict relationship among record |
| Ex:- AXML and XAML use this Model. | Ex: VAX- DBMS. | Ex: Mostly used in real world applicatio Oracle, SQL |

## 3. Physical Data Model

This data model is used to describe the data at low level.

- Low level data model or physical data model describes how the data is stored on the physical storage.
- Low level data model is not for common end users.
- Physical data model is for computer specialist.




- Hardware & Software dependent.
- gives data types information

---

## # Conceptual Modeling

- A conceptual model is the model of an application that the designers want users to understand.
- It is the process of developing an abstract model or graphical representation using real-world concept or idea.
- The 3 basic tenants of conceptual Data Model..
  1) Entity
  2) Attribute.
  3) Relationship

Ex

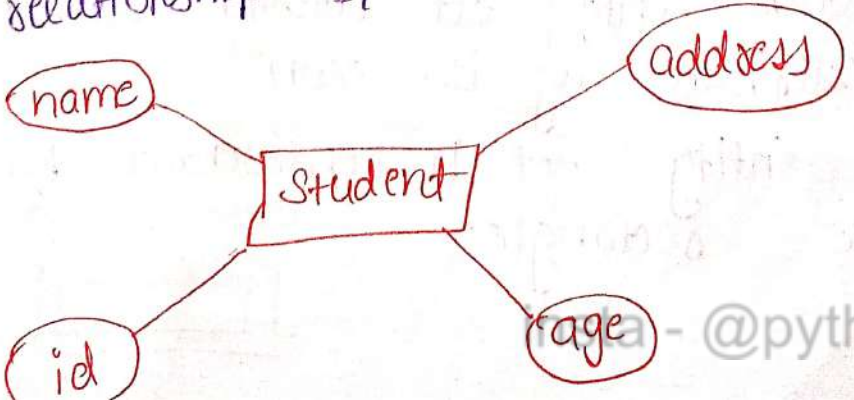

- Customer and product are two entities. number and names are attributes of the customer entity.
- Sales is the relationship b/w customer and product
- This type of Data Models are designed and developed for a business audience.

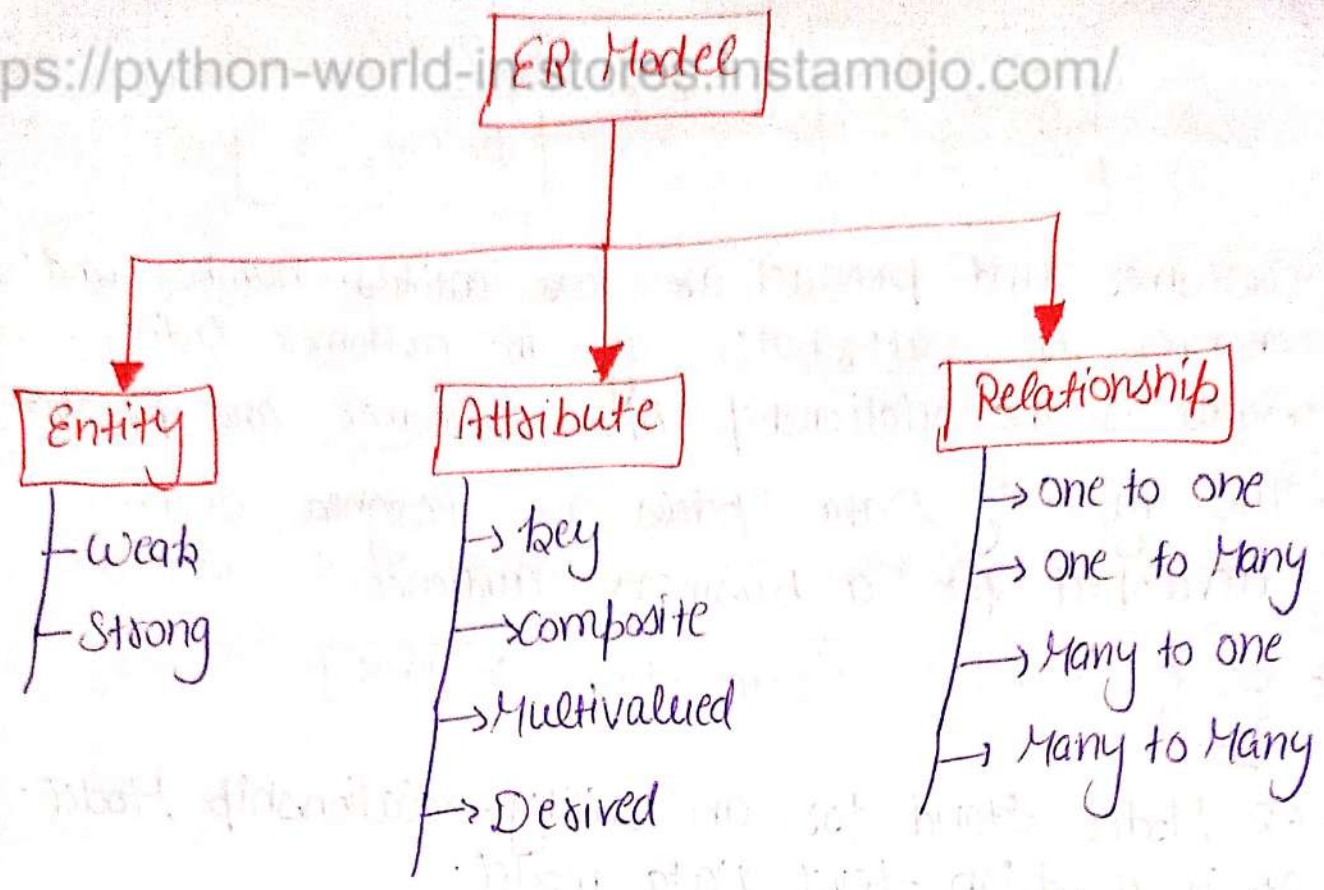
---

# E-R Model [Peter Chenn in 1976]

- ER Model stand for an Entity-relationship Model.
- It is a high-level Data model.
- This Model is used to define the data elements and relationship for a specified system.
- It develops a conceptual design for the database. It also develop a very simple and easy to design view of data.
- It is based on the notation of real-world entities and relationships among them.

Ex: Suppose we design a school database. In this database, the student will be an entity with attributes like address, name, id, age etc. The address can be another entity with attributes like city, name, pin code etc. and these will be a relationship b/w them.






1) Entity :→ It is a thing or object in the real world that is distinguishable from all other object.
- Anything about which we store information is called an Entity
- An entity can be represented as rectangles

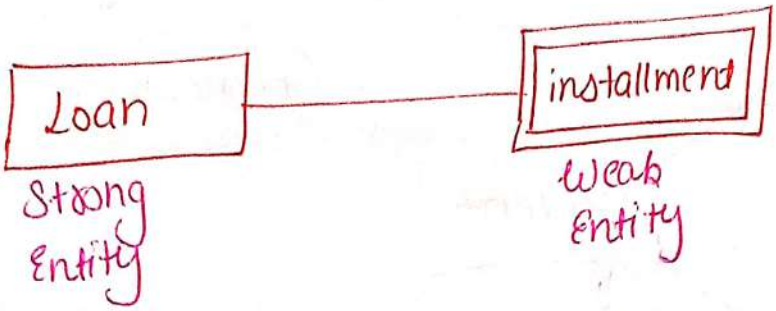
- These are classified into two types

1) Strong entity :→ An entity that depend on another entity called a Weak Entity Set.
- The weak entity set doesn't contain any key attribute of its own
- Weak Entity Set is represented by a double rectangle.

2) Strong Entity :- A strong entity not depends on any other entity to exist

- Single rectangle is used & uniquely identify all its entities, primary key of strong entity set

Eg:-

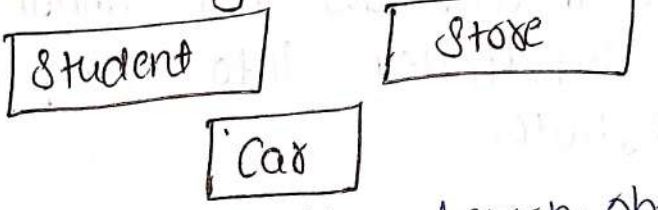



# Entity Set

It is a set of entities of the same type that share the same properties or attributes.

- Also entity set are represented as rectangles

Ex: Person: Student, employee.
Place: building, Store.




- An entity can be place, person, object etc

1) Tangible entities: The entities that physically exist in the real world.

Eg:- Entity of car, books etc.

2) Non-Tangible entities: The entities do not physically exist in the real world.

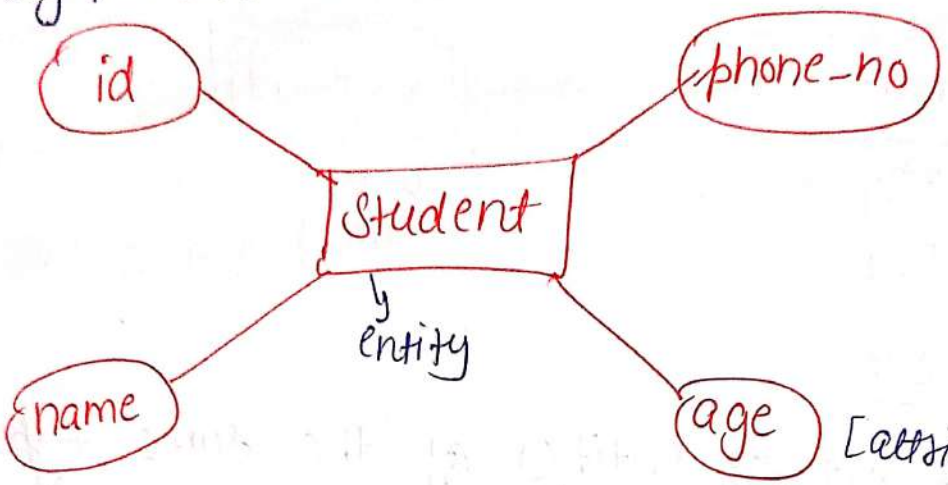
for ex: entity of email id's etc

## 2) Attributes

The attribute is used to describe the property of an entity.

- Represented by ellipse

Ex: id, age, contact no. can be attributes of a Student




[attributes]

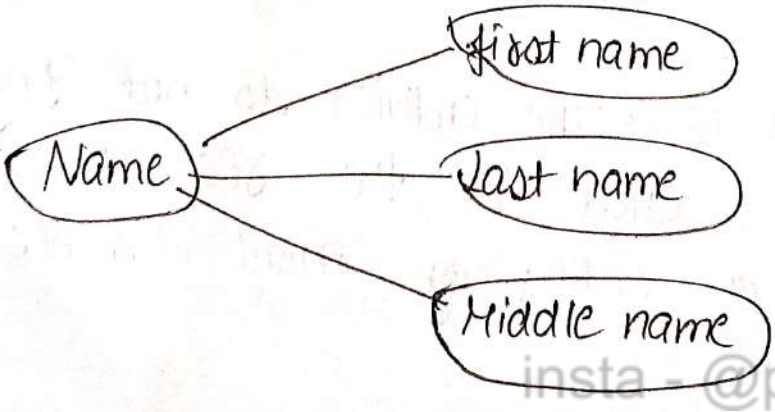
→ every attributes are directly connected to its entity

→ Types of attribute

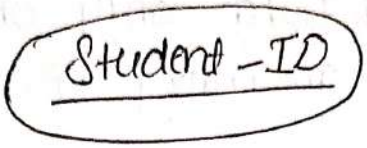
1) Simple attribute: An attributes that cannot be further subdivided into components in a simple attributes.

Ex:


2) Composite attribute: An attribute that can be split into components is a composite attribute.



3) Key Attribute :- The key attribute is used to represent the main characteristics of an entity.
- It represent a primary key.
- [ ____ ]
- Uniquely identify each entity



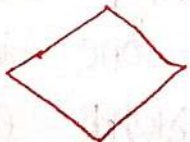

4) Multivalued attribute :- An attributes can have more than one value.
- The Double no. ellipse is represented.

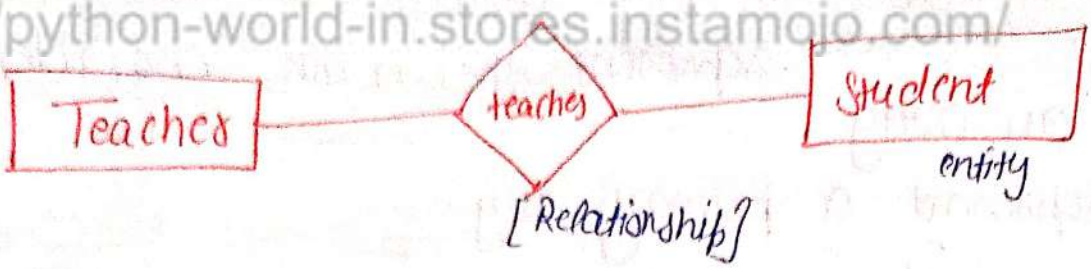

- A Student can have More than one phone number

5) Derived attribute :- An attribute that can be derived from other attribute.
- represented by dashed Ellipse.


- A person age changes over time and can be derived from another attribute like. date of birth

3) Relationship :- It is used to describes the relation b/w entities.
- Diamond Shape is used to represent

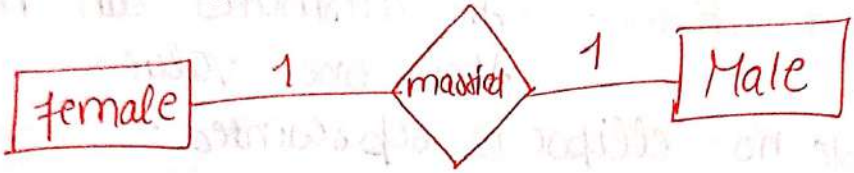





- These are four types

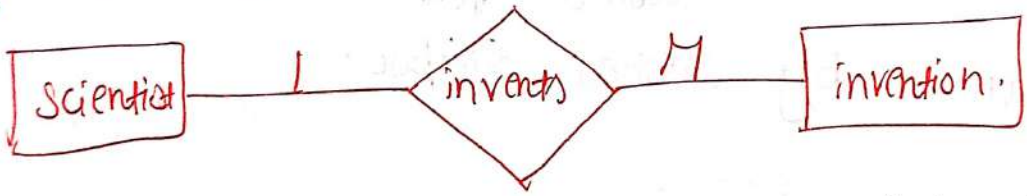
1) one to one relationship When only one instance of an entity is associated with the relationship.

Ex:-



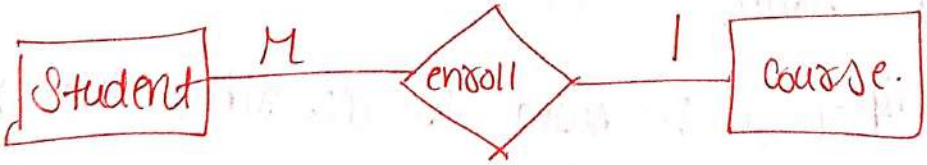

2) one to Many relationship: When only one instance of the entity on the left and More than one instance of an entity on the right associated with the relationsh

Eg:-




3) Many to one :- When More than one instance of the entity on the left and only one instance of an entity on the right associates with the relationship.

Ex:



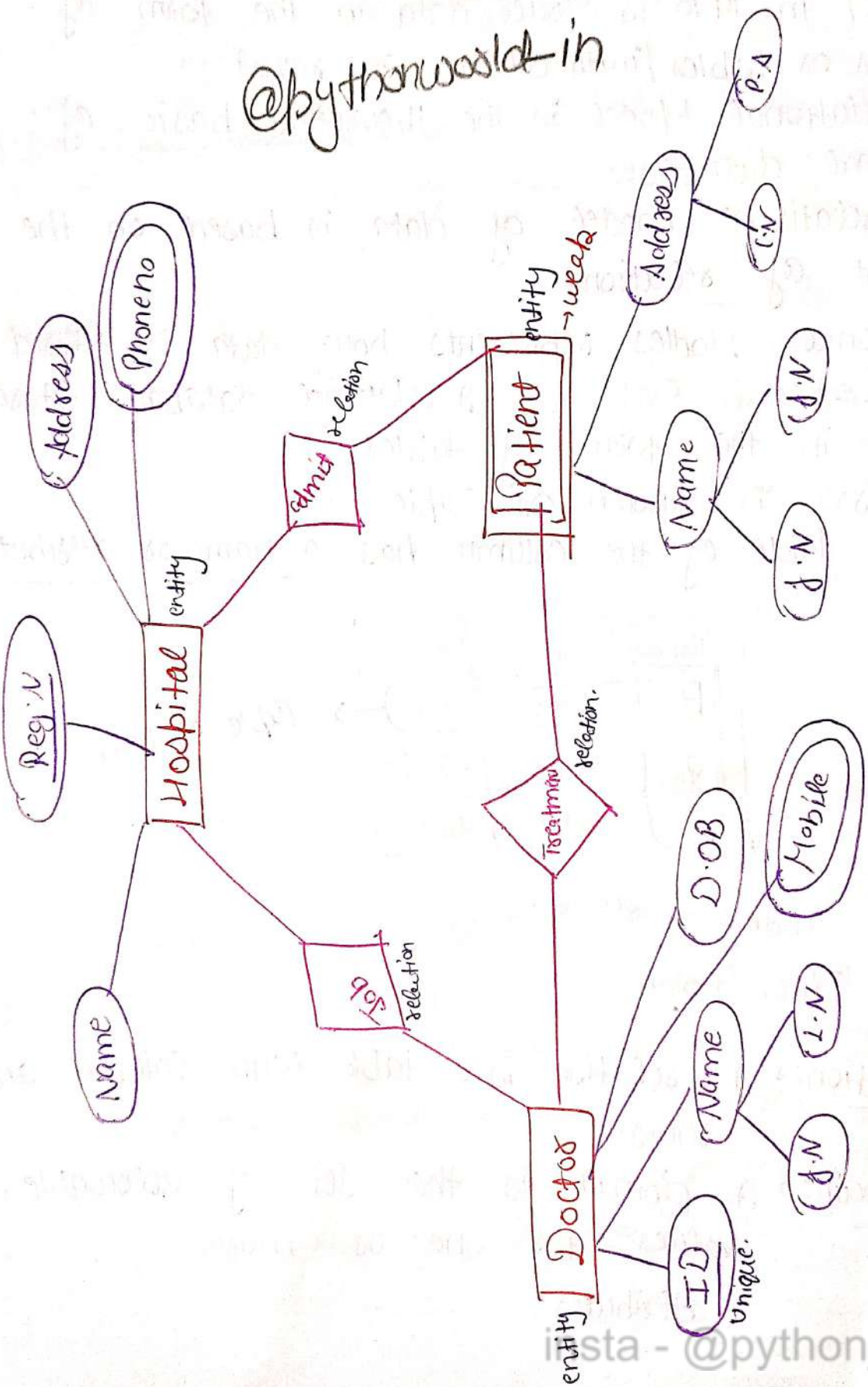

4) Many to Many :- When more than one instance of the entity on the left, and More than one instances of an entity on the right associates with the.

relationship.

Ex!:-


# Example → Degree → Unary Ternary
Binary N-ary

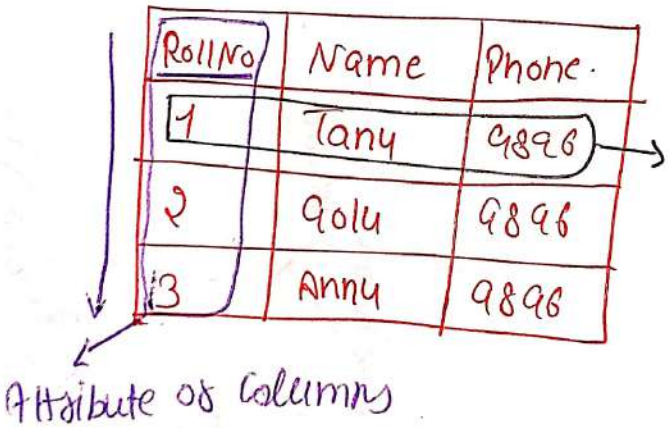




Unit-4

# Relational Model

- The relational Model was proposed by E.f Codd for IBM in 1970 to Model data in the form of relations or tables [with columns and rows]
- The relational Model is the theoretical basic of relational databases
- The relational model of data is based on the concept of relations.
- "Relational Modles represents how data is stored in relational Databases. A relational database stores data in the form of tables."
- Each row is known as Tuple.
- Each table of the column has a name or attribute.




# key Terminology

1) Relation:→ A relation is a table with column and rows.

2) Domain:→ A domain is the set of allowable values for one or more attributes.

3) Attribute: It contains the name of a column in a particular table. Each attribute is a domain value

4) Tuple: A typle is a row of a relation

5) Relation Schema: A relational Schema contains the name of the relation and name of all columns or attributes

6) Relation instance: Relation instance is a finite set of type. Relation instance never have duplicate typles

7) Relation key: Every row has one or multiple attributes, that can uniquely identify the row in the relation [Primary key]

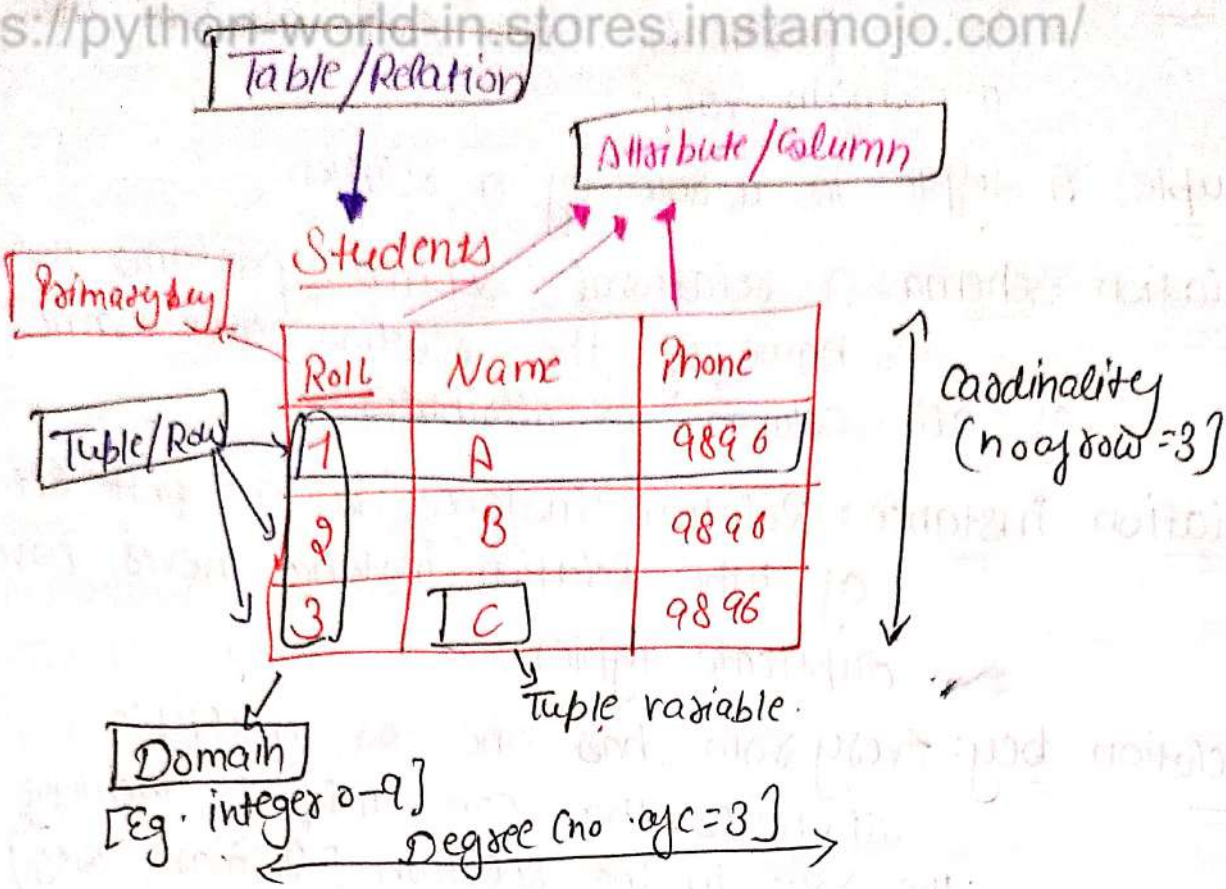
8) Degree: The total number of columns or attribute in the relation

9) Cardinality: Total number of rows present in the Table.

10) Tuple variable: It is the data stored in a record of the table

# Properties

1) Each relation has unique name.
2) Each tuple/row is unique. No duplicate row
3) Entries in any column have the same domain.
4) Each attribute /column has a unique name.
5) Order of the columns or row is irrelevent.
   i.e relations are unordered




Relation Schema :→ Students(Roll, Name, Phone)

---

# Keys

- A key is a value which can always be used to uniquely identify an object instance
- key is used to identify any record or row of data from the table.
- It also used to establish and identify relationships b/w tables.

Ex:→ Id is used as a key in the Student table because it is unique for each Student.

| Student |
| --- |
| Id |
| Name |
| Address |

# Types of keys

1) Super key → It is a combination of all possible attributes that can uniquely identify the rows in the given relation.

→ Super key is a superset of a candidate key.
→ A table can have many super keys.
→ A super key may have additional attribute that are not needed for unique identity.

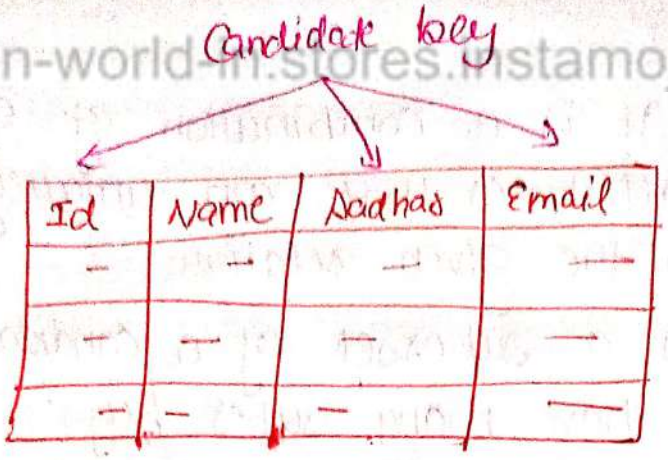
| Emp-Id | Name | Aadha-no | Email-id |
|---|---|---|---|
| 01 | A | 788976--- | aa@gmail.com |
| 02 | B | 88966--- | bb@gmail... |
| 03 | B | 67456--- | cc@gmail.com |

1){Emp-id} 2){Aadhar-no} 3){Email-id}
4){Emp-id, Aadha-no} etc.
→ But can not [Name] make a super key because one person name also related to same other per
5){Empid-Name}

2) Candidate key

- A candidate key is an attribute or set of an attribute which can uniquely identify a tuple.
- A candidate key is a Minimal Super key or a super key with no redundant attributes.
- Candidate key are not allowed to have NULL values.
- Candidate key are defined as distinct set of attributes from which primary key can be Selected.

Ex:




- { Emp-id }
  { Aadhar-no }
  { Email-id }

## 3) Primary key

→ A primary key is one of the candidate key chosen by the database designer to uniquely identify the tuple in the relation

- The value of primary key can never be NULL
- " " " " " must always be unique.
- " " " " " can never be changed.
- " " " " " must be assigned when inserting a record.
- A relation is allowed to have only one primary key

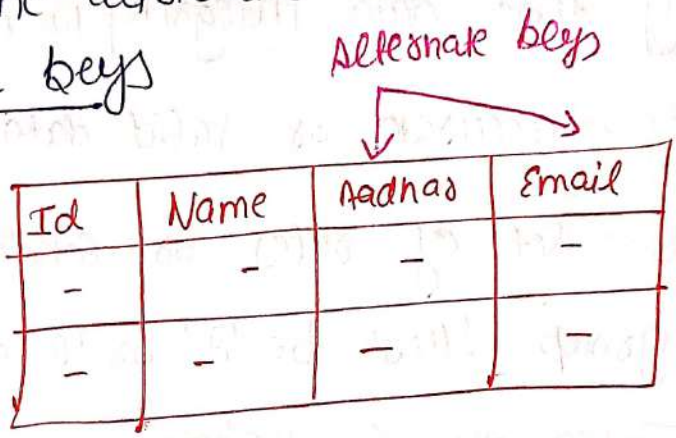
Primary key →

| Id | Name. | Aadhar |
|---|---|---|
| – | – | – |
| – | – | – |

- 1) { Emp-id } → Unique id.

## 4) Alternate keys

- Out of all candidate keys, only one gets selected as primary key, remaining keys are known as alternate keys
- Emp-id is best suited for the primary key
- Rest of the attributes like Aadhar-no etc as a alternate keys

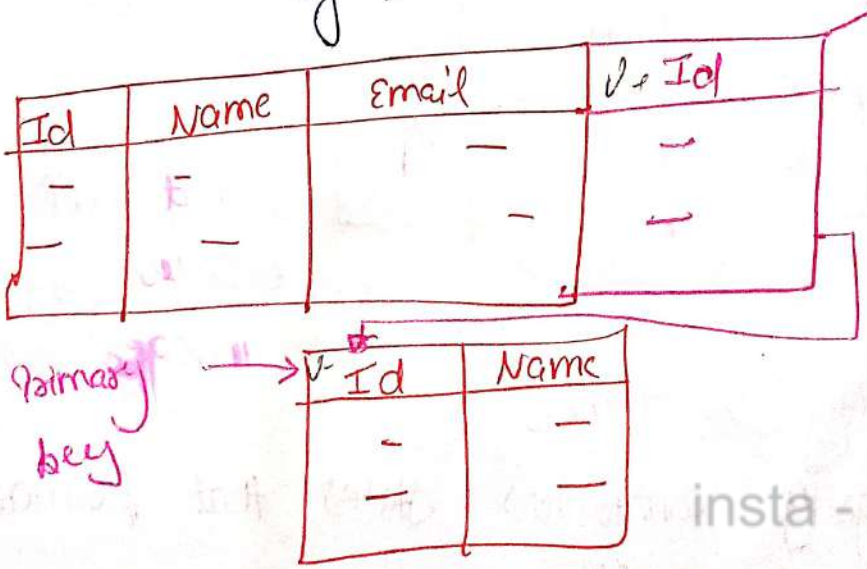



- 1) {Aadhar-no}
  2) {Email-id}

## 5) foreign key

A foreign key is :→ A key used to link two table together
- An attribute in one table that refers to the primary key in another table.

- purpose :→ to ensure (or maintain) referential integrity of the data.



# Integrity Constraints over relation

- Integrity constraints are a set of rules. It is used to maintain the quality of information
- Integrity Constraints ensure that the data insertion, updating and other process have to be performed in such a way that data integrity is not affected

Integrity → accurate or valid data are there.

Constraints → Set of rules or condition

Ex :- A blood group must be 'A' or 'B' or 'AB' only

# Types of Integrity Constraints

## 1) Domain Constraints

It can be define as valid set of value for an attributes.

- The datatype of Domain includes string, character, integer, time, Date etc. The value of the attribute must be available in the corresponding domain

| Student-id | Name | Semester | Age |
|---|---|---|---|
| 101 | A | 1st | 18 |
| 102 | B | 3rd | 19 |
| 103 | C | 7th | 17 |

→ Not allowed. because age is an integer value

## 2) Entity - integrity Constraints

The entity integrity constraints states that primary key value can't be null.

- This is because the primary key value is used to identify individual rows in relation and if the primary key has a null value, then we can't identify those rows.
- A table can contain a null value other than primary key field

Primary key →

| Emp-id | Emp-Name | Salary |
|---|---|---|
| 123 | A | 30000 |
| 345 | B | 4000 |
| NULL | C | 8000 |



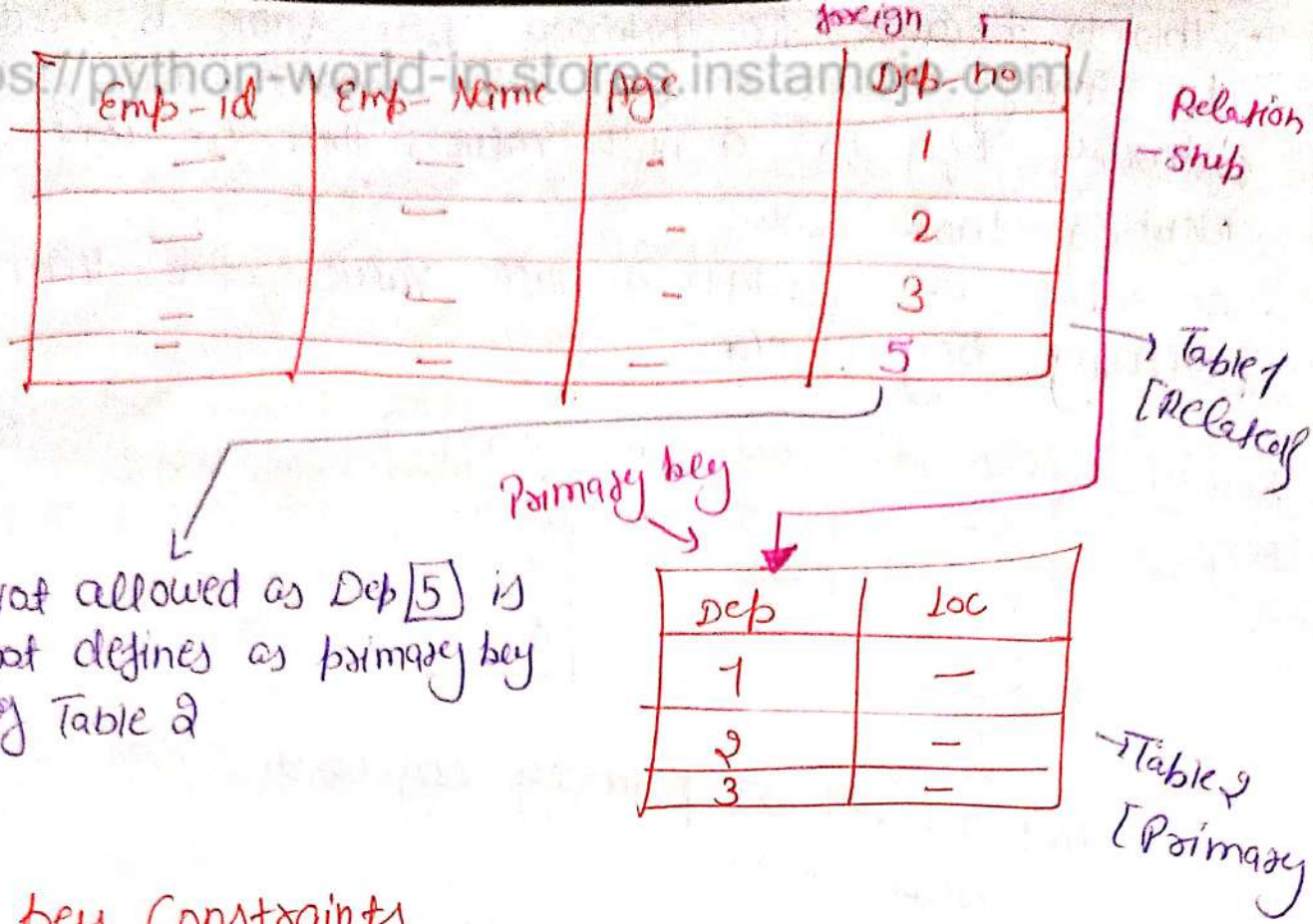
↓

Not allowed as primary key can't contain a NULL value.

## 3) Referential integrity constraints

- It is specified b/w two tables
- In the referential integrity constraints, if a foreign key in Table 1 refers to the primary key on Table 2, then every value of the foreign key in Table 1, must be null or be available in table 2.

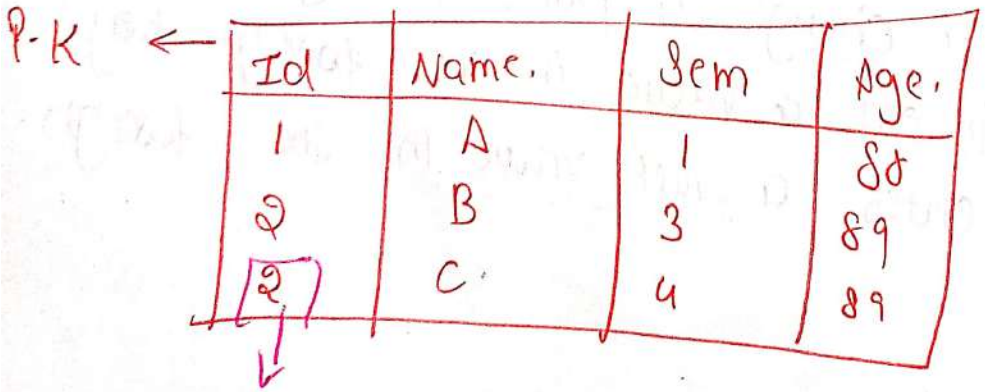
Rules
- You can't delete a record from a primary table
- You " " change a primary key value.
- " " " insert a value in the foreign key.
- " You can enter a Null value in the foreign key




## 4) Key Constraints

An entity set can have multiple key or candidate keys (minimal, Superkey), but out of which one key will be primary keys

- key constraint specifies that in any relation
  1) all the values of primary key must be unique.
  2) The value of primary key must not be null



# Views

- "Views" are Virtual relations, through which a selective portion of the data from one or more relations can be ~~tables~~ seen.
- Views do not contain data of their own.
- Views do not exist physically.

→ Uses of view

→ It helps in query processing like simplify command
→ to hide data complexity

- A view is defined using the create view statements

| create view v as <query expression> |
|---|

where <query expression> is any legal relational algebra / SQL query expression and v represents the view name

Ex: In SQL

```
Create view v as
Select column-list
from table-name [where condition];
```

[used as security mechanism]

→ Types of views

1) Read-only view

- Used only to read data
- In SQL, it allows only SELECT operation.

2) Updateable view

- Used to read and update the data.
- In SQL, it allow SELECT as well as INSERT, UPDATE and DELETE operations.

# Table

- Table is a collection of data which is organized in terms of rows and columns.
- Table is a simple form of data storage

Operation on Table

1) Create Table.
2) Drop Table.
3) Delete Table.
4) Rename Table

- We need three things to create a table
  - Table name
  - Columns / field name.
  - Definations.

→ CREATE TABLE [IF NOT EXISTS] table_name (
column-definition 1,
column_definition 2,
....,
table_constraints
);

| Table | View |
|---|---|
| 1) It is used to store the data. | 1) It is used to extract Data from the Table. |
| 2) It generates a fast result | 2) Slow result |
| 3) Independent data object | 3) Depend on the Table. Thus we ~~Can~~ can not create a view without using Tables. |
| 4) It occupies space on the System. | 4) It does not occupy Space on the System. |

# Unit-1

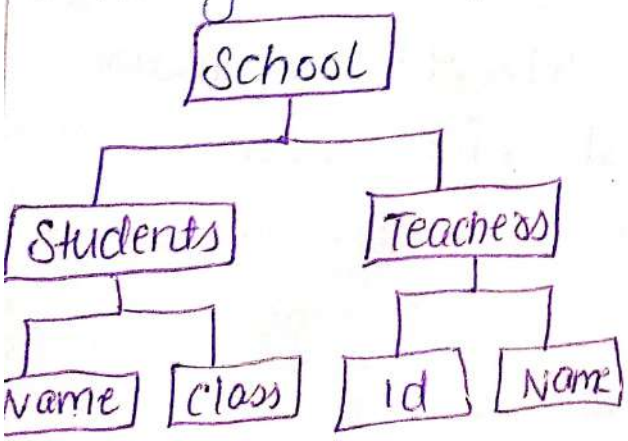
## #Difference between DBMS and RDBMS?

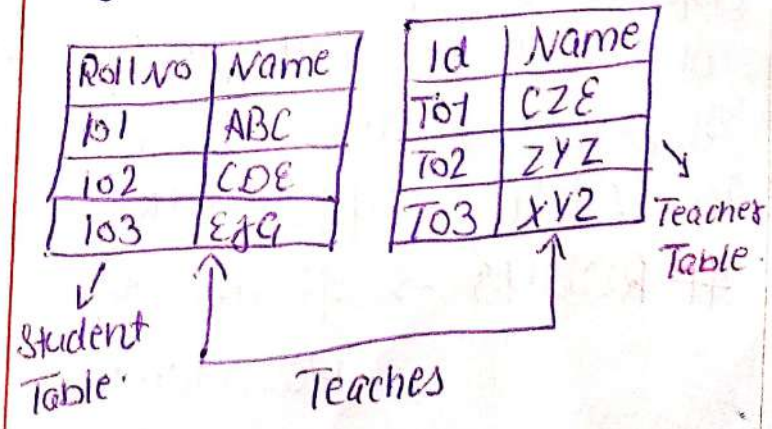
Ans Although DBMS and RDBMS both are used to store information in physical database but these are some remarkable difference b/w them.

**Database Management System:->** It is a Software that is used to define, create and maintain a database and provides controlled access to the data.

**Relational Database Management System:->** It is an advanced version of DBMS.

| DBMS | RDBMS |
|---|---|
| 1) DBMS stands for database Management System | 1) RDBMS stands for relational Database Management System |
| 2) Data is stored as file | 2) Data is stored as tables. |
| 3) Data is generally stored in either a hierarchical form or a navigational form. | 3) The data values are stored in the form of tables. |



| | |
|---|---|
| 4) Normalization is not present in DBMS. | 4) Normalization is present in RDBMS |
| 5) DBMS uses file System to store data, so there will be no relation b/w the tables. | 5) In RDBMS data values are stored in the form of tables, so a relationship b/w these data values. |
| 6) DBMS does not support distributed database. | 6) RDBMS supports distributed database. |
| 7) DBMS is design for small organization and deal with small data. It supports single users. | 7) RDBMS is designed to handle large amount of data. It supports Multiple users. |
| 8) DBMS needs minimum Software and hardware requirements. | 8) In RDBMS, hardware and Software requirement are highes than the classic DBMS. |
| 9) Examples: file System, xml etc. | 9) Examples: Mysql, postgre, sql server, Oracle. etc. |

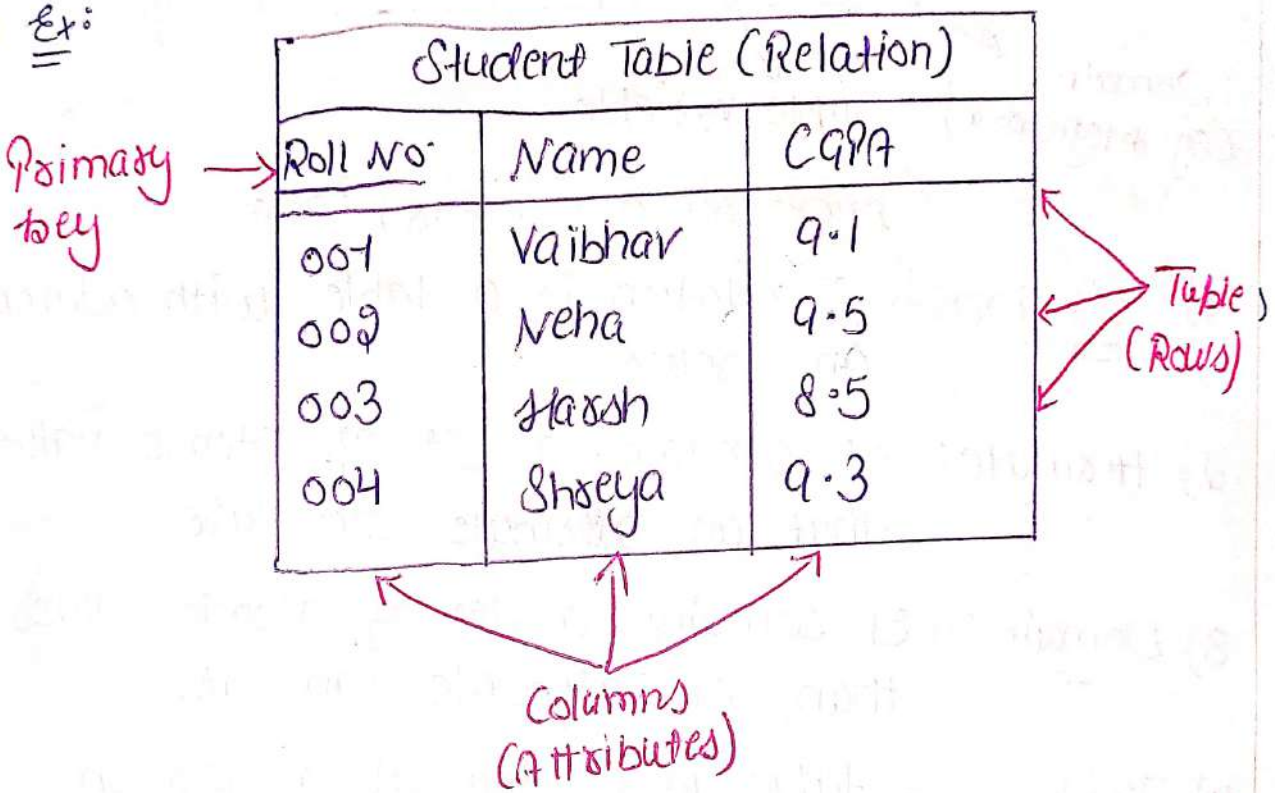
# after observing the diffference b/w DBMS and RDBMS, you can say that RDBMS is an extension of DBMS. There are many Software products in the market today who are compatible for both DBMS and RDBMS. Means today a RDBMS application is DBMS application and vice-versa.

# RDBMS → flexibility
Maintenance
Data Structure.

# Relational Models

- The relational Models was proposed by E.f Codd's for IBM in 1970 to Model data in the form of relations or tables
- Relational Model of data is based on the concept of relations.
- "The relational Models represents how data is stored in relational databases. A relational database stores data in the form of tables.

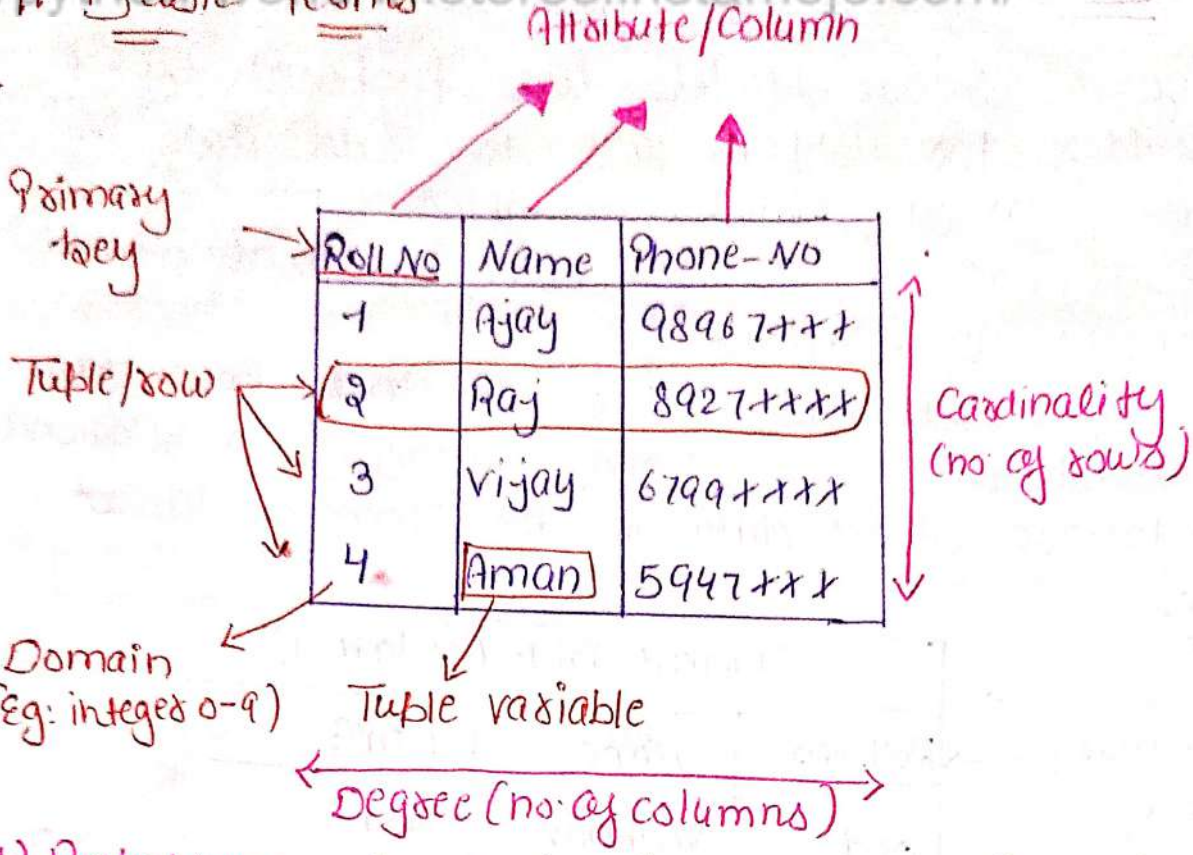
- Ex:




## => Objective of relational Model

1) To provide a high degree of data independence.
2) To provide a community view of the data.
3) To introduce a theoretical foundation.
4) To merge the fact retrieval and file Management.

# Basic terms




1) **Relation**:-> A relation is a table with columns and rows.

2) **Attribute**:-> It contains a set of atomic values that an attribute can take.

3) **Domain**:-> It contains a set of atomic values than an attribute can take.

4) **Tuple** :-> A tuple is a row of a relation.

5) **Degree** :-> The total number of column or attributes in the relation.

6) **Cardinality**:-> Total number of rows present in the table

7) **Relational Schema**:-> A relational Schema contains the name of the relation and name of all columns or attributes

8) **Relational instance**:-> It is a finite set of tuples. relation instances Never have duplicate tuples.

9) Relational key :-> In the relational key each row has one or more attributes.
It can identify the row in the relation uniquely!
Ex:-> Student relation

1) Candidate key
2) Super key
3) Composite key
4) Primary key
5) Alternate key
6) Foreign key.

# Advantages of relational Model

1) Easy to use :-> This is simple to use because information is stored in tables so if one is handling it for the first time.

2) Flexibility
3) Precision [No ambiguity in data]
4) Security

# Disadvantages of relational Model

1) Performance
2) Physical storage.
3) Complexity
4) Information loss

# Components of relational Model

1) Data Structure. [Tables contain row and columns]
2) Data Manipulation [SQL Languages are used to perform powerful operation]
3) Data integrity [It possible to specify certain rules]

# Codd's rules for relational Model

- These rules are developed by Dr. Edgar F. Codd in 1985
- Codd presents his 13 rules for a database to test the concept of DBMS against his relational Model and if a database follows the rule, it is called a true relational database (RDBMS) [that Manages Stores data]
- These 13 rules are popular in RDBMS known as CODD's 12 rules.

## 1) Rule 0: Foundation rule

- [ information are Stored in Table ] i.e row/colun
- [ Tables are connected/ Mapping ]

- According to this rule any database System Should have characteristics as relational, as a database, as a Management System to be are RDBMS. [The database Must be in relational form]
- These Should not be any independent table hanging in the database.

## 2) Rule 1: Information rule

- [ row / column order range but not affect the Meaning of the table ]
- [ It include only single data not Multiple data ]
- A database consists of lot of data - may be uses data and the data about these data or Metadata.
- Data Must be Stored in a table in the form of rows and columns.

- Information rules states that the order of rows and columns in the table should not affect the meaning of table.
- Each cell should have single data there should not be any group / range of values separated by comma, space or hyphen.
  Ex:- 2 Phone No not exist in one cell.

## 3) Rule 2: Guaranteed Access Rule

- This rules states to the primary key refers.
- It states that any data / column / attribute in the table should be able logically accessed by using the primary key table.
- Column / attribute
  Table Name
  Primary key

  ⟷ when combination of these 3 is used, it should give the correct result.

- Any column / cell value should not be directly accessed without specifying the table and primary key.

Ex: Table Name → Student
RollNo → 820 of city ] find

Select city from Student where RollNo
↓ ↓ ↓
Column Table Name Primary key

Student Table.

| No. | Name | City | Roll |
|---|---|---|---|
| 1 | | | |
| 2 | | | 820 → find. |
| 3 | | | |

## 4) Rule 3: Systematic Treatment of NULL

[*** If any value miss during entry then value not make NULL, blank space leave ]

- otherwise data not give correct.
- This rules states about handling the NULL value in the database.
- According to this if any of the cell value is unknown or not applicable or missing, it cannot be represent as zero or empty.
- It will be always represented as NULL.
- This NULL should be acting irrespective of the datatype used for the cell.
- When used in logical or arithmetical operation, it should result the value correctly.

Ex:→

5 + unknown

5 + NULL = NULL [✓]

5 + NULL ! = 5 or 0 [✗]

- It should not result in any zero or numeric value.

## 5) Rule 4: Active Online Catalog

[- This rules illustrates data dictionary.

- Metadata should be maintained for all the data in the database.
- These Metadata should also be stored as tables [rows and columns]
- In short, Metadata stored in the data dictionary should obey all the characteristics of a database.

- Users can use the same query language to access the catalog which they use to access the database itself.

6) Rule 5: Comprehensive Data Sub language Rule

- According to this rule RDBMS database should not be directly accessed.
- It should always be assist be using some strong query language.
- This query language should be able to assessed the data Manipulate the data and Maintain the consistency and integrity of the database.

Ex:- किसी भी query language की help से इस database को Accessed कर सकते है।

Database + query language = All organisation formed.
Have Uses

- Any database without any query language is not a RDBMS. Database can be accessed by using query language directly.

7) Rule 6: View updating Rule (partical possible Not)

- Views are the virtual tables created by using queries to show the partial view of the table.
- All view that are theoretically updatable should be updatable by the system.
- It is only partial table with few rows and columns. [If views are permission to update table then database integrity destroy]

· This rules states that views are also be able to get updated as we do with its table.

## Rule 7: High-level insert, update and delete

[ if any change in a system, apply all command in a one time i.e all query works one one time] [Eg:- bonous given in one time to all deparment employee]

· This rule states that every query language used by the database should support INSERT, DELETE and UPDATE on the records.
· It should also support set operations like UNION, UNION ALL, MINUS.
· All these operation should not be restricted to single table or row at a time. It should be able to handle Multiple tables and rows in its operation.

## 9) Rule 8: Physical data independence

· If there is any change in the physical storage of the data, it should not affect the data at the logical or external view.

Ex:- 20mb to 70mb data shift, then data is increase but logical view not affect the application [Not known user]

· Data shift one disk to another disk.

## 10) Rule 9: Logical Data indepence [This rule is practically not possible]

· According to this rule, if there is any changes in logical view its should not be reflected in the user views.

· 10 field → 2 field separate. but this working not known user.

- This rule is practically not possible because logical views and users views are strongly connected as much as the same.

## Rule 10: Integrity Independence

- According to this rule data should be able to apply integrity rule by using query language.

[Integrity -> Correctness]

- It should not be dependent on any external factor or application to Maintain the integrity.

Ex:

| No | Name | Course |
|---|---|---|
| 1 | A | BCA |
| 2 | C | MCA |
| 3 | D | BBA |

Base Table.

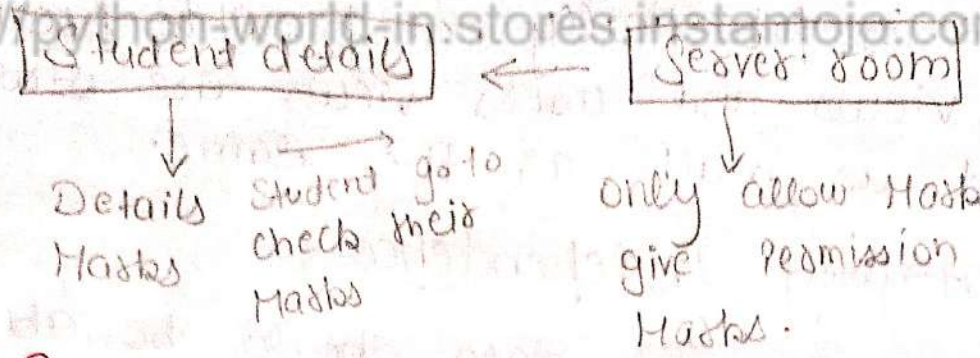
| No | Name | Course |
|---|---|---|
| 1 | A | BCA |
| 2 | C | MCA |
| 3 | D | BBA |
| 4 | E | B-TECH |

Reference Table.

B-Teach Not in base Table but they loose the integrity of Base Table.

## 12) Rule 11: Distribution Independence

- This database can be located at the user server or at any other Network. The end user should not able to know about the database servers.
- He should be able to get the records as if he is pulling the records locally [retrieve data]
- Even if the database is located in different servers.




## 13) Rule 12:- Non-Subversion Rule

- When a query is fired in the database, it will be converted into the low level language so that it can be understood by the underlying system to retrieve the data.

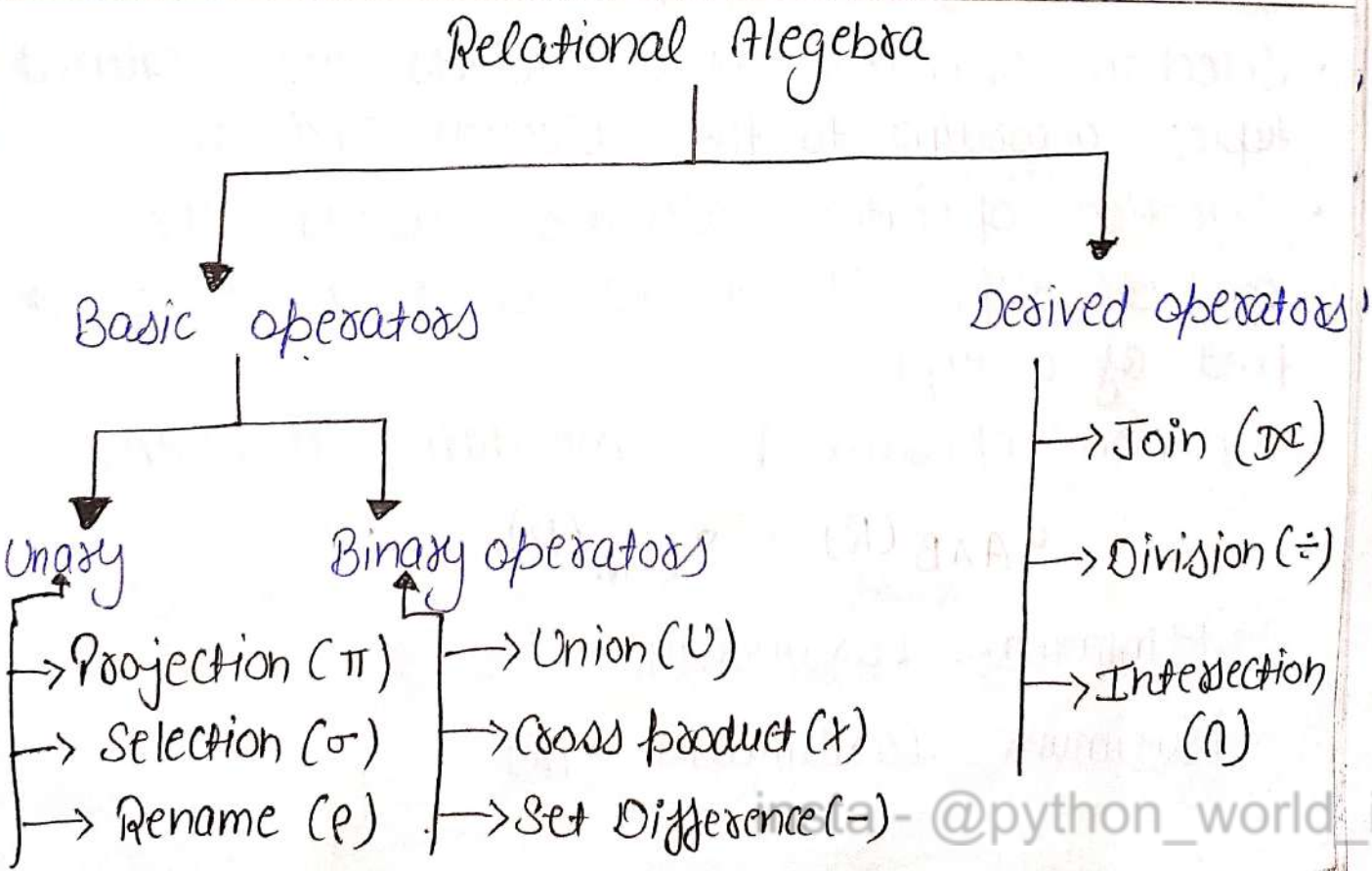
[ All information firstly converted into low level language ]

- When access the records at low level language, there should not be any loopholes that alter the integrity of the database.
- even thought the query written does not change the integrity of the tables, the converted low level language should be same as the query written.
- It should not be converted into some other low level language which changes the data integrity in the database or performs some unwanted actions in the database.

Ex :->

# Relational Algebra

- It is a procedural query language which takes a relations as an input and generates a relation as an output.
- It gives a step by step process to obtain the result of the query.
- It uses operators to perform queries. an operator can be either unary or binary.
- Relational algebra operations work on one or more relations to define another relation without changing the original relations.
- Relational algebra mainly provides theoretical foundation for relational databases and SQL.
  [Helpful to understand the queries]
- Types of operators in relational algebra
  a) Basic operators
  b) Derived operators.



# 1) Selection operator

- Selection operator ($\sigma$) is a unary operator in relational algebra that performs a selection operation.
- It selects tuples (or rows) that satisfy the given condition from a relation. [what you need]
- It is denoted by Sigma ($\sigma$).

# Notation :-> $\sigma_p(\gamma)$ or $\sigma_{(condn)}(\text{Relation Name})$

- $\sigma$ → used for selection operator
- $p$ → used as logic formula propositional or condition.
- $\gamma$ → used for relation.

# Syntax: $\sigma_{<selection\_condition>}(R)$

- We use logical operators like $\wedge$ (And), $\vee$ (Or), ! and relational operators like $=, \neq, >, <, <=, >=$ with the selection condition.
- Selection operator only selects the required tuples according to the selection condition.
- Selection operator always selects the complete tuple. It can not select a section or part of a tuple.
- Selection operator is commutative in nature.

$$\sigma_{A \wedge B}(R) = \sigma_{B \wedge A}(R)$$

- Minimum Cardinality = 0
- Maximum Cardinality = $|R|$

Example: Select tuple from a Student table whose age is greater than 17

$\sigma_{age>17}(Student)$

Select ← $\sigma$ age > 17 (Student)

Cond^n ↓ Relation

[is means =]

Student

| rollno | Name | age | address |
|---|---|---|---|
| 1 | A | 20 ✓ | Bhopal |
| 2 | B | 17 | Mumbai |
| 3 | C | 16 | Mumbai |
| 4 | D | 19 ✓ | Delhi |
| 5 | E | 18 ✓ | Delhi |

Output

| roll_No | name | age | address |
|---|---|---|---|
| 1 | A | 20 | Bhopal |
| 4 | D | 19 | Delhi |
| 5 | E | 18 | Delhi |

- Select tuples from a relation "Boobs" where Subject is "database".

$\sigma_{Subject = "database"}(Boobs)$

- Select tuples from a relation "Boobs" where Subject is "database" and price is "450" or have a publication year after 2010

$\sigma_{Subject = "database" \wedge price = 450 \vee year > 2010}(Book)$

## 2) Projection operator

- Projection operator is a unary operator in relational algebra that performs a projection operation.
- It displays the particular columns from a relation and it delete column that are not in the projection list.
- It is denoted by $\pi$

# Notation: $\pi_{A_1, A_2 \ldots A_n}(R)$

- $\pi$ used to represent the projection operation
- Where $A_1 A_2, A_n$ are attribute name of relation R.

# Syntax: $\pi_{<attribute\ list>}(R)$

- Duplicate rows are automatically eliminated from result.
- Projection operator does not obey commutative property.
- Example: Display the column roll_no and name from the relation Student

$\pi_{rollno, name}(Student)$

Query 1: Display the roll no and name of Students whose age is greater than 17.

Ans → $\pi_{rollno, name}(age > 17(Student))$

Student

| roll_no | name | age |
|---|---|---|
| 1 | A | 20 |
| 2 | B | 17 |
| 3 | C | 16 |
| 4 | D | 19 |

→

| roll_no | Name |
|---|---|
| 1 | A |
| 4 | D |

## 3) Rename operator

- The RENAME operator is used to rename the output of a relation.
- Sometimes it is simple and suitable to break a complicated sequence of operations and rename it as a relation with different names.

Reason to rename a relation can be many like:

→ We may want to save the result of a relational algebra expression as a relation so that we can use it later.

→ We may want to join (or cartesian product) a relation with itself in that case, we rename one of the tables and perform join operation on them.

Eg:→ R x R → Same name then rename one table.

↓

Rename to S.

R x S → It provide the clearity of tables result

# Symbol:→ rho

# Notation:→ $\rho_x(E)$

- $\rho$ → used to RENAME operator
- E is the result of expression or sequence of operation
- x → which is saved with the name.

- In SQL, we use AS keyword.

Ex:→ R x R → Ambiguity will be there.

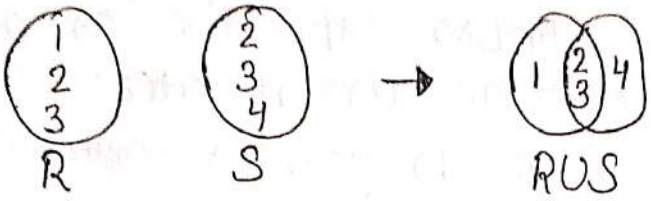
↓

R x $\rho_S(R)$ [Rename R to S]

* Binary operators

## 4) Union operator / fundamental

- It performs binary union b/w two given relations.
- Suppose R and S are two relation. The union operation select all the tuples are either in relation R or S or in both relations R & S.




- For a union operation to be valid, the following cond^n must hold -
  - → Two relations R and S both have same no of attributes.
  - → Corresponding attribute have the same domain
  - → Duplicates tuples should be automatically removed.

- Symbol :→ ∪

# Notation :→ R ∪ S

- R A → R ∪ S
- Union is Commutative.

Ex :→ Student

| Roll No | Name |
|---|---|
| 1 | A |
| 2 | B |
| 3 | C |
| 4 | D |

Employee.

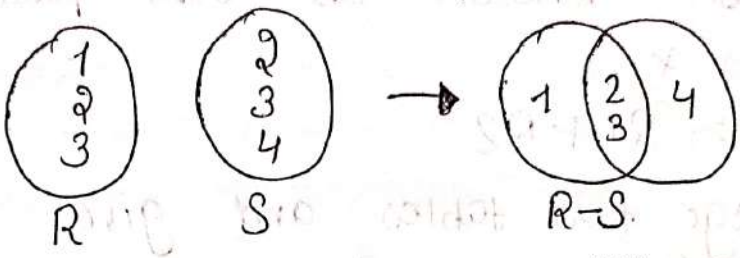
| Emp-no | Name |
|---|---|
| 2 | B |
| 4 | G |
| 8 | H |

(Student) ∪ (Employee).

| RollNo | Name |
|---|---|
| 1 | A |
| 2 | B |
| 3 | C |
| 4 | D |

## 5) Set Difference (-) Operator

- Suppose R and S are two relations. The Set Difference operation select all the tuples that are present in first relation R but not in second relation S.




- for a Set difference to be valid, the following Cond$^n$ must hold -
  - → Two relations R and S both have same no. of attributes.
  - → Corresponding attributes have the same domain

# Symbol : −

# Syntax : R − S

- RA :→ R − S

- Example: Student

| Roll_No | Name |
|---|---|
| 1 | A |
| 2 | B |
| 3 | C |

Employee.

| Emp | Name |
|---|---|
| 2 | B |
| 4 | C |

$\Pi_{Name}(Student) - \Pi_{Name}(Emp)$

Name

| Name |
|---|
| A |
| C |

## 6) Cross product or Cartesian

- Cartesian product is fundamental operator in relational algebra.
- Cartesian product combines information of two different relations into one.
- It is also known as cross product.

# Symbol :- x

# Notation :- R1 x R2

[It merge two tables and gives results in a single tables].

- Generally, a Cartesian product is never meaningful operation when it is performed alone. but it becomes meaningfull when it used operations after cross product like followed by select operations

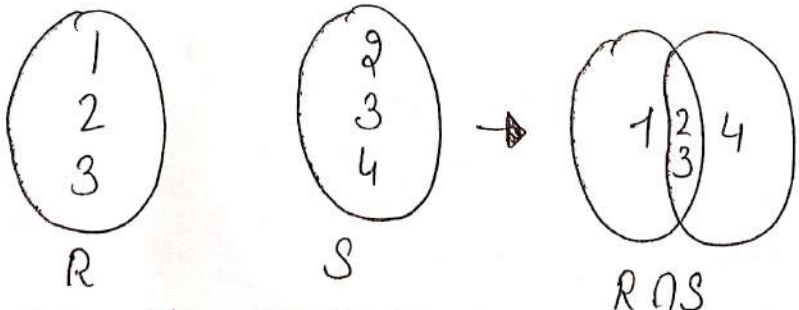
Ex :-

$$\sigma_{author = 'fp'}(Book \times Articles)$$

→ Derived operators

## 5) Set intersection operator

- Suppose R and S are two relations. The set intersection operator selects all the tuples that are in both relations R & S.




→ Same no. of attributes.

→ Corresponding attribute are in same order.

# Symbol: ∩

# Syntax: R ∩ S

- Set interaction is commutative.

$$A \cap B = B \cap A$$

Ex: Student Employee (Student) ∩ (Employee)

| Roll | Name |
|---|---|
| 1 | A |
| 2 | B |
| 3 | C |
| 4 | D |

| Emp | Name |
|---|---|
| 2 | B |
| 8 | G |

→

| Roll No | Name |
|---|---|
| 2 | B |

→ Common numbers write.

# Set operators

- Union, intersection, difference, binary operators as they take to input relation.
- We take a relation and given answer in 1 relation

- The two relations must be compatible.
- Duplicates tuples are automatically eliminate
- Previous valid Cond^n

Set operators

- Union operator (∪)
- Intersection (∩)
- Difference (−)

## 8) Join operator

- Join is an Additional / Derived operator which simplies the queries but does not add any new power.
- Join is a combination of a cartesain product followed by a selection process.

> Join = Cartesian product + Selection

- A join operation pairs two tuple from different relations if and only if a given cond$^n$ is satisfied

# Symbol: $\bowtie$

# Notation: $A \bowtie_C B = \sigma_C (A \times B)$

Types of joins

1) Inner join
   a) Theta join
   b) Equi join
   c) Natural join
2) Outer join / Extension
   a) Left outer join
   b) Right outer join
   c) Full outer join

1) Inner join :→ It includes only those tuples that satisfy the matching criteria, while the rest of tuples are excluded.

a) Theta join / Conditional join

- It combines tuples from different relation provided they satisfy the theta ($\theta$) cond$^n$
- Symbol :- $\theta$
- Notation :- $A \bowtie_\theta B$

  $\theta$ is predicate cond$^n$
- Comparison operator use $<, >, <=, >=$ etc.

b) Equi join

- When a theta join use only equivalance ($=$) cond$^n$ it becomes a Equi join.
- Notation : $A \bowtie_{A.a_1 = B.B_n} B$

c) Natural join

- Based on common attributes in both relation.
- Does not use any comparison operator
- Notation: $A \bowtie B$

2) Outer join:

- It includes Matching tuples and also includes rest of the tuples.
- It uses NULL Value

> Outer join = Natural join + Extra info.

- Contain all row from either one or both relation.

a) Left outer join

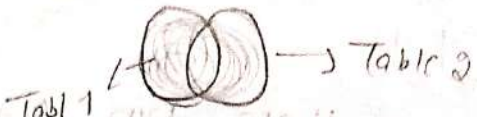
- all the tuples from the left relation R1. are included in resulting relation
  - → All record from left table.
  - → only matching records from right table.
- Symbol :- ⟕

b) right join :→ All record from right table.
• only matching record from left table.

• Symbol: ⟖

c) full join: all the tuples from both left relation R1 and right relation R2 are included in the resulting relation.

• Symbol: ⟗




---

9) Division operator

• Suited to queries that keywood "all" or "every" etc.

• Notation → A ÷ or A/B

10) Assignment operator

The assignment operator (←) provide a convient way to express complex queries.

# Characteristics of relational Model

1) Relational Model always work on one or more relational tables.

2) Relational operations always produce another relational Table.

---

# Relational Calculus

- It is a non-procedural query language.
- It uses mathematical predicate calculus.
- Relational Calculus tells what to do but never explain how to do.
- Relational Calculus provides description about the query to get the result where as relational algebra gives the method to get the result.
- When applied to database, it has two types.
  1) Tuple Relational Calculus.
  2) Domain relational Calculus

# Calculus has variables, constants, comparison (<, >, =) operators, logical connectives (∧, ∨) and quantifiers [∀, ∃] Universal, Existential

TRC :→ variables range over tuples

DRC :→ variable range over domain element.

## 1) Tuple Relational Calculus

- It is a non-procedural query language.
- proposed by Codd in the year 1972.
- Works on tuples (or rows).
- Tuple relational Calculus is used for selecting the tuples in a relation that satisfy the given Cond^n (predicate). The result of the relation can have one or more tuples.
- A query in TRC as expressed:

$$\{t \mid P(t)\}$$

t → resulting tuple.
P(t) → predicate used to fetch T
• result :→ it is the set of all tuples t such that predicate P is true for t

# Predicate Calculus Formula

1) Set of attributes and constants
2) Set of Comparison operator [ <, ≤, = ]
3) Set of Connectives [and (∧), or (∨)]
4) Implication (⇒y) [x ⇒ y, if x is true, y is true]
5) Quantifiers [∃ , ∀]

• Free and Bound Variables

1) The use of quantifiers ∃x and ∀x in a formula is said to bind x in the formula.
2) A variable that is not bound is free

∃x → bind
x → free

$$\{t \mid P(t)\}$$

free variable ↘ quantifier bound.

• The variable t that appears to the left of '|' must be only free formula.
• all other tuple variables, must be bound using a quantifier

Ex: like SQL

• $\{T.name \mid Author(T) \text{ AND } T.article = 'database'\}$

→ This query select the tuples from the tuple author relation. It returns a tuple with 'name' from author who has written an article on 'database'.

## 2) Domain relational Calculus

- It is a non-procedural query language.
- Works on domain of attributes (or columns)
- DRC uses list of attributes to be selected from relation based on the condn.
- DRC is same as but differs by selecting the attributes rather than selecting whole tuples
- DRC, each query expression.

$$\{<a_1, a_2, \ldots, a_n> / P(a_1, a_2, \ldots, a_n)\}$$

- $a_1, a_2, \ldots a_n$ represent domain variables.
- P represents a predicate.
- Result of Query: It is the set of all tuples $a_1, a_2, a_n$ such that predicate P is true for $(a_1, a_2 \ldots a_n)$ tuples.

Ex:- $\{<article, page, subject> / \in j.T \wedge subj = "DB"\}$

output: This query will yield the article, page and subject from the relational javapoint, where the subject is a database

---

Quantifiers are used in quantified expression in which the free variables are bound by the quantifiers.

1) Universal → [∀] all tuples satisfy give condn

2) Existential → [∃] → set of tuples there is at least one occurrences whose satisfy a given condn.

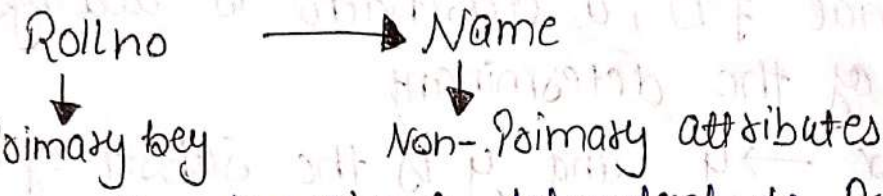
# Difference b/w Relational Algebra and Relational Calculus

| Relational Algebra | Relational Calculus |
|---|---|
| 1) It is a procedural language | 1) It is a Declarative language. |
| 2) Relational Algebra states how the obtain the result. | 2) Relational Calculus states what result we have to obtain. |
| 3) The order in which operations have to be performed. | 3) Does not specify the order of operations. |
| 4) It is not domain dependent | 4) It is domain dependent. |
| 5) It is close to a programming language. | 5) It is close to the natural language |

# Unit-2

## #Function Dependencies

- The functional Dependencies is a relationship that exists b/w two attributes.
- It typically exists b/w the primary key and non-primary key attributes within a table.




- Name is functional dependent to Roll no

$$X \longrightarrow Y$$

- The left side of FD is known as Determinant
- The right side of FD is known as Dependent

[If x value change then same as y value change also ∴ y dependent to x]

- functional dependencies are used to mathmatically express relations among database entities and are very important to understand advanced concepts in RDBS.

for ex:- Assume we have an employee table with attributes :- Emp-id, Emp-name.

Emp-id ⟶ Emp-Name.

- Emp-Name is functionally dependent on Emp-id.

# Types of functional Dependencies

1) Trivial functional Dependencies.
2) Non-Trivial functional Dependencies.
3) Multivalued functional Dependencies
4) Transitive functional Dependencies
5) fully functional Dependencies.
6) Partially functional Dependencies

## 1) Trivial functional Dependencies

- In Trivial f.D, a dependent is always a subset of the determinant.

i.e if $x \rightarrow y$ and y is the subset of x, then it is called trivial functional dependency

- The following dependencies are also trivial like: $A \rightarrow A$, $B \rightarrow B$.

for ex:→ Consider a table with two columns Roll no and Name.

- {Roll-no, name} ⟶ Name. is a trivial functional dependency, since the dependent name is a subset of determinant set {roll no-, name}.

-lly, roll_no ⟶ roll-no and name ⟶ name is also an example of Trivial functional dependency.

- In Trivial f.D we uses reflexive property

## 2) Non-Trivial functional Dependency

- In non-Trivial f.D, the dependent is strictly not a subset of the determinant.
  i.e if $x \rightarrow y$ and y is not a subset of x, then it is called Non-Trivial functional Depend.
- When A intersection B is NULL. then $A \rightarrow B$ is called as Complete non-Trivial.

$A = \{P, Q, R\}$

$B = \{S, T\}$

$A \rightarrow B$ , $A \cap B = \{NULL\}$

Ex:-> Consider a table with two column, Emp-id, name and age.

- roll-no $\longrightarrow$ name is non-trival f.D, since name is not a subset of age
- 111y, {roll_no, name} $\longrightarrow$ age is also a non-trivial f.D, since age is not a subset of {roll_no, name}.

## 3) Multivalued functional Dependency

- In Multivalued functional Dependency entities of the dependent set are not dependent on each other.
- if $a \rightarrow \{b, c\}$ and there exists no functional dependeneny blw b and c, then it is called f Multivalued f.D

Ex:-> Consider a table with three column, roll-no name, age.

- rollno → {name, age} is a multivalued f.D, since the dependents name & age are not dependent on each other.

(name → age , age → name)

Not exist

## 4) Transitive functional Dependency

- In Transitive functional Dependency, dependent is indirectly dependent on determinant.
- if a → b & b → c, then according to axiom of Transitivity, a → c, This is a Transitive functional dependency.
- example: enrol-no → dept and dept → bul-no

↓

enrol-no → building-no is valid functional dependency.

## 5) fully functional dependency

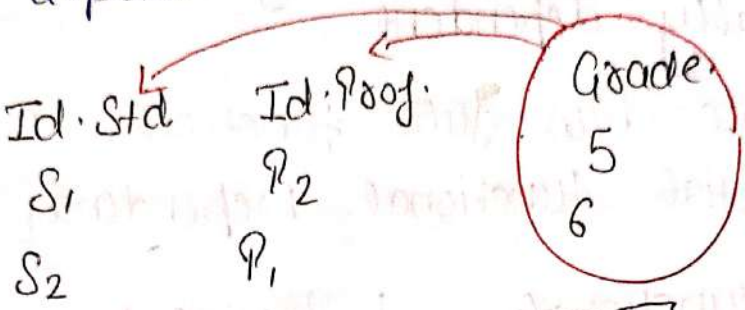
- If x and y are an attribute set of a relation, Y is fully functional dependent on X, if y is functionally dependent on x but not on any proper subset of x.
- In the relation ABC → D, attribute D is fully functionally dependent on ABC and not on any proper subset of ABC. That means that Subsets of ABC like: AB, BC, A, B etc cannot determine D.

- $ABC \rightarrow D$ [D is fully ∮ D on ABC]
- But D cannot dependent on any subset of ABC

$BC \rightarrow D$ not possible because BC cannot determine D
$C \rightarrow D$ C cannot determine D
$A \rightarrow D$ A cannot determine D

- only ABC determine D means D is fully dependent ABC.

Example:




$\{Id.Std, Id.Proj\} \rightarrow Grade$

- Id.Std, Id.Proj can uniquely determine the price but both Id.Std & Id.Proj together.
- So we can say that Grade is fully functionally dependent on {Id.Std, Proj}.

## 6) Partial Functional Dependency

A Functional Dependency $x \rightarrow y$ is a partial dependency if y is functionally dependent on x and y can be determined by any proper subset of x.

$AC \rightarrow B$
$A \rightarrow D$ ⟩ Relationship
$D \rightarrow B$

$\{A\}^+ = ADB$ [Closure Compute]

Here A is alone capable of determining B, which means B is partially dependent on AC.

Example: Student table

| name | roll-no | course |
|---|---|---|
| Tanu | 2 | DBMS |
| Anu | 3 | OS |

Here, we can see that both the attributes name and roll-no alone are able to uniquely identify a course. Hence we can say that the relationship is partially dependent.

## # Difference b/w full functional Dependency and Partial functional Dependency

| Full functional | Partial functional Dependen. |
|---|---|
| 1) A functional dependency $x \to y$ is a full functional dependency if y is functional dependent on x and y is not functionally dependent on any proper subset of x. | 1) A functional dependency $x \to y$ is a partial dependency if y is functionally dependent on x and y can be determined by any proper subset of x. |
| 2) In full functional dependency, the non-prime attribute is functionally dependent on the candidate key | 2) In partial functional dependency, the non-prime attribute is functionally dependent on part of a candidate key |
| 3) enhances the quality of data in our database. | 3) does not enhance the data quality. |

4) In fully functional dependency, if we remove any attribute of $x$, then the dependency will not exist.

5) fully functional Dependency equates to the normalization standard of second Normal form.

4) In partial functional dependency, if we remove any attribute of $x$, then the dependency will still exist.

5) Partial functional Dependency does not equate to the normalization standard of second Normal form

## #Closure of functional Dependency

- The Closure of functional dependency means the complete set of all possible attributes that can be functionally desired from given functional dependency using the inference rules known as Armstrong's Rules.
- If "F" is a functional dependency then closure of functional dependency can be denoted using "$F^+$".
- These are three steps to calculate closure of functional dependency.

Step 1: Add the attributes which are present on left Hand Side in the orginal functional dependency

Step 2: Now, add the attributes present on the right hand side of the functional dependency.

Step 3: With the help of attributes present on right hand side, check the other attributes that can be derived from the other given functional dependencies.

Repeat this process until all the possible attribute which can be derived are added in the Closure.

Example: Consider the tables Student - details having (Roll_No, Name, Marks, location) as the attributes and having two functional dependencies.

FD1 → Roll_No :→ Name, Marks

FD2 → Name :→ Marks, location.

Step1: Add the attributes present on the LHS of the first dependency to the Closure.

$\{Roll\_no\}^+ = \{Roll\_No\}$

Step2: Add the attributes present on the RHS of the original functional dependency to the Closure.

$\{Roll\_no\}^+ = \{Roll\_No, Marks\}$

Step3: $\{Roll\_no\}^+ = \{Roll\_No, Marks, Name, Location\}$

Next FD2 same process

# Prime attributes :→ Attributes which are indispensable part of Candidate key.

# Non-Prime attributes :→ Attributes other than prime attributes which does not take part in formation of Candidate key.

# Extraneous Attributes :→ Attributes which does not make any effect on removal from Candidate key.

# Inference Rule

- Armstrong's axioms / properties of functional dependencies
- Armstrong's axioms property was developed by William Armstrong in 1974 to reason about functional dependencies.
- The inference rule is a type of assertion. It can apply to a set of functional to derive other FD.

1) The property suggest rule that hold true if the following are satisfied.

**# Transitivity:** In the Transitive rule, if X determine Y and Y determine Z, then X also determine Z. by this rule.

$\therefore X \to Y$ and $Y \to Z \implies X \to Z$

for ex: roll_no → dep_name, dept_name → builder
then roll_no → dept_building

**# Reflexivity:** In the reflexive rule, if Y is a subset of X, then X determines Y.

$Y \subseteq X$ then $X \to Y$

for ex: {roll_no, name} → name

**# Augmentation:** This rule is also called as partial dependency. If X determines Y, then XZ determines YZ for any Z.

∴ $X \rightarrow Y$ then $XZ \rightarrow YZ$

Ex:- for R(A B C D)

if $A \rightarrow B$ then

$AC \rightarrow BC$ [C add both in side]

4) Union rule:- if x determines Y and x determines Z, then x must also determines Y and Z

∴ $X \rightarrow X$ and $X \rightarrow Z$ then $X \rightarrow YZ$

5) Decomposition rule:- also known as project rule. It is the reverse of union rule

- If x determines Y and Z then X determine Y and x determines Z.

∴ $X \rightarrow YZ$ then $X \rightarrow Y$ , $X \rightarrow Z$

---

# Advantages of functional Dependency

1) Prevent data redundancy
2) Maintain the quality and integrity of data.
3) Reduce the risk of error
4) Identify poor designs
5) Define Meaning and Constraints of databases

# Main Characteristics used in normalization for F.D:- There is a one-to-one relationship b/w the attribute on the left-hand side and those on RHS of a F.D

# Data Redundancy

- Data redundancy is when an organization stores the same data in multiple places at the same time. This problem arises when a database is not normalized.
- Data redundancy can occur in databases and file-based storage system
  - Databases: It is a systematic collection of data stored electronically on a computer in which a database management system controls and can manipulate the stored data.
  - A file-based storage system: It is an ordered and nested folder-based method to store and organize data on devices that include a hard drive, flash drive.

# Benefits of data redundancy

1) Creates data backups
2) Improves data protection
3) Provides data access speed
4) Data recovery 5) Provides Data Security

# Drawbacks of data redundancy

1) Increases data inconsistencies
2) Allows for data corruption
3) Increase data maintenance costs
4) Wastage of Storage

→ Types of data redundancy

# How to reduce data redundancy

1) Database Normalization: → we can normalize the data using the normalization method. In this method, the data is broken down into pieces, which means a large table is divided into two or more small tables to remove redundancy.
- Normalization removes insert anomaly, update anomaly and delete anomaly

2) Deleting Unused Data: → It is important to remove redundant data from the database as it generates data redundancy in the DBMS. It is a good practice to remove unwanted data to reduce redundancy.

3) Master data: → The data administrators shares master data across multiple systems. Although it does not remove data redundancy, but it updates the redundant data whenever the data is changed.

Example:

| Student id | Name | Contact | College |
|---|---|---|---|
| 100 | Tanmay | 98xxxx | GEU |
| 101 | A | xx0142x | GEU |
| 102 | B | 740xxx | GEU |
| 103 | C | 989xx1 | GEU |
| 104 | D | 01x01x | GEW |

- As it can be observed that values of attribute college name is being repeated which can lead to problems.

- Problem causes due to redundancy are:
  - → Insertion anomaly
  - → Deletion anomaly
  - → Updation anamaly

# Anomalies

- A database anomaly is a data inconsistency caused by an operation such as an insertion, deletion and updation. When a record is held in Multiple time, and not all copies are updated, inconsistencies can occur.
- Anamalies in DBMS occur when the data in the database contains too much redundancy and the tables that comprise the database are poorly constructed.
- It can be removed by process of Normalization, which generally splits the database which results in reducing the anomalies in the database.

| Emp - id | Emp - name | Emp-dept | Emp-city |
|---|---|---|---|
| 102 | A | D001 | Delhi |
| 103 | B | D001 | Delhi |
| 104 | C | D001 | Delhi |
| 105 | D | D002 | Delhi |
| 106 | E | D002 | Punjab |
| 107 | F | D002 | Punjab |
| 108 | G | D002 | Punjab |

- all data are stored in one table so some problem are created which is Anomalies.

## 1) Insertion Anomaly

- Insertion Anomaly arises when you are try to insert some data into the database, but you are not able to insert it.

Ex:

| Emp-id | Emp. Name | Emp. dept | Emp. City |
|---|---|---|---|
| 101 | A | D001 | Delhi |
| 102 | B | D001 | Delhi |
| 103 | C | D00 | Bangloge |
| 104 | D | D00 | " |

- Suppose add a new entry in above table

| 108 | Shilpti | | Delhi |
|---|---|---|---|

Not allot any dept so we can not insert dept

apply a constraints [NULL not]

due to some data not insert a new entry.

## 2) Deletion Anomaly

Deletion anomaly arises when you delete some data from the database, but some unrelated data is also deleted, that is, these will be a loss of data due to deletion anomaly.

Ex:

| 103 | C | D00 | Bangloge |
|---|---|---|---|

This column leave the company, so That data delete from the table

[Delete from Employee where Emp-id 103]

3) Updating Anomaly

An update anomaly arises when you update some data in the database, but the data is partially updated, which causes data inconsistency.

Eg: If a single value update, but changes are going in Multiple value

- Doo1 change Doo6 due to some organizational in above table Doo6 only change in 2 column not change in other table, that causes data inconsistency

---

# Normalization

- It is a step-by-step process to eliminate anomalies.
- Normalization is the process of organizing data in the database
  1) Minimize redundancy in relation.
  2) Decompose the table and link using relationships [large table converted into small tables]
- The normal form is used to reduce redundancy from the database table.

# Why do we need Normalization?

- To remove anomalies. failure to eliminate anomalies leads to data redundancy and can cause data integrity and other problems as the database grows.
- Normalization consists of a series of guidelines that help to guide you in creating a good database structure.

- If there is no normalization in SQL, there will be many problem such as

1) Insert Anomaly :- This happens when we cannot insert data into the table without another.

2) Update Anomaly :- This is due to data inconsistency caused by data redundancy and data update

3) Delete exception :- occurs when some attribute are lost due to the deletion of other attributes

- Normalization involves organizing the columns and tables in the database to ensure that their dependencies are correctly implemented using database constraints
- Normalization is the process of organzing data in a proper manner. It is used to minimize the duplication of various relationship in the database.
- It helps to split a large table into several small normalized tables. Relational links and links are used to reduce redundancy. Normalization also known as database normalization or data normalization because it helps to improve the speed, accuracy and efficiency of the database.

# Objective of Normalization

- To avoid creating and updating any unwanted data connections and dependancies
- Used to remove the duplicate data and database anomalies from the relational table.
- helps to reduce redundancy and complexity by examining new data type used in the table.

# Advantages

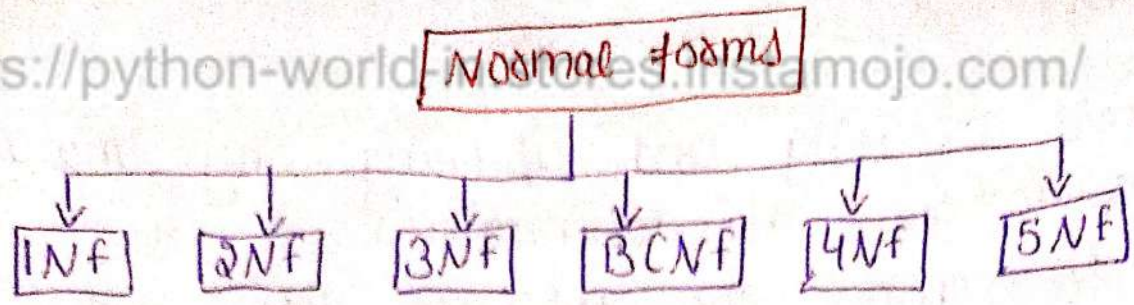
1) Reduces data redundancy
2) Increase data integrity
3) Improves query performance

# Disadvantages

1) Increased Complexity
2) Slower write performance
3) Careless decomposition may lead to a bad database design.

# Types of Normal forms

- When a database has been normalized, it is said to be in normal form.
- The normal forms apply to individual relations.
- There are Six Normal forms in DBMS.




1) 1NF :→ A relation is in 1NF if it contains an atomic value
   1) Each table cell should contain a single value.
   2) Each record needs to be unique

- Attribute of a table cannot hold multiple value.
- first normal form disallow the mult-valued attribute, composite attribute and their combinations

Example:

| Name | Rollno | address |
|---|---|---|
| Tanny | 01 | Delhi, Punjab |
| Anny | 02 | Hardwar, |
| Manu | 03 | Haryana |

- In address, two address are stored, so it is mult-valued attribute ∴ not 1NF relation.

↓ convert into 1NF.

| Name | Roll no | address |
|---|---|---|
| Tanny | 01 | Delhi |
| Tanny | 01 | Punjab |
| Anu | 02 | Hardwar |
| Manu | 03 | Haryana. |

Now this is first normal form. 1Nf wants to store unique information in table without data repeatation

2) 2Nf:- A relation will be in 2Nf if it is in 1Nf and all non-key attributes are fully functional dependent on the primary key.

- No attributes of the table should be functionally dependent on only one part of a Concatenated primary key.

Primary key ← $AB \rightarrow C$ ✓ → attributes

∴ AB determine C

$A \rightarrow C$ ✗
$B \rightarrow C$

Example: 2Nf is based on the concept of full functional dependency $X \rightarrow Y$ is a fully function dependency if removal of any attribute (x) from (x) means that the

$X \rightarrow Y$

$(AB) \rightarrow C$

- If we remove B, then $A \rightarrow C$ ✗
- If we remove A, then $B \rightarrow C$ ✗

dependency does not hold any more.

- Let us assume, a School can store the data Student and Subject.

Student Table

| Student id | Subject | Studentage |
|---|---|---|
| 501 | Physices | 22 |
| 501 | Math | 22 |
| 502 | Zoology | 24 |
| 503 | Sanskrit | 27 |
| 503 | Botany | 27 |

- It is not in 2NF because it hold partial function dependency and fully function dependency.

Student-id → Subject [P.f.D]

(Student id, age → Subject ] [f.f.D]

id → Subject ✗

age → Subject ✗

- Convert this table in two parts

Student detail table

| Student-Id | Studentage |
|---|---|
| 501 | 22 |
| 502 | 24 |
| 503 | 27 |

Student-Subject table

| Student-Id | Subject |
|---|---|
| 501 | Phy. |
| 501 | Math |
| 502 | Zoology |
| 503 | Sanskrit |
| 503 | Botany |

- The following two tables are satisfy the condⁿ of 2NF relation. It also in 1NF form and every non-prime attribute is dependant on primary key.

3) 3NF: A relation will be in 3NF if it is in 2NF and no transition dependency exists.

| |
|---|
| A → B → C ✗ |
| A ⟶ C ✓ |

- 3NF is used to reduce the data duplication. It also used to achieve the data integrity
- If there is no transitive dependency for non-prime attributes, then the relation must be in thired normal form.

→ A relation is in third normal form if it holds atleast one of the following condⁿ for every non-trivial function dependency $X \rightarrow Y$

1) X is Super key
2) Y is a prime attribute such as each element of y is part of some candidate key

Example: Employee – Department – Location table.

| Emp NO | Emp-name | Sal | Dept-Name | Depart-Locat |
|---|---|---|---|---|
| 1001 | A | 7500 | Account | 102 |
| 1002 | B | 5000 | Sales | 104 |
| 1003 | C | 1000 | Accounts | 102 |
| 1004 | D | 4500 | Sales | 104 |
| 1005 | E | 6500 | Store | 106 |

- table Not in 3NF because it hold the transitive dependency.
  - EmpNo → Depname → Dept-Location
  - EmpNo → Dept-Location.
- To make it in 3NF we decompose and remove the transitive dependency. So we convert the given table in 3NF decompose two sub table such as →

(Table 1) Emp – Department

| Emp-No | E-Name | Sal | Dep-Name |
|---|---|---|---|
| 1001 | Vishal | 7500 | Account |
| 1002 | Amit | 5000 | Sales |
| 1003 | Anuj | 10000 | Accounts |
| 1004 | Vikas | 4500 | Sales |
| 1005 | Sumit | 65000 | Store |

(Table 2) Department-location

| Dept-name | Dept-Location |
|---|---|
| Accounts | 102 |
| Sales | 104 |
| Store | 106 |

- This 2 table is converted in 3NF, they don't have transitive dependancy.
- Some dependancies cause redundancy in database
- Redundancy removed by BCNF.

4) BCNF:) A stronger definition of 3NF is known as Boyce Codd's normal form.

- It is stricter than 3NF.
- A table is in BCNF if every functional dependency $x \rightarrow y$, x is the super key of the table.
- for BCNF, the table should be in 3NF, and for every FD, LHS is super key

Example: let's assume there is a company where employee work in more than one department

Employee table

| Emp-id | Emp-country | Emp-dept | Dept-type | Dept no |
|---|---|---|---|---|
| 264 | India | Designing | D394 | 283 |
| 264 | India | Testing | D394 | 300 |
| 364 | UK | Stores | D283 | 232 |
| 364 | UK | Developing | D283 | 549 |

FD → Emp-id ⟶ Emp-country

Emp-dept → {Dept-type, Dep-no}

Candidate key = {Emp-id, Emp-Dept}

- the table is note in BCNF because neither Emp-dept nor Emp-id alone are keys.
- To convert the given table into BCNF, we decompose into three tables.
  - → Emp-Country table.
  - → Emp. Dept table
  - → Emp Dep-mapping table

Emp - Country table

| Emp - id | Emp - Country |
|---|---|
| 264 | india |
| 364 | UK |

Emp - Dept table

| Emp - dept | Dept type | Emp - Dept - no |
|---|---|---|
| Designing | D394 | 283 |
| Testing | D394 | 300 |
| Stores | D283 | 232 |
| Developing | D283 | 549 |

Emp - Dept - mapping Table

| Emp - id | Emp - Dept |
|---|---|
| D394 | 283 |
| D394 | 300 |
| D283 | 232 |
| D283 | 549 |

- Functional Dependencies:

  Emp - id → Emp - Country

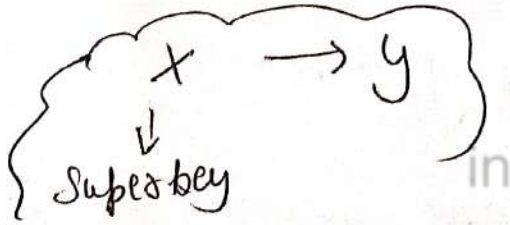
  Emp - Dept → {Dept - type, Emp, Dept - No}

- Candidate keys:

  for the first table : Emp - id

  " " Second " : Emp - dept

  " " Third " : {Emp - id, Emp, Dept}




→ BCNF

- Now, this is in BCNf because left side part of both the functional dependencies is a key.

4) 4NF:) A relation will be in 4NF if it is in Boyce Codd normal form (BCNf) and has no multi-valued dependency.

- MVD occuss when two or more independent multi-valued facts about the same attribute occuss within the same relation.
- MVD is denoted by

$$X \longrightarrow\longrightarrow Y$$

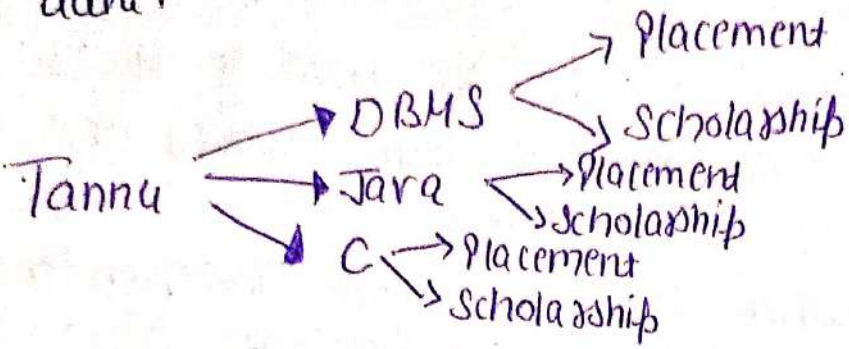
It will be readess "there is a mult-valued dependency of Y" or "X" mult-detesmines Y".

Example: faculty

| faculty | Subject | Committee |
|---|---|---|
| Tannu | DBMS | Placement |
| Tannu | Java | Placement |
| Tannu | C | Placement |
| Tannu | DBMS | Scholasship |
| Tannu | Java | Scholasship |
| Tannu | C | Scholasship |

- The given faculty table is in 3Nf, but the Subject and Committee ase two independent entity. Hence there is no relationship blw Subject and Committee.
- In the faculty relation a faculty with faculty name Tannu contains three Subject DBMS, Java and C and two committee placement and Scholasship.

So there is a mult-valued dependency on faculty name, which leads to unnecessary repeation of data.




- Tannu ⟶ ⟶ placement
- Tannu ⟶ ⟶ Scholarship

So to Make the above table into 4NF, we can dempose it into two tables.

Table 1 Faculty - Cource

| Faculty | Subject |
|---|---|
| Tannu | DBMS |
| Tannu | Java |
| Tannu | C |

Table 2 Faculty - Committee.

| Faculty | Committee |
|---|---|
| Tannu | Placement |
| Tannu | Scholarship |

[ remove Multivalued Value ] ↘ 4NF.

6) 5NF:-

The 5NF [fifth normal form] is also known as project - join normal form.

- A relations is in fifth normal form, if it is in 4NF, and won't have lossless decomposition into smaller tables
- 5NF is satisfied when all the tables are broken into a many tables as possible in order to avoid redundancy. After that you combined these all tables if it is equal to original table then 5NF

→ $R = \{A, B, C, D\}$ → Subpart.

$R_1 = (A, B)$ $(C, D)$

Join table.

→ $R \bowtie R_1$

→ $(A, B, C, D)$

Both value are same it known as 5NF.

[No data is losseless]

→ $R = \{A, B, C, D\}$

$R_1$ $(A)$ $(C, D)$

→ $R \bowtie R_1$

$\{A, B, C, D\}$

Not Sames loss one data.

---

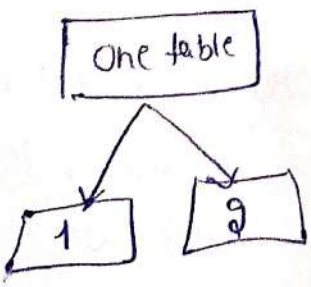
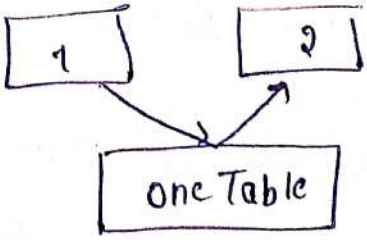
Question Why BCNF considered to be Stronger than 3NF.

→ because it eliminates the second condition for 3NF, which allowed the right side of the FD to be a prime attribute.

# Denormalization

- When we normalize tables, we break them into multiple smaller tables. So when we want to retrieve data from multiple tables, we need to perform some kind of join operation on them. In that case, we use the denormalization technique that eliminates the drawback of the normalization.

Denormalization is a technique used by database administrators to optimize the efficiency of their database infrastructure.

| Normalization | Denormalization |
|---|---|
| 1) It is used to delete redundant data from a database and replace it with non-redundant and reliable data. | 1) Denormalization is a technique used to merge data from multiple tables into a single table that can be queried quickly. |
| 2) Normalization eliminate redundancy. | 2) Denormalization added redundancy. |
| 3) It maintain data integrity | 3) It does not maintain integrity |
| 4) Uses optimized memory and hence faster in performance | 4) introduces some sort of wastage of memory. |
| 5)   | 5)   |

# Prime attribute

A prime attribute is one of attributes that make up the candidate key.
- also known as key attribute.
- A set of attributes that uniquely identify tuples in a table is known as Candidate key
- Candiate key is a Supes key with no attributes that are repeated

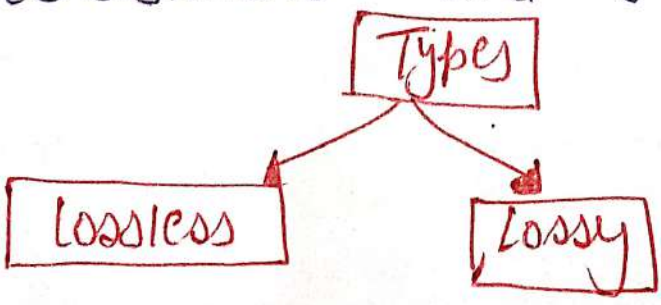
# Non-Prime attribute

- Non-Prime attribute are those attributes of the relationship that do not present in any of the possible candidate key of relation.
- also known as non-key attributes.

---

# Decomposition

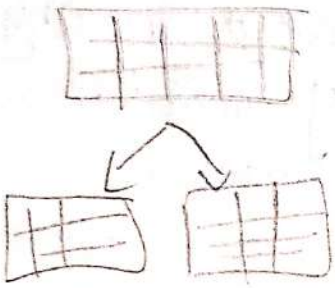
- When a relation in the relational model is not in appropriate normal form then the decomposition of a relation is required.
- In a database, it breaks the table into multiple tables
- If the relation has no proper decomposition, then it may Lead to problems like loss of information.
- Decomposition is used to Eliminate Some of the problems of bad design like anomalies inconsistencies and redundancy



1) Lossless Decomposition :- If the information is not lost from the relation that is decomposed, then the decomposition will be lossless

- The lossless decomposition guarantees that the join of relations will result the same relation as it was decomposed.
- The relation is said to be lossless decomposition if natural join of all the decomposition given the original relation.

Ex:



→ All data

→ Decompose table in two form

after applying join operation

→ No data loss of information if we join all Decompose table

2) Lossy Decomposition :- When a relation is decomposed into two or more relational schemas, the loss of information unavailable when the original relation is retrieved

Ex :-

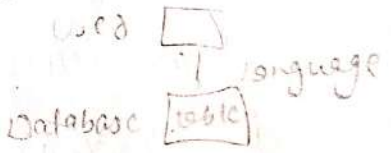
→ Table

Decompose the table

→ Not able to join. Because one relation has left in one table

# Unit-3

#SQL :→ SQL Stands for Structured Query Language
- It is used for storing and managing data in RDBMS
- It is a Standard language for relational Database System. It enables a users to create, read, update and delete relational database and tables.
- SQL is not a database System, but it is a query language.
- Suppose you want to perform the queries of SQL language on the stored data in the database. you are required to install any DBMS in your systems. for ex:→ Oracle, MySQL
- This query language became the standard of ANSI in the year of 1986 and ISO in the years of 1987.




→ Why SQL used

1) SQL can execute queries against a databa
2) SQL can retrieve data from a database.
3) SQL can insert, update, delete, records from a datase.
4) SQL can create views in a tables.
5) Mainly represent in tables