DBMS Concepts Explained (Simple + Examples)

Made for app testing: clear explanations, examples, mini diagrams in text, and Q&A.;

1) What is a Database vs DBMS?

Database = data ka store (like tables/files).
DBMS = software jo database ko manage karta hai (create, update, secure, backup).

Example: College ka data (students, courses, marks) ek database hai. MySQL/PostgreSQL/Oracle jaisa software DBMS hai.

2) Why DBMS? (Problems in File System)

- **Redundancy**: same data multiple files me repeat.
- **Inconsistency**: update ek file me hua, dusri me nahi.
- **Data isolation**: data alag-alag format me scattered.
- **Security**: access control weak.
- **Concurrency**: multiple users ek sath update kare to conflict.

3) Three-Level Architecture (Easy)

DBMS me data ko 3 levels me represent kiya jata hai:

- **External Level**: user view (Student portal me sirf marks show).
- **Conceptual Level**: full logical schema (tables + relationships).
- **Internal Level**: physical storage (indexing, files, blocks).

Benefit: Data Independence (structure change ho to app ko break na kare).

4) Keys in DBMS (with Examples)

Keys ka use uniquely identify karne ke liye hota hai.

Key Type	Meaning (Simple)	Example
Super Key	Any attribute set that uniquely identifies	{sid}, {sid,name}
Candidate Key	Minimal super key	{sid}
Primary Key	Chosen candidate key	sid
Alternate Key	Other candidate keys	email (if unique)
Foreign Key	Refers PK of another table	Enroll.sid → Student.sid

5) ER Model (Entity-Relationship) Explained

Entity = object (Student), **Attribute** = property (name), **Relationship** = link (Student enrolls Course).

```
Text ER example:
Student(sid, name, dept)
Course(cid, title)
Relationship: Enrolls(Student, Course)
```

6) Cardinality & Participation (Most Confusing Part)

- **1:1** One-to-One: Person ↔ Passport (usually).
- **1:N** One-to-Many: Department → Students.
- **M:N** Many-to-Many: Students ↔ Courses (needs new table).
- **Total participation**: entity must participate (every student must have dept).
- **Partial participation**: optional participation (employee may have parking).

7) Relational Algebra (Concept + Use)

Relational Algebra is a **procedural query language** used internally by DBMS for query processing.

- Selection σ: rows filter
- Projection π: columns select
- Join ■: combine tables
- Union ∪ / Difference –: set operations

```
Example:
σ(dept='CSE')(Student)
π(name)(Student)
Student ■ Enroll
```

8) SQL Explained with Real Meaning

SQL is declarative: you tell *what* you want, DBMS decides *how*.

```
CREATE TABLE Student(
  sid INT PRIMARY KEY,
  name VARCHAR(50),
  dept VARCHAR(10)
);

SELECT dept, COUNT(*) AS total_students
FROM Student
GROUP BY dept;
```

9) Functional Dependencies (FD) Explained

FD means: **X** → **Y** (if you know X, you can find Y).

Example: sid → name, dept (because sid is unique).

Anomalies: insertion, deletion, update anomalies happen when table is not well designed.

10) Normalization (Super Simple)

- **Goal**: redundancy reduce + anomalies avoid.
- **1NF**: atomic values
- **2NF**: no partial dependency (composite key case)
- **3NF**: no transitive dependency
- **BCNF**: stricter than 3NF

Thumb rule: Agar non-key attribute kisi aur non-key pe depend kare, 3NF break.

11) Transactions & ACID (with Story)

Transaction = operations ka group jo either **fully** complete hoga ya **rollback**.

- **Atomicity**: all or nothing
- **Consistency**: rules break na ho
- **Isolation**: parallel transactions interfere na kare
- **Durability**: commit ke baad data safe

Example: Money Transfer

```
T1: Transfer 100 from A to B
read(A)
A = A - 100
write(A)
read(B)
B = B + 100
write(B)
commit
```

12) Concurrency Control (Locks Explained)

- **S-lock** (Shared): read allowed, write not allowed.
- **X-lock** (Exclusive): read+write allowed for one transaction.
- **Deadlock**: T1 waits for T2 and T2 waits for T1.
- **2PL**: growing phase + shrinking phase (ensures serializability).

Mini Q&A; (Interview Style)

Q: Primary key vs Unique key?
A: PK is one per table + not null. Unique can be multiple + may allow null (DBMS dependent).

Q: Why M:N needs new table?
A: Because relational tables can’t store repeating groups cleanly.

Q: What is a join?
A: Combine rows from 2 tables using a common attribute.