→ Organized in a way to make querying fast & efficient.
→ Ex :- A retail company stores all its past sales data in a data warehouse to find out which product sells the most in winter.

(7) Data Mining :- → statistics
→ ML
→ AI

→ The process of analyzing large sets of data to ~~the~~ find patterns, relationships or trends.
→ Also, called Knowledge Discovery in DB (KDD).
→ uses techniques from statistics, machine learning, and AI.
→ Helps in decision-making.
→ Netflix uses data mining to understand your viewing habits & recommend similar shows.

SAP → Systems, Applic^n., Products

→ Ex:- In SAP, a logical DB allows users to read employee records without accessing the entire HR database.

(4) Web DB :-

→ DB that are connected to web applications, allowing users to access and manipulate data over the internet.

→ users interact via web browsers using forms or interfaces.

→ Popular in e-commerce, social media, online banking, etc.

→ uses HTML, PHP, JavaScript, SQL, etc. to communicate with the DB.

→ Ex - When you search on Amazon, your req. queries a web DB that returns product details.

(5) Distributed DB :-

→ A DB that is stored across multiple locations or computers but appears to users as a single DB.

→ Each location is called a site.

→ Data is replicated or partitioned across sites.

→ ↑ reliability, availability, & speed.

→ Ex- Google uses distributed DB to store data across the world for faster access.

(6) Data warehousing -

→ A centralized storage system where data is collected from diff. sources for analysis and reporting.

→ used for business intelligence (BI).

→ Stores historical data (old data) to analyze trends.

CAD - Computer Aided Design.
ERP - Enterprise Resource Planning.

MOD-VI

① Object - Oriented Database (OODBMS)
→ These DB store data in the form of objects, just like how data is represented in OOP (eg. Java, C++)
→ An object has attributes (data) & methods (functions).
→ It supports concepts like Inheritance, encapsulation & polymorphism.
→ useful in applications where complex data is involved (eg. CAD, multimedia databases).
→ Ex - Storing a student object with name, age, and methods like calculateCGPA().

② ~~Objection~~ - Relational Databases (ORDBMS)
→ Hybrid DB that combines features of relational DB (RDBMS) and OODBMS.
→ Stores data in tables like RDMS but allows you to use objects & classes like OODBMS.
→ Allows custom data types, inheritance, & encapsulation within relational models.
→ Ex:- PostgreSQL supports ~~oo~~ object - relational features where you can define your own complex data types and use them like a normal table column.

③ Logical DB
→ not a physical DB but a virtual view created from data in existing DB.
→ Provides a logical view (like a customised lens) of the data for specific purposes.
→ Often used in large systems like ERP to improve security & usability.
→ Logical DB are read-only & can't store data physically.

SELECT * FROM users WHERE username = 'admin' AND password = '' OR '1' = '1';
This always returns true, & the hacker gets access without password.

Prevention:→ Use prepared statements / Parameterized queries.
→ Always validate user inputs.
→ Avoid building SQL queries using raw user input.

2. Easier to manage - Especially useful in big systems with many users (like schools or companies).
3. Improve Security - limits access to data by assigning minimum necessary rights to each role.
4. No user-based permission Access isn't given to individuals but to roles they play.
5. Widely used in organizations: like hospitals, offices, universities, banks.
6. Ex:- In a college system, only "Admin" can add students, "Teachers" can update marks, "Students" can view marks.

4) Intrusion Detection - is like security camera for your DB. It watches for any suspicious activity.
Ex - Someone trying to log in 10 times with wrong password.
- unusual query running at midnight.
Such activities are flagged as potential attacks.
Tools like Snort, OSSEC, or built-in DBMS features help in detection.

5) SQL Injection - hacking technique where the attacker inserts harmful SQL code into your application to access the DB illegally.
Ex:- Suppose a login page has this code:
SELECT * FROM users WHERE username = 'input' AND password = 'input';
If the attacker enters:
Username: admin
Password: 'OR' 1' = '1 'OR' 1 '=' 1
The query becomes:

DAC (Discretionary Access control)

1. Data Owner controls access - The person who creates the data decides who can access it.
2. Users can share permissions: Owners can give or take access (read / write / delete) to / from others.
3. Flexible system: Access control is simple & easy to manage by users.
4. Security is weaker: Bcz users can give permissions to others, it's less secure.
5. Used in general systems: Commonly used in personal computers, file-sharing apps.
6. Ex: You create a file & allow your friend to read or edit it.

MAC (Mandatory Access Control)

1. System enforces access - OS or administrator sets strict access rules.
2. Based on security levels - Both users & data are assigned levels like "Confidential", "Top Secret".
3. Users can't change permissions: Unlike DAC, users can't share or change access.
4. Highly secure: Prevents unauthorized access even from users with partial rights.
5. Used in sensitive environments: Such as military, defense, or govt. systems.
6. Example: A file marked "Secret" can only be opened by users with "Secret" clearance or higher.

RBAC (Role Based Access Control)

1. Access based on roles: Users are assigned roles (like Admin, Teacher, Student), and roles have permissions.

vijeta

MOD-V

Database Security :- all about protecting the data from :-
- Unauthorized access.
- Data leaks
- Hacking
- Malicious activities.

1) Authentication - Process of verifying the identity of a user.

Ex :- When you log into Gmail, you provide your username & password.
The system checks if it's really you. If correct - you're authenticated.

Purpose :- Makes sure that only valid users can access the DB.

2) Authorization and access control - Once a user is authenticated, authoriz^n. decides what they can do.

For ex :-
- Admin : can read, write, delete.
- Student : can only read their results.

Access control - means giving permissions like :
Read, write, update, delete
So, access is controlled based on user roles.

3) DAC (Discretionary Access Control)

Owner of the data decides who can access it.

ex :- If you created a file / table, you can decide :
- who can read it
- who can change it.

* Flexible, but less secure (users can give permission to others).

1. Read phase → Transac^n reads data & performs operations in memory.
2. Validation phase → Before committing, DB checks if this transaction conflicts with any other.
3. Write phase → If no conflict - changes written to the DB, If conflict - transition rolls back.

Ex:- T1 & T2 both read the same product price. ₹100
T1 changes price to ₹110, T2 changes to ₹105.
Both try to commit!
- If T1 commits 1st, DB checks T2's version, T2 is rejected as it worked on outdated data.

Benefits:- No locks needed, Great for reads, Simple to implement

(6) Database Recovery:-
Sometimes the system crashes during a transaction. DBMS must restore the DB to a consistent state.
Why does recovery matter?
To handle:- Power failure, hardware crash, sw crash

Recovery Technique -
a) Log-based Recovery - DBMS maintains a log file (a record of all actions).
In case of crash: • undo uncommitted transactions
• Redo committed ones.

b) Checkpointing - DBMS saves a snapshot of the DB at regular times (called checkpoints). If a crash happens
- It starts recovering from the last checkpoint, not from the beginning.
* faster recovery!

Why MVCC? To improve performance when many users have reading & writing at the same time.
Instead of locking rows, MVCC gives each user their own view of the data.

How it works?
Let's we have a row:
Student: {Name: Anu, Marks: 85}
Now:
1. Reader Transaction (T1) starts → sees the current version (Marks = 85).
2. Writer Transaction (T2) starts later → updates Marks = 90
But instead of changing the original row, it creates a new version.
3. T1 still sees Marks = 85 & T2 sees Marks = 90.
This ensures: No waiting, No conflict, No locking needed for reads.

Benefits of MVCC: - high performance, no read locks, consistent views.

Optimistic concurrency control (OCC)
OCC assumes that conflicts are rare.
So, it lets transactions run freely without locking.
At the end, it checks, if there's any conflict.
If conflict is found:
- The transaction is rolled back & retired.

Why OCC? few conflicts, mostly read operations, many users working simultaneously.

How it works? There as 3 phases: -

Feature	Locking-based	Timestamp-based
i) Basic idea.	Uses locks (shared or exclusive) to control access.	use Timestamps to decide the order of operations.
ii) Control type	Pessimistic - assumes conflict will occur.	Optimistic - assumes conflict is rare.
iii) How it works	Transactions lock data before accessing it.	Transactions are assigned timestamps when they begin.
iv) Read/write rules	Must acquire lock before reading / writing.	Follows timestamp rules: older transac. get priority
v) Data Access	May block other transactions.	Transactions don't block each other.
vi) Ex:-	2-Phase Locking	Thomas Write Rule.
vii) Concurrency	Moderate (due to locks)	High (as there's no waiting due to locks).

(5) Multi-Version Concurrency Control (MVCC).

MVCC means DB keeps multiple versions of the same data instead of overwriting it immediately.

This way:-
- Readers don't block writers
- Writers don't block readers.

Used in PostgreSQL, MySQL (InnoDB), and Oracle.

(2) Concurrency control - When multiple transactions happen at the same time, they can interfere with each other & cause errors.
To avoid conflicts, DBMS uses concurrency control.
Problems without concurrency control:-
- Lost updates,
- Dirty reads
- Unrepeatable reads.

Goal:- let many transactions run together safely & correctly.

(3) Serializability of Scheduling
Schedule → order in which operations of transactions are executed.
Some schedules can cause problems, while others give correct results.

Serializable schedule: A schedule is serializable if it gives the same result as some serial execution (i.e., one after another),
There are 2 types:-
- conflict serializable
- view serializable.

Serializability ensures that even if transactions run together, the final result is correct - just like running them one by one.

(4) Locking and Time Stamp-based Schedulers.

Featu
i) Basic
ii) Cont
iii) Ho
iv) R
v) D
vi)
vii)
(5)

MOD-IV

Transaction Processing :-

Transaction is a unit of work done in a DB. It could be a single query or multiple operations (like updating a bank balance or placing an order).

Ex:- check balance, Deduct amount, update account.

A good DB must ensure that:

- All steps of a transaction complete successfully, or
- If there's any error, nothing should change.

This is handled by Transaction Management in DBMS.

(1) ACID Properties :-

Atomicity Consistency Isolation Durability.

i) Atomicity - "All or Nothing" - Either the entire transaction is completed or nothing is done.

Ex:- In money transfer, if money is deducted from A but not added to B, the DBMS should cancel the whole transaction.

ii) Consistency - DB should always remain in a valid state before & after the transaction.

Ex: Total money in the system should remain same before and after transfer.

iii) Isolation - Multiple transaction can happen at the same time, but they should not affect each other.

Ex - 2 people booking the last seat in a train - Only 1 should get it.

iv) Durability - Once a transaction is committed, the changes should be permanent, even if the system crashes.

vijeta

How it works?

DBMS uses a Query Optimizer, which:

1. Takes the SQL query.
2. Converts it into relational algebra.
3. Generates many possible query plans.
4. Estimates the cost of each plan.
5. Chooses the lowest-cost plan.

Techniques used:-

a) Heuristic-based Optimization

Applies a fixed set of rules (heuristics) to transform queries.

Ex rules:- Apply selections early
combine projections
Avoid unnecessary joins.

fast but may not find the best plan.

b) Cost-based Optimization

- Calculates the estimated cost (CPU, I/O, memory) of each plan.
- chooses the lowest-cost one.
- More accurate but takes more time.

- used in most modern DBMS like Oracle, MySQL, etc.

Affect query cost → Size of tables
→ Indexes available
→ Data distribution
→ Available memory.

iii) Join Strategies – Join combines rows from 2 tables based on a related column (like foreign key).

Ex:- SELECT * FROM student JOIN department ON student.dept_id = department.dept_id;

This will join the student and department tables.

DBMS has many ways to do this. Some common Join strategies are:-

a) Nested Loop Join → For every row in table A, check all rows in table B.
→ Simple but slow for large tables.
→ Good when one table is small.

cost: $O(m \times n)$

b) Sort-Merge Join → First, sort both tables on join key.
→ Then merge them like in merge sort.
→ works well if data is already sorted.

cost:- $O(m \log m + n \log n + m + n)$

c) Hash Join → Build a hash table on the smaller table using the join key.
→ Then scan the larger table & match rows using the hash.

- fast & commonly used for large Joins.

iv) Query Optimization Algorithm – process where the DBMS choose the best executive plan among many possible ones. This step happens before actual query execution.

Goals:- Minimize disk I/O
Reduce CPU cost
Decrease query response Time.

Evaluation means: How the DBMS executes these operations (like select, project, join) & in what order.
Some operations are costly (like join), and some are cheap (like projection).
So, DBMS must choose an efficient evaluation plan, which:- → use less CPU time
→ uses less memory and disk I/O
→ Gives results faster
~~→ Giv~~

Instead of:-

Join → Filter → Project.

A better plan could be:

Filter → Join → Project.

This avoids joining unnecessary rows, saving time & space.

ii) Query Equivalence

Sometimes, 2 diff. queries give the same result but have diff. performance. They are called equivalent queries.

Ex:- 2 sql queries are equivalent:

SELECT name FROM student WHERE marks > 80;

and SELECT name FROM (SELECT * FROM student WHERE marks > 80);

But one might be faster than the other.

* Query optimizer in DBMS tries to find equivalent forms of a query that are faster.

The goal is to choose the best way to execute the query.

If we loose some dependencies, then we may not be able to enforce all data integrity rules using the decomposed tables.

Ex:- Suppose the original table has these FDs :-

- $A \rightarrow B$
- $B \rightarrow C$.

After splitting, if we lose $B \rightarrow C$ in the smaller tables, we can't check that rule anymore. That's bad. This is called dependency loss.

v) Lossless Design (Lossless Decomposition)

→ When we break a large table into smaller ones, we must be able to join them back and get exactly the same data as before - no extra or missing rows. This is called Lossless Join or Lossless Decomposition.

Ex - If you split a table & after joining them you get more / less depth, it's lossy decomposition, which is bad.

A good DB design always ensures lossless decomposition.

③ Query Processing and Optimization

When we write SQL query, DBMS performs many int. ~~storage~~ steps to convert it into results. These steps are part of query processing.

1. Evaluation of relational algebra expressions:

When we write SQL queries like:

SELECT name FROM Student WHERE marks > 80;

Internally, the DBMS converts it into relational algebra operations like:

π name (σ marks > 80 (Student)).

1NF (First Normal Form)

Rule: No repeating groups or arrays; every cell should contain atomic (single) values.

Ex:-

Roll No	Name	Subjects
101	Ravi	Math, Science ✗ (Not 1NF)
101	Ravi	Math ✓ (1NF)

2NF (Second Normal Form)

Rule:- Must be in 1NF & no partial dependency.

- Partial dependency means a non-key column depends only on part of the 1° key.
- Apply when the table has composite 1° keys.

3NF (3rd Normal Form)

Rule:- Must be in 2NF and no transitive dependency.

- Transitive means: $A \to B \to C$, then $A \to C$ (which is not allowed in 3NF for non-key attributes).
- In 3NF, non-key columns must depend only on the 1° key.

BCNF (Boyce-Codd Normal Form)

Stronger version of 3NF.

Rule:- For every functional dependency $A \to B$, A should be a super key.

Used when there are anomalies even in 3NF.

iv) Dependency Preservation

When we split a large table into smaller ones during normalization, we must ensure that all functional dependencies are still preserved in the smaller tables.

vijeta

Most imp. type is :- Functional Dependency (FD)
If $A \rightarrow B$, it means if we know A, we can find B.
Ex - If in a Student table, Roll No → Name, then Knowing the Roll No is enough to uniquely identify the Name. This is used in normalization & is essential for maintaining consistency.

ii) Armstrong's axioms → are a set of rules used to infer all functional dependencies from a given set. These rules helps in understanding how attributes depend on each other.

3 Basic Rules :-

→ Reflexivity - If B is a subset of A, then $A \rightarrow B$
Ex - (RollNo, Name) → Name.

→ Augmentation - If $A \rightarrow B$, then $AC \rightarrow BC$ (add the same attribute on both sides)
Ex :- If RollNo → Name, then RollNo, Marks → Name, Marks.

→ Transitivity - If $A \rightarrow B$ & $B \rightarrow C$, then $A \rightarrow C$
Ex :- If RollNo → Name and Name → Email, then RollNo → Email.

These rules help derive all possible dependencies and are the foundation for normalization.

iii) Normal Forms (Normalization)
Process of organizing data to :- Remove redundancy (duplicate data)
- Ensure data integrity.
- Improve efficiency.

There are several normal forms (NFs), & each one is stricter than the previous.

b) DB2 (by IBM)
- Used in banking and mainframe environments.
- highly secure & reliable.

c) SQL Server (by Microsoft)
- Easy integration with Windows.
- Good for BI (Business Intelligence) tools.
- uses T-SQL (Transact-SQL).

② Relational DB Design:-
When designing a DB, the goal is to organize the data in such a way that;
- It avoids repetition (reduces redundancy),
- Prevents errors or inconsistency,
- Makes querying easy and efficient.

To achieve this, we use several theoretical concepts and design rules.

i) Domain & Data Dependency.
ii) Armstrong's axioms
iii) Normal forms
iv) Dependency preservation
v) Lossless design.

i) Domain & Data Dependency:-

Set of valid values that a column (attribute) can have
Ex - for a column Age, the domain might be all positive integers.

Data Dependency → This refers to a relationship between attributes (columns) in a table, where the value of one attribute depends on another.

vijeta

DML - used to manipulate the data inside tables.
- INSERT: Add new records.
- UPDATE: Modify existing records.
- DELETE: Remove records.
- SELECT: Fetch data.

```
INSERT INTO Student VALUES (1, 'Sanjana', 95);
SELECT * FROM Student WHERE Marks > 80;
```

vi) Open Source and commercial DBMS.

What is a DBMS?

→ A Database Management System (DBMS) is software that manage database and provides a way to create, read, update, and delete data.

Open Source DBMS - free to use & modify.

a) MySQL
→ most popular open-source DBMS.
→ fast and lightweight.
→ used in web applications (PHP + MySQL)
→ Supports SQL standards & basic transactions.

b) PostgreSQL (not metioned in your list but important)
→ Advanced open-source DBMS.
→ Supports complex queries, indexing and objects-relational features.

Commercial DBMS - paid and proprietary.

a) Oracle DB
- Most widely used commercial DBMS.
- Powerful & scalable.
- used in enterprise applications.
- Supports PL/SQL (its own language)

TRC → rows (tuples) variable
DRC → columns (col. value)

(iii) Domain Relational Calculus (Non-Procedural)
- DRC is similar to TRC,
- it uses domain variables (col. values) instead of entire tuples.

Structures :- $\{ \langle X, Y, Z \rangle \mid P(X, Y, Z) \}$

Each variable represents a single column value.

ex:- $\{ \langle n, m \rangle \mid \exists r \in Student (r.Name = n \text{ AND } r.Marks = m \text{ AND } m > 80) \}$

(iv) SQL3 (Structured Query Language)

SQL3 (also called SQL:1999) is an advanced version of SQL that includes support for:
- Object-oriented features.
- User-defined types
- Recursive queries
- Triggers & stored procedures.

Features →
- New data types :- BOOLEAN, ARRAY, BLOB, etc.
- Object-Relational Support: User-defined types & inheritance
- WITH RECURSIVE clause for recursion.
- Enhanced Triggers & functions.

made SQL much more powerful & modern.

v.) DDL and DML constructs (2 main parts of SQL)

DDL (used to define the str. of the DB).
- CREATE → Create table, view, etc.
- ALTER → Modify existing structure.
- DROP → Delete a table or view.
- TRUNCATE → Delete all rows quickly.

- Select (σ): Picks rows that satisfy a condition.
  σ (Marks > 80)(Student)

- Project (π): Picks specific columns
  π Name, Marks (Student)
  Anu 95

- Union ($\cup$): Combines two tables with the same structure.
  Student $\cup$ Teacher.

- Set Difference (−): Returns rows in one table but not the other. Student − Teacher.

- Cartesian Product (x): Combines all rows of one table with all rows of another.

- Rename (ρ): Change the name of table or colums.

- Join ($\bowtie$): Combines related rows from two tables based on a condition.

ii) Tuple Relational calculus (Non-Procedural)
(TRC) - non-procedural query lang. where you describe what you want, not how to get it.

Structure: TRC uses tuples (rows) as variables.
$\{ t \mid P(t) \}$

- t is a tuple variable.
- P(t) is a condition that must be true.

ex:- $\{ t \mid t \in \text{Student AND } t.\text{Marks} > 80 \}$
Give me all tuples from Student where Marks > 80.

vijeta

Types of hashing - Static
hash table size stays the same.

Dynamic
hash table grows / shrinks as needed (eg. Extendible Hashing)

Drawbacks - Not good for range queries (like finding all students with marks > 80).
- can lead to collisions (when 2 keys hash to the same index).

MOD-II

① Relational query languages:-
- to interact with relational DB, meaning DB that store data in tables (relations).
- These languages allow us to:-
  - Retrieve data
  - Insert, update, or delete data
  - Create & modify the str. of DB.

Types of relational query:-
→ Procedural (you tell how to get the result)
→ Non-Procedural (you tell what you want, not how to get it).

foundation of modern querying DB

SQL is based on these principles

i) Relational Algebra (Procedural Lang.)
- procedural query lang.
- we have to specify the sequence of operations to get the desired result

Main Operations in Relational Algebra:
- σ
- ∪
- ρ
- Π
- X
- ⋈

key (value) $\xrightarrow{\text{convert}}$ hash code (fixed-size value) (index) $\xrightarrow{\text{use}}$ hash function.

~~hash table.~~

Values <20 go to left child.
Values b/w 20-50 go to middle-left
B/w 50-80 to middle-right
80 to the right child.

Adv. of B-trees - very fast for range queries
- great for large datasets.
- widely used in DB & file sys.

#. B+ trees (a type of B-tree) are commonly used in real-world DBMS bcz. all values are stored in the leaf nodes only, & leaves are connected like linked list for faster sequential access.

key → ~~hash code~~ → hash funct.

③ Hashing

- technique where we convert a value (key) into a fixed-size value (hash code) using a function known as hash function.
- hash code - used as index - store the record in a table called a hash table.

ex- we want to store a record for a student with Roll_No = 125
we use hash functions like :-

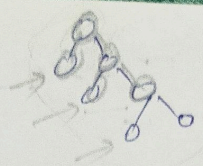
$$\boxed{\text{has}(125) = 5}$$

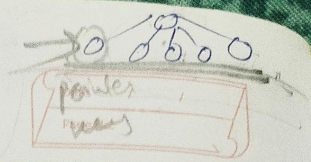
This means the data will be stored at posn. 5 in hash table.

O(1)

Benefits:- very fast access time - almost const. time
- Great for exact match searches (eg. Search by ID).

O(1)





- Without an index, if we want to find "Anjali", the system might have to search the entire table (full table scan)
- But with index on Name, it directly jumps to the memory location where "Anjali" is stored — saving time.

Types of indexes -

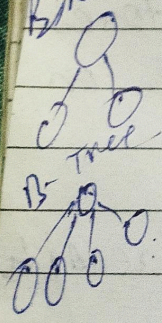
i) 1° index - Built on 1° key (Unique values).
ii) 2° index - Built on non-1° key columns.
iii) Clustered index - Reorders the table data physically.
iv) Non-clustered index - keeps index separate from actual data.

Benefits:	Drawbacks: -
• Faster searches	• Takes extra memory.
• Quick data retrieval	• Needs to be updated when data changes.
• Efficient sorting.	

(2) B-TREES (Balanced Tree)

- Self-balancing tree data str.
- used to store sorted data in a way that allows for efficient searching, insertion, & deletion.
- B-trees commonly used to implement indexing in DB.
- Features:- all values are stored in sorted order.
- every node can have multiple children (not just 2 like in Binary Trees)
- all leaf nodes are at the same level, ensuring balance
- each node has keys & pointers to child nodes.

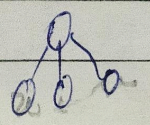
Binary Tree



B-Tree

ex:- Let's say a B-Tree has 3 keys in the root:-

| 20 | 50 | 80 |



b) INSERT: add new data.
INSERT INTO Student VALUES (3, 'Anu', 88);

c) DELETE: to remove data.
DELETE FROM Student WHERE Roll_No = 2;

use: form core of user interact. with the DB.
help maintain, analyze & manage data efficiently.

MOD - III

In DB, we often deal with huge amounts of data. If we go on searching for data without any smart technique, it would be very slow & inefficient.
To quickly access & manage data, DB uses storage strategies like:- 1. Indices (Indexing)
2. B-Trees
3. Hashing.
} to organize & locate data efficiently.

Storage Strategies in DBMS

① INDICES (INDEXING)
- Index is like table of contents in a book -
- it keeps you find info. quickly without scanning every page.
- in DBMS, index is separate data str. that stores pointers (address) to actual records in table based on one/more columns.

ex:- Imagine a student table with 1000s of records.

Roll_NO	Name	Marks
1	Sanjana	90
2	Anjali	85
...	...	...

Ex:- class Student {
int rollNO;
String name;
int marks;
void calculateGrade() {...}
}

v) Integrity constraints → prevents wrong / duplicate data entries
→ keeps the DB reliable & trustworthy.
- Rules applied to the data in a DB to maintain correctness, consistency, and accuracy.
- Types of constraints
a) Key (primary) constraints :- uniquely identifies each record.
ex:- Roll_No in Student table.
b) Foreign key constraints :- maintains a relationship b/w 2 tables.
ex- Student. Course ID refers to Course. Course ID.
c) Not Null constraint :- Ensures a col. can't have NULL (empty values).
d) Unique constraint :- Ensures all values in a column are different.
e) check constraint :- Ensures data satisfies a condition
Ex:- Age > 18

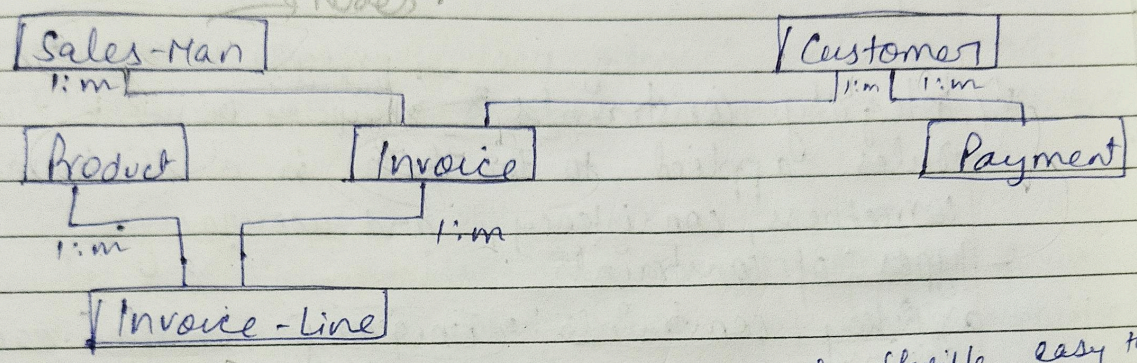
vi) Data Manipulation Operations :-
- operations used to retrieve, insert, delete, and modify data stored in DB.
Main Operations :-

a) SELECT :- fetch data from a table.
SELECT * FROM Student;

SQL, MySQL, MongoDB

- more flexible than the hierarchical model.
- efficient for complex relationships, but difficult to maintain & understand.

- Ex - Network model for Finance Department.



iii) Relational Data Model → simple, flexible, easy to use. → supports powerful query lang. like SQL.
- Organizes data into tables (relations).
- used in modern DB like MySQL, Oracle, Postgre SQL.
- Table (Relation) = collection of rows & columns.
- Row (Tuple) = Single record.
- column (Attribute) = property of data.
- Student table with columns: Roll No, Name, Marks.

(tuple row) →

Roll NO	Name	Marks
1	Sanjana	90
2	Anjali	85

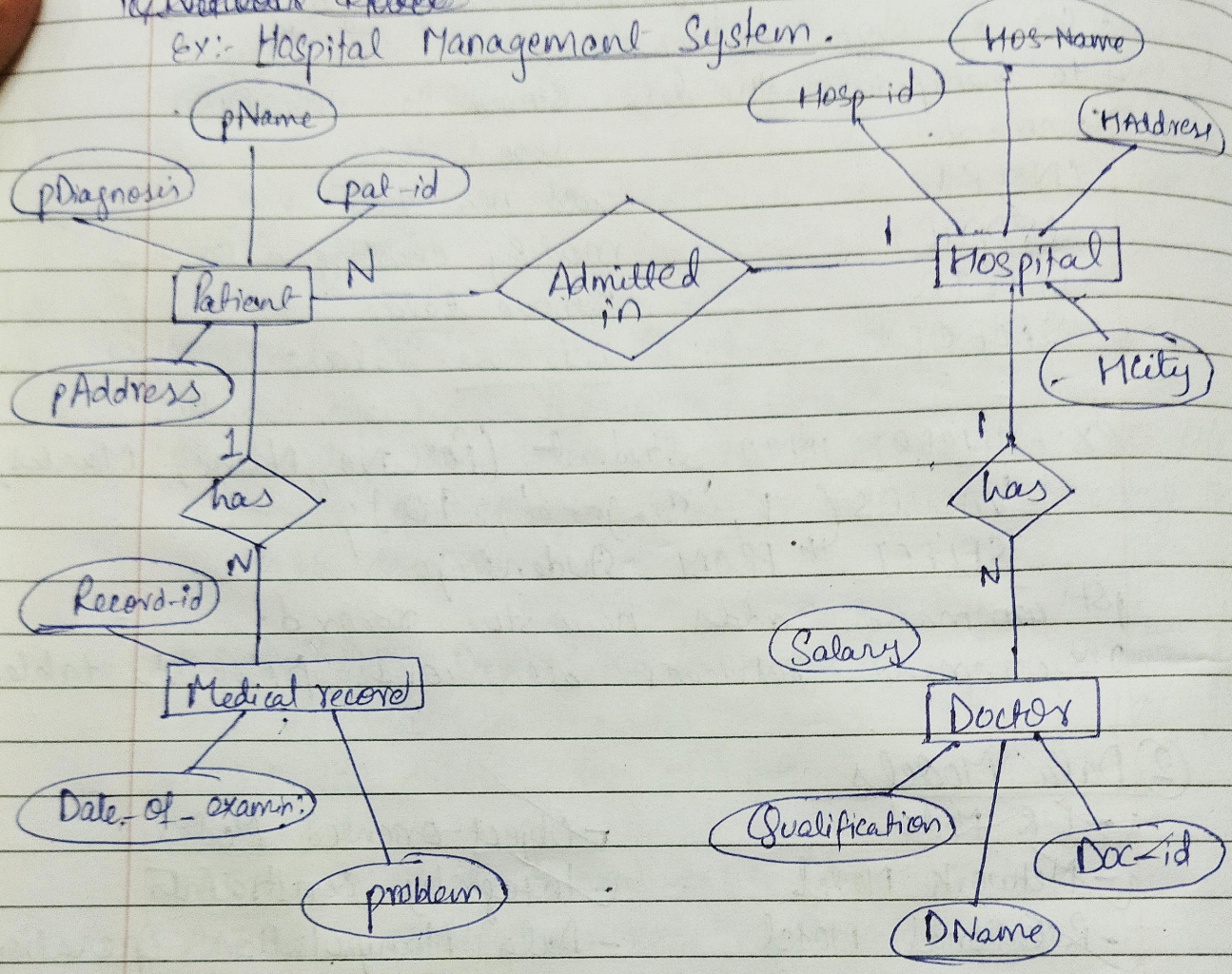
Attribute.

iv) OO Data Model.
- combines features of DB with OO prog. concepts like classes, inheritance, encapsulation.
- Objects → data (attributes), functions (methods).
- Supports complex data types - images, audio, video.
- Ideal for multimedia, engg., CAD applic^n.
- Closer to real-world modeling

records
links

ii) Network Model

Ex:- Hospital Management System.



iii) Network-Data Model

- Organizes data using records & links in the form of graph where each record can have multiple parent & child records.
- data-stored as nodes (records).
- Relationships are called sets, connecting multiple records.
- one-to-many & many-to-many relationships are supported.

Ex- Department can have multiple employees & an employee can work in multiple projects.

vijeta

DML (Data Manipulation Language)
- Set of SQL commands
- to manipulate the data stored in the DB.

Command.	use
INSERT	add new data
UPDATE	modify existing data.
DELETE	remove data
SELECT	retrieve data.

ex:- INSERT INTO Student (Roll_No, Name, Marks)
VALUES (1, 'Sanjana', 90);
SELECT * FROM Student;

1st command adds new stu. record.
2nd command retrieves all records from stu. table.

(2) Data Models

- E-R Model
- Network Model
- Relational Model
- Object-oriented Model
- Integrity constraints
- Data Manipulation Operations.

i) ER Model - high level data model
- design logical str. of DB.

Entity → object / thing in real world. ex- Stu, Teacher, Book.
Attributes → Properties of entity. Ex- Student - Roll. NO, Name, age.
Entity Set → Grp. of similar entities. Ex - Student in college.
Relationship → how entities are related. ex- Student enrolls in a course
ER-Diagram → Visual repres". of entities, attributes, relationship

- ▭ - entities
- ⬭ - Attributes
- ◇ → Relationship
- — → connections.

Ex:- Student is related to a Course via enrollment relation

Student - Enrolls → course.

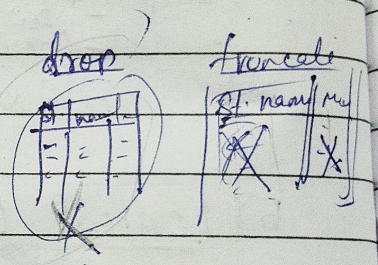
→ easy to understand & design DB str. before actual creation.
→ helps identify data requirements clearly.

Types of data independence:-

a) Logical DI :- ability to change the logical str. (adding new columns / tables) without affecting the view level.
Ex- add a new column email in the Student table but it doesn't affect users who don't use this field.

b) Physical DI :- Ability to change physical storage without changing the logical level.
Ex- changing how data is stored internally (like file format or storage device) without affecting the table design.

iii > DDL (Data Definition Language)
→ set of SQL commands
→ used to define str. of DB.

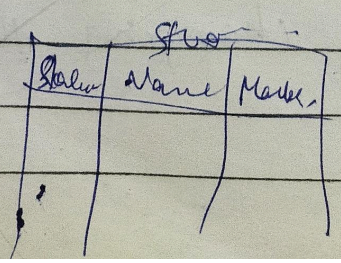


Command

CREATE → create a new table / DB.
ALTER → modify an existing table.
DROP → delete table / database.
TRUNCATE → delete all records from table but keep this str.
RENAME → rename an object.

ex:-
```
CREATE TABLE Student (
    Roll_No INT,
    Name VARCHAR(100),
    Marks INT
);
```

This command defines a new table named Student.



MOD-I (DBMS)

① DB system architecture :-
→ design & str. of DB system.
→ Show how data is stored, accessed, managed in DBMS
→ To manage this efficiently, DB sys. follows concepts like
- data abstraction
- data independence
- Special languages
  - DDL, DML

i) Data abstraction → 1) make syst. easy to use, flexible, secure
→ 2) user - interact - DB - without worrying
- hiding the complex details of how data is stored & maintained in DBMS. Showing only necessary part to user.

Levels of DA → 3 schema arch of DBMS

a) Physical level (lowest level) -
- how data actually stored in DB.
- focus on storage using files, blocks, indexing.
- Ex :- user doesn't need to know whether the data is stored in hard disk sectors or SSDs.

b) Logical Level (Middle level)
- describes what data is stored & what relationship exist among the data.
- Ex → A user knows there is a table named Student with fields Roll_No, Name, Marks.

c) View Level (Highest level)
- (user-level View.)
- shows only part of the data relevant to the user.
- Ex - A college staff might see only the student's name & roll no., not their personal details.

ii) Data Independence → makes system more flexible & easier to maintain
→ improve performance.
- Ability to change the schema (str.) at one level of DB system without changing the schema at the next higher level.