Unit - I

Introduction.

Database Management System :-

Data is a collection of facts & figures that can be processed to produce the information.

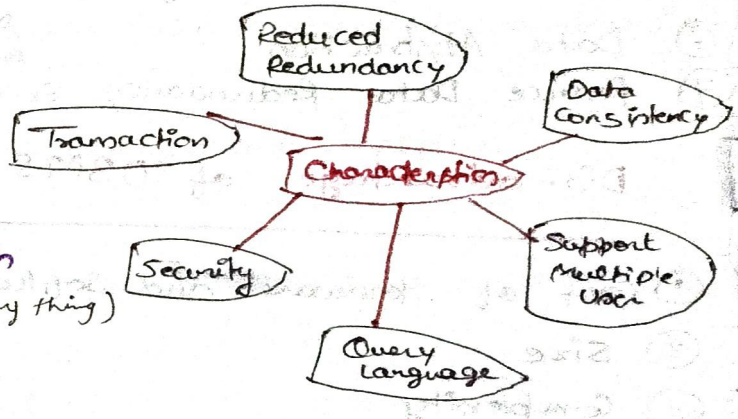
Processed data is known as information.

Database is collection of inter-related data.

Database Management System = Database + A set of program to access the data/ information from the database.

A DBMS stores data in which it become easier to retrieve, manipulate, & produce information.

Characteristics of DBMS



1. Real-world entity
2. Relation-based tables
3. Isolation of data & application
4. Less Redundancy (Repetition of any thing)
5. Consistency
6. Query Language
7. ACID Properties (Atomicity, Consistency, Isolation & Durability)
8. Multiuser & Concurrent Access
9. Multiple views
10. Security
11. Back up & Recovery
12. Concept of Normalization to minimize the redundancy of relation

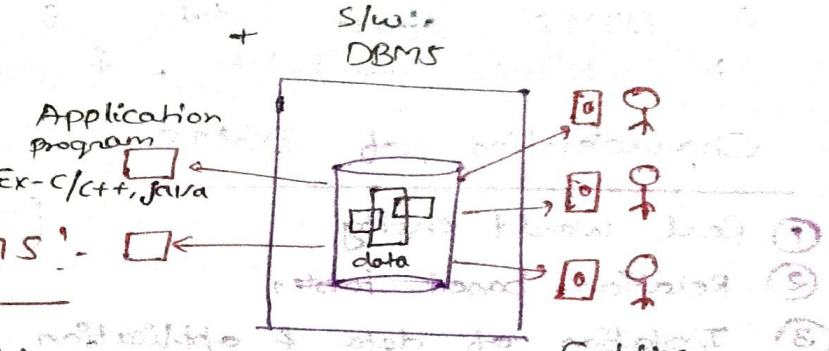
Example of DBMS MySQL Database, MS Access, Oracle, DB2, Microsoft SQL Server

Needs of DBMS :-

1. Creation a database.
2. Retrieval information from the database.
3. Updating the database
4. Managing the database

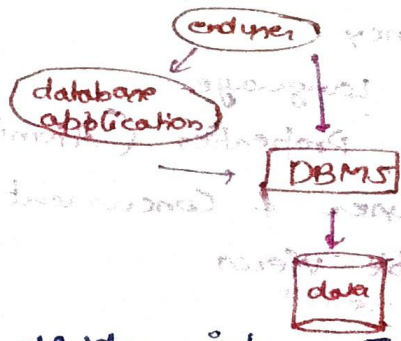
Advantage of DBMS :-

1. Reduce time :- It reduce development time & maintenance need.
2. Backup :- It provides backup & recovery subsystem which create automatic backup data from hardware & software & restore that data if req.
3. Multiple User Interface :- It provides different types of user interface like graphical user interface, application interface.
4. Better Data Transfering
5. Better Data Security
6. Data Abstraction
7. Reduce Data Redundancy



Dis-advantage of DBMS :-

1. Cost of Hardware and Software
2. Size
3. Complexity
4. Higher impact of failure
5. Increased Staff Cost
6. Req. of technical staff



Components of DBMS :-

If divided into 6 major components -

1. Hardware
2. Software
3. Data
4. Procedures
5. Data Acess Language
6. User

Characteristics -

1. Shared
2. persistent
3. Security
4. integrity
5. Consistency

Database Designers:- Design the appropiate structure of the database where we store data.

System Analyst:- Analyses the req of end users, especially native & parametric end user.

Application Programmer:- are computer professionals, who write application programs.

Native Users / Parametric Users:- Native User are Un-sophisticated users, which has no knowledge of the database. These users are like a layman, which has a little bit knowledge of the database.

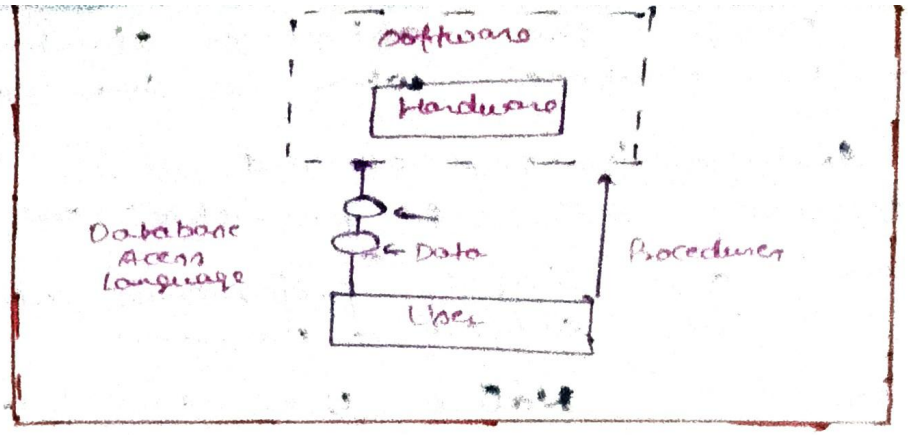
Naive User are just to work on developed applications & get the desired result.

Sophestic Users:- are can be engineers, scientists, buisness analyst, who are familiar with the database. These users intract with the database but they do not write programs.

Casual Users / Temporary Users:- These types of users communicate with the database for a little period of time.

Database Management Vs File Management System

File Mangement System	Database Management System
Smalls system like C++, Java program	Large System like Oracle, MySQL.
Relatively Cheap	Relatively expensive
Simple structure	Complex Structure
Not Secure	Secure
Single User Oriented	Multi-User oriented
The file System is software that manage & organizes the files in a storage medium with in a computer.	DBMS is software for managing the database
• Redundancy • provide back & Recovery	• No Redundant Data • Provide Back & Recovery



① Hardware!- When we say Hardware, we mean computer, Hard disk I/O channels for data, and any other physical component involved before any data is sucessfully stored into the memory.

② Software!- This is the main component, which controls everything. The DBMS software which used to manage the database.

③ Data!- Data is the resource, which can used for to make database. Meta data is data about da

④ Procedures!- It refers to general instruction to use a database management system. This includes to setup & install a DBMS, To login & logout of DBMS, to manage databases to take backups, generatin reports etc..

⑤ Data Acess Language!- Simple language designed to write commands to acess, insert update & delete data stored in any database.

⑥ Users!- Database user who can acess the database. It is typen -

a. Database Administrator (DBA)!- A person or a team, who is responsible for managing the overall database management System.
It is the leader of the database & superuser of the system.
It is responsible for the data administration of the all the 3 levels of database.
Responsible for deciding instances of database, defining the schem Liasing with Users, Security, Back-Recovery, Monitering the performance

DBMS Architecture :- The DBMS design depends upon its architecture. The basic client / server architecture is used to deal with a large no of PCs, web server, database server & other components that are connected with networks.

- The client/server architecture consist of many PCs & a workstation which are connected via the network.
- DBMS architecture depends upon how users are connected to the database, to get their req done.

Types :- 1-tier Architecture
2-tier Architecture
3-tier Architecture

1-Tier Architecture :- The database is directly available to the user.

It means user can directly have the DBMS & use it. Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end user.

The 1-tier architecture is used to development of the local application, where programmers can directly communicate with database for the quick response.

2-Tier Architecture :- It is same as basic client-server.

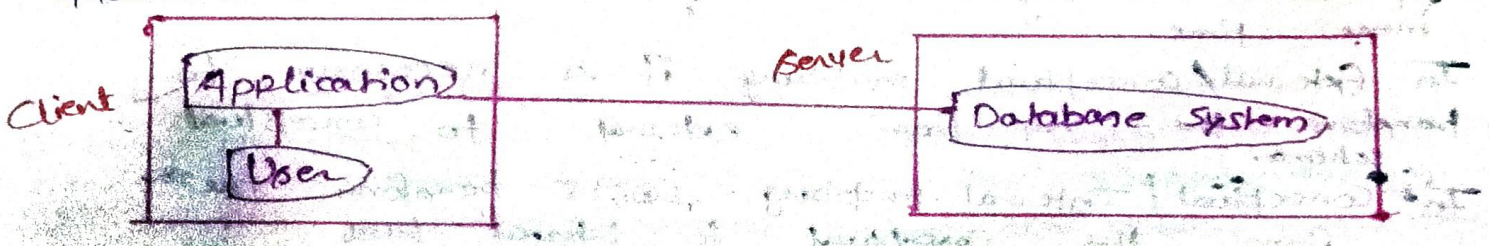
In this, application on the client end can directly communicate with the database at the server side.

For this interaction : API like ODBC, JDBC are used. (Open database Connectivity) (Java database Connectivity)

The user interfaces & application programs are run on the client side.

The server side is responsible to provide the functionalities like query processing & transaction management.

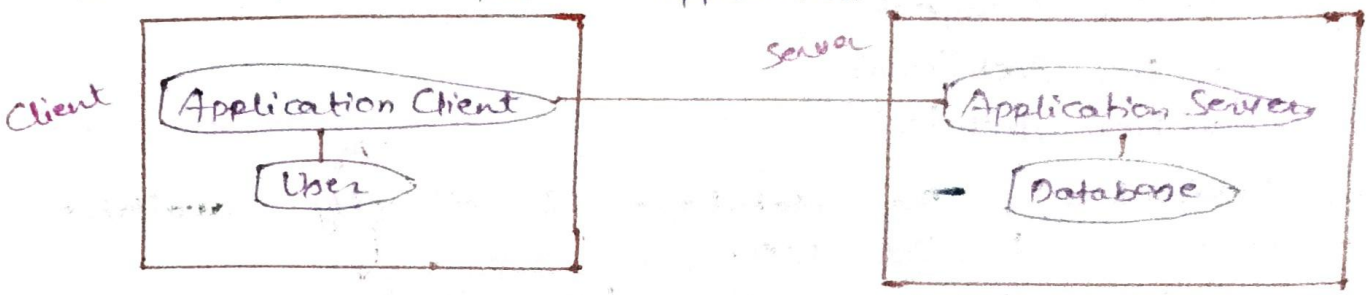
To communicate with the DBMS, client side application establish a connection with the server side.



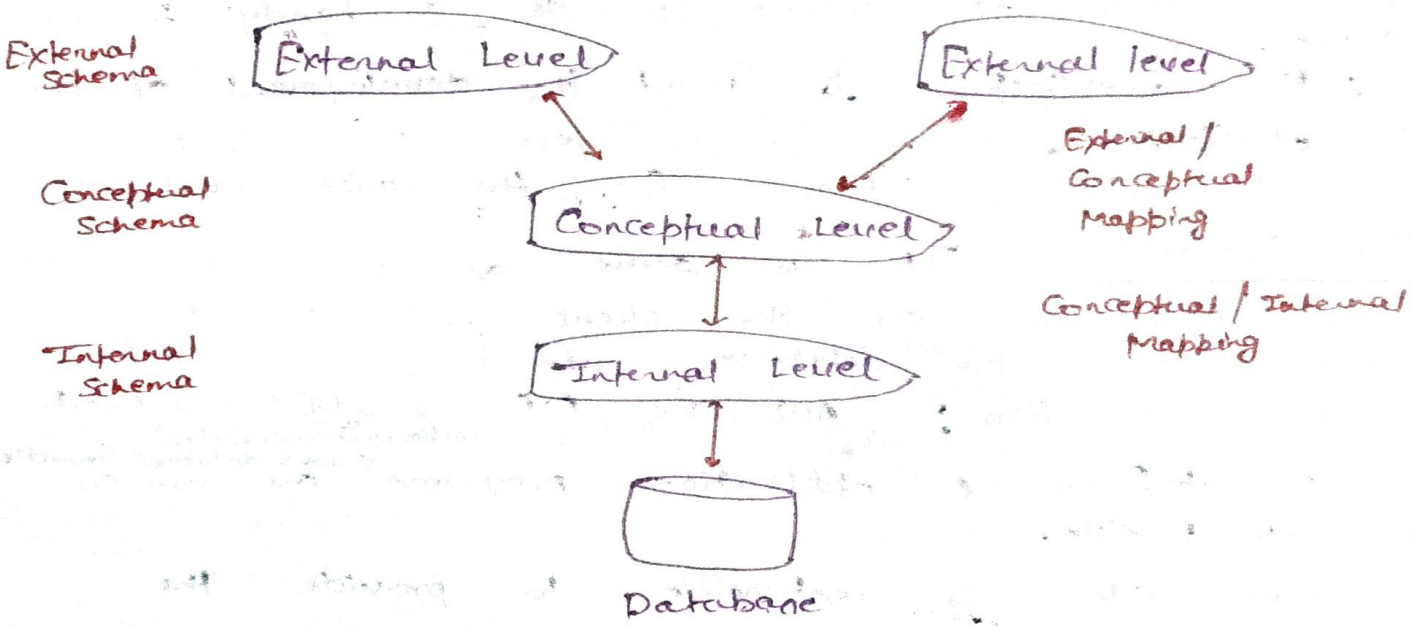
3-Tier Architecture :- It contains another layer b/w the client & server. In this, client can't directly communicate with server.

The application on the client-end interacts with an application server which further communicate with the database system.

End user has no idea about the existence of the database beyond the application server. The database also has no idea about any other user beyond the application.



Three level Architecture of DBMS :-



It is also known as 3 schema architecture of DBMS.

Mappings is used to transform the req. & response b/w various database levels of architecture.

It is not good for small database bcz it takes more time.

In External/conceptual mapping it is necessary to transform the req from external to conceptual schema.

In Conceptual / Internal mapping, DBMS transform the req from the conceptual to internal level.

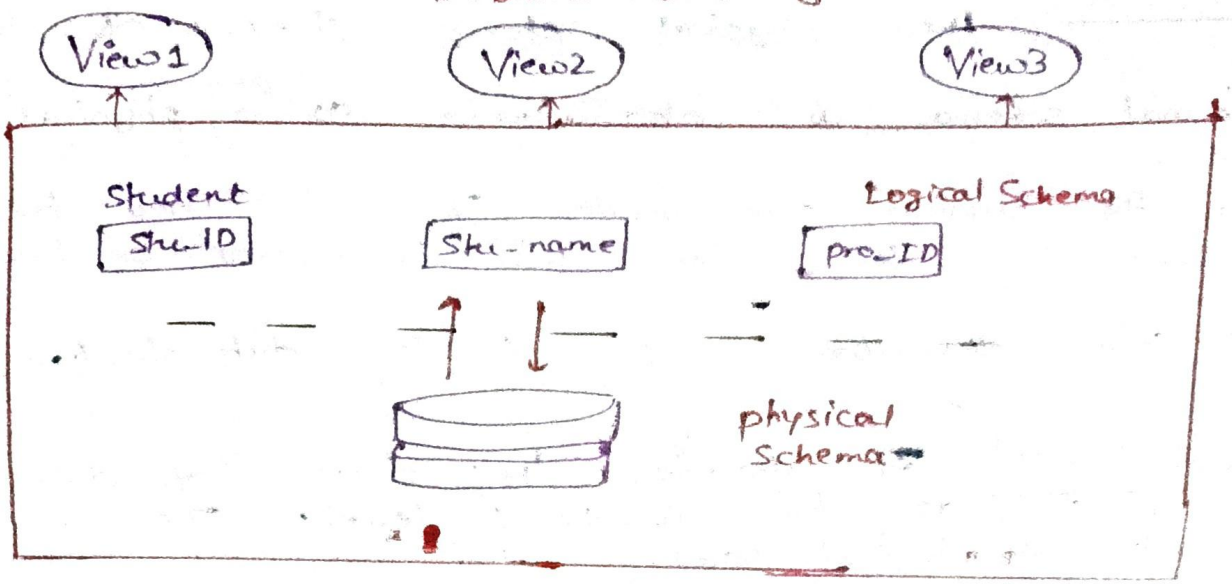
Internal Level:- It has internal schema which describes the physical storage structure of the database.
The internal schema is also known as a physical schema.
It uses the physical data model. It is used to define that how the data will be stored on the disk.
Physical level describes complex low-level data structure in detail.

Conceptual Level:- It describes the design of database at the conceptual level. It is also known as logical level.
It also describes the structure of whole database.
It also describes what data are to be stored in the database & what relationship exist among those data.
In conceptual level, internal details such as an implementation of the data structure are hidden.
Programmers & database administrators work at this level.

External level:- A database contains several schemas that sometimes called as subschema.
The subschema is used to describe the different view of the database.
An External schema is also know as View schema.
Each view schema describes the database part that a particular user group is interested & hides the remaining database from that user group.
The view schema describes the end user interactions with database systems.

Database Schema:- Skeleton that represent the logical view of the entire database.
Defines how the data is organized & how the relations among them are associated.
formulates all the constraints that are to be applied on the data.
Defines its entities & relationship among them.
Contains descriptive detail of database, which can depicted by means of schema diagrams. It's the database designer the schema to help programmers to understand database & make useful

Database Schema



A database schema divided into 2 categories-

① physical Database Schema :- This schema pertains to the actual storage of data & its form of storage like files, indices etc. It defines how the data will be stored in a secondary storage.

② Logical Database Schema :- This schema defines all the logical constraints that need to be applied on the data stored. It defines tables, views, & integrity constraints.

Database Instance :- Database Schema is the skelton structure of database. It is designed when the database doesn't exist at all.

Once the database is operational, it is very difficult to make any change to it.

A database schema does not contains any data or information.

A database instance is a state of operational database with data at any given time.

A DBMS ensures that its every instance (state) is in a valid state, by deligently following all the validations, constraints & conditions that the database designers have imposed.

If a database system is not multi-layered then it becomes difficult to make any change in the

the database system. Database system are designed multilayers.

The collection of information stored in a database at a particular moment is called an an instance of the database.

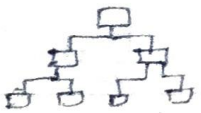
Data Models: Data models how the logical structure of database is modeled.

Data Models define how data is connected to each other & how they are processed & store inside the system.
First model was flat data-models, where all the data used are to be kept in the same plane.
Earlier data models were not so scientific. hence they were prone to introduce lots of duplication and update anamolies.

Significance → {computer efficient access easily manage structured}

Types of Data Models —

1. Hierarchical Data Model
2. Network Data Model
3. Relational Data Model
4. Object : Oriented Data Model
5. Graph Data Model
6. ER Data Model
7. Document Data Model

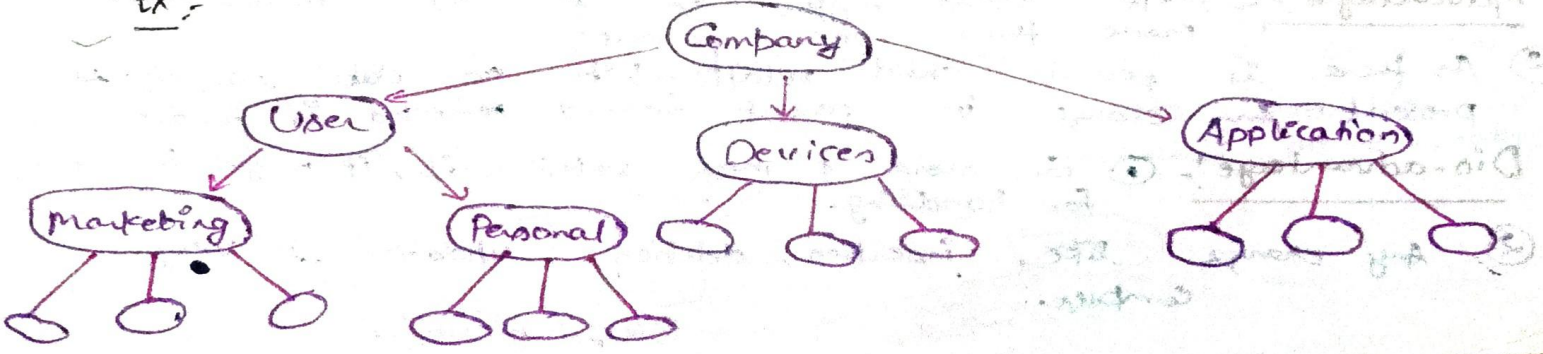


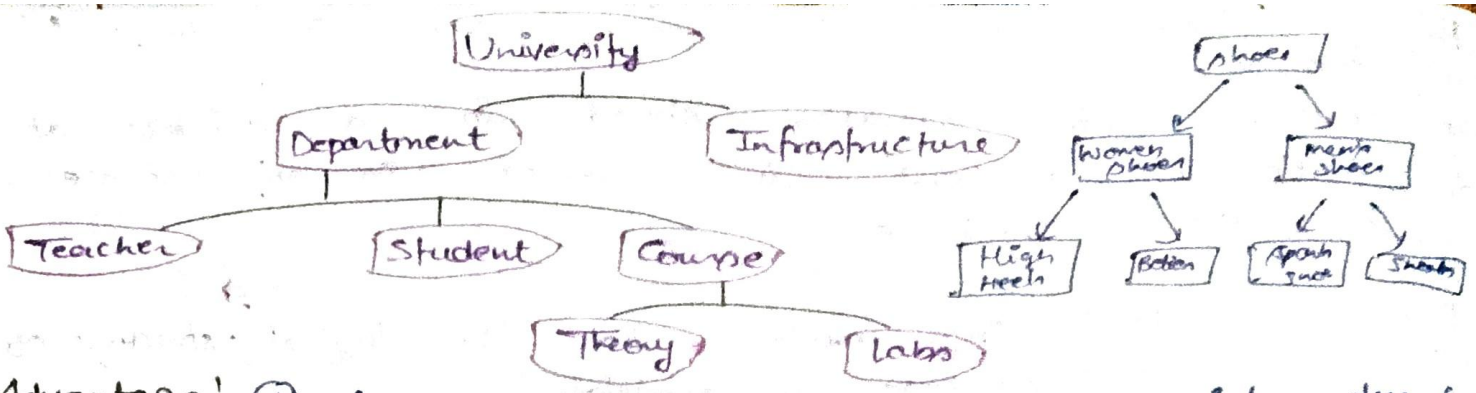
Ad → ① quick to navigate by linking nodes ② Simple ③ Efficient ④ lack of structural independence ⑤ Complex for big database

Dis → Insertion/D Anamolies ② Not flexible ③ many to many ↔ (IRM), Simplest & oldest models

Hierarchical Data Model: → IMS
- files are related in parent/child format.
- Like file system, this model also had some limitations like complex implementation, lack structural independence.
- It can't handle many-many relationships etc.
- It is also called IMS (Information Management System).

Ex:





Advantage!- ① An user retrieve data very quicly due to presence of explicit linking.

② All the link & sub links are depend on eachother. This can make simple to access the child nodes.

Dis-advantage!- ① When a user to add new child whose linking is not in structure, this couldn't be done.

② This type of linking becomes complex relationship. & problem of redundancy, which affects on accuracy due to inconsistency.

Network Data Model!- It was invented by charles Bachman. It uses network database to create a relationship b/w entities.

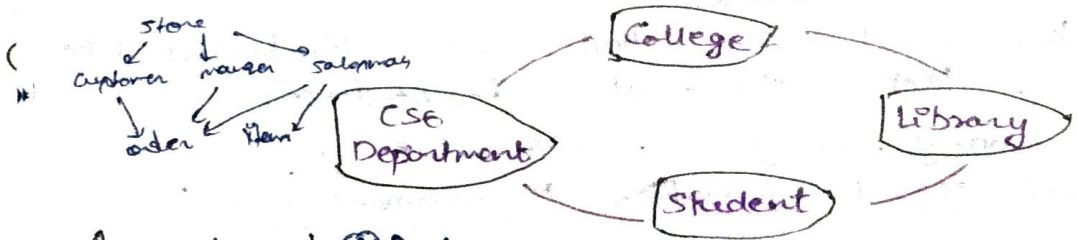
Network databases are similar to hierarchical databases differs with one key point that in network database one node can have a relationship with multiple entities In Network database, parent are occupier & children are members.

If organize many to many relationships.

Ex:- Integrated Data Store (IDS)

Adv → ① many to many
② Quick to navigate
③ Flexible
④ Data independency

Dis → ① for more relation, it complex
② lack of structural independence
③ Insertion & deletion anomalies



Advantage!- ① Data access faster as the child nodes have more than one parent.

② As there is parent child relationship so data integrity is present. Any change in parent record reflected in child record.

Dis-advantage!- ① for more & more relationship, it might be complex for handling.

② Any change like, insertion, deletion, updation is very complex.

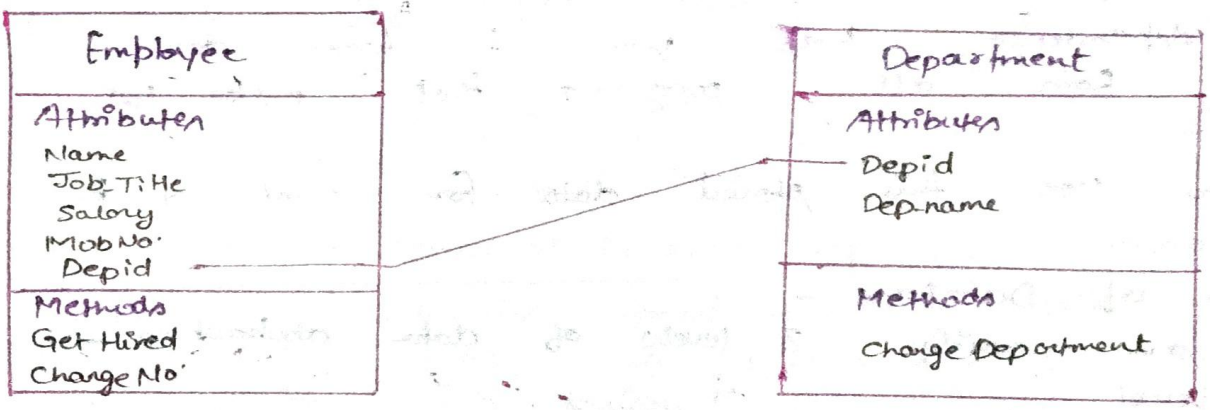
Relational Data Model:- It most widely used model. The data is maintained in the form of a two dimentional table. All information is stored in the form of row & columns. Basic structures is Table. So, Tables are called Relation in Relational Data Model.

Empid	Emp-name	Jobname	Salary	Mobile no	Dep-id	projectid
001	John	Engineer	70k	9187653210	2	99
002	Adam	Analyst	80k	9119740058	3	100
003	Kande	Manager	60k	8193063848	2	65

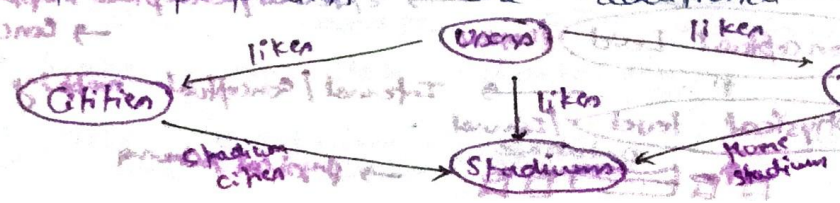
Advantage:- ① Simple, as compared to Network & Heirarichal.
② Scalable, we can add as many rows & columns.
③ Structure Independence.

Dis-advantage:- ① For hiding complexity & making things easier for user this more req. powerful hardware computers & data storage device.
② Bad Design

Object-Oriented Database:- Real world problems are represented through this model. In this, the data & relationship are present in a single structure known as object. two or more objects are connect through links.

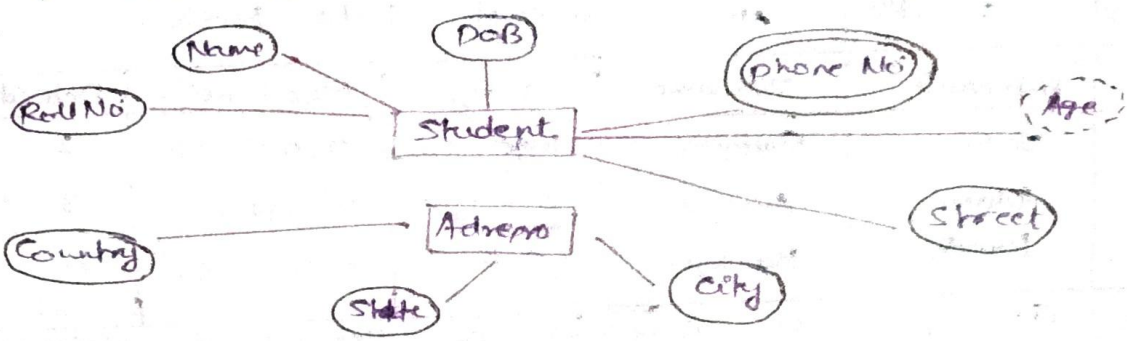


Graph Data Model:- No SQL database & it use the graphical structure for semantic queries. Data is stored in the form of nodes, edges & properties in which node eg. to record, the edge link between nodes & properties are additional information added in nodes.



ER Model Database :- Entity - Relationship Database Model was developed by Peter Chen 1976.

Each row in the table represent one instance of an object type, & each column in a table represent an attribute type.



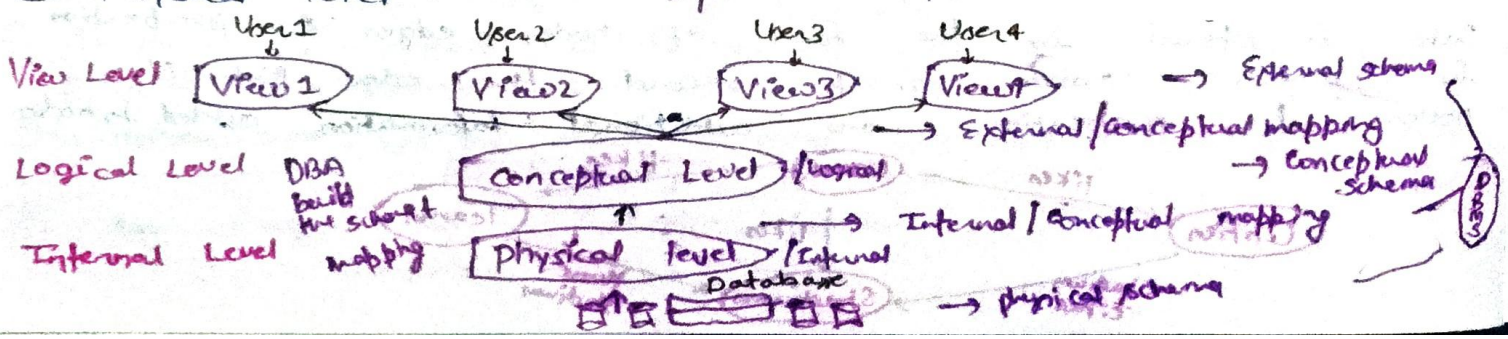
Document Database :- Also no SQL Database. Stores data in the form of documents which are key values. Each document makes relationship of the data with other data elements and attributes.
It popular due to its storage of documents & NO SQL properties. The speciality of NO SQL data storage is that it provides a faster mechanism for storing & searching for documents.

Data Independence :- It helps you to change the database schema at one level of database system without requiring to change the schema at the next heigher level.
Data independence helps you to keep data seprated from all programs that make use of it.
You can use this stored data for computing & presentation.

Levels of Database –
There are mainly 3 levels of data abstraction –

1. View level
2. Conceptual level
3. Physical level

3 tier level Architecture



View Level or External level :- This level tells the application about how the data should be shown to user. Example, if we have login id & password in a university system, then as a student, we can view our marks, fee, roll no etc.. If we talk about the staffs or faculty they have different view like salary, edit marks, attendence enter etc.. So, student & faculty view is different. This will increase security. So, different user will have different view & they are capable to acess their own view bcz they have no privilages to acess other's view.

Conceptual Level or Logical Level :- Logical data independence means that the conceptual schema (middle level) can be changed without affecting the existing external schema's.
This leveh tells how the data is actually stored & structured We have different data models by which we can store the data.
Example, of Relational data model, to store data. We have to store the data of student, the columns in the student name, age, mail-id, roll-no etc.. We have define all these while we are creating the database.
Though the data is stored in database, but the structure of the tables like the student table, teacher tables, books tables etc are defined here in the conceptual or logical level. And also how tables related to each other.
Overall we can say, we can creating a blue print of the data at the conceptual level.

physical Level or Internal Level :- It tells us that where the data is actually stored i.e., it tells the actual location of data that is being stored by user. The DBA decide that which data should be kept at which particular disk drive, & where it is stored. They decide if the data has to be centralized or distributed. Though we see the data in the form of tables at view level the data here is actually stored in the form of files only. It totally depends on DBA how he/she manage the database.

Two types of data Independence -

① Logical Data Independence (operations like updation, deletion)
higher level can't affected

② physical data Independence (new hdd, ssd, ide)

Database language And Interfaces:-

A DBMS has appropiate language and interfaces to express database queries & updates.

Database language can be used to read store & update the data in the database. It can be classified as:-

1. DDL
2. DML
3. DCL
4. TCL

- DML → procedural → Relational Algebra, Relational Calculus
- DML → Non-procedural → Formal QL, Commercial QL

5. DSL (Data Sub language)
   DSL = DDL + DML
6. SDL - (Storage Definational language) physical schema on physical storage or physical level
7. VDL → (View Definational language) → External schema or user level at high level of data abstraction.

DDL:- It stands for Data Definition Language. It is used to define database structure or pattern.

It is used to create schemas, tables, indices, constraints etc. in database. We can create Skelton of database using it.

The task performed by DDL are -

6. Host language → DML statement embedded into HLL.
4. 4GL → SQL, Spreadsheet, Report output & generator

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

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

DML:- It stands for Data Manipulation language. It is used for accessing & manipulating data in database. It handles user req.

The task performed by DML are:-

1. Select :- It is used to retrieve data from a database.
2. Insert :- It is used to insert in a table.
3. Update :- It is used to update existing data with in a table.
4. Delete :- It is used to delete all records / data from a table.
5. Merge :- It performs UPSERT operation i.e., insert or update operations.

⑥ Call :- It is used to call a SQL or Java sub program.

⑦ Explain Plan :- It has the parameter of explaining data.

⑧ Lock Table :- It controls concurrency.

DCL :- It stands for Data Control language. It is used to retrieve the stored or saved data.

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

(But in Oracle, the DCL language not have the feature of rolling back).

The task performed by DCL are –

① Grant :- Used to give a user access privileges to a database.

② Revoke :- It is used to take back permission from the user.

TCL :- TCL is used to run the changes made by DML statement. TCL can be grouped into a logical transaction.

The task performed by TCL are –

① Commit :- It used to save the transaction of database. permanently.

② Rollback :- It is used to get data or restore from the last save point or last commit state.

③ Savepoint :- It used to save data at a particular point temporily, so that whenever needed can be rollback to that particular point.

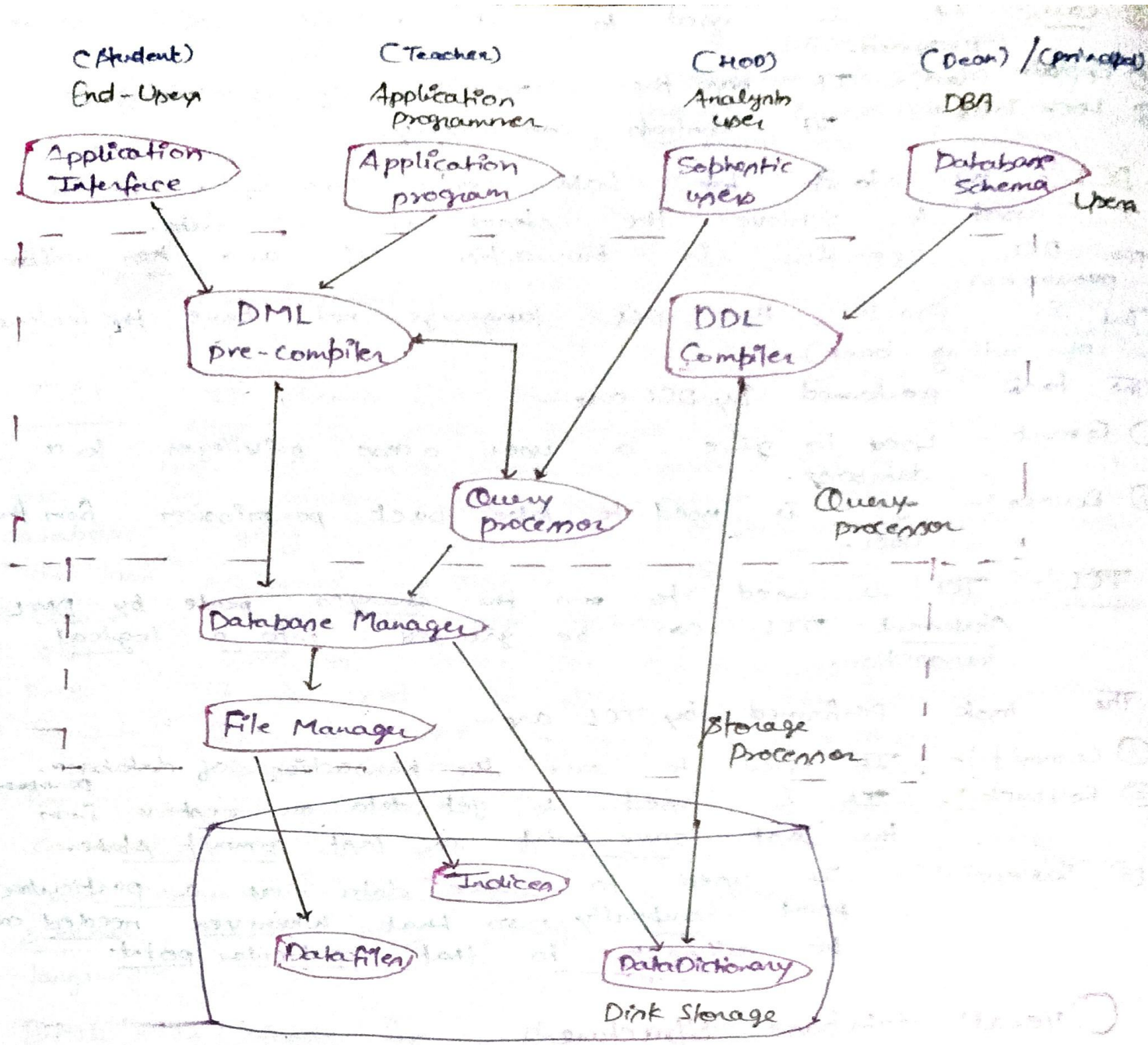
Overall database Structure :-

A database system is partitioned into modules that deal with each of the responsibilities of the overall system.

The functional components of a database system can be broadly into storage manager & the query processor components.

The storage manager is important b/z database typically req. a large amount of storage space.

The query processor is important b/z it helps the database system simply facilitate acess to data.



It divides into 4 parts :-

1. DBMS Users
2. Query Processor
3. Storage Processor
4. Disk Storage

DBMS User :- Database users are categorized based upon their interaction with the database.

There are 4 main types of DBMS Users-

① Naive / end User :- End user are unsophesticated who don't have any DBMS knowledge but they frequently used database application in their daily life to get their desired results.

Ex :- Railway ticket booking users are naive users. bcz they don't DBMS knowledge but they still use database & perform their given tasks.

② Application Programmer :- These are an backend programmers, who writes the code for the application program.
They are computer profesional.
These program could be written in programming language such as C, C++, Java etc..

③ Sophestic User :- These users are can be engineer, scientists, They don't write the program code but they interact the database by writing SQL queries directly through the query processor.
These are familiar with database.
They can develop their own database application a/c to their req.

④ DBA :- (Database Administrator), DBA is a person/team who defines the schema & also controls the 3 levels of database. [• Design the Conceptual database].

- The DBA will then create a new account id / password for user if he/she need to acess the database.
- DBA is responsible for providing security to the database & he allows only the authorized users to acess/ modify the database. • [physical Database design → • Sequential, Random (Contiguous) • Linked, Index (Non-contiguous)]
- DBA monitors the recovery & backup & provide technical support. • [check Security & integrity] (physical, macchine user)
- The DBA has a account in the DBMS which called a system or superuser account
- DBA repairs damage caused due to hardware / software failures. • Design backup & recovery procedure • Granting Acess.

Query Processor :- It interprets the req. (queries) received from end user via an application program into instruction.

- If also execute the user req. which is received from the DML compiler.

Query processor contains the following component -

① DDL Compiler :- The DDL statements are sent to DDL compiler, which converts these statements to a set of tables.
These tables contains the metadata concerning the database & are in the form that can be used by other components of the DBMS.

② DML Pre-compiler and Query Processor :- The DML Pre-compiler converts the DML statement embedded in an application program to normal procedure calls in the host language.
The Query processor components include -

- DDL Interpreter :- It processes the DDL statement into a set of table containing meta data (data about data).
- DML Compiler :- It processes the DML statements into low level instructions (machine language), so that they can be executed.
- Query Evaluation Engine :- Which executes low-level instructions, generated by the DML compiler.

Storage Manager / Processor :- It is a program that provides an interface btw the data stored in the database & the query received.
It is also known as database control system.
It maintains the consistency & integrity of the database by applying the constraints and executes DCL statements.
It is responsible for updating, storing, deleting and retrieving data in the database.
It contains following components -

① Authorization Manager :- It tests for satisfication of integrity constraints & checks the authority of users to access data.
② Transaction Manager :- Which ensure that database remains consistent state despite system failures,

4 that concurrent transaction execution proceed without conflict.

③ File Manager :- Which manages the allocation of space on disk storage & data structures used to represents information stored on disk.

④ Buffer Manager :- It responsible for access methods / encapsulates relational operation to read / write / allocate / deallocate pages.

Disk Storage :- It contains following components —

① Data files - It stores data.

② Data Dictionary :- It contains the information about the structure of any database object.
It is repository of information that governs metadata.
It is collection of names, definations & attributes about data elements that are being used or captured in a database, information system, or part of a research project.

③ Indices :- It provides fast access to data items that hold particular values.

Entity :- An entity can be a real-world object, that can be easily identifiable.
For ex- in a school database, students, teachers & courses offered can be considered as entities.
All these entities have some attributes or properties that give them their identify.
An entity set is a collection of similar types of entities.
An entity set may contain entities with attribute sharing similar values.
Ex- a Students set may contain all the students of University, Similarly a Teacher set may contain all the teachers of University from all faculties.

Attributes :- Entities are represented by means of their properties. called as attributes.
All attributes have value. Ex- a student entity may have name, class & age as attributes.

There exists a domain or range of values that can be assigned to attributes. Ex, a student name can't be a numeric value. It has to be alphabetic.
OR A student age can't be negative.

Types of Attributes:-

① Simple Attributes:- Are atomic values, which can't be divide further.
Ex- a student phone no in an atomic value of 10 digits.

② Composite Attributes:- It is an made of more than one simple attribute.
Ex- student name contain first name & last name.

③ Derived Attribute:- Attributes that do not exist in physical database, but their values are derived from other attributes present in database.
Ex- average-salary in a department should not be saved directly in the database, instead it can be derived. OR age can be derived data-of-birth.

④ Single Value attribute:- Contain single value.
Ex- Social-Security-Number.

⑤ Multi-Valued Attribute:- Contain may contain more than one values.
Ex- a person can have more than one phone no, email-adress.

Keys:- It is an attribute or collection of attribute that uniquely identifies an entity among entity set.
Ex- the roll-no of a student makes him/her identifiable among students.

Types of keys:-

- Super key
- Unique key
- Primary key
- Alternate key
- Candidate key
- Foreign key

① Candidate key – A set of attributes that uniquely identifies a record. Among set of candidate key is chosen as a Primary key. So, a table can have multiple candidate key but each table can have maximum one primary key.

② Primary key :- It uniquely identifies a record & must never be the same for the 2 records. (Not null + no redundant value.)
A table can have only one primary key & one candidate key can select as a primary key. The primary key should be choosen such that attribute are never or rarely changed.

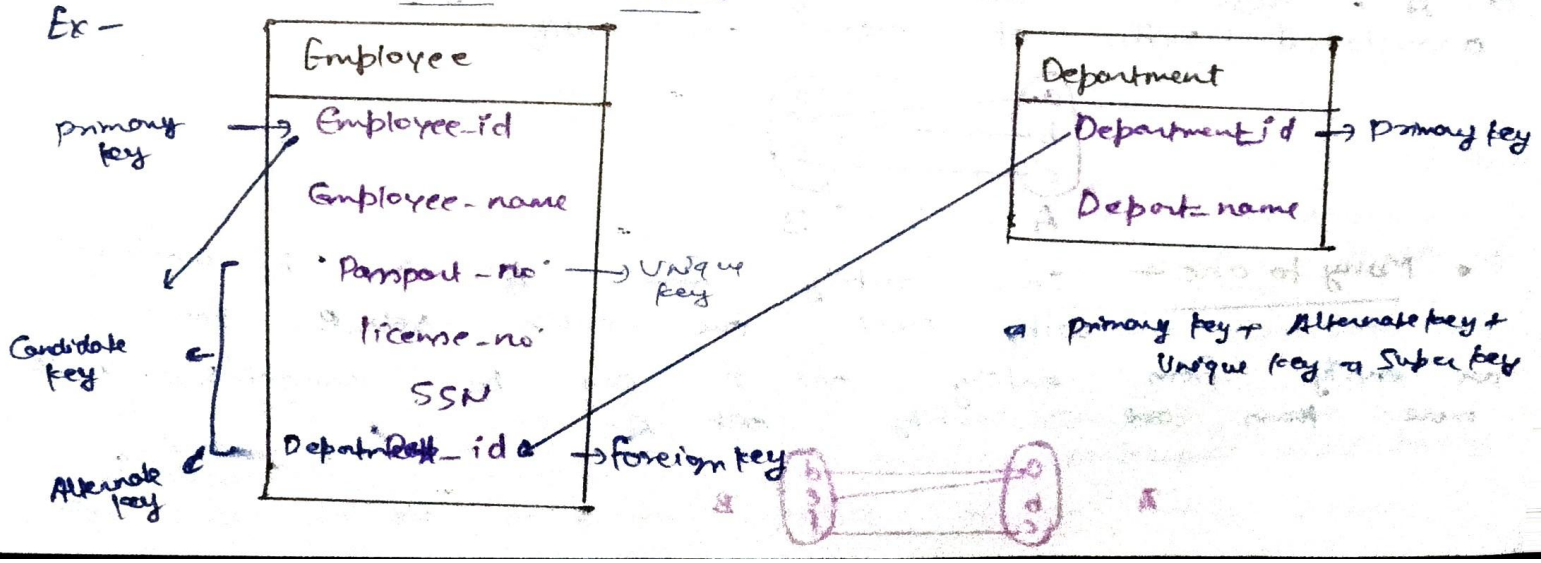
③ Alternate key :- Alternate key are candidate keys that are not selected as primary key.
Alternate key is also called "Secondary key".

④ Unique key :- It uniquely identifies record from a table, similiar to primary key but it contains "NULL" value whereas primary key doesn't allow "NULL".

⑤ Super key :- It is an set of more than one keys can be used to find uniquely record in a table. It combine form of primary key, Alternate key, Unique key. These key are subset of super key.

⑥ Foreign key :- Foreign keys are the column of a table which is used to point primary key of another table.
It can connect tables.

Ex –



Relationship:- The association among entities called Relationship.

Ex:- An employee works_at a department, a student enroll in a course. Here works-at & enroll are called relationship.

Degree of Relationship:- The no of participating entities in a relationship defines the degree of the relationship.

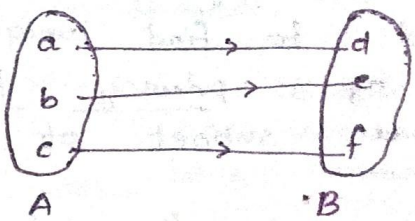
Ex- Binary = degree 2, Ternary = degree 3

Relationship Set:- A set of relationship of similiar type called a relationship set.

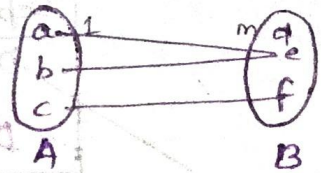
Like entities, a relationship too can have attributes. These attributes are called descriptive attributes.

Mapping Cardinalities:- It defines the no of entities in one entity set, which can be associated with the no of entities of other set via relationship set.

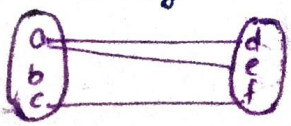
- **one-one** — One entity from entity set A can be associated with at most one entity of entity set B & vice versa.



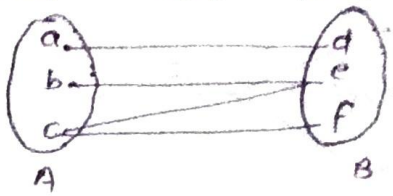
- **one-to-many** — One entity from entity set B can be associated with more than one entities of entity set A, however an entity from entity set A, can be associated with at most one entity



- **Many to one** — One entity from A can be associated with most one entity of set B, however an entity from entity set B can be associated with more than one entity set A.

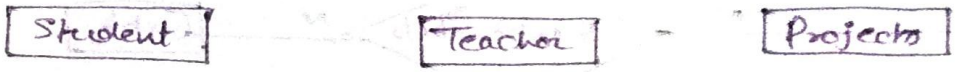


- **many to many**: one entity from A can be associated with more than one entity from B vice-versa.

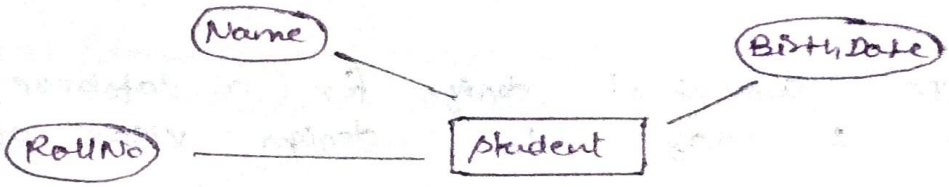


Entity - Relationship Diagram:- ER Model is represented by means of ER diagram. Any object, for ex- entities, attributes of entity, relationship sets, & attribute of relationship sets, can be represented with the help of E-R diagram.

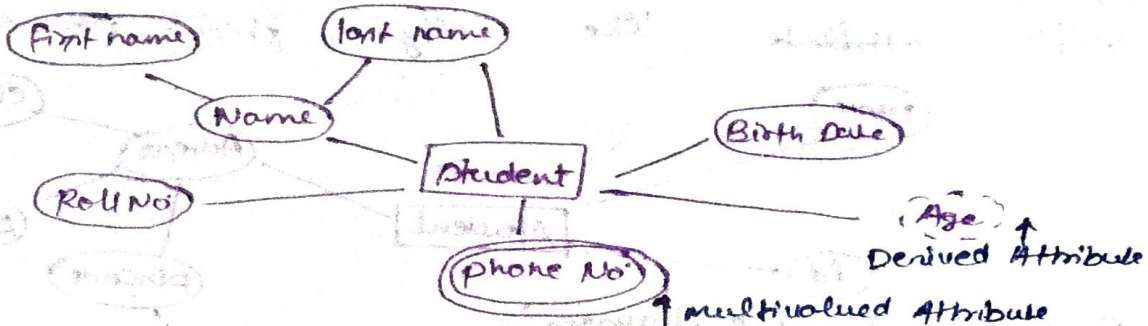
Entity:- Entities are represent by means of Rectangles.



Attributes:- Properties of entities. Represent by means of ellipse. Every ellipse represent one attribute & is directly connected to its entity (rectangle).



If the attributes are composite, they are further divided in tree like structure. Every node is then connected to its attribute. That is, composite attribute that are represented by ellipse that are connected with an ellipse.



Relationship:- Represented in diamond box, written inside of box. (name)

All entity participating in a relationship, are connected to it by line.

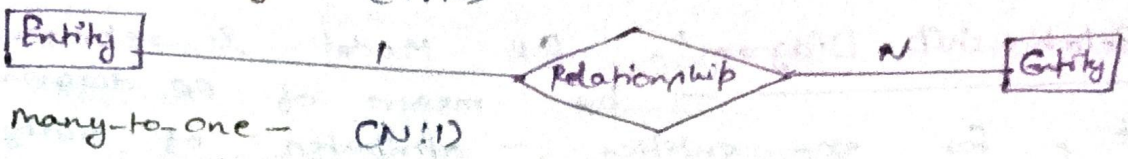
Binary Relationship And Cardinality:- A Relationship where two entities participate called binary relationship.

Cardinality is the no. of instance of an entity from a

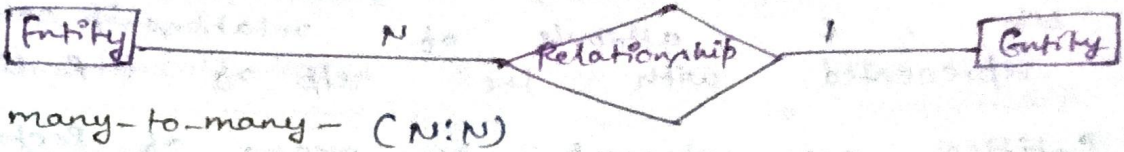
relation that can be associated with the relation.

✓ one-one - (1:1)

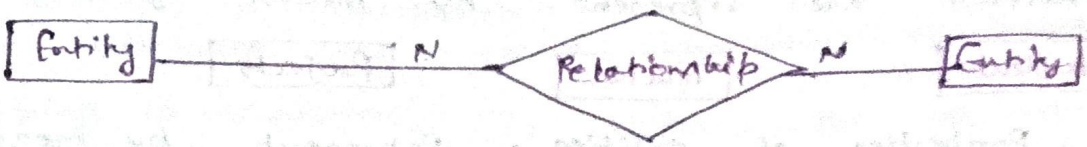
✓ one-to-many - (1:N)



✓ many-to-one - (N:1)



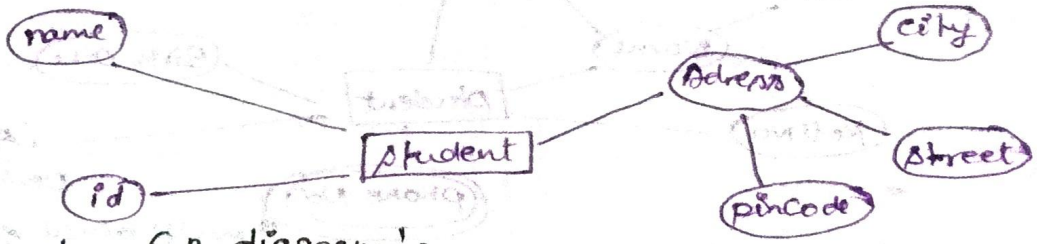
✓ many-to-many - (N:N)



E-R Model stands for Entity-Relationship model, high-level data model used to define the data elements & relationship for a specified system.

It develops conceptual design for a database, & simple & easy to design view of data.

Example:- Suppose, we design a school database. In this database the student will be an entity with attributes like adress, name, id, age, etc.. The adress can be another entity with attribute like city, street, phcode etc..



Components of E-R diagram:-

ER Model

- Entity
  - Weak Entity → ▭
  - Strong Entity
- Attribute
  - key Attribute
  - Composite Attribute
  - Multivalued Attribute
  - Derived Attribute
- Relation
  - one-one
  - one-many
  - many-one
  - many-many

Notation :-

 → Entity

 → Relationship

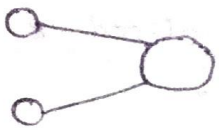
 → Attribute

 → Weak Entity

 → Weak Entity Relationship

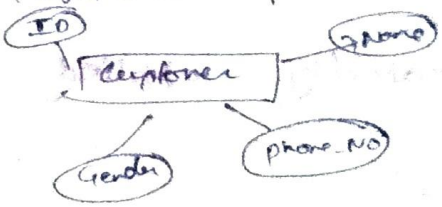
 → Multivalued Attribute

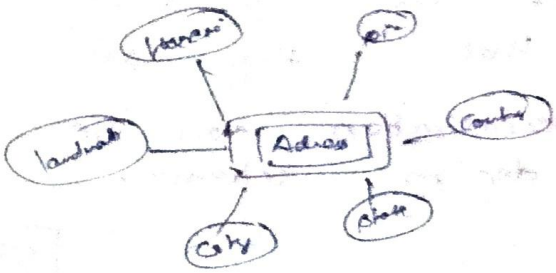
 → key Multivalued Attribute

 → Composite Attribute

- A Strong Entity is independent, whereas a weak entity is relyant on another point.
- A Weak Strong Entity does not have any key attribute on its own, but strong has key.

Entity → Tangible (object touch or see) Ex- a mobile
Intangible (object can't see or touch) Ex- bank account



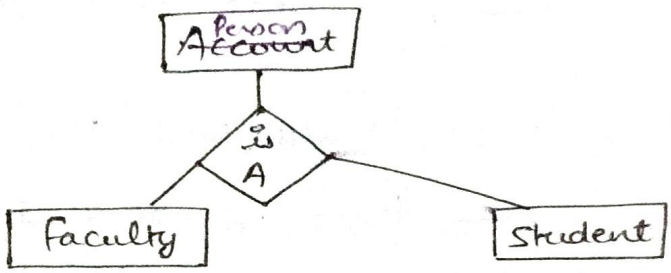


Generalization :- • It is like bottom-up approach in which two or more entities of lower level combine to form heigher entity.

* In Generalization, the heigher level entity can also combine with other lower level entity to make further high level entity.

• Generalization proceeds from the recognition that a no' of entity sets share some common features. On the basis of the commonalities, generalization synthesis these entity sets into a single, heigher-level entity set.

• Subclass combine to make superclass.



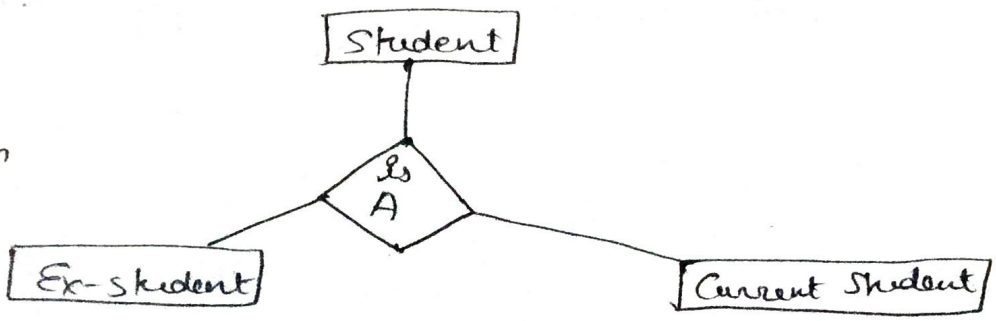
Bottom Up Approach

• Faculty & student entities generalized & create a heigher level entity person.

Specialization :- • Opposite to Generalization, it is a top-down approach in which one heigher level entity can be broken into lower level entity. super breakdown & make sub class

• Ex- The specialization of Student to recognize that they are Ex-student or Current student.

• Specialization can be repeatedly applied refine a design schema.

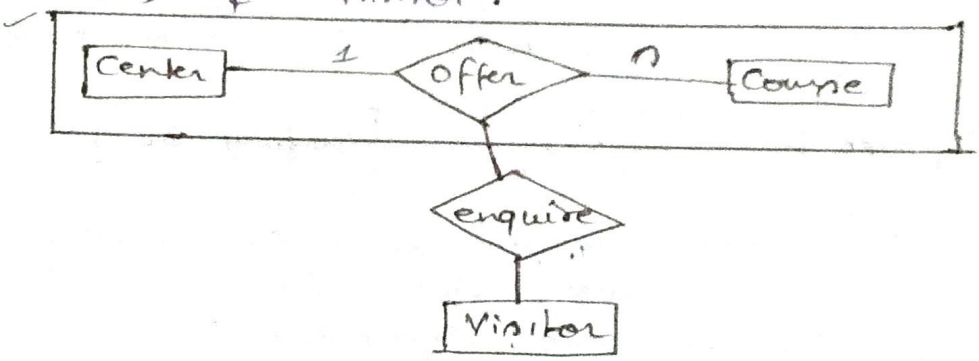


Top Down Approach

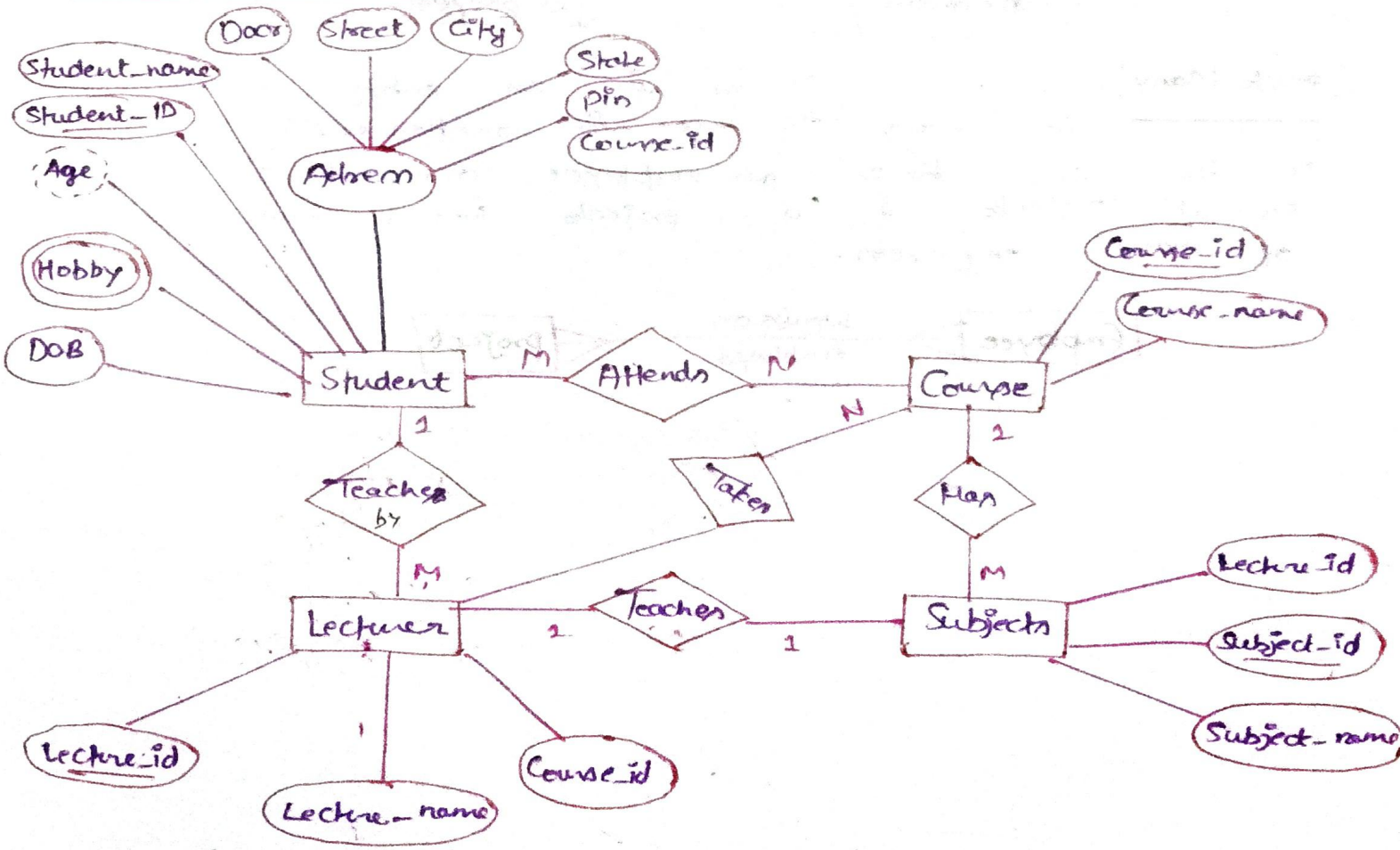
Aggregation!- One limitation of E-R model is that it can't express relationships. To illustrate the need for such an a construct, quaternary relationship are used which lead to redundancy in data storage.

- The best way to make such situation to use aggregation.

- Aggregation is an abstraction through which relationship are treated as heigher level entities.
- Ex - Relation blw offer (which is binary relation blw center & course) & visitor.



Example of E-R Diagram!-



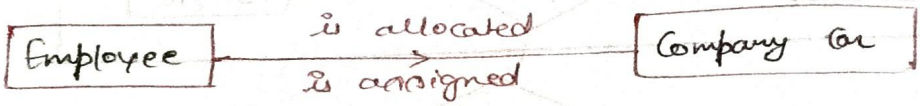
Relationship of Heigher Degree :-

The degree of relationship can be defined as the no of occurences in one entity that is associated with the no of occurence in another entity.

There are 3 degree of relationship -

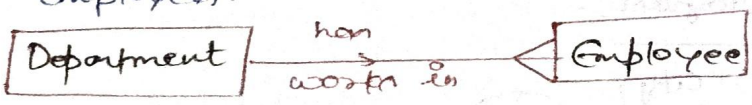
- one - to - one (1:1)
- one - to - many (1:M)
- Many - to - many (M:N)

One-to-One :- One occurence of an entity relates to only one occurence in another entity.



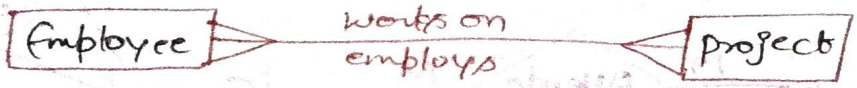
One - to - many :- One occurence in an entity relates to many occurences in another entity.

An Employee work in one department, but a department has many employees.



Many to Many :- Many occurences in an entity relate to many occurence in another entity.

At the same time, an employee can work on several projects, & a projects has a team of many employees.



Relational Data Model and Languages:-

- It can be represented on a table with columns & rows. Each row known as tuple & columns as an attribute. (Structured format)
- **Relational Instance**: In relational database system, the relational instance represents by a finite set of tuples. Relational instances do not have a duplicate tuples.
- **Relational Schema**:- A relational schema contains the name of relation & name of all columns or attributes.
- **Relational key**: In relational key, each row has one or more attributes. It can identify the row in the relational uniquely.

Name	Roll.No	Course	AGE
Rohit	1	MCA	21
Vaman	2	B.TECH	20
Ayush	3	BBA	20
Shiv	4	MBA	21

→ Attributes

Cardinality = no of tuples = 4

tuples

degree of schema = no of columns = 4

Properties:
- Name of the relation is distinct from all other relations.
- Each relation cell contains atomic values.
- Each attribute contains distinct name.
- Tuple has no duplicate value.
- Order of tuple can have different sequence.
- Values of tables are related to each other.
- If possible to run multiples queries across table at once.

Integrity Constraints:-
- Set of rules used to maintain quality of information.
- It ensures insertion, deletion, updating & other processes have to be performed in such a way that data integrity is not affected.

Types:-
- Domain Constraint
- Entity Integrity Constraint
- Referential Integrity Constraint
- key Constraint

Domain Constraints:- • It defined as the definition of a valid set of values for an attribute.

- The data type of domain includes string, character, integer, time, date, currency etc. The value of the attribute must be available in the corresponding domain.

Ex:-

ID	NAME	SEM	AGE
1	Rohit	1st	17
2	Ram	2nd	22
3	Raman	3rd	19

Not allowed, bcz AGE is an integer attribute but it contain character value.

Entity Integrity Constraints:- • It states that primary key value can't be null.

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

Ex:-

Emp_id	Emp_Name	Salary
123	Jack	30k
142	Harry	60k
164	Jack	70k
	Honey	20k

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

Referential Integrity Constraints:- It specified b/w two tables.

- If a foreign key in Table 1 refers to the primary key of Table 2, then every value of foreign key in Table 1 must be null or be available in Table 2.

Ex:

Table 1

Emp_Name	Name	AGE	D_No → foreign key
1	Jack	21	11
2	Ram	20	24
3	Sam	23	10 → Not allowed, D_No is not defined in primary key of Table 2.
4	John	14	13

primary key

DNO	DLocation
11	Mumbai
24	Delhi
13	Noida

Table 2

Key-Constraints: The entity set that is used to identify an entity with in its entity set uniquely.

An entity set can have multiple keys, but out of which one key will be the primary key. A primary key can contain a unique value in the relational table.

Ex:-

ID	NAME	SEM	AGE
1001	Ram	1st	17
1002	Raman	2nd	24
1003	Vishnu	3rd	21
1002	Vishal	1st	19
	Rohit	2nd	22

If contain duplicate value, which violates primary key rule, bcz primary key must be unique.

Primary can't be NULL.

Relational Calculus: (domain dependent)

- It is an non-procedural query language. In NPQL, the user is not concerned with the details of how to obtain the end results.
- It tells what to do but never explains how to do.

There are two types -

- Touple Relational Calculus
- Domain Relational Calculus

Touple Relational Calculus:

- It is specified to select the touples in a relation.
- In TRC, filtering variable uses the touples of a relation.
- The result of the relation can have one or more touples.

Logical → OR, AND, NOT

Notation: $\{T \mid P(T)$ or $T \mid \text{Condition}(T)\}$

T = Resulting Touples

P(T) → condition used to fetch T.

Example: $T.name \mid Author(T)$ AND $T.article = 'database'\}$

Output → The query select the touples from the AUTHOR relation. It returns a touple with 'name' from Author who has written on 'database'.

2. $\{R \mid \exists T \in Author(T.article = 'database' \text{ AND } R.name = T.name)\}$

* **Domain Relational Calculus (DRC)**:- The second form of relation is known as Domain relational calculus. In domain relational calculus, filtering variable uses the domain of attributes.

- Domain relational calculus uses the same operators as touple calculus. It uses logical connective $\wedge$ (and), $\vee$ (or) & $\neg$ (not).
- It uses Existential ($\exists$) & Universal Quantifiers ($\forall$) to bind the variable.

Notation:- $\{a_1, a_2, a_3 \ldots a_n \mid P(a_1, a_2, a_3 \ldots, a_n)\}$

where a_1, a_2 attributes

P stands for formula by Prime attributes

Example:- 1. $\{<$article, page, subject$>\} \in$ Javapoint $\wedge$ subject = 'database'$\}$

Output → The article, page, & subject from the relation javatpoint, where subject database.

Relational Algebra:- → Theoritical foundation for SQL queries

Relation algebra is a procedural query language, which takes instance of relations as input & yields instances of relations as output.

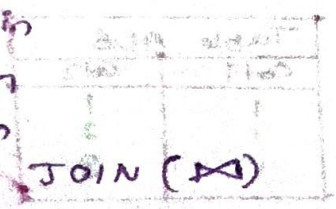
- It uses operator to perform queries. (on relations & we can get a relation as a result which has req. data). An operator can be either uniary or binary. They accept their relation as their input & yield relation as their output.

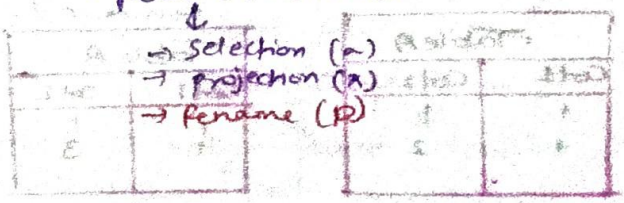
Fundamental Operation of relational algebra –

- SELECT (σ)
- Projection (π)
- Rename (ρ)
- Union Operation ($\cup$)
- Set Difference ($-$)
- Intersection
- Cartesian Product ($\times$)
- Join Operations
- Inner Join
- Theta Join
- EQUI Join
- NATURAL JOIN ($\bowtie$)
- OUTER JOIN
- Left Outer Join (A B)
- Right Outer Join (A B)
- Full Outer Join (A B)

It supports two types of operations

- Set-oriented → $\cup, \cap, -, \times$
- Relation-oriented
  - → Selection (σ)
  - → Projection (π)
  - → Rename (ρ)





① SELECTION (σ) :-

It used to select subset of the tuples a/c to given selection condition.

$\sigma_p(r)$ σ → is the predicate
r → relation
P → prepositional logic

Ex- σ topic = "Database" AND author = "Keith" (Tutorials)

② Projection (π) :-

It defines a relation that contains a vertical subset of Relation.

This helps to extract the values of specified attributes to eliminates duplicate values. (π) symbol to choose attributes from a relation.

$\pi_p(\sigma)$

Ex-

Cusid	Custname	Status
1	Google	Active
2	Amazon	Active
3	Apple	Inactive

π custname, status (Customers)

Custname	status
Google	Active
Amazon	Active
Apple	Inactive

③ Rename (p) :-

It is a unary operations used for renaming attributes of a relation.

$\rho(a/b)$ R will rename the attribute 'b' of relation by 'a'.

④ Union Operation (U) :-

It includes all the tuples are in tables A or in B.

It also eliminates duplicate tuples. So, set UNION set B would be AUB.

Rules → • R & S must be same no of attributes
• Attributes domain need to be compatible.
• Duplicate tuples should be automatically removed.

Table A

Col1	Col2
1	1
1	2

Table B

Col1	Col2
1	1
1	3

→

Table AUB

Col1	Col2
1	1
1	2
1	3

(5) Set Difference (−) :- The result (A-B), is a relation which includes all tuples that are in A but not in B.
The attribute name of A has to match with the attribute name in B.
The two - operand relation A & B should be either compatible or union - compatible.

Table A-B	
col 1	col 2
1	2

(6) Intersection (∩) :- A Relation consisting of a set of all tuples that are in both A & B. However, A & B must be union compatible.

Table A∩B	
col 1	col 2
1	1

(7) Cartesian Product (X) :- It is an operation used to merge columns from two relations.
It is also called Cross Product or Cross Join.

Table A	
col 1	col 2
A	B
B	C

Table B	
col 1	col 2
C	D
E	f

Table A X B			
col 1	col 2	col 3	col 4
A	B	C	D
A	B	E	f
B	C	C	D
B	C	E	f

Join Operations :- Join operations is essential a cartesian product followed by a selection criterion. It denoted by ⋈.
Join operation also allows joining variously related tuples from different relations.

Types :- Various forms —
- Inner Joins
  - ⇒ Theta Join
  - ⇒ EQUI Join
  - ⇒ Natural Join
- Outer Join
  - ⇒ Left Outer Join
  - ⇒ Right Outer Join
  - ⇒ Full Outer Join

INNER JOIN :- In an inner join, only those tuples that satisfy the matching criteria are included, while rest are excluded.

1. Theta Join!- Allows us to merge two tables based on the condition represent by theta.

Theta join works for all comparision operator.
Denoted by ⋈.

Ex- STUDENT ⋈ Student.Std = Subject.Class SUBJECT

$T_1 \bowtie_{T_1.colname1 = T_2.colname2} T_2$

2. Equi Join!- When theta operator uses only equality comparision operator, it is said to be equijoin.

3. Natural join!- • Doesn't use comparision operator.

- Doesn't concatenate the way cartesian product does.
- We can perform a Natural join only if there at least one common attribute that exist b/w two relations.
- In addition, the attributes must have same name & domain.
- Natural join acts on those matching attributes where the values of attributes in both the relations are same.

Outer Join!- We need to use outer joins to include all the tuples from the participating relations in the resulting relation. There are 3 kinds —

- left outerjoin
- right outerjoin
- full outerjoin

1. Left Outer Join!- All the tuples from the left relation, R, are included in the resulting relation. If there are tuples in R without any matching tuple in the Right relation S, then the S-attribute of the resulting are made NULL.

2. Right Outer Join!- All the tuples from the Right relation S, are included in the resulting relation. If there are tuples in S without any matching tuple in R, then the

R- attributes of resulting are made NULL.

3. FULL - Outer Join:- All the tuples from both participating relations are included in resulting relation. If there are no matching tuples for both relations, their respectives unmatched attributes are made NULL.

Features of RDBMS: ① Every piece of information is stored in the form of tables.

② Has primary key for unique identification of rows & foreign key to relate tables each other.

③ Provide SQL for data access.

④ Use indexes for faster data retrieval.

⑤ Give access privileges to ensure data security.

RDBMS Vs TRADITIONAL APPROACH: • The key differences is that RDBMS applications to store data in a tabular form, whereas in traditional approach, applications store data as files.

• There can be, but there will no "relation" b/w the tables, like in a RDBMS. In traditional approach, data is generally stored in either heirarchial form / navigational form. This means that a single data unit will have 0, 1s or more children nodes with, 1 parent node.

SQL: (Structured Query language)

• It is used to perform insertion, deletion, updation etc. some operation on the data stored in database.
• It design & managing the RDBMS.
• It provides interaction of user to database. Ex Oracle, MySQL etc.
• It is a database query language, it is based on relational algebra & tuple relational calculus.

Needs: • To create new, database & views.
• To insert records in a database.
• To update records in a database.
• To delete records in a database.
• To retrieve records in a database.

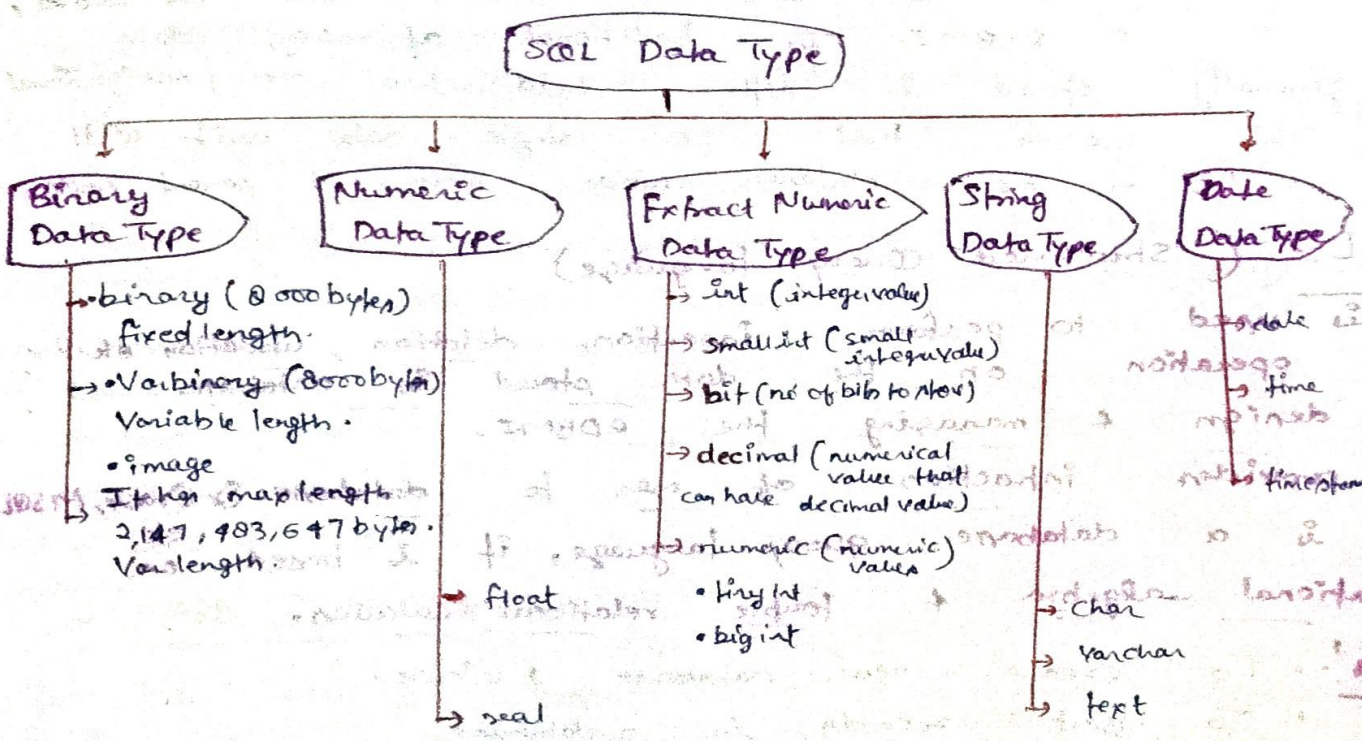
Functions: • It can be helped us to SQL queries, using like english statements.
• It allows user to create & drop database data.
• It allows the user to create a views, stored procedure & function.

- It gaves some privilledges to users.

Characteristics:
- Easy to learn, used for accessing data from RDBMS.
- Used to define data & manipulate when it needed.
- It allows us to set permissions on tables / on procedures, views.
- It provides backup & recovery.
- We can query for our database, using English like statement.

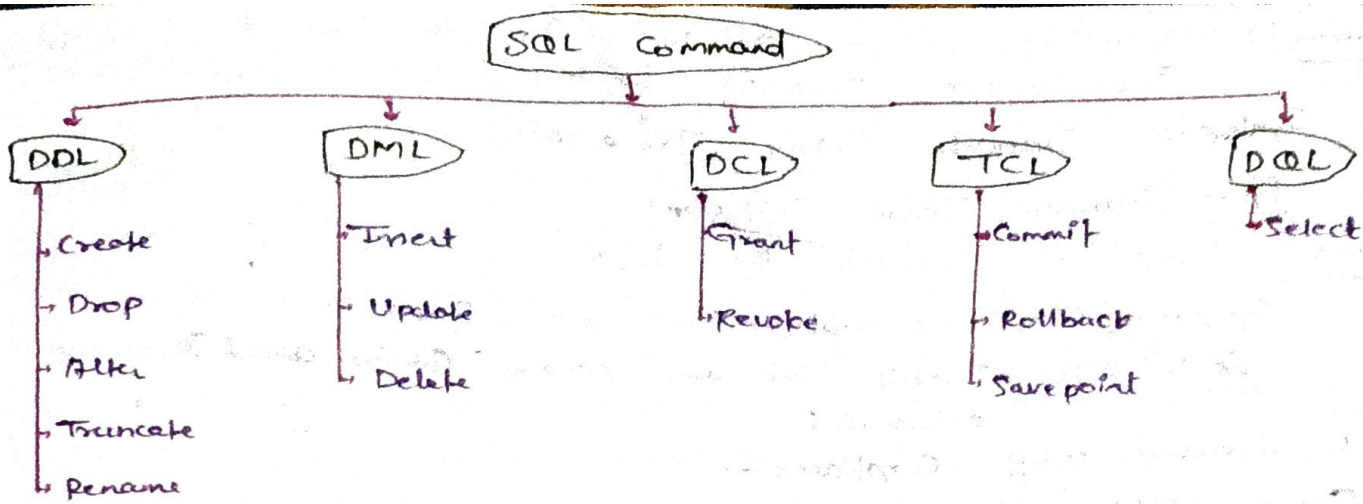
Advantage:
- High speed
- No coding needed
- Portability
- Interactive language
- Multiple data view

SQL Data Types & Literals: It is used to define the values that a column can contain. These are different types.



SQL Commands: SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission to users.

There are 5 types of SQL commands —



SQL Constraints:

Constraints	Description
Not NULL	Column doesn't have NULL value.
Default	When none is specified provides default value
Unique	Values in column is different.
Primary	Uniquely identify data, Notnull, noredundancy in that column.
Check	All values in column satisfy certain conditions.
index	Creates & retrieves database very quickly.

① DDL:- It used create an structure of database, table, or deleting, altering a table.

- **Create**: Create a new table in database.

  Syntax→ CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPE [size],-);

  Ex- Create Table Employee (Name VARCHAR (20), DOB Date);

- **Drop**:- It is used to delete structure & record store in database.

  Syntax→ DROP TABLE Table_Name;

  Ex:- Drop Table Employee;

② **Alter**:- It is used to add new attribute in existing table.

Syntax → • Alter Table table_name ADD (Column_name Datatype (size));
• Alter Table Table_name Modify (column defination);

Ex → • Alter Table Student ADD (Age Number (2));
• Alter Table Student Modify (Name Varchar (20));

- **Truncate**:- Used to delete all rows from the table & free space containing the table.

  Syntax → Truncate Table Table_name;

  Ex - Truncate Table Employee;

- **Rename**:- Used to rename the table.

  Syntax → Alter Table employeeen Rename Column Contact To Job_code;

  Ex :- Alter Table Tablename Rename Column column1 to column2;

Create Command using Constraints –

```
Creak Table Employees (
    emp_id INT(10) NOT NULL,
    first_name varchar(20),
    last_name Varchar(20) NOTNULL,
    salary int(10) NOTNULL,
    primary key (emp_id));
```

DML:- It is used to do manipulation operation on stored data in a table using insert, delete, Update.

- **Insert**:- Used to insert data values.

  Syntax → Insert into Table_name VALUES (column_value1, column_value2 ...);

  Ex → Insert into employeen Values (101, 'Harry', 'Potter', 2000);

  OR

  Insert into employee (emp_id, first_name, last_name, Salary) Values (102, 'Rohit', 'kumar', 3000);

- **Update**:- Used to update data values.

  Syntax → Update table_name SET Column_name = 'value' where column name = Value;

  Ex → Update employeen SET last_name = 'kumar' where emp_id = 101;

- **Delete**:- Used to delete data.

  Syntax → Delete From table_name where column_name = Values;

  Ex → Delete From employeen Where emp_id IN (101, 103);

DCL :- Used to give acess & withdraw acess privileges with the grant command, & revoke command.

- Grant :- Give acess privileges to database
- Revoke :- Withdraw acess privileges given with the grant command.

Syntax :-

Grant <privilege list> ON <Relation name> To <USER>

Revoke <privilege list> ON <relation name> From <USER>

TCL :- Used in transaction operations such as commit, savepoint, rollback.

- Commit :- Saves the work done
- Rollback :- Restore database to origin state since the last commit.
- Savepoint :- Identify a point in a transaction to which you can roll back later.

DQL :- Used to acessing the database using select Command.

Syntax :- Select columnname, columnname2 relationname ;

Ex :- Select * From Students ;

SQL Operators - Filter :- Where clause :- Used to fetching the data from a table.

- We can use this clause with SELECT, UPDATE, DELETE Statement.

Ex :- Select * From Students Where emp_id = 101 ;

SQL Operator - Logical :-

Operator	Example	Result
AND	(5<2) AND (5>2)	FALSE
OR	(5<2) OR (5>3)	TRUE
NOT	NOT (5<2)	TRUE

Ex :-

- Select * From Students Where f_name = 'Rohit' and salary = '15000 ;
- Select * From Students Where f_name = 'Rohit' or salary = 15000 ;
- Select * From students where f_name = 'Rohit and salary != 10,000 ;

SQL Operators - Comparision:

Ex:- • Select * from employees where first_name = 'Ronit' AND salary <= 10000;

Comparision Operators

Symbol	Meaning
=	Equal to
>=	Greater than equal to
<	less than
<=	greater than equal to
<> or !=	not equal to
>	greater than equal to

SQL Operators - Special:

Ex:- • Select * from employeen where salary between 10000 and 20000;

• select * from employeen where firstname like '%s'; '_s%'
y can be any character

• select * from employeen where salary is null;

• Select * from employeen where salary in (10000, 12000, 20000);

• Select DISTINCT (first_name) from employees;

Special Operators

Between	Check an attribute value with in range
Like	Check an attribute value matches a given string pattern
ISNULL	Check an attribute value is NULL
IN, NOT IN	Check an attribute value matches any value with in value list
DISTINCT	Gives unique values

SQL Operators - Aggregations:

Ex:- • Select avg(salary) from employeen;

• select count(*) from employeen;

• select min(salary) from employees;

• select max(salary) from employeen;

• Select sum(salary) from employees;

Aggregation function

Avg()	Returns the average value from specified columns.
Count()	Returns a no of tables rows.
Max()	Return largest value among the records.
Min()	Return smallest value among the records.
Sum()	Return the sum specified column values.

SQL Group By Clause:- • Arrange idential data into groups.

Ex:- • Select max(salary), ~~dept~~ emp_id from employees Group By dep_id;

emp_id	f_name	l_name	salary	dep_id
103	Harry	Potter	20k	12
102	Edwin	Thomas	30k	11
101	Steven	Cohen	10k	10
100	Erik	John	10k	12

Output →

[illegible]	max(salary)
103	20k
102	30k
101	10k

SQL Having Clause :-

- Used with aggregate functions due to its non-performence in the WHERE clause.
- Must follow the GROUP BY Clause in a query & must also precede the ORDER BY clause if used.

Name	Age	Salary
Jhon	24	25K
Nick	22	22K
Amara	25	15K
Nick	22	22K
Jhon	24	25K

→ Ex- Select Name, Sum(salary) from Employee Group by Name Having Sum(salary) > 45k;

Name	SUM(salary)
Jhon	50k

SQL Order By Clause :-

- Used to sort output of select statement.
- Default is to sort in ASC (Ascending)
- Can sort in Reverse (Descending) Order with "DESC" after the column name.

emp_id	name	salary
101	Steven	10k
102	Edwin	20k
103	Harry	30k

→

Ex- Select * from employee order By salary DESC;

emp_id	name	salary
103	Harry	30k
102	Edwin	20k
101	Steven	10k

SQL Union :-

- Used to combine set of two or more select statements removing duplication.
- Must have the same no. of columns.
- Must be similar with datatypes & must be in the same order in each select statement.
- More than two queries can be clubbed using more than one Union statements.

Product1

Cat_id	P_name
1	Nokia
2	Samsung
3	HP
6	Nikon

Product2

Cat_id	P_name
1	Samsung
2	LG
3	HP
5	Dell
6	Apple
10	playstation

e.g → Select product_name
from product1
UNION
Select p_name
from product2;

Output →

P_name
Nokia
Samsung
HP
Nikon
LG
Dell
Apple

SQL Union All:- Used to combine the result of two SELECT statements including duplicate rows.

- The same rules that apply to the UNION clause will apply to UNION ALL operator.

Syntax- Select col1, col2 ... FROM table1
UNION ALL
Select col1, col2 ... from table2;

product1

Catid	P_name
1	Nokia
2	Samsung
3	HP
6	Nikon

product2

Catid	P_name
1	Samsung
2	LG
3	HP
5	Dell
6	Apple
10	playstation

Output →

P_name
Nokia
Samsung
HP
Nikon
Samsung
LG
HP
Dell
Apple
playstation

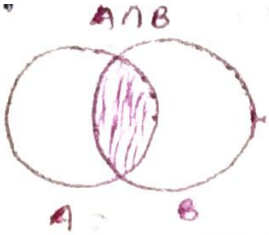
e.g → Select pname from product1 UNION ALL
Select pname from product2;

SQL Joins:- Combines rows/columns from two or more tables, based on a related column b/w them in a database.

- **INNER JOINS**:- Return rows when there is a match in both tables.

It creates a new result table by combining values of two tables (table1 & table2) based upon the join predicate.

Syntax- SELECT table1.col1, table2.col2 ..., table1.coln
FROM table1
INNER JOIN table2
ON table1.commonfield = table2.commonfield



Employeen

emp_id	ename	salary	dep_id
103	Harry	Potter	12
102	Edwin	Thomas	11
101	Steven	Cohen	10
100	Erik	John	12

Departments

dep_id	dep_name	man_id	loc_id
10	IT	200	1700
11	Marketing	201	1800
13	Renovation	203	2400
14	Shipping	121	1500

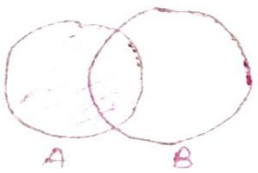
e.g → Select e.emp_id, e.ename, d.dep_id, d.dep_name
FROM employee e
INNER JOIN departments d
ON e.dep_id = d_dep_id;

↓ Output

emp_id	ename	dep_id	dep_name
101	Steven	10	IT
102	Edwin	11	Marketing

- **SQL LEFT Joins**:- The left JOIN returns all the values from the left table, plus matched values from the right table in NULL in case of no matching join predicate.

Syntax- SELECT table1.col1, table2.col2 ... table1.coln
from table1
LEFT JOIN table2
ON table1.common field = table2.commonfield



Only A + A∩B

Employeen

emp_id	ename	salary	dep_id
103	Harry	Potter	12
102	Edwin	Thomas	11
101	Steven	Cohen	10
100	Erik	John	12

Departments

dep_id	d_name	man_id	loc_id
10	IT	200	1700
11	Marketing	201	1800
13	Renovation	203	2400
14	Shipping	121	1500

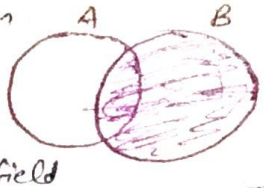
e.g → Select e.employeen e emp_id, e.ename, e.lname, d.dep_id, d.dep_name.
FROM employeen e
LEFT OUTER JOIN departments d
ON e.dep_id = d.dep_id;

↓ output

emp_id	e_name	e_lname	dep_id	dep_name
101	Steven	Cohen	10	IT
102	Edwin	Thomas	11	Marketing
103	Harry	Potter	NULL	NULL
100	Erik	John	NULL	NULL

- **SQL Right Joins**:- Returns all the values from the right table, plus matched values from the left table or NULL in case of no matching join predicate.

Syntax:- Select table1.col1, table2.col2, ... table1.coln
FROM table1
RIGHT JOIN table2
ON table1.common field = table2.common field



Employeeen

emp_id	e_name	salary	dep_id
103	Harry	20,000	12
102	Edwin	15,000	11
101	Steven	10,000	10
100	Erik	10,000	12

Department only B + A ∩ B

dep_id	dep_name	man_id	loc_id
10	IT	200	1700
11	Marketing	201	1800
13	Resource	203	2400
14	Shipping	121	1500

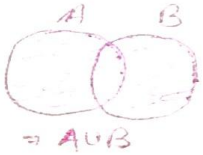
e.g→ Select e.emp_id, e.ename,
d.dep_id, d.dep_name
FROM employees e
RIGHT JOIN department d
ON e.dep_id = d.dep_id;

↓ output

emp_id	e_name	dep_id	d_name
101	Steven	10	IT
102	Edwin	11	marketing
NULL	NULL	13	Resources
NULL	NULL	14	Shipping

- SQL FULL Outer Joins:- It combines the result of both left & right joins. The joined table will contain all records from both the tables & fill in NULLs for missing matches on either side.

Syntax:-
Select table1.col1, table2.col2...
FROM table1
LEFT JOIN table2
ON table1.commonfield = table2.commonfield;
UNION
Select table1.col1, table2.col2 ... table1.coln
From table1
Right JOIN table2
ON table1.common field = table2.common field;



⇒ A∪B

Employees

emp_id	e_name	salary	dep_id
103	Harry	20,000	12
102	Edwin	15,000	11
101	Steven	10,000	10
100	Erik	10,000	12

Departments

dep_id	dep_name	man_id	loc_id
10	IT	200	1700
11	marketing	201	1800
13	Resources	203	2400
14	Shipping	121	1500

e.g→ Select e.emp_id, e.ename,
d.dep_id, d.dep_name
From employee e
LEFT JOIN departments d
ON e.dep_id = d.dep_id
UNION
Select e.emp_id, e.name, d.dep_id, d.dep_name
From employee e
RIGHT JOIN department d
ON d.dep_id = e.emp_id;

emp_id	e_name	dep_id	dep_name
101	Steven	10	IT
102	Edwin	11	Marketing
103	Harry	NULL	NULL
100	Erik	NULL	NULL
NULL	NULL	13	Resource

→ Output

- **SQL Cross Join!** Assumes three 4 records in table1 & 3 records in table2. If produces a result set with the no of rows in 1st table multiplied by the no of rows in the second.

e.g →

```
Select * from table 2
Cross Join Table 2;
```

alpha
A
B
C
D

num
1
2
3

Syntax:

```
Select table.col1, table2.col2 ... table2.coln
FROM table1
CROSS JOIN table2;
```

Output →

A	1
A	2
A	3
B	1
B	2
B	3
C	1
C	2
C	3
D	1
D	2
D	3

SQL Views! It is an virtual table based on the result-set of an SQL statement.

A view contains rows, columns just like a real table. The fields in a view are fields from one or more real tables in the database.

You can add SQL functions & JOIN statements to a view & present data as if the data were coming from one single table.

Syntax— `Create VIEW view_name AS SELECT Column1, Column2 ... FROM table_name WHERE Condition;`

EMP TABLE

EID	ENAME	SALARY
1	Jhon	20K
2	Smith	30K
3	Shane	25K
4	Ricky	28K
5	Tom	30K

e.g → `Create View V1 As Select * FROM EMP WHERE eid < 4;`

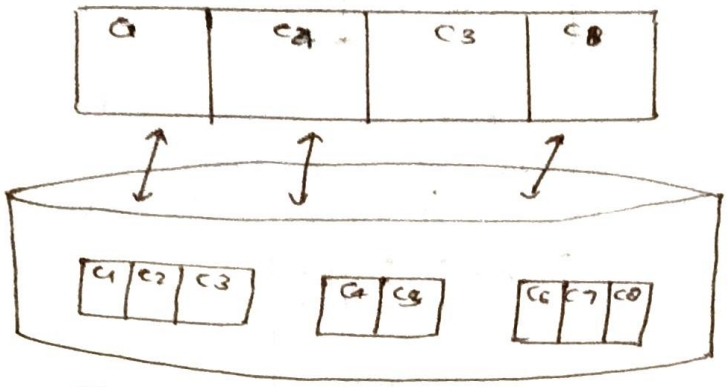
Output →

V1 (VIEW)

EID	ENAME	SALARY
1	Jhon	20K
2	Smith	30K
3	Shane	25K

A view may consist of columns from multiple tables using join or just a subset of columns of a single table. This makes views useful for abstracting or hiding complex queries.

A view that includes columns from multiple tables —



Syntax for creating views from multiple table:-

```
CREATE VIEW MarkView AS
SELECT Student Detail.NAME,
 Student Detail.ADRESS, Student Detail.Course,
  Student Marks.Marks.
  FROM StudentDetails, Student Marks
  WHERE StudentDetails.NAME = StudentMarks.NAME;
```

ADAVANTAGE:

① If creates secure environment, by restricting users to access directly to a table and allow them to access a subset of data via views.

② Relational database have many tables with complex relations (one to many etc..) It makes difficult to navigate data, however you can simplify the complex queries with join & condition using a set of views.

③ Sometimes, we need to write complex queries, to make it consistent, you can hide queries in views.

→ Using Drop command we can delete the SQL view.

```
DROP View View_name;
```

→ Using CREATE OR REPLACE VIEW statement to add or remove fields from a view -

```
CREATE OR REPLACE VIEW view_name AS SELECT
 column1, column2.. FROM table_name where condition;
```

Subquery in SQL: Innerquery or Nested query is a query within another SQL query & embedden with in the WHERE clause.

- It can be used with SELECT, INSERT, UPDATE, & DELETE with operator =, >, <=, etc OR IN, BETWEEN etc.

Rules –

- Must enclosed with paranthesis.
- ORDER BY command can't used with it, but Grop by command can be used.
- Between can't with a subquery, but loc can used with in a subquery.

Syntax:- Select clumn name [, column name] from table1 [, table2] WHERE Column_name OPERATOR [SELECT column_name [, column name] from table [, table2] [WHERE]]

Ex- Select emp_name from Employee where salary = (Select max(salary) from Employee);

Practice Query

Datasets Used: Cricket_1, cricket2

cricket2 → 2 table for cricket test match2

Cricket_1 → 1 table for cricket test match 1

① Find all the players who were present in test match 1 or test match 2;

Match 1

player_id	player_name	runs	popularity
P1	Virat	50	10
P2	Rohit	41	7
P3	Dhoni	33	6
P4	Jadeja	35	15
P5	Dhawan	65	12

Match 2

player_id	player_name	runs	popularity	charisma
P1	Virat	50	10	2
P2	Rohit	41	7	3
P3	Dhoni	33	8	1
P4	Jadeja	35	15	2
P15	Dravid	90	13	

Sol^n:-

```
Select * from cricket1
UNION
Select * from cricket2;
```

② Find player_name from cricket1 where average popularity greater than its popularity.

Sol^n:-

```
Select p_name from cricket1 where popularity >
( select avg (popularity) From cricket1);
```

③ Find player_id & player_name that are common in test match1 & test match2.

Sol^n

```
Select player_id, player_name from cricket2 where player_id
IN ( select player_id from cricket2);
```

④ Write a query the player_id, runs, player_name from table "cricket 1" where player_name starts with "y" & end with "v":

Ans: select player_id, runs, player_name from cricket1 where player_name LIKE "y%v";

⑤ Write a query to extract all data from cricket1 where player_name does not end with 't'.

Ans: select * from cricket1 where player_name NOT LIKE '%t';

⑥ Extract the player_id and player_name of the players Cricket2 where Charisma is NULL.

Ans: select player_id, player_name from cricket2 where Charisma IS NULL;

⑦ Seprate all player_id into single numeric ids (ex- P1 = 1)

Ans: select player_id, SUBSTR(player_id, 3) from cricket2;

⑧ Display all columns & their datatypes size in Cricket2.

Ans: ~~Selec~~ Describe Cricket2;

⑨ ~~Change~~ Add the attribute with id data type size, and adds the ~~dat~~ attribute as a primary key:

Ans: ~~Select~~ Alter table cricket2 ~~alter~~ add column ~~id~~ id INT(2);
OR
Alter table cricket ~~alter~~ add ~~columns~~ primary key (Id);

⑩ Alter the ~~data~~ id from INT TO INTEGER in Cricket2.

Ans: Alter table cricket2 modify player_id INTEGER(10);