1)

DBMS	RDBMS
Managing, Retrieving, storing from collect of logic rel info	Provides with Relational Integrity (Tables)

2) Advantages of DBMS:

* Sharing Data * Reduced Redundancy

* Data Independence

3) Types:

DDL : Definition

DML : Manipulation

DCL : Data Ctrl

TCL : Transac Ctrl

4) Trigger -> TC.

Stored proc -> CGPA

View -> mult tab
temp oper

4) Levels of Abstraction:

* Physical : HOW data stored

* Logical : Rel b/w data

* View : Part of entire DB

5) Concurrency ctrl:

Process of controlling simultaneous operations in a DataBase

6) Keys in DB:

* Primary key : Key uniquely identifies very Tuple

* Unique Key : similar to 1° key, allows null value

* Foreign Key : Take values from some other attribute (other Table)

* Composite key : combn of 2 / ↑ columns to idetify each Tuple uniquely

* Candidate key : set of attribute to uniq ident. out of all cand key -> 1 can be chosen as 1° key.

Eg:
cust Id + Pancard NO } -> cand key

4) Triggers   Stored Procedures

Trigger is stored Procedure automatically invoked when a spl event occurs in the DB

Eg: enter CGPA, GPA automatic updates.

5) Stored procedure:

Can be reused over & over again

Eg: even after giving TC cert info abt cand is imp written as stored procedure

6). JOINS:

Combine rows from 2/more Tables based on a related column b/w them

orders Table

common

order ID	cust ID	orderdate

customer's Table

custID	cust Name	Cont Name	Country

```
select orders.orderID, customers.custname,
    orders.OrderDate, ~~customers.~~
from orders    INNER JOIN
customers on    Orders.customerID = Customers.customerID
```

op:

Order ID	cust_Name	order Date

* Inner Join: Returns records having matching values in Both Tables.

* Outer (left) join: Ret records from left Table, Matched records from Right Table

* Right outer join: Return records from Right Table, Matched Records from Left Table

* Full outer join: Returns all records when there is a Match in either Left/Right Table

* self join → Regular join.

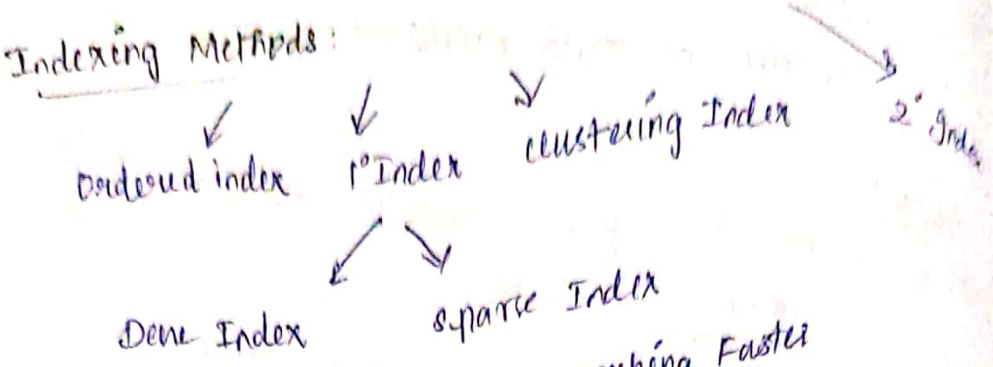
Indexing in DBMS:

Optimize the performance of DataBase by Minimizing the number of Disk access required when a query is processed.

Index structure:

Search Key	Data Reference

Indexing Methods:



Ordered Index: sorted to Make searching Faster

1° Index : created on Basis of 1° key of Table

Dense Index : contains Index Record for every search key.

Sparse Index: Index record appears only for Few Items

clustered Index: can be def as Ordered Data File.
Index may be created on non 1° key column
May not be unique for each record.

Records with similar char are grouped, index created for each Record.

2° Index → As size of Table grows, size of Mapping ↑.
So reduce size of Mapping.

Analytical functions:

computes values over group of Rows and returns single result for Each Row.

Aggregate: → single result for group of Rows

Includes over clause, which defines windows of Rows around the row being evaluated.

1NF:

- * single (Atomic) valued attributes / columns.
- * Vals stored in col should be same domain
- * All col in Table → unique names.
- * order in which data is stored doesnt matter

2NF:

→ Be in 1NF.

→ should not have Partial dependency.

Functional dependency:

stu-id	name	reg-no	Branch	address
10	-	-	-	-

stud id is 1° Key.

with 1° key ⟨ dept / name ⟩ ... ly can be accesed.

∴ every other column that depends on it is FD.

Partial dependency:

stuname	stu-id	subj-name	subj-id
1	suppy	4	Java
2	Cat	5	c++
3	Arthi.	6	C

stud-id → 1° key for stud

subj-id → 1° key for sub

studid + subjid → Candidate key.

Partial dependency only on part of 1° key & not on whole key

To Remove Partial dependency:

Divide Table, Remove attrib that causes PD, move it to other Table

3NF :

It is in 2NF

Doesnt have transitive dependency.

3 Tables:

Student Table :

stu-id	name	reg-no	Branch	Address
1° Key				

Subject Table :

subj-id	subj-name	Teacher
1° Key		

score Table :

score-id	stud-id	subj-id	marks
1° Key			

combo 1° keys

But Marks not 1° key but depends on other Table's 1° key.

when Non-prime attributes depends on other Non-prime attributes -> transitive dependency

How to remove transitive dep:

Remove columns exam-name, Total marks from score-Table & put them in exam Table, use examid whenever required.

Adv of remov TD :

* Data Integrity achieved.
* Amt of data replication ↓.

SQL :

DB language for storing, Managing data in Relational DBMS.

DDL : Data Definition Language : → Auto-committed

→ Create : create new DB /Table.
→ Alter : for alteration
→ Truncate : delete data from Table
→ Drop : drop Table
→ Rename

DML : Data Manipulation Language : → Not Auto committed (changes are x permanent to DB)

Insert : insert a new row
Update : update existing Row
Delete : Delete Row
Merge : Merge 2 Rows /Tables.

TCL: Transaction control language.

Keep a check on other commands & their effect on the DB.

commit	permanently save
RollBack	undo change
Savepoint	save Temporarily.

DCL: Data control language:

grant	grant permission of right
Revoke	take Back permission.

DQL : Data Query Language:

select	Retrieve records from 1 /More Table

INT, FLOAT, Double, varchar, char, Date, Text

integer Float characterse Integers single char vals Date

TEXT: storing profile information of a social networking website, ...

ALTER:

- * Add col to existing Table
- * Rename Existing column
- * Change datatype of any col / Modify its size
- * Drop column from Table

Spl features:

Add Multiple columns

Add column with default value

Eg:

```
ALTER TABLE table_name modify
(
    column_name datatype;

);
```

TRUNCATE:

→ Removes all records from Table
It x not destroy Table's structure.

→ More or less same as Delete

→ Of

DROP:

→ completely removes a Table from DB.

→ Destroys the Table structure

INSERT:

Managing data in DB.

They are not Autocommited.

changes made by DML are not permanent to DB.

INSERT INTO Table-name VALUES (data1, data2, ...)

Insert into specific columns:

INSERT INTO student (id, name) values (102, 'Alex').

null, default values: 3rd parameter (102, 'Alex', null)

(. . . , default)

UPDATE:

Real world example: update status in Facebook

Eg:

Update table_name SET col_name = new_value WHERE condition;

Eg:

Update student SET age = 18 where id = 102;

Increment Integer value:

update student set age = age +1;

DELETE:

DELETE TABLE TABLE-NAME

Delete particular record:

Delete from student where id = 103;

Delete Truncate

Deletes all Rows from Table . Deletes all records stored in Table

Eg: 10 rows, autoincrement 1° key

If deleted, 1° key again starts from 11.

But in truncate, 1° key is reinitialized, starts from 1.

COMMIT:

save Transaction permanently to the DB.

↓ SP

Dn using DML commands like

Insert, Delete, Update

↓

changes are not permanent.

changes made by these commands can be rolled back

COMMIT;

ROLLBACK:

Restores DB to last committed site.

Also used with savepoint command to jump to SP

If we update ch to DB, but realise those changes are not required, Rollback those changes (if they were not commited using commit command)

SAVEPOINT:

Temporarily save a transaction so that you can rollback to the point wherever required

DCL: (Data ctrl Language):

It is used to control priveleges in DB.

To perform any operation in DB, such as creating Tables, sequences / views we need priviliges.

Types: * system
* object.

commands:

→ grant : when we create user in SQL, it doesnot allow to login, create session until unless proper permission / priveleges are granted.
→ Revoke

Eg: GRANT CREATE SESSION TO username;
GRANT CREATE TABLE TO user;

Revoke: To Take Back permissions.
Revoke create Table from username;

DCL commands:

select

where : specify a condition for execution

like : compares data with expression using wildcard ops to match patt in exec.

% _
>1ch single

order by

group By

Having

District

AND, OR

LIKE

select * from student where sname LIKE 'A%';
↓
A character starts with

ORDER BY :

arranging retrieved data in sorted order

By default sorts in ascending order

DESC keyword in descending order

op:

Eg: select * from EMP ~~whe~~ order by salary .
DESC

GROUP BY:

group the results of select Query Based on 1/more columns.

Also used with SQL func to grp result from 1 / ↑ Tables.

Eg:

select col name, function (col-name)

from Table-name

where condn

groupby col name

Eg: select Name, Age
from employee
groupby salary.

We get a dataset with unique salaries listed.

Emp Table :

Eid	Name	Age	salary .
401	Anou	22	9000
402	shane	29	8000
403	Rohan	34	6000
404	Scott	44	9000
405	Tiger	35	8000

Select Name, Age form
Emp groupBy salary

Name	Age
Rohan	34
Shane	29
Anu	22

HAVING:

give more precise condition just like where used with ↳ select

Having used with groupBy.

Eg: syntax:

select colname, funname
from Tab-name
where col-name condn
groupBy col_name
Having function(col-name) condn

Eg : select *
from Emp groupby Name
having sum(prevBal) > 3000 .

oid	order_name	prevBal	Name
11	01	2000	Alex
12	02	1000	Adam
13	03	2000	Abhi
14	04	1000	Adam
15	05	2000	Alex

Op:

oid	o-name	prev	Nam
11	01	2000	Alex

// Alex had PrevBal of 4000 .

DISTINCT:

Used with select stmt to retrieve unique values from Table.

Removes all duplicate records while retrieving from any Table in DB.

- Syntax:

Select distinct col-name from Table name

// Refere prev Page emp Table

- Eg:

select distinct salary from Employee

SALARY
9000
6000
8000

// retrieves unique salary values from Table.

AND: And is used to set Multiple conditions with where clause, alongside SELECT, UPDATE, Delete sql queries.

select * from Emp where salary <1000 and age >25.

SQL constraints:

Rules used to limit Type of data that can go in atable to maintain → Accuracy
→ integrity

columnlevel const : Limits only column - data
Table level const : limits only Table data

Constraints → Make sure that Integrity of data is maintained DB.

Notnull, unique, 1° key, Foreign key, check, Default

Notnull:
once not null is applied to a column, you cannot pass a null value to that column.

Unique:
It doesnot have duplicate data.
Field / column has unique values.

1° key:
uniquely identifies each Record in, DB.
1° key → contains unique value
→ x cont null value

Foreign key:
Relate 2 Tables
Used to Restrict actions that would destroy links b/w Tables

CHECK CONSTRAINT:

```
Create Table Student (
    s-id not NULL CHECK (s-id >0)
    NAME varchar (60) not NULL,
    Age int
);
```

QL functions:

i) Aggregate functions:
Returns single value after performing calculations on group of value

i) Avg // select Avg (col-name) from Table-name

ii) count // Returns the No of rows present in Table based on some condn/ w/o any condition.
select count (name) from Emp
where salary = 5000;

select count (distinct salary) from Emp;

iii) First first } value of selected column
iv) Last last } value of selected column
v) Max maximum value from selected col.
vi) MIN
vii) SUM Total sum of selected columns numeric value

ii) scalar functions:
returns single value from an i/p value

i) UCASE: converts value of str-col to uppercase
select ucase (name) from Emp

ii) LCASE()
iii) MID: extract substring from column-values of str-type in a Table
iv) ROUND: round field to nearest integer

SQL JOIN:

↳ Fetch Data from 2 /more Tables which is joined to appear as a single set of data.

↳ Join → query for joining 2/more Tables.

Types of join:

Inner, outer, Left, Right

Inner join:

(or) Equi join in which the Result is Based on Matched data as per the Equality condition in the SQL query

class - Table

ID	NAME
1	Abhi
2	Adam
3	Alex
4	Anu

classinfo Table

1	Delhi
2	Mumbai
3	chennai.

select * from inner join class_info where class.id = class_info.id

Natural join:

Inner join Based on column having same name & same dataType present in Both Tables to be joined.

select * from Tablename1 NATURAL JOIN Tablename2.

Book

1	Abhi
2	Adam
3	Alex
4	Anu

Student

1	Delhi
2	cheni
3	Mumbai.

select * from Book NATURAL JOIN student

op of Natural join

1	Abhi	Delhi
2	Adam	Mumbai
3	Alex	Chni.

INNER JOIN

↳ Based on Matched Data as per equality condn

OUTER JOIN.

↳ Based on Matched & Un Matched data

↳ left Outer Join

↳ Right outer join

↳ Full outer join

left outer join :

Returns Result set Table with Matched Data from 2 Tables & remaining rows of left Table & null from remaing Right Table's columns .

syntax :

select col-name from TAD-1 , Tab-2 on tab1.colname=tab2.colname

Eg:

class Table

ID	Name
1	Abhi
2	Adam
3	Alex
4	Anu
5	Ashish

class-info Table

ID	Addrs ~~Name~~
1	Delhi
2	Mumbai
5	Chennai
7	Noida
8	Panipat

select * from left outer join class-info on (class.id = class-info.id);

Right outer join:

Returns Resultset Table with Matched data from 2 Tables being joined,
Remaining rows of Right Table /
Null for rem left rows.

op for Right outer join:

ID	NAME	ID	ADDRESS
1	Abhi	1	Delhi
2	Adam	2	Mumbai
3	Alex	3	chennai
null	Null	7	Noida
null	Null	8	Panipat

op for left outer join:

ID	NAME	ID	Address
1	Abhi	1	Delhi
2	Adam	2	Mumbai
3	Alex	3	chennai
4	Anu	null	null
5	Ashish	null	null

Full outer join:

Resultset Table with Matched data of 2 table then remaining Rows of both left & Right Table.

op fullout:

ID	NAME	ID	ADDRS
1	Abhi	1	Delh
2	Adam	2	Mum
3	Alex	3	chn
4	Anu	null	null
5	Ashish	null	null
null	null	7	NOIDA
null	null	8	PANIPAT

ALIAS:

→ gives an Alias name To Table / column, which can be a Resultset Table.

→ Useful incase of large/complex queries.

→ gives short alias name for column/tables

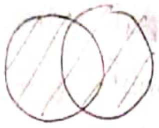
Eg:

select * from employee-name as emp;

SET operations in SQL:

- * UNION
- * UNION ALL.
- * INTERSECT
- * MINUS .

UNION:



combine Results of 2/more select stmts.

eliminates duplicate Res from O/p.

Number of col & datatype must be same

UNION ALL:

similar to Union.

But also shows duplicate Rows

First Table

ID	NAME
1	Abhi
2	Adam

second Table

ID	Name
2	Adam
2	chester

```
SELECT * from FIRST
UNION
SELECT * From second
```

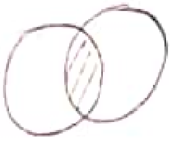
UNION op:

id	Name
1	Abhi
2	Adam
3	chester

UNION ALL:

id	Name
1	Abhi
2	Adam
2	Adam
3	chester

INTERSECT:

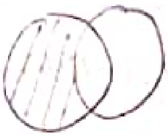


But Returns Record which are common from Both select statements

op:

ID	NAME
2	Adam

MINUS:



combines results of 2 select stmts & return those in final result, belongs only to 1st set

op:

ID	NAME
1	Abhi.

SQL Sequence :

SQL Sequence used in some DB (such as MySQL) to Autoincrement .

Autoincrement:

Increments the col value by 1 each time a new recorded in The Table

syntax to create sequence;

```
CREATE SEQUENCE seq-name
START WITH initial-value
INCREMENT BY increment-value
MAXVALUE maximum-value
CYCLE | NOCYCLE;
```

initial value → starting val for seq

increment value → upper limit for Max val upto which seq will ↑ .

maximum value → value by which seq is inc.

Eg:

```
Create sequence seq-1
START WITH 1
INCREMENT BY 1
MAXVAL 999.
CYCLE;
```

we have class Table

ID	NAME
1	Abhi
2	Adam
4	Alex

```
INSERT INTO class VALUE (seq1.nextval, 'anu');
```

op Resultset Table :

ID	NAME
1	Abhi
2	Adam
4	Alex
1	Anu .

SQL VIEW:

Logical subset of data from 1/↑ Tables.
Used to Restrict data Access.

↳ view is created using data fetched from some other Tables.

2 Types: < simple
complex

SIMPLE VIEW	COMPLEX VIEW
created from 1 Table	created from 1/↑ Table
Donot contain function	contain functions
× cont group of Data	cont ✓ grp of Data

UN

Operations:
- create view
- Display view
- Force view
- Update view
- Read-only view

* UNIQUE INDEX:
ensures the values in index key col are unique.
No duplicate records can be inserted in col.

* Retrieve duplicate Records:
→ Using groupby clause to find duplicates
↓ (or)
ROW_number

→ select a,b, count(*) from t1 group by a,b
having count (*) >1.

* ROW_number: Assigns sequential integer to each row of result set

```
Select colname(s)
from Table name
where condn
groupby colname(s).
having condn
order by colname(s)
```

Constraints:

↳ Set of rules for all records in Table.

↳ Any const gets violated abort action that caus

Types of constraints

↳ Not null : col value cant be left null

↳ Unique : each Row & col has unique val, cant be null

↳ 1° Key : identify partic record as unique key

↳ Foreign key : Referential Integrity in DB

↳ Check : column fulfills specific condn.

JOINS:

* Inner join: All rows from Both Table, if it has atleast 1 matching column

* Full join/ Full outer : All rows if there is match in either left / right Table

* Right outj : All right + Matched rows in left

PIFF

↳ Inner joins : doesnt include non-matching rows.

↳ outer joins : Include them

* Equijoin: Match column values of associated Tables. equal sign → used as comparison operator in where clause to refere equality.

* Crossjoin: prod Result set : Multpl no of no of Rows in 1st Table × 2nd Table.

* Self join: Table joined with itself. When Table has foreign key that references its 1° key.

One-Many: splitting the data into 2 Tables with 1° key, Foreign key.

Many-Many: Junction Table with keys from both Tables forming composite 1° key of June Table

One to one: single Table, Rarely as 2 Tables with 1°, Foreign key.

Transaction: sequence of Task performed on DB in a logical manner. 4 TCLs: commit, Rollback, setTransaction, save point

properties: * ACID

Aggregate functions: 7 agg. Avg() Max() Sum() count() Min() First() Last().

Scalar Functions: Ucase() Mid() Format() Lcase() Len() Round().

Trigger: kind of stored procedures to create response to specific action perf on Table. Eg:

View: Virtual Tables that cont Rows/Tables from 1 or more Tables Eg: supraja Arthi from all Tabl Stud/cours

Clusterd index:

1° key constraint creates clust index.

Def order in which data is physically stored in Table

Non-clust:

Doesn't sort physical data in Table.

>1 NCI per Table.

NCIndex has address of record, col values of index

(DML)
Delete : specific Row. where clause.

(DDL) Truncate: All Rows.

Drop : All Rows (Can't be Retrieved back)
Removes entire Table from DB.
↓
(DDL)
Integ const removed.
frees up Memory.

DBMS:

Tech of storing & retrieving users data with eff & acc

Trad Data → Files.

Features:

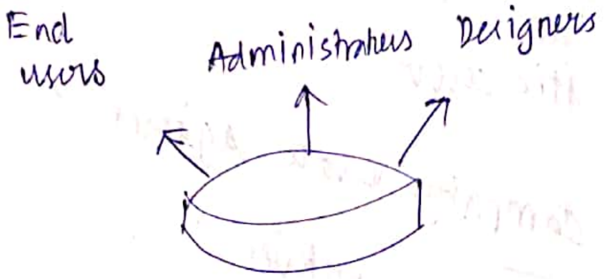
- Real World Entity. → Realistic, RWE to design architecture
- Relation Based Tables → Entities & Relations = Tables
- Isolation of data & Appln
- Less Redundancy → Normalization, splits relation, attrib to Redundancy
- consistency → state where every rel rem cons
- Query Language
  ↓
  Efficient to retrieve & Manipulate Data

Applns of DBMS:

Data of students, conclude abt Toppers & Avg Marks

Atomicity
consistency
Isolation
Durability

Users:



3tier Architecture:

Presentation Tier ⇅ End users.

Application tier ⇅ Application server & Prog that access the DataBase

DataBase tier Query processing languages

Data Models:

- * Logical structure of DB is Modelled
- * fundamental entities to Entroduce abstraction

ER Model:

- * Based on notion of Real world Entities
- * Real world scenario into DB Model.
- * It creates Entity, Rel, Attrib & constraints.

Relational Model:

- * Based on 1st order predicate Logic.
- * Data is stored in Tables called Relations.
- * Relations can be Normalized.
- * each Row has unique value.
- * Each column cont values from same domain

Comp of relational DB:

*Table *Record/Tuple *Instance * schema

* keys * Field/column name/ Attribute

Distributed DB:

Data is distributed among diff dB systems of an organization

Types:

→Homogenous DDB : DB systems execute on same OS, same appln process, same H/W devices

→Heterogenous DDB : diff → Hetero.

Centralized DB:

stores DB at cent DB systems

Allows users to store data through sev applns.

Eg: central DB of library in college/univ

Indexing:

- We know that Data is stored in form of 'Records'
- Every record has a key field → recognise uniq
- Indexing is a datastructure technique to efficiently retrieve records from DB files based on some attributes

- Types:

↳ Primary Index:

→ Defined on an ordered data file.

→ Data field is ordered on a key field.

→ Key Field → 1° key.

↳ Secondary Index:

→ generated from field which has and key + unique value in every record.

(non-key with duplicate value.

↳ clustering:

Defined on an ordered Data file.

Two Types: Dense Index
sparse Index