422CIT03 Database Management System

Unit - I

Relational Database

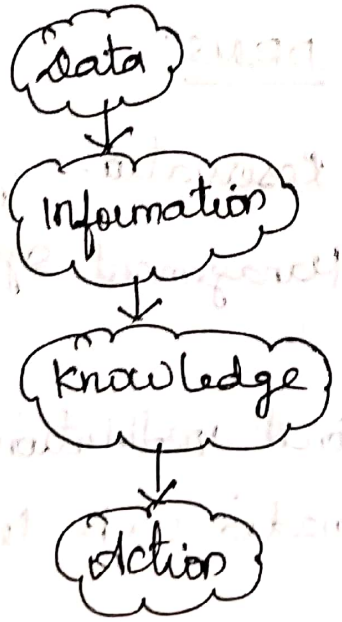
Purpose of database system - Views of data - data models.

The DBMS is defined as a S/w system that allows the user to define, Create and maintain the db and provide Control access to the data.

It is a Collection of programs used for managing data and simultaneously it supports different types of users to Create, manage, retrieve, update and store information.

Purpose of DBMS:

* Data into information
* information into knowledge
* knowledge to the action.



Drawbacks of File s/m:

* Data redundancy and inConsistancy
* difficulty in accessing data.
* Data isolation.
* Integrity problem
* Atomicity of updates
* Concurrent acces by multiple users
* Security problem.

Uses of DBMS:

⇒ Data independence and efficient acces of data.

⇒ Application development time reduces.

⇒ Security and data integrity.

⇒ Uniform data administration

⇒ Concurrent acces and recovery from Crashes

Application of DBMS:

⇒ Railway Reservation system.

⇒ Library Management system

⇒ Banking

⇒ Educational institutions

⇒ Social media websites

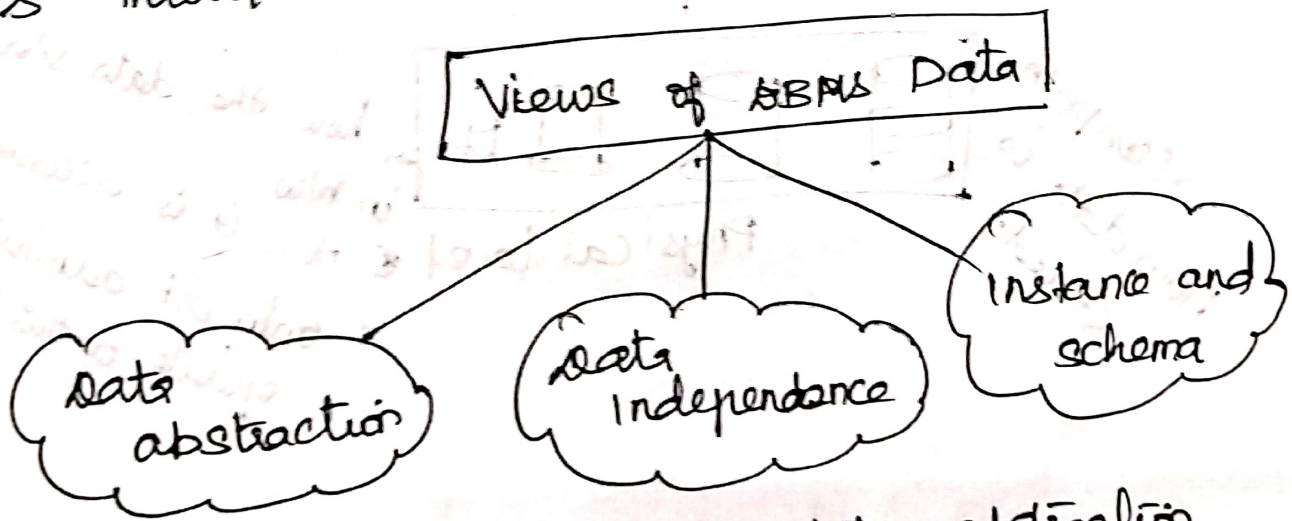
Types of DBMS:

* MySQL
* Oracle
* MySQL Server
* Microsoft Access
* SQL
* DB2.

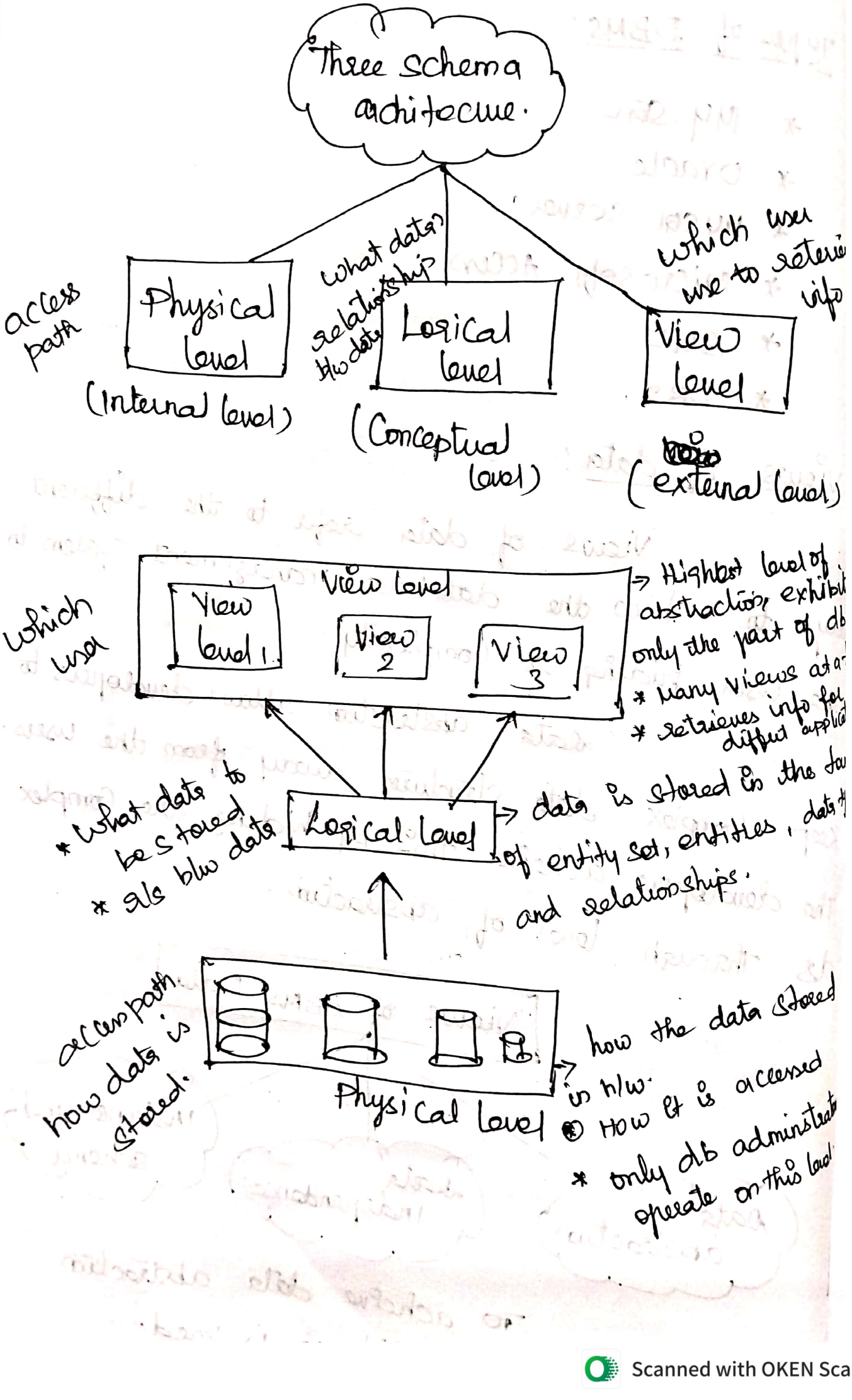
Views of data:

Views of data refer to the different ways to show the database management system to the user hiding its complexity.

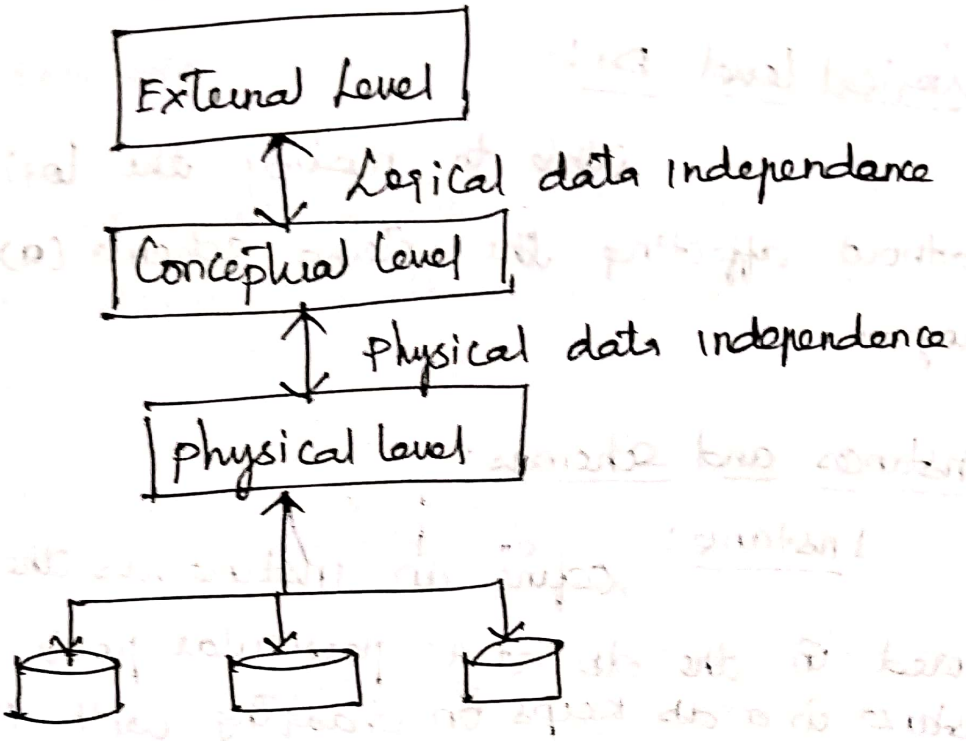
Data abstraction allows developers to keep complex data structures away from the users. The developers achieve this by hiding the complex ds through level of abstraction.



To achieve data abstraction the three-schema architecture is used.

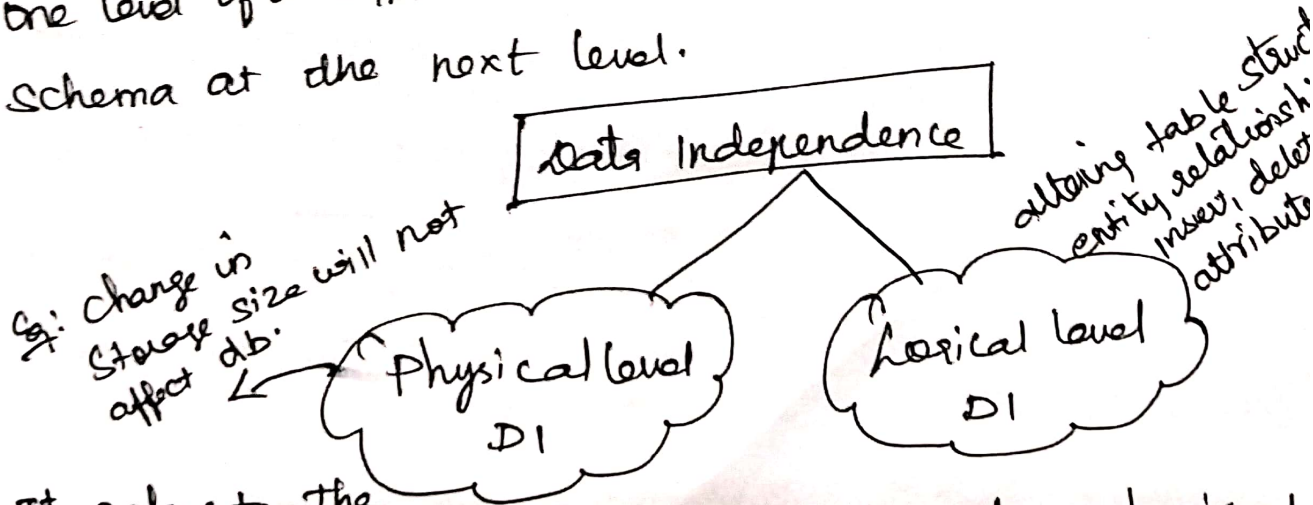

Three schema architecture.
access path
Physical level
(Internal level)
what data? relationship? blw data
Logical level
(Conceptual level)
which user use to retrieve info
View level
(external level)
which user
View level
View level 1
View 2
View 3
→ Highest level of abstraction, exhibit only the part of db
* many views at a
* retrieves info for different applica
* what data to be stored
* rls blw data
Logical level
→ data is stored in the for of entity set, entities, data and relationships.
access path
how data is stored.
Physical level
→ how the data stored in h/w.
* How it is accessed
* only db administrato operate on this leve

Data Independance:



The data schema can be changed at one level without modifying the data schema at the next level.

DI is mainly defined as a property of DBMS that helps to change the db schema at one level of a s/m without requiring to change the schema at the next level.



It refers to the characteristics of being able to modify the physical schema without any alterations to the Conceptual (or) logical schema for the optimization purpose.

4. Change in storage size

Logical level DI:

idbb to modify the logical schema without affecting the extend schema (a) applicatio program.

Instances and schemas:

Instance:

Define an instance as the informatio Stored in the db at a particular point of time. A instance in a db keeps on changing with time.

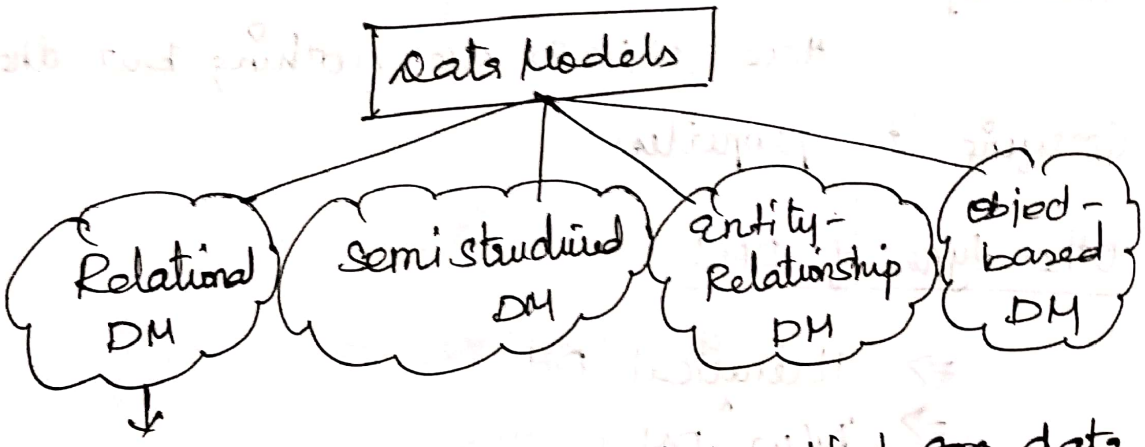
Schemas:

Schema is a overall design of the entir db. Schema of the db is not changed frequently

Data Models:

Data model is the modeling of the data description, data semantics and Consistency Constraints of the data.

It provides the Conceptual tools for describing the design of a db at each level of data abstraction.



Relational DM:

This type of model designs that ~~data~~ data is in the form of rows and Columns within a table.

Relational model uses tables for representing data and in-between r/s. Tables are also called relations.

Entity-Relationship DM:

An E-R model is the logical representation of data as objects and r/s among them. These objects are known as entities and r/s is an association among these entities.

- ⇒ A set of attributes are entities.
- ⇒ A set of same-type of entities is known as Entity Set.

⇒ A set of same type of r/s is known as entity relationship set.

Object based DM:

An extension of E-R model with notation of functions, encapsulation and object identity.

Here objects are nothing but the data carrying its properties.

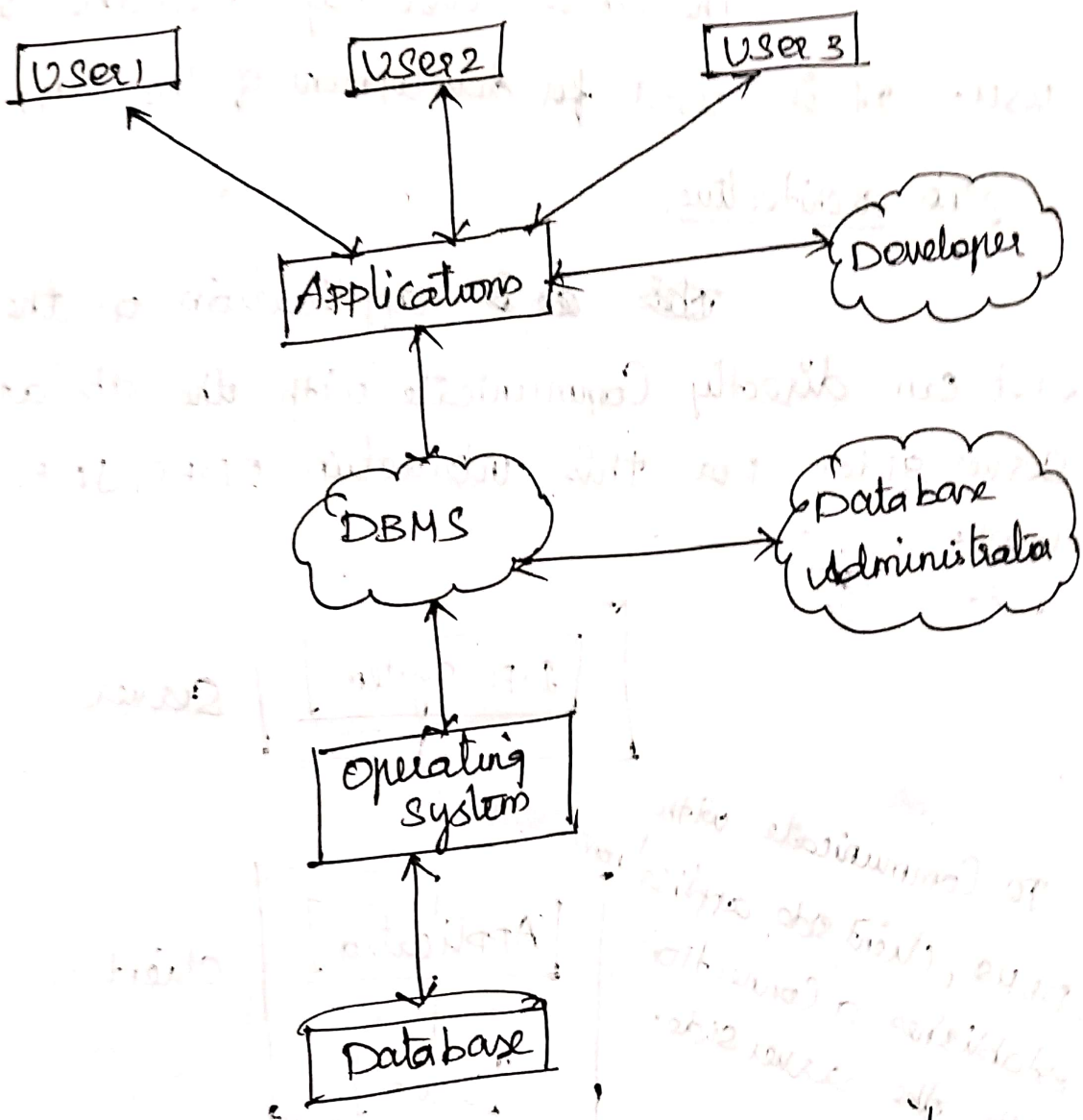
Other types of DM:

⇒ Hierarchical DM
⇒ N/w DM
⇒ Object oriented DM
⇒ Float DM
⇒ Context DM
⇒ So

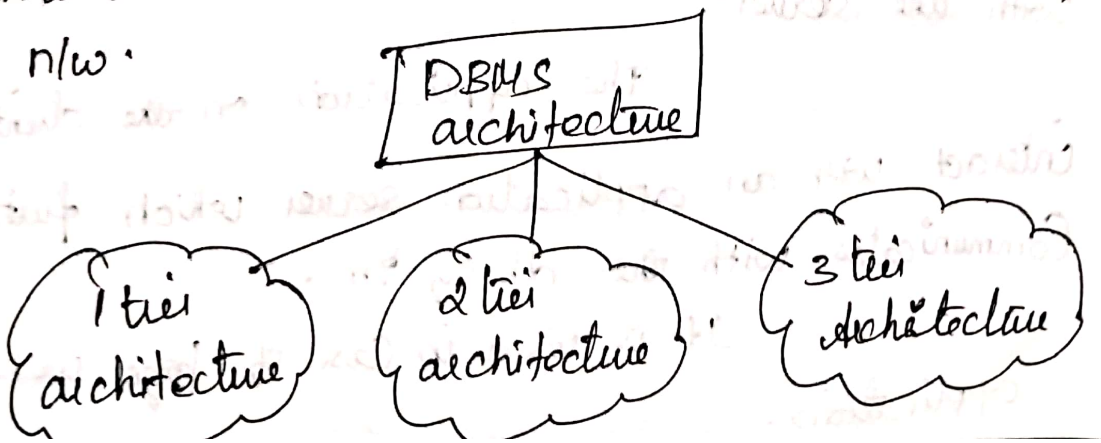
Advantages:

* Helps us to present data accurately
* helps in finding missing data and minimizing data redundency
* security
*

DBMS Architecture:



The DBMS design depends on its architecture. The basic client/server architecture is used to deal with large number of pc's, web servers, db servers and other components that are connected with n/w.

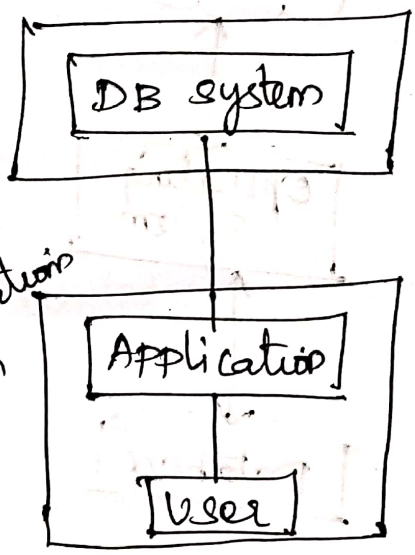


1 tier architecture:

The db is directly available to the user. It is used for development of local applications.

2 tier architecture:

~~db~~ application on the client end can directly communicate with the db at the server side. For this interaction ODBC, JDBC are used.



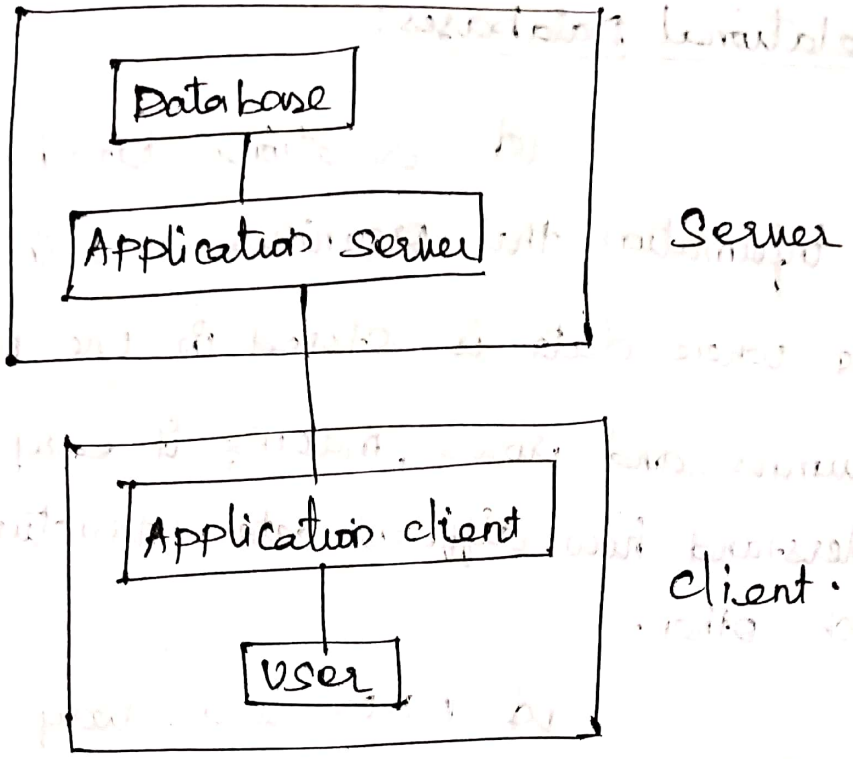
To communicate with DBMS, client side application establishes a connection with the server side.

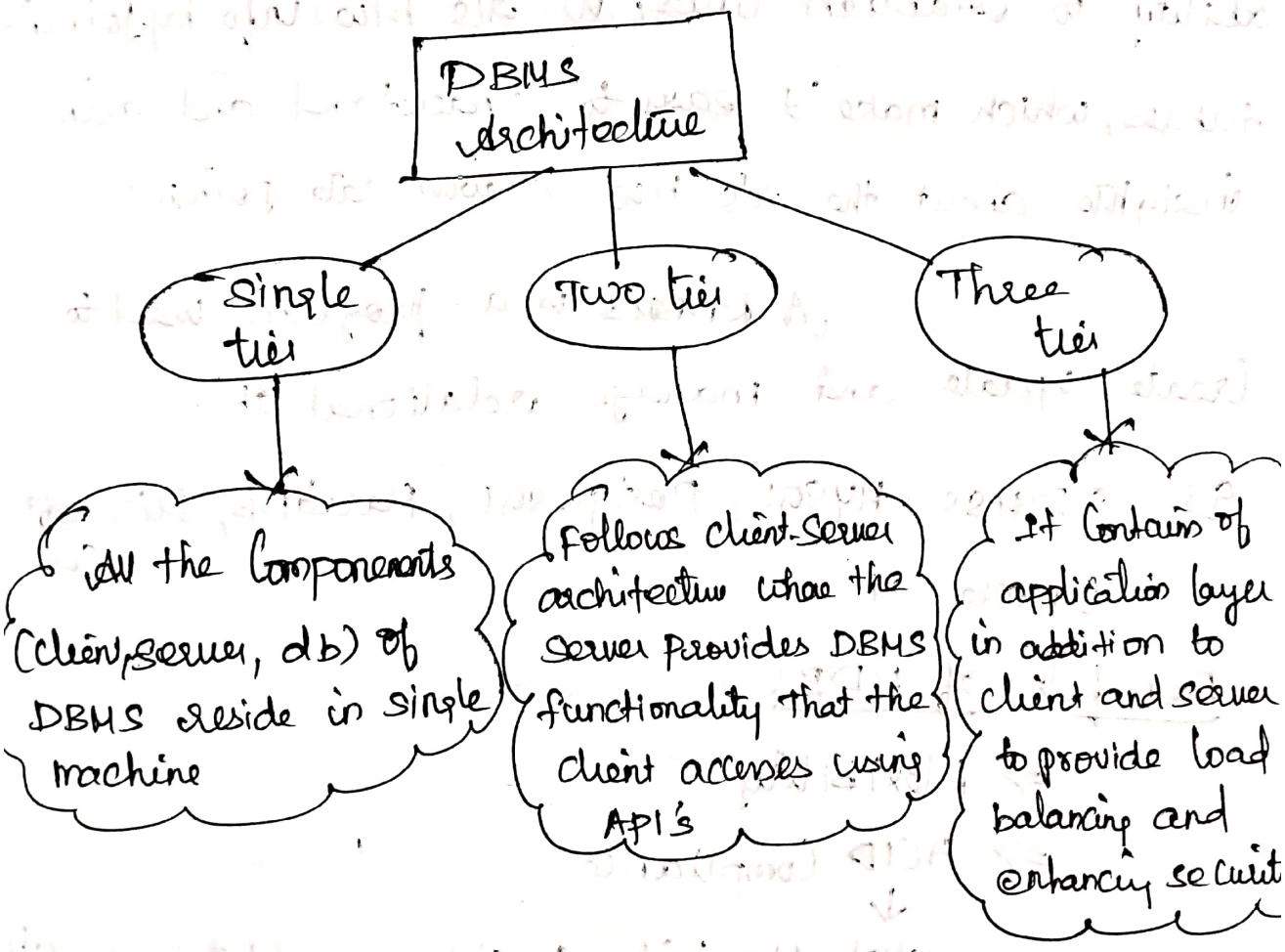
3 tier architecture:

This contains another layer b/w the client and server. Here client can't directly communicate with the server.

The application on the client side interact with an application server which further communicates with the db system.

It is used in case of large web applications.





Relational Databases:

A relational data base is a Collection of information that organizes data in predefined r/s where data is stored in one or more tables Columns and rows, making it easy to see and understand how different data structures relate to each other.

- A RDB is a way of structuring info in tables, rows and Columns. An RDB has the ability to establish links, r/s b/w info by joining tables, which makes it easy to understand and gain insights about the r/s b/w various data points.

A RDBMS is a program used to create, update and manage relational db.

eg: RDBMSs, MySQL, PostgreSQL, MariaDB, Microsoft SQL Oracle db.

Benefits of RDB:

=> Flexibility
=> ACID Compliance
↓
Atomicity, Consistency, Isolation, durability
=> Ease of use
=> Collaboration
=> Built-in Secu.t..

Relational Model:

A relational model represents how data is stored in Relational database. A relational db consists of a collection of tables each of which is assigned a unique name, Consider a relation Student with attributes ROLL-NO, NAME, ADDRESS, PH and AGE.

eg: Student table.

ROLLNO	NAME	ADDRESS	AGE	PH. NO
1	AAA	Delhi	18	111 111
2	BBB	Tamilnadu	18	222 222
3	CCC	Karnataka	20	333 333
4	DDD	Kerala	18	444 444

Attribute: Properties that define an entity.

eg: ROLL-NO, NAME, ADDRESS

Relation schema: A relation schema defines the structure of the relation and represents the name of the r/s with its attributes.

eg: Student (ROLL-NO, NAME, ADDRESS) is the relation schema for student.

Tuple: Each row in the relation is known as a tuple.

eg:

1	AAA	Delhi	18	111 111

Relational Instance: The set of tuples of a relation at a particular instance of time is called a relation Instance.

Degree: The no. of attributes in the relation is known as the degree of relation. The Student relation have 5 degree.

Cardinality: The no. of tuples in a relation is known as Cardinality. Eg: Student has 4 Cardinality

Column: The Column represents a set of values for a particular attribute:

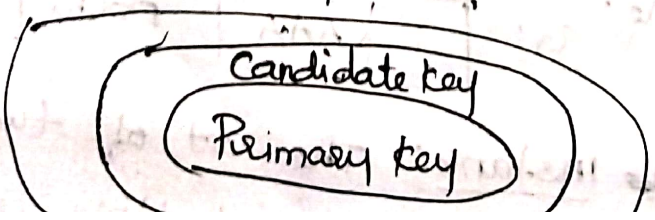
Eg:

Roll No
1
2
3
4

Relation keys:

Keys are used to identify the rows uniquely & also help in identifying tables.

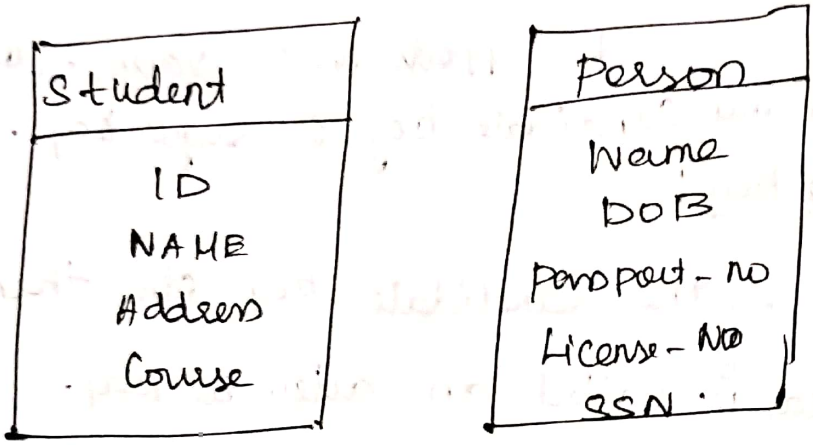
=> Primary key
=> Candidate key
=> super key
=> Foreign key
=> Alternate key
=> Composite key



⇒ Keys play an important role in the relational db.

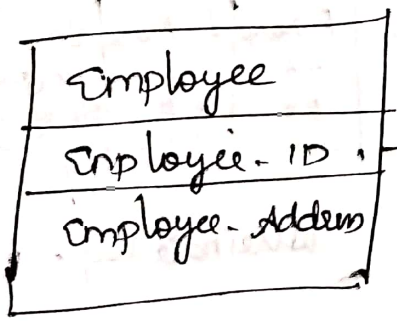
⇒ It is used to uniquely identify any record (or) row of data from the table.

Eg:



⇒ Primary key:

⇒ It is the first key used to identify one and only one instance of an entity uniquely.



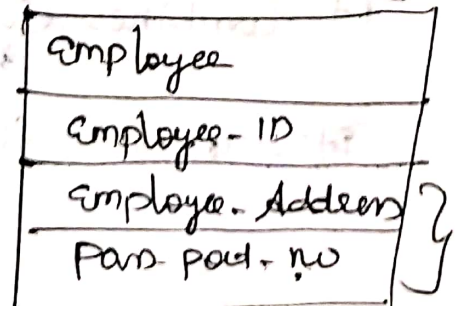
→ Primary key.

It cannot be a null value.

⇒ Candidate key:

A Candidate Key is an attribute (or) set of attributes that can uniquely identify a tuple.

Except for the primary key, the remaining attributes are associated (or) considered as Candidate key.



It can have multiple Candidate keys.

} Candidate Key

⇒ Super Key:

Super key is an attribute set that uniquely identify a tuple. A super key is a super of a Candidate key.

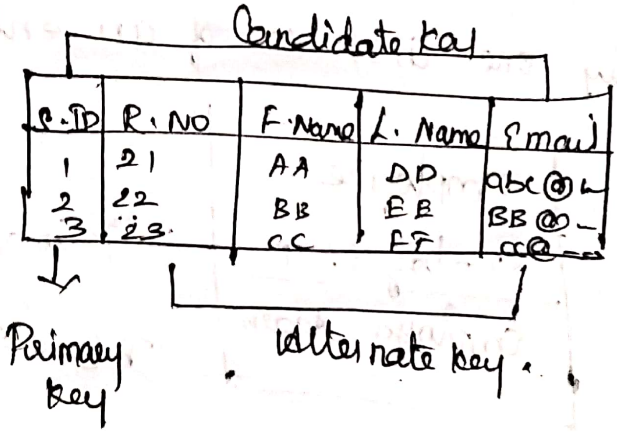
It supports NULL values. Any key added to my Candidate key is super key..

Alternate Key:

⇒ The Candidate key other than the primary key is called an alternate key.

⇒ It is a secondary key.

⇒ These values are repeated.



Foreign Key:

It is a key it acts as a primary key one table and it act as secondary key in another table.

It Combines two (or more relation a time.

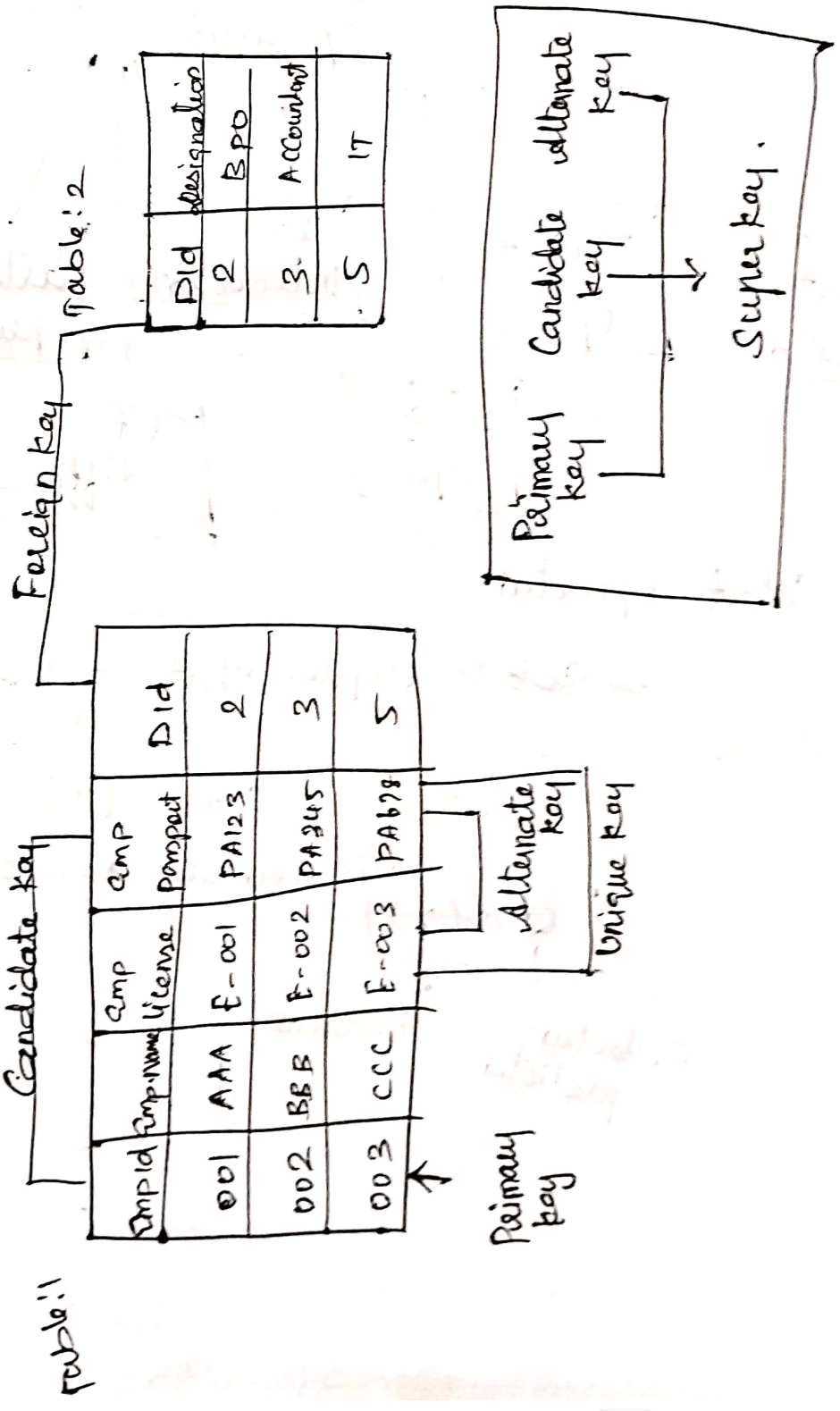
It acts as a Cross-reference b/w tab

Composite Key:

⇒ It act as a primary key if there is no primary key in a table.

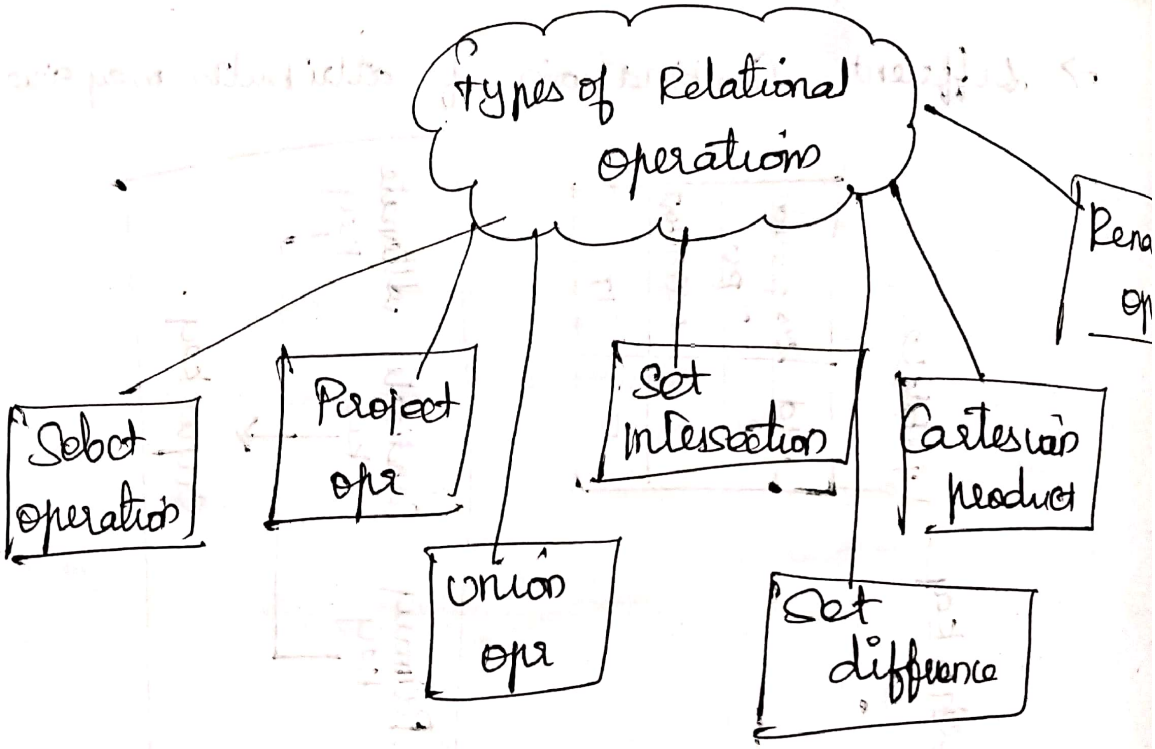
⇒ Two (or) more attributes are used together to make a Composite key.

⇒ Different Combinations of attributes may give



Relational Algebra:

RA is a procedural query lang
It gives a step by step proces to obtain the
result of the query.



=> Select operation :

=> Selects tuples that satisfy a giv
predicate.

=> denoted by sigma (σ)

$\sigma_{p(r)}$

=> logic formula AND, OR, NOT.

Selection predicate

relation

Loan table

Branch Name	Loan No	Loan Amount
AAA	L-17	1000
BBB	L-18	2000
CCC	L-19	3000
DDD	L-20	4000
AAA	L-21	5000

σ Branch_Name = "AAA" (LOAN)

AAA	L-17	1000
AAA	L-21	5000

⇒ Project operation :

Shows the list of those attributes that we wish to appear in the result.

Rest of the attributes are eliminate from the table.

$$\Pi\ A1, A2, An(r)$$

Π Branch_name , Loan_no (LOAN)

AAA	L-17
BBB	L-18
CCC	L-19
DDD	L-20
AAA	L-21

→ Union operation:

There are two tuples R & S. The union operation contains all the tuples that are either in R or S or both R & S.

It eliminates the duplicate tuples. It is denoted by ∪.

$$R \cup S$$

Customer Table

Branch-name	ACC-NO
XXX	L-22
YYY	L-23
ZZZ	L-24
AAA	L-17
BBB	L-18
CCC	L-19

$\Pi_{branch\text{-}name}$(Loan-Table) ∪ $\Pi_{Branch\text{-}name}$(Cust

-ta

Branch_Name
AAA
BBB
CCC
DDD
XXX
YYY
ZZZ

⇒ Set Intersection:

⇒ Suppose there are two tuples R & S.

The set intersection operation contains all tuples that are in both R & S.

It is denoted by intersection ∩.

$\Pi_{\text{branch-name}}$ (loan-table) ∩ Π branch-name (Cust-table

AAA
BBB
CCC

⇒ Cartesian Product:

The Cartesian product is used to Combine each row in one table with each row in the other table. It is also known as a Cross product.

It is denoted by X.

E X D

Eg:

Employee

Emp-ID	Emp-name	Emp-dept
1	AAA	A
2	BBB	B
3	CCC	C

Department

Dept-no	Dept-name
A	Marketing
B	Sales
C	Legal

Employee × department

Emp-ID	Emp-name	Emp-dept	dept-No	dept-name
1	AAA	A	A	Marketing
2	BBB	B	B	Sales
3	CCC	C	C	Legal.

⇒ Rename operation :

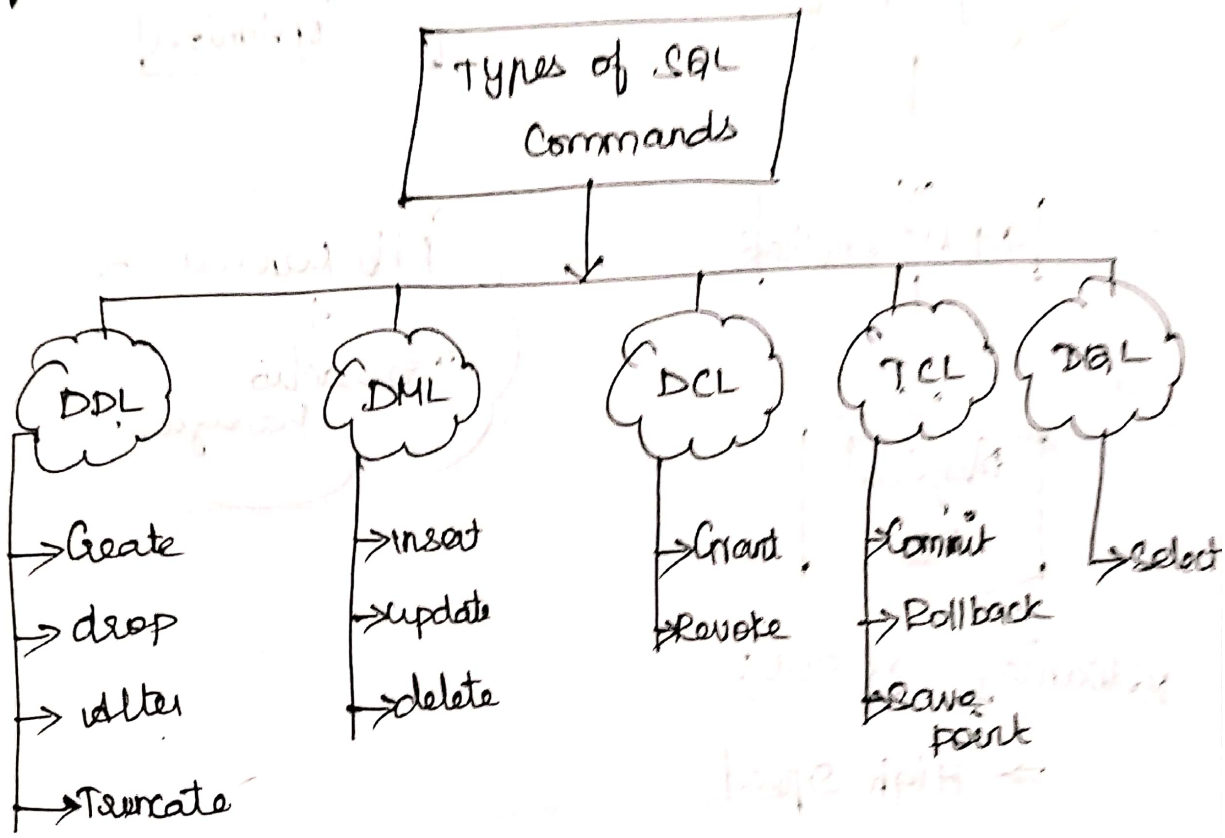
The rename operation is used to rename the output relation. It is denoted by.

rho(P)

Eg: P(student1, student)

SQL Fundamentals:

SQL Commands are instructions. It is used to communicate with the db. It is also used to perform specific tasks, functions and queries of data.



Sql stands for Structured Query language. It is used for storing and managing data in RDBMS.

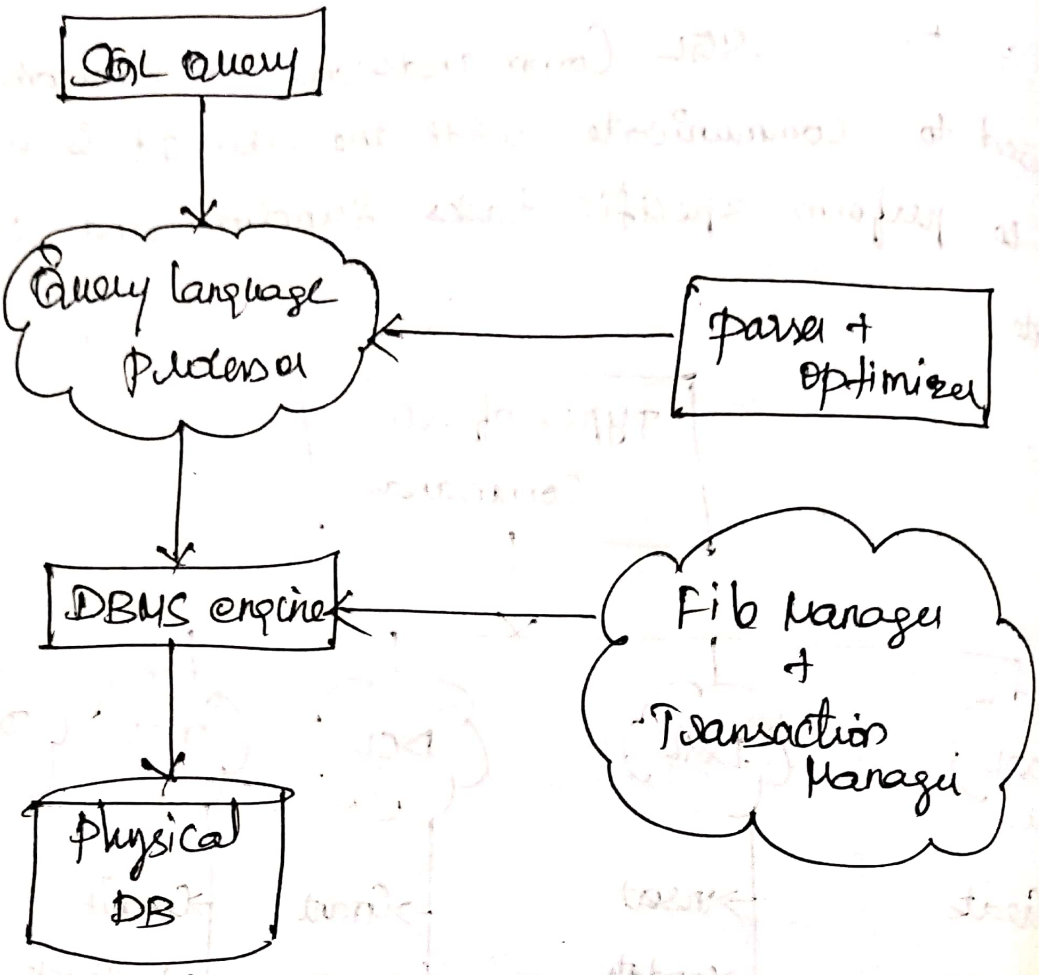
It enables a user to create, read, update and delete relational database and tables.

SQL rules:

=> SQL is not case sensitive.

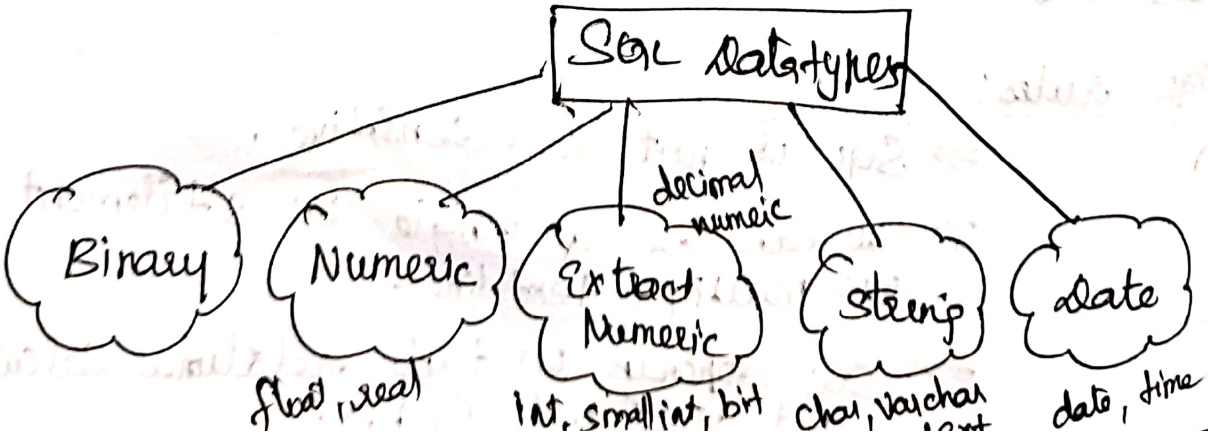
=> We can use a single SQL statement (or) multiple text line.

=> SQL depends on tuple relational calculus and relational algebra.



Advantages of SQL:

- ⇒ High speed
- ⇒ No Coding needed.
- ⇒ Well defined standards
- ⇒ Portability
- ⇒ Interactive language.
- ⇒ Multiple data view.



DDL Commands: [Create, Alter, Drop, Truncate]

=> Create: It is used to Create a new table in db.

Syntax:

```
CREATE TABLE TABLE_NAME (Column Name
                         Datatype [ ]);
```

eg:

```
Create Table Employee (Name Varchar2 (20),
                       email Varchar2 (100),
                       DOB Date);
```

=> Drop: It is used to delete both the structure and record stored in the table.

Syntax:

```
Drop Table table_name;
```

eg:

```
Drop Table Employee;
```

=> Alter: It is used to alter the structure of the db.

Syntax:

① ALTER TABLE table_name ADD Column_name COLUMN definition;

② ALTER TABLE table name Modify (Column_definition);

eg:

```
ALTER TABLE STUDENT ADD (Address Varchar2 (20));
ALTER TABLE STUDENT MODIFY (Name Varchar2 (20));
```

⇒ TRUNCATE: It is used to delete all the from the table and free the space Containing table.

Syntax:

```
TRUNCATE TABLE table_name;
```

Eg:

```
TRUNCATE TABLE EMPLOYEE;
```

DML: [insert, update, delete]

DML Commands are used to modify database. It is responsible for all the form of changes in db.

⇒ Insert: It is used to insert data into th row of a table.

Syntax:

```
INSERT INTO TABLE_NAME (col1, col2... coln)
                        VALUES (value1, v
                        value 3... value N);
```

Eg:

```
INSERT INTO STUDENT (Std-ID, Name) values ('1
```

⇒ UPDATE: This Command is used to upda modify the value of a Column in the table.

Syntax:

```
UPDATE table_name SET (Col_name1 = value1,
```

eg: UPDATE Student SET Name = 'BBB' WHERE Std-ID = '1';

=> **DELETE**: It is used to remove one (or) more row from a table.

syntax:

```
DELETE FROM table-name (WHERE Condition);
```

eg: DELETE FROM Student WHERE Name = "AAA";

DCL: [Grant, Revoke]

DCL Commands are used to grant and take back authority from any db user.

Grant: It is used to give user access privilages to a db.

syntax:

```
GRANT SELECT, UPDATE ON MY_TABLE TO
SOME_USER, ANOTHER_USER;
```

=> **REVOKE**: It is used to take back permissions from the user.

syntax:

```
REVOKE SELECT, UPDATE ON MY_TABLE FROM
USER1, USER2;
```

TCL: [Commit, Rollback, Save point]

=> Commit: It is used to save all the transa to the db.

Syntax:

```
Commit ;
```

Eg:

```
DELETE FROM CUSTOMERS WHERE AGE =2
COMMIT ;
```

=> ROLLBACK: It is used to undo transac that have not already been saved to the db.

Syntax:

```
ROLLBACK;
```

Eg:

```
DELETE FROM CUSTOMERS WHERE AGE = 25
ROLLBACK;
```

=> SAVEPOINT: It is used to roll the transa back to a certain point without rolling back t entire transaction.

Syntax:

```
SAVEPOINT savepoint_name;
```

=> DQL: [SELECT]

DQL is used to fetch the data fro the db.

⇒ SELECT: It is used to select the attributes based on the condition described by WHERE clause.

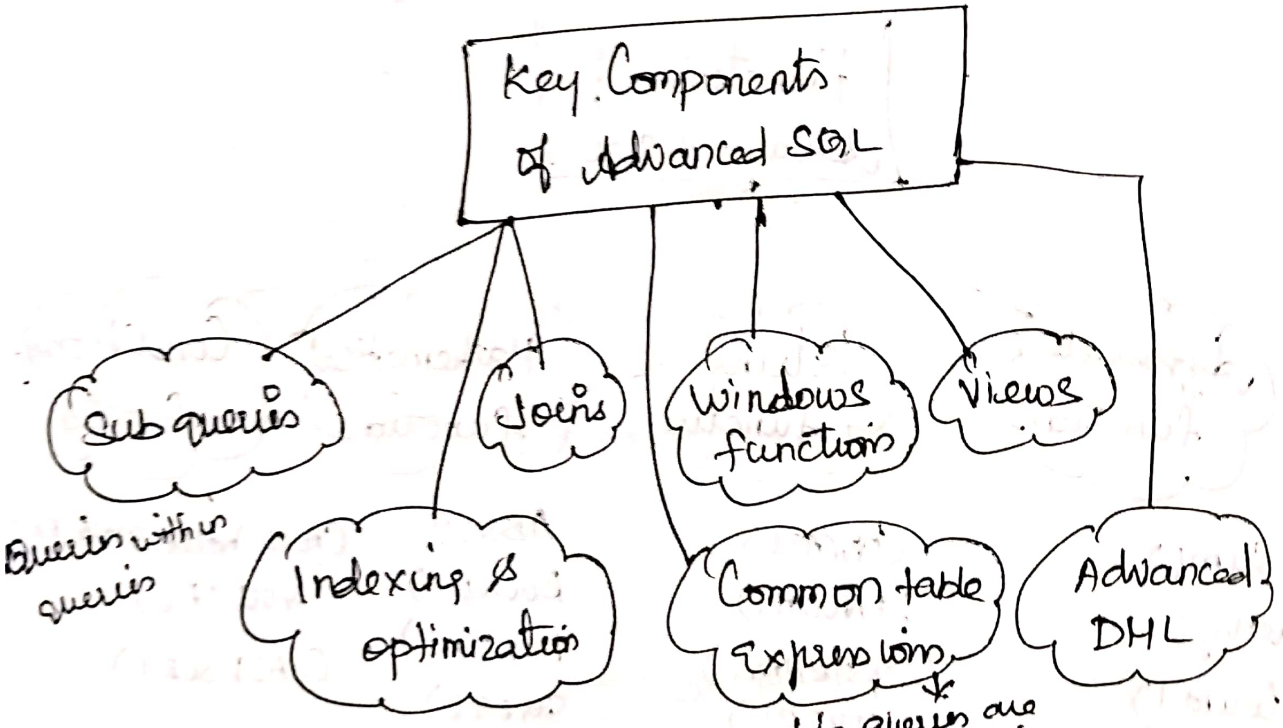
Syntax:

```
SELECT expressions FROM TABLES
  WHERE Conditions;
```

Eg: SELECT emp_name FROM employee
WHERE age > 20;

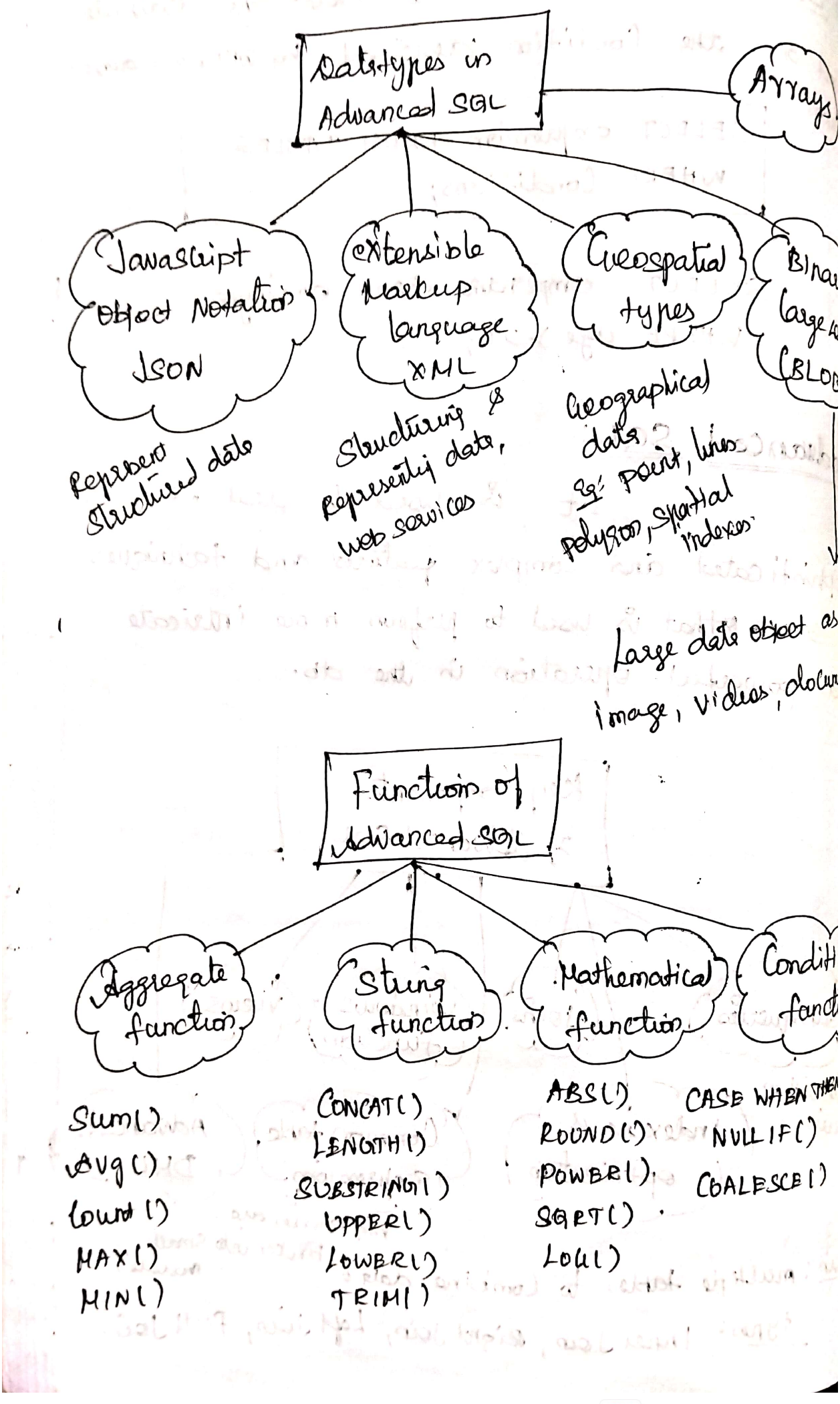
Advanced SQL:

It is used to deal with sophisticated and complex features and techniques in SQL, that is used to perform more intricate and powerful operation in the db.



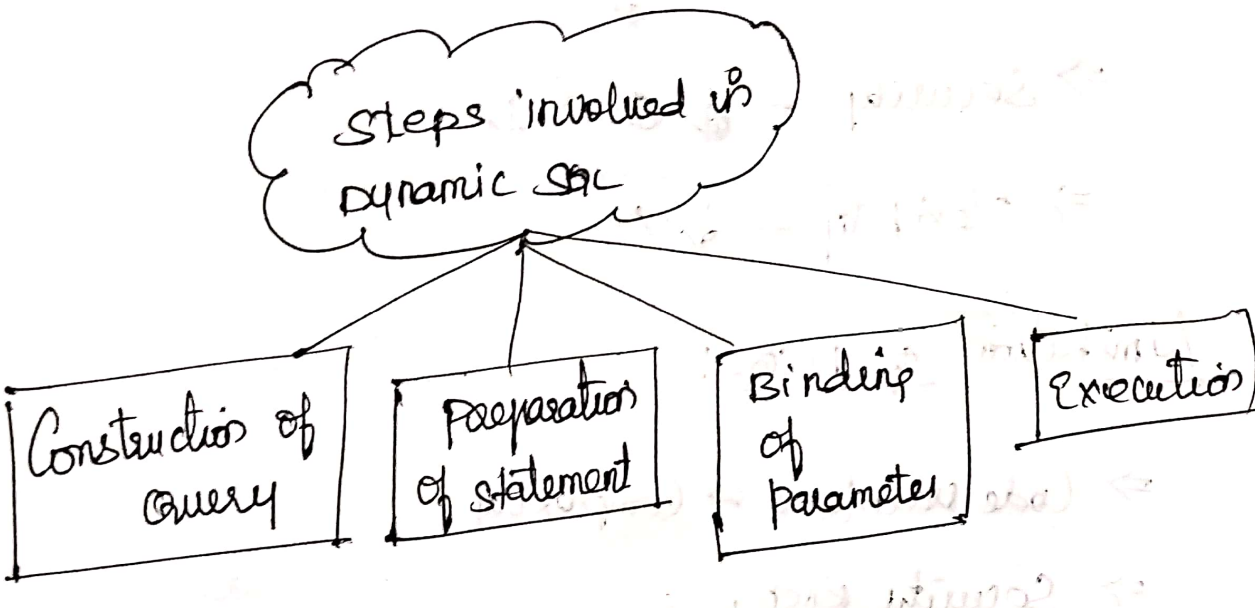
Join: multiple tables to combine data

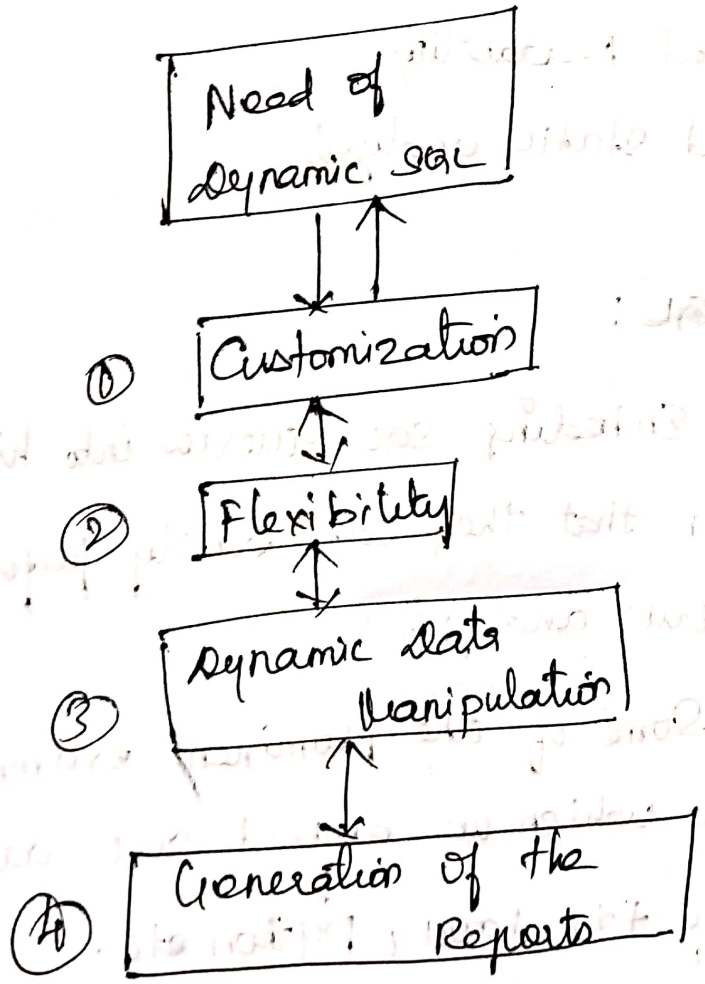
types: Inner Join, Right Join, Left Join, Full Join


Datatypes in Advanced SQL
Arrays
Javascript Object Notation JSON
Represent Structured data
eXtensible Markup language XML
Structuring & representing data, web services
Geospatial types
Geographical data eg: point, lines, polygon, spatial indexes
Large data object as image, videos, docu
Functions of Advanced SQL
Aggregate function
Sum()
AVG()
Count()
MAX()
MIN()
String function
CONCAT()
LENGTH()
SUBSTRING()
UPPER()
LOWER()
TRIM()
Mathematical function
ABS()
ROUND()
POWER()
SQRT()
LOG()
CASE WHEN THEN
NULLIF()
COALESCE()

Dynamic SQL:

DSQL is the SQL statement that is being formed dynamically during runtime at the time of execution of the SQL Statement.





Advantages over Static from Dynamic:

S D

=> Construction - ↓ ↑

=> Performance - ↓ ↑

=> Maintain - ↓ ↑

=> Security - ↓ ↑ ↓

=> Flexibity - ↓ ↑

Limitation of DSQL:

=> Code maintenice & Complexity

=> Security Risk.

=> Reduced Readability.

=> Limited static analysis

Embeded SQL:

Embeding SQL queries into high-level languages such that they can easily perform the logic part of our analysis.

Some of the prominent examples of languages with which we embed SQL are as follows: C++, Java, Python etc.

It is a technique which allows SQL Statement to be directly included with in

Programming language. DBMS provides some API calls or special syntax that allows developers to embed the SQL statements directly in their source programming code.

Advantages of embedded SQL:

* Seamless Integration [Tight integration]
* Enhancement of Performance
* Managing Transaction
* Manipulation of data and business logic
* Parameterized queries
* Code can be reusable
* Debugging and handling errors.
* Data Consistence
* Database Security