1) * Database:
- Collection of data in format that can be easily accessed.

2) Database management system:
- Software that manages database.

User → [DBMS] → [DB]

3) Types of DB

Relational	Non-Relational (NoSQL)
- Data stored in tables	- Data not stored in tables
- Eg: MySQL, Oracle, PostgreSQL	- Eg: MongoDB
- Use SQL to work with DBMS	

4) SQL - Programming language to interact with relational db.

* Advantages of DBMS
1) Data Sharing: Data from single db can be simultaneously shared by multiple user - Enables end user to react changes quickly in db environment
2) Integrity constraints: Existance of constrains allows storing data in organized & refined manner

3) Controlling redundancy in db: Eliminates redundancy by providing mechanism that integrates data in single db

a) Data independance: Allows changing data structure without altering the composition of any existing appl^n program.

5) Provides backup & recovery facility can be configured auto create the backup of data & restored data in db

6) Data security :

6) Different languages in DB

- DDL (Data defindt^n) - Commands required to define db - create, alter, drop, truncate, rename
- DML (Data manipul^n) - Commands to manipulate data in db - select, update, insert, delete
- DCL (Data Control) - Commands required to deal with user permiss^n & controls db = Grant & revoke
- TCL (Transaction control) - Commands to deal with transaction of db - Commit, rollback, savepo^n

- Attributes
  - Piece of data that describes entity
  - Eg: in student db: rollno, dob, age, name, mobile
  - Types
    1. Simple attributes: Attr that cannot be divided further in sub-attributes
       Eg: Age / Roll no
    2. Composite attr: Attr made up of 2/more simple attr
       Eg: Name (Fname, Lname)
       Addr (Street, city, state, country)
    3. Multi-valued attribute
       Attr with more than 1 values
       Eg: Phone no., email addr
    4. Single-valued attr: - Attr with only 1 value
       Eg: Roll_no, dob, age
    5. Derived attr: can be derived from other attr.
       Eg: Age from DOB
    6. Key attr: Unique value, used to uniquely identify each row in table
       Eg: stud_id, roll_no,

o Keys in SQL
- Used to uniquely identify any row/ record of data from table.
- Used to establish & identify relationship between table.
- Types
① Primary key: It uniquely identifies each record in the table.
must be unique for each record / No null
- Eg: E_id (Employee table)
② Candidate key: Attr / set of attr that can uniquely identify tuple
- Except primary key, all others are candidate
- As strong as primary.
- Eg: Passport no, Visa no, (Employee table)
③ Super key: Attr set to uniquely identify tuple
- superset of candidate key
Emp_Id, E_name (in emp table)
④ Foreign key (combo): Field that refers to Primary key in another table
- Establishes link between 2 tables
- Dept id (Dept table in emp)
⑤ Composite key: Primary key composed of multiple columns each of which is individually not unique but unique when combined
- emp_id, e_mail, proj_id
⑥ Alternate key:

* constraints in SQL
- used to specify rules for data in table
- Ensure accuracy & reliability

1] Primary key constraint:
- Ensure each row in table is uniquely identified
- must contain unique values
- only 1 primary key for table

2] foreign key constraint
- Enforces referential integrity between 2 tables
- Ensure that value in column (/ columns) of one table match the values in another table's primary key / unique constraint

3] Unique constraints
- Ensures that all values in a column are distinct from one another

4] Check constraint
- Specifies a condⁿ that must be met for a row to be inserted or updated on a table

5] Not Null constraint: Ensures column cannot contain null value

* Aggregate functions:
- Functions that operate on a set of values & return a single aggregated result.
- Used in queries to perform calculations across multiple rows of a table

1] Count: count the no. of rows in a result set or no. of non-null val in column
SELECT COUNT(*) FROM employees;

2] SUM: Calculates sum
SELECT SUM(SALARY) from employees;

3] AVG: Calculates average
select AVG(salary) from employees

4] MIN: Return minimum val in col
select MIN(salary) from employee;

5) MAX: Return max val in col
select MAX(salary) from employee;

* SQL query to find nth highest salary

Select salary from employee
order by salary desc limit (n-1), 1);

CREATE copy of table

only schema — create table emp (new) like employees; (old)

Data + schema — create table emp select * from employees

ACID properties

- Set of 4 key characteristics, that gurantee reliability, durability & consistency of db transactions.
- Ensure db transactions are processed reliably in a way that preserves integrity of data

1] **Atomicity**: Ensures transactions are treated as single unit of work.
- means, all ~~transaction~~ operations within the transactions are successfully completed / none of them are.
- No partial completion in trasn
- If any part fails, whole transacn is rolled back to initial state & db remains consistent

2] **Consistency**: Ensures that db remains in a consistent state before & after transactions
- Means, transactn must follow all rules & constraints defined in db schema

3] Isolation: Ensures concurrent execution of transactions does not result in data inconsistency.
- Transactions should operate independently of each other, even when executed simultaneously.
- Prevents issues like dirty reads, non-repeatable reads & phantom reads

4] Durability: Ensures changes made by committed transactions are permanently saved & persisted in db, even in the event of a system fails / crash.
- Once transaction committed, changes should be durable & not lost

* Data integrity
- Refers to overall accuracy, completeness & reliability of data
① Domain integrity: int, char, etc
- Restricts format, type & value of data recorded in db

② Entity integrity:
- Ensures that data is not recorded multiple times
- Eg: By using primary key

③ Referential integrity: Remove duplicate data record
- Uses foreign key

④ User-defined integrity
- Fulfill their specific requirements
- Eg: for roll no, I want ICB75, 70--
- User defines the constraints

* Difference between:

1] DELETE, TRUNCATE & DROP

DELETE	TRUNCATE	DROP
- Delete one / more rows at time.	- Deletes all the data at once	- Whole table / structure is deleted.
- can undo the delete	- cannot undo	- Cannot undo.
- (Temp)	(Temp)	(Permanent)
- ~~DDL~~ DML	- DDL	- DDL

2] WHERE & HAVING

WHERE	HAVING
- Puts a cond^n on a record	- Puts a cond^n on group of records
- Individual	- Group
- Cannot used with agg fun	- Involves agg func
	- ~~Applied after grp by~~

3] Group By & ORDER By

Group By	ORDER By
- Group the data	- Sorts data in asc / desc order
- Perform agg fun on each ~~row~~ grp (used with)	- used with SELECT stmt
- Transforms result set into summary rows	- Does not Transform
- Group rows based on specified	- Sorts rows based on specified

4) CHAR vs VARCHAR

	CHAR	VARCHAR
length	- Fixed, specified at time of creation	- Variable length, specified max. length
Storage	- Fixed length, pads shorter strings with spaces	- Variable length, store actual data
Efficiency	- More efficient for fixed	- Maybe more for var-len data
Space usage	- Wastes space for short val	- Space efficient

5) UNION & JOIN

UNION	JOIN
- 2 tables combined column wise	- 2 tables combined row wise
- Combines RS of 2/ more queries	- Retrives related data from multiple tables
- Combines & removes duplicate rows from RS	- Retrives matching rows from joined tables

6) IN & EXISTS

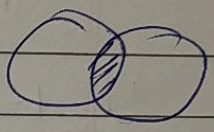
IN	EXISTS
- Checks if value exists in set of values	- Checks if subquery returns any row
- Compares a value with a list of vals returned by subquery	- Checks existence of rows returned by sub query
- Select * from emp where dept in ('IT', 'CSE')	- Select * from emp e where exists (select 1 from dept d ... e.dept

JOINS

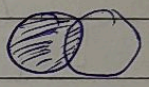
- Used to combines rows from 2/more tables based on related col^n between them

(1) (INNER) Join:

- Returns records that have matching val in both tables
- If no match, rows not included in R,
- Select orders.o_id, cust.c_id
  from orders
  inner join cust on o.c_id = c.cid;



(2) LEFT (OUTER) Join:

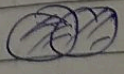


- Returns all rows from left table (table1) & matched records from right table
- Eg: select order.oid, cust.cname
  from order
  left join cust on orders.cid = cust.cid

(3) RIGHT (OUTER) JOIN:

- Returns all records from right table & matched records from left table
- select orders.o_id, cust.cname
  from orders
  right join cust on orders.cid = cust.cid

(4) FULL (OUTER) Join



- Returns all records when there is a match in either left/right table

(5) CROSS JOIN - returns cartesian prod of 2 tables

- Generates all possible combn of rows from tab

* NORMALISATION
- Process of organizing a db schema to minimize redundancy & dependancy.
- Aims: Structure data in way that ensures data integrity, reduces data redundancy, improves query performance

1) 1-NF (First normal form)
- Each col should contain atomic val, (NO multivalued attr) Java, CPP (sug)
- 1 val for each row

2) 2NF
- meets all requirements of 1NF
- No partial dependancy
- Eliminates partial dependancy by ensuring each non-key attr is fully functionally dependent on the PK

3) 3NF
- meets 2NF
- Eliminates transitive dependancy, by ensuring non-key attr are not functionally dependant on other non-key attr.

4) BCNF (Boyce codd)
- Stricter form of 3NF
- Eliminate non-trivial functional dependancy attr on PK

* PATTERN matching.

- 'like' clause is used.
- Allows to use wildcard characters:
  % - Represents 0/more
  _ - Represent single character

eg: select * from emp where lname like 'sm%';

- Name start with p & has 3 char in name

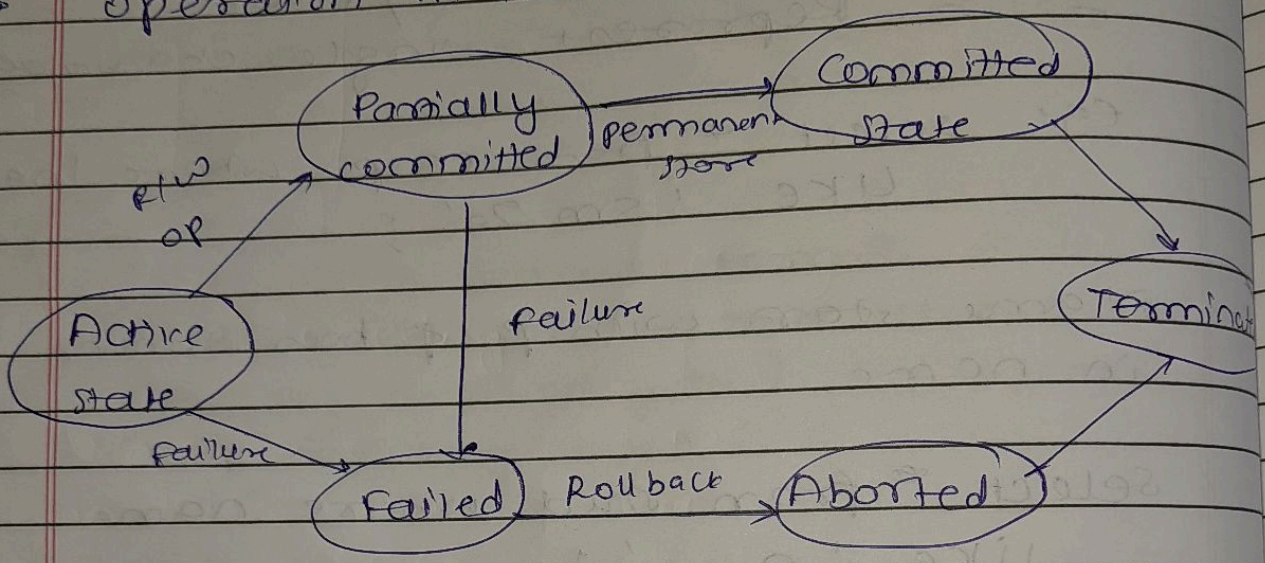
Select * from stud where name like 'p__';

* Character manipulation:
  - Lower() arya
  - Upper() ARYA
  - Initcap() Arya
  - Length() 4
  - Concat() Anya + Tanja

* View
  - Virtual table based on 'SELECT' query
  - Does not store data itself
  - Dynamically retrive data from underlying tables every time it is queried

- Create view abc as select col1, col2 from student where cond^n.

→ operation in transaction



→ Indexing:
- Technique to improve the speed & efficiency of data retrival operation, such as SELECT queries, by creating special ds that allow faster access to data
- Index: Special structure associated with tables. Stores a sorted list of vals for 1/more cols in table. Along with ptrs to corresponding rows in table.

• Purpose:
- Idx created to speed up retrieval of data from tbl by providing quick access to rows based on indexed cols

- Improve performance of 'SELECT' queries, specially when filtering, sorting / joining data.

- Structure:
  - They implemented as B-tree / hash ds, depending upon db system.
  - B-tree idx most used, well suited for range queries & sorting.
  - Hash idx are faster for exact match queries but does not support range queries & sorting.

- CREATE index idx_name on table_name (c1, c2, ...);

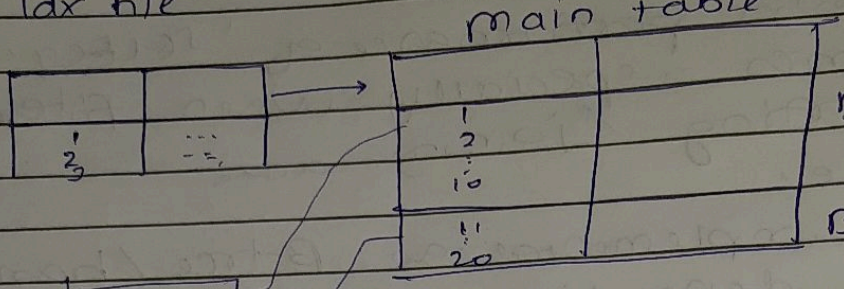
- create index idx_lname on employees (lname);

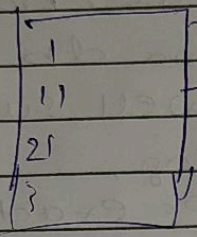
- Usage:
  - Automatically used by db optimizer to speed up query execution.
  - Excessive indexing can lead to increased storage overhead & decreased performance for data modification op (insert, update, delete)

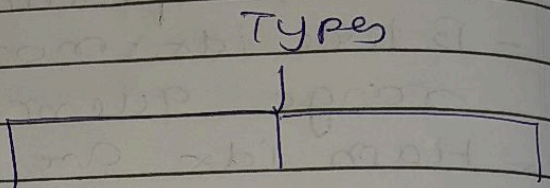
- Types:
  - Just like indexes in books
  - Used to optimize performance by minimizing no. of disk accesses required when query processing search of disk block

PK

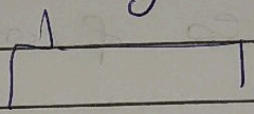
search key	data reference



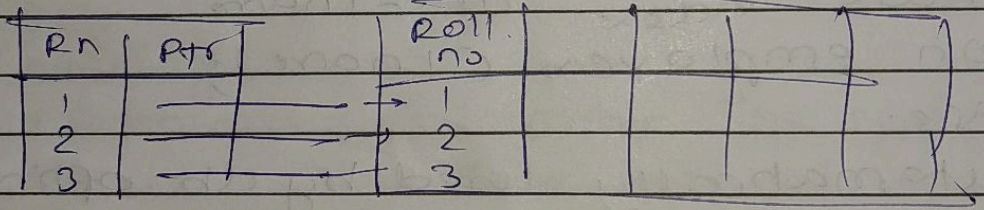




1] Dense indexing:

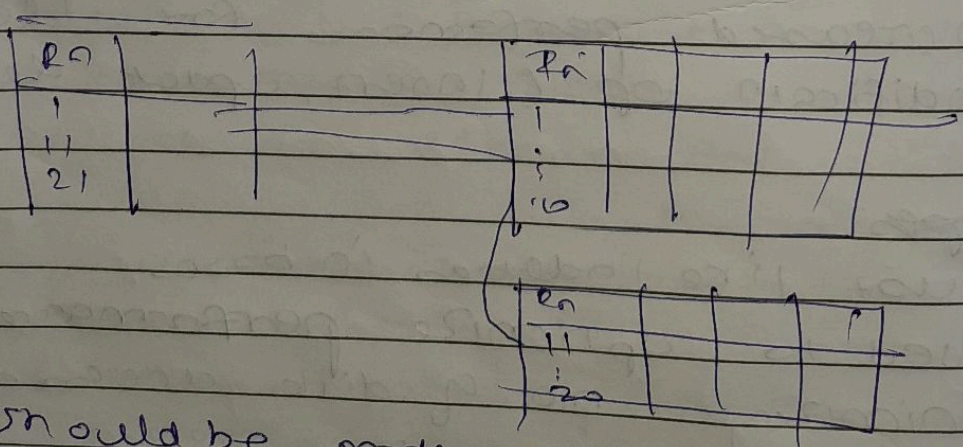


- No. of entries in index table is same as the no of entries in main table



2] Sparse

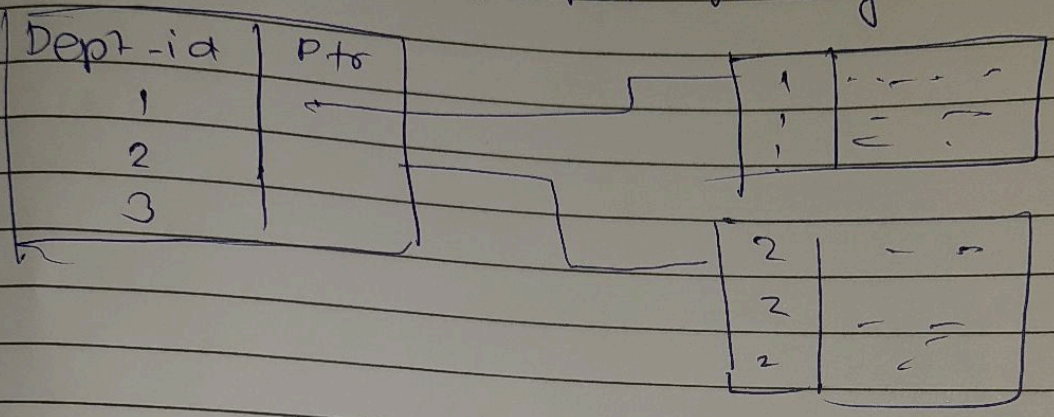
- Not same no. of rows as idx table



- Should be ordered

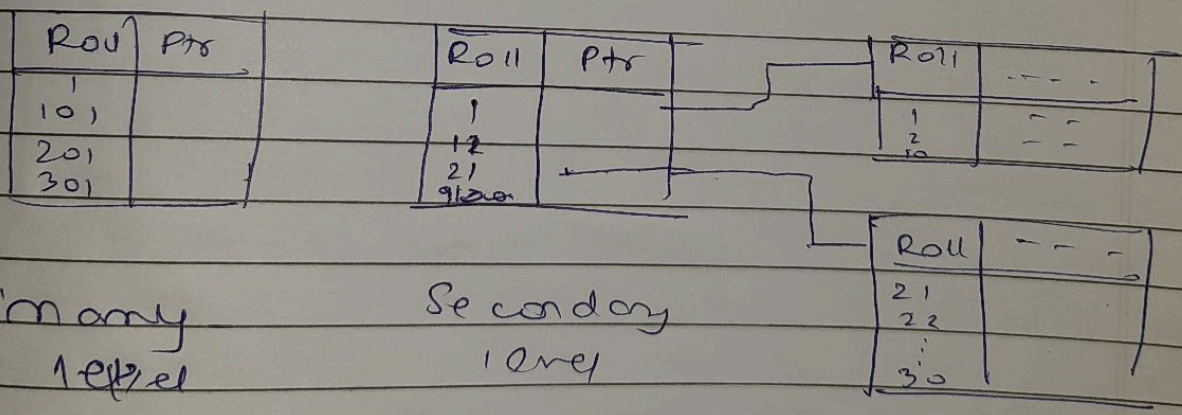
2] clustering

- In case of a non-unique key, such as dept-id (same for many stud)



3] Secondary indexing

- 2 level index



Primary level

Secondary level