Database Management Systems

Short notes for B.Tech CSE / IT · BCA · MCA · GATE CS

Uncodemy Handwritten Notes · free for students

1. Introduction

- **Database** = an organised collection of related data. **DBMS** = software to create, store, query and manage it.
- Examples: MySQL, PostgreSQL, Oracle, SQL Server.

➤ DBMS vs file system

File system	DBMS
Data redundancy and inconsistency	Controlled redundancy
No easy querying	Query language (SQL)
Weak security, no concurrency control	Access control, concurrency, recovery

➤ Three-schema architecture

- **External level** (user views) → **Conceptual level** (logical structure) → **Internal level** (physical storage).
- **Logical data independence**: change conceptual schema without changing views.
- **Physical data independence**: change storage without changing conceptual schema (easier to achieve).

➤ Languages

- **DDL**: CREATE, ALTER, DROP, TRUNCATE. **DML**: SELECT, INSERT, UPDATE, DELETE.
- **DCL**: GRANT, REVOKE. **TCL**: COMMIT, ROLLBACK, SAVEPOINT.

2. ER Model

- **Entity** = a real-world object (Student). **Attribute** = its property (roll_no).

- Attribute types: simple / composite, single / multi-valued, stored / derived (age from DOB), key.
- **Weak entity**: has no key of its own, depends on an owner (strong) entity. Shown with a double rectangle.
- **Cardinality**: 1:1, 1:N, M:N. **Participation**: total (double line) or partial.

➤ ER to tables (rules)

ER element	Table
Strong entity	One table, key = its key
Weak entity	Table with owner's key + partial key
1:N relationship	Add key of "1" side as foreign key in "N" side
M:N relationship	New table with keys of both sides
Multi-valued attribute	Separate table

EXAM POINT Minimum tables for M:N with two entities = 3. For 1:N = 2 (relationship merged into the N side).

- **Generalization** (bottom-up), **specialization** (top-down), **aggregation** (relationship treated as an entity).

3. Relational Model and Keys

- **Relation** = table. **Tuple** = row. **Attribute** = column. **Degree** = number of columns. **Cardinality** = number of rows.

Key	Meaning
Super key	Any set of attributes that uniquely identifies a row
Candidate key	Minimal super key
Primary key	Candidate key chosen by the designer (not NULL)
Alternate key	Candidate keys not chosen as primary
Foreign key	Refers to the primary key of another table

FORMULA If a relation has n attributes and one candidate key of 1 attribute, number of super keys = $2^{(n-1)}$.

- **Integrity constraints**: domain, entity integrity (PK not NULL), referential integrity (FK must match a PK or be NULL).

➤ Relational algebra

Operator	Symbol	Job
Select	σ	Choose rows
Project	π	Choose columns (removes duplicates)
Union, Difference, Intersection	∪ − ∩	Need union-compatible relations
Cartesian product	x	All combinations
Rename	ρ	Rename relation / attributes
Natural join	⋈	Join on common attributes
Division	÷	"For all" queries

EXAMPLE Names of students in CSE: π name (σ dept = 'CSE' (Student)).

REMEMBER R x S has |R| x |S| rows and degree(R) + degree(S) columns.

4. SQL

```
SELECT dept, COUNT(*) AS total
FROM   Student
WHERE  marks > 60
GROUP BY dept
HAVING COUNT(*) > 5
ORDER BY total DESC;
```

- Order of execution: FROM → WHERE → GROUP BY → HAVING → SELECT → ORDER BY.
- WHERE filters rows before grouping. HAVING filters groups.
- Aggregates: COUNT, SUM, AVG, MIN, MAX. COUNT(*) counts all rows; COUNT(col) skips NULL.

Joins

Join	Result
INNER JOIN	Only matching rows
LEFT OUTER	All rows of left + matching of right (else NULL)
RIGHT OUTER	All rows of right + matching of left
FULL OUTER	All rows of both
SELF JOIN	Table joined with itself

Nested and other queries

```
-- Second highest salary
SELECT MAX(salary) FROM Emp
WHERE salary < (SELECT MAX(salary) FROM Emp);
```

- `IN`, `EXISTS`, `ANY`, `ALL` work with subqueries. A **correlated subquery** runs once per outer row.
- `DELETE` removes rows (can roll back), `TRUNCATE` removes all rows (DDL), `DROP` removes the table.
- **View** = a virtual table from a query.

> **TIP** NULL compared with anything gives UNKNOWN. Use `IS NULL`, never `= NULL`.

5. Functional Dependency and Normalization

- **Functional dependency (FD)** $X \to Y$: same X value always gives the same Y value.
- **Armstrong's axioms**: reflexivity (if $Y \subseteq X$ then $X \to Y$), augmentation ($X \to Y$ gives $XZ \to YZ$), transitivity ($X \to Y$, $Y \to Z$ gives $X \to Z$).
- **Attribute closure X+** = all attributes X can determine. If X+ has all attributes, X is a super key.

Solved example

- $R(A, B, C, D, E)$, FDs: $A \to B$, $B \to C$, $CD \to E$.
- $A+ = \{A, B, C\}$. $(AD)+ = \{A, D, B, C, E\}$ = all attributes.

- D and A never appear on the right side, so both must be in every key. Candidate key = AD.

➤ Normal forms

Normal form	Rule
INF	All values atomic (no repeating groups)
2NF	INF + no partial dependency (non-prime depends on part of a key)
3NF	2NF + no transitive dependency. For every X → A: X is super key OR A is prime
BCNF	For every X → A: X is a super key
4NF	BCNF + no non-trivial multi-valued dependency

- In the example above: A → B is a partial dependency (A is part of key AD), so R is only in INF.
- BCNF is stronger than 3NF. 3NF decomposition can always keep dependencies; BCNF may not.

➤ Decomposition properties

- **Lossless join**: R1 ∩ R2 → R1 or R1 ∩ R2 → R2 must hold.
- **Dependency preserving**: union of FDs in the parts gives back all original FDs.

> **EXAM POINT** Minimal cover: split right sides, remove extra attributes on the left, remove redundant FDs.

6. Transactions and Concurrency Control

- **Transaction** = a unit of work that reads / writes data.

ACID	Meaning
Atomicity	All or nothing
Consistency	Database stays valid
Isolation	Transactions do not disturb each other
Durability	Committed changes survive failure

- States: active → partially committed → committed, or failed → aborted.

Schedules

- **Serial**: one transaction after another. **Serializable**: result same as some serial schedule.
- **Conflict**: two operations of different transactions on the same item, at least one is a write (RW, WR, WW).
- **Conflict serializable**: build a precedence graph. No cycle = conflict serializable.
- Every conflict serializable schedule is view serializable, not the reverse (blind writes).
- **Recoverable**: a transaction commits only after those it read from commit. **Cascadeless**: read only committed data. **Strict**: no read / write of uncommitted data.

EXAMPLE S: R1(A) W2(A) W1(A). Edges T1 → T2 (R1 A then W2 A) and T2 → T1 (W2 A then W1 A). Cycle, so not conflict serializable.

Locking

- **Shared lock (S)** for read, **exclusive lock (X)** for write. Only S is compatible with S.
- **Two-phase locking (2PL)**: growing phase (only take locks), then shrinking phase (only release). Ensures conflict serializability but can deadlock.
- **Strict 2PL**: hold X locks till commit (cascadeless). **Rigorous 2PL**: hold all locks till commit.
- **Timestamp ordering**: older transaction gets priority. Thomas write rule ignores outdated writes.
- Deadlock prevention: **wait-die** (older waits, younger dies) and **wound-wait** (older wounds younger, younger waits).

7. Recovery

- **Log-based recovery**: write log record before the data change (**WAL**, write-ahead logging).
- **Undo** uncommitted transactions, **redo** committed ones after a crash.
- **Checkpoint** reduces how much of the log is scanned.
- **Deferred update**: write to DB only after commit (no undo needed). **Immediate update**: needs undo and redo.

8. Indexing and File Organization

- **Primary index**: on ordered key field, sparse. **Clustering index**: on ordered non-key field. **Secondary index**: on unordered field, dense.
- **Dense index**: entry for every record. **Sparse index**: entry for some records (one per block).
- **B+ tree**: balanced, all records at leaves, leaves linked for range queries. Used by most DBMS.

FORMULA B+ tree order p for internal node: p x (pointer size) + (p − 1) x (key size) <= block size.

EXAMPLE Block 512 B, key 8 B, pointer 6 B. 6p + 8(p − 1) <= 512, so 14p <= 520, p = 37.

FORMULA Blocking factor = ⌊block size / record size⌋. Blocks for file = ⌈records / blocking factor⌉.

9. Most Asked Questions (practice)

1. Explain three-schema architecture and data independence.
2. Draw an ER diagram for a library / hospital and convert it to tables.
3. Define all keys. Find candidate keys using attribute closure.
4. Write relational algebra and SQL for given queries.
5. Explain INF, 2NF, 3NF and BCNF with examples. Normalise a given relation.
6. Check lossless join and dependency preservation of a decomposition.
7. Explain ACID properties and transaction states.
8. Test conflict serializability using a precedence graph.
9. Explain 2PL, strict 2PL and timestamp protocols.
10. B tree vs B+ tree. Find the order of a B+ tree.

TIP GATE DBMS focuses on candidate keys, normal form checks, serializability, SQL output questions and B+ tree order. Practise closure calculation until it is quick.