DATABASE MANAGEMENT SYSTEM (DBMS)

Complete Revision & Study Notes | Target: STET Exam

1. BASIC DEFINITIONS

Data kya hota hai?
Data = Raw facts / information jo abhi process nahi hui hai.

Database kya hai?
Database = Related data ka organized collection.

DBMS kya hai?
DBMS = **Database Management System**
Ye ek software/system hai jo database ko:

- **Create** karta hai
- **Store** karta hai
- **Retrieve** karta hai
- **Update** karta hai
- **Delete** karta hai
- **Manage** karta hai

Simple language mein: Database mein data pada hai, aur DBMS us data ko manage karta hai.

Examples of DBMS

MySQL | Oracle | PostgreSQL | Microsoft SQL Server | SQLite

2. DBMS KE MAIN FUNCTIONS STET IMPORTANT ⭐⭐⭐

STET ke liye ye yaad rakho:

FUNCTION	MEANING
Data Storage	Data store karna
Data Retrieval	Data nikalna
Data Update	Data change karna
Data Deletion	Data remove karna
Security	Unauthorized access rokna
Backup & Recovery	Data loss hone par recover karna
Concurrency Control	Multiple users ko simultaneously handle karna

3. DBMS VS DATABASE

ENTITY	ROLE / DESCRIPTION
Database	Data ka collection
DBMS	Database ko manage karne wala software

4. DBMS ADVANTAGES — EXAMPLES KE SAATH

ADVANTAGE	SIMPLE MEANING	EXAMPLE
1. Data Redundancy Control ⭐	Same data ko unnecessary baar-baar store hone se reduce karta hai	College mein student ka address 5 alag files mein store karne ke bajay ek central database mein rakha ja sakta hai.
2. Data Security ⭐	Unauthorized person ko data access karne se rokta hai	Student apne marks dekh sakta hai, lekin marks change sirf authorized teacher kar sakta hai.
3. Data Integrity ⭐	Data ko correct aur valid rakhta hai	Student ki age -10 enter karne par constraint invalid data ko reject kar sakta hai.
4. Data Sharing ⭐	Multiple users same database ko access kar sakte hain	College mein teacher marks update karta hai, accountant fees update karta hai aur student apna result dekh sakta hai.
5. Data Independence ⭐	Database mein changes karne par application par minimum effect hota hai	Student table mein ek naya column email add karne par existing application ka main logic same reh sakta hai.
6. Backup & Recovery ⭐	Data loss hone par backup se data recover kar sakte hain	Server crash hone par database ka backup restore karke student records wapas laaye ja sakte hain.
7. Concurrency Control ⭐	Multiple users ko simultaneously database use karne mein manage karta hai	Teacher aur accountant ek hi time par database mein different records update kar rahe hain.
8. Data Consistency	Data ki values ko consistent rakhta hai	Agar student ka mobile number update hua, to system mein uski latest value maintain ki ja sakti hai.
9. Centralized Data Management	Data ko ek central database se manage kiya ja sakta hai	College ke students, teachers, fees aur marks ka data ek database mein manage hota hai.

ADVANTAGE	SIMPLE MEANING	EXAMPLE
10. Easy Data Retrieval	Required data ko query ke through quickly find kar sakte hain	`SELECT * FROM students WHERE city='Patna';` se Patna ke students mil jayenge.
11. Reduced Data Inconsistency	Different places par conflicting data hone ki possibility reduce karta hai	Ek student ka address ek jagah Patna aur doosri jagah Delhi hone ki problem reduce hoti hai.
12. Transaction Management	Database transactions ko properly manage karta hai	Bank transfer mein ₹5,000 sender se deduct aur receiver ko credit hona properly manage hota hai.

5. FILE SYSTEM KYA HOTA HAI?

File System ek traditional method hai jisme data ko files aur folders ke form mein store aur manage kiya jata hai.

Simple words mein:
File System = Data ko separate files mein store karne ka method.

🏫 Example — College

Maan lo college ke paas ye files hain:

Students.txt **Teachers.txt** **Fees.txt** **Marks.txt**

6. DBMS VS FILE SYSTEM — STET IMPORTANT

STET IMPORTANT ⭐⭐⭐

Sabse pehle simple difference:

- **File System** → Data files mein store/manage hota hai
- **DBMS** → Database ko systematically manage karne ka software/system hai

BASIS	FILE SYSTEM	DBMS
1. Data Storage	Data separate files mein store hota hai	Data database/tables mein organized hota hai
2. Redundancy ⭐	Data duplication zyada ho sakta hai	Redundancy ko reduce karta hai
3. Data Consistency ⭐	Maintain karna difficult	Better consistency maintain karta hai
4. Data Security ⭐	Comparatively less secure	Better security provide karta hai
5. Data Sharing	Multiple users ke liye difficult	Multiple users easily access kar sakte hain

BASIS	FILE SYSTEM	DBMS
6. Data Integrity ⭐	Integrity maintain karna difficult	Constraints ke through integrity maintain karta hai
7. Backup & Recovery	Difficult / limited	Better backup & recovery
8. Concurrency ⭐	Multiple users ko handle karna difficult	Concurrency control available
9. Data Relationship	Files ke beech relationship manage karna difficult	Tables ke relationships easily manage hote hain
10. Data Independence ⭐	Data aur application tightly coupled ho sakte hain	Higher data independence
11. Query	Complex searching difficult	SQL ke through easy searching
12. Transaction Management	Limited	Proper transaction management
13. Scalability	Large data ke liye difficult	Large databases ko efficiently handle kar sakta hai
14. Example	.txt, .csv files	MySQL, Oracle, SQL Server

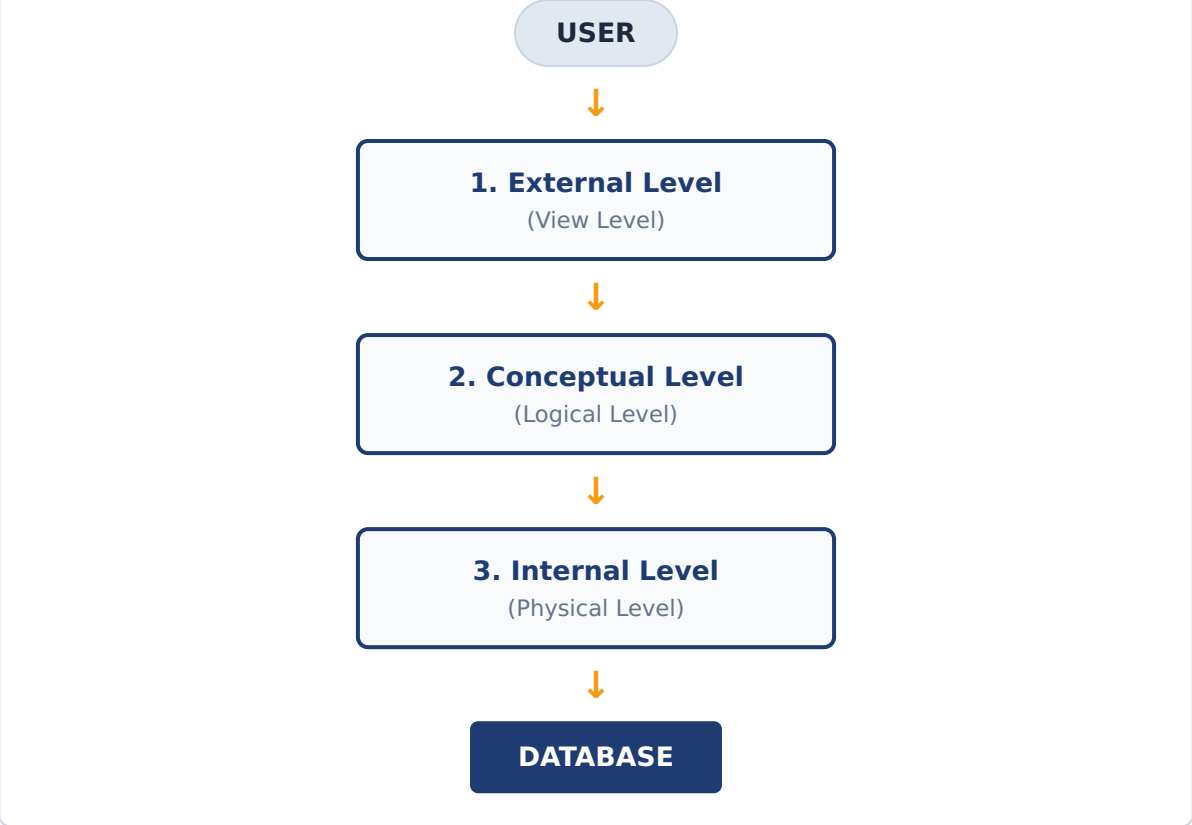
DBMS TOPIC 2 — 3-LEVEL ARCHITECTURE

Complete Study Notes | Target: STET Exam

> Isko **Three-Schema Architecture** bhi kaha jata hai.

BASIC IDEA

Database ko 3 levels mein divide kiya gaya hai:



1️⃣ EXTERNAL LEVEL — VIEW LEVEL

> Ye user ko kya data dikhana hai ye define karta hai.

Example: College Database Mein

- **Student** → apne marks dekhe
- **Teacher** → students ke marks update kare
- **Accountant** → fees information dekhe

Har user ko database ka different view mil sakta hai.

Exam keyword: External = User View

2️⃣ CONCEPTUAL LEVEL — LOGICAL LEVEL ⭐

> Ye batata hai database mein kaunsa data hai aur data ke beech relationships kya hain.

Example Structure:

```
Student
  ├── Student_ID
  ├── Name
  └── Course_ID

Course
  ├── Course_ID
  └── Course_Name
```

Yahan database ka overall logical structure define hota hai.

Exam keyword: Conceptual = Logical Structure / Complete Database

3️⃣ INTERNAL LEVEL — PHYSICAL LEVEL ⭐

> Ye batata hai data actually storage mein kaise stored hai.

Example Storage Details:

- Files
- Disk blocks
- Indexes
- Storage locations

User ko normally ye details nahi dikhayi deti.

3-LEVEL ARCHITECTURE KA MAIN PURPOSE

> **Data Independence**
> Yahi iska sabse important exam point hai.
>
> **Data independence ka matlab:**
> Ek level par change karne se higher level par minimum/no effect ho.

1. Physical Data Independence

Internal/Physical level mein change → Conceptual level par effect nahi.

> **Example:** Storage method change kar diya, lekin tables ka logical structure same hai.

2. Logical Data Independence

Conceptual level mein change → External/View level par minimum effect.

> **Example:** Database mein new field add kiya, lekin users ke existing views unaffected reh sakte hain.

STET KE LIYE YE 5 CHEEZEIN PAKKI KARO ⭐⭐⭐

CONCEPT	KEY RULE / MEANING
External Level	**View**
Conceptual Level	**Logical**
Internal Level	**Physical**
Physical Data Independence	Internal changes without affecting conceptual level
Logical Data Independence	Conceptual changes without affecting external views

DBMS QUICK REVISION NOTES

Relational Model & Database Keys

Complete Concept Guide with Hinglish Explanations & Examples

1. Relational Model Kya Hai?

Relational Model mein data ko **Tables** ke form mein represent kiya jata hai.

Term (DBMS Name)	Standard Meaning
Relation	Table
Tuple / Record	Row
Attribute	Column

Keys का Importance: Key ka main purpose hai record ko uniquely identify karna aur tables ke beech relationship establish karna.

2. Step 1: Super Key

Definition: Super Key aisa attribute ya attributes ka combination hota hai jo table ke har record ko uniquely identify kare.

> 💡 **Simple Hinglish:** Jo kisi bhi row ko uniquely pehchan de = **Super Key**

Example: Student Table

Roll_No	Name	Email	City
101	Abhishek	abhi@gmail.com	Patna
102	Rahul	rahul@gmail.com	Delhi
103	Amit	amit@gmail.com	Patna

- **Roll_No:** Har student ka Roll_No unique hai (101 → Abhishek, 102 → Rahul).
  ⇒ `Roll_No` → **Super Key** ✓
- **Email:** Sabhi students ka Email unique hai.
  ⇒ `Email` → **Super Key** ✓
- **Roll_No + Name:** Ye combination bhi har student ko uniquely identify kar raha hai.
  ⇒ `Roll_No + Name` → **Super Key** ✓

> 🔥 **Important Point:** Super Key mein extra/unnecessary attribute ho sakta hai. Example: `Roll_No` unique hai, par `Roll_No + Name` bhi unique hai. Dono Super Keys hain, lekin `Roll_No + Name` mein *Name* extra attribute hai.

3. Step 2: Candidate Key

Definition: Candidate Key = **Minimal Super Key**. Yaani jo record ko uniquely identify kare aur usme koi extra/unnecessary attribute na ho.

Student Table ke Analysis:

- `Roll_No` alone student ko uniquely identify karta hai, standard zero redundancy.
  ⇒ **Roll_No = Candidate Key** ✓
- `Email` alone bhi har student ke liye unique hai without extra attribute.
  ⇒ **Email = Candidate Key** ✓

Why is `Roll_No + Name` NOT a Candidate Key?

Ye combination unique toh hai (Super Key ✓), par kyunki `Roll_No` alone hi unique identity define kar deta hai, isliye `Name` ki extra zarurat nahi hai. Candidate Key minimal honi chahiye, isliye:
⇒ `Roll_No + Name` → **Candidate Key** ✗

⭐ Super Key vs Candidate Key Comparison

Super Key	Candidate Key
Record ko uniquely identify kare	Record ko uniquely identify kare
Extra attributes ho sakte hain	Extra attributes nahi hone chahiye
Minimal hona zaroori nahi	Minimal hona zaroori hai
Example: Roll_No + Name	*Example:* Roll_No

🧠 **Shortcut Memory Trick:**
- **Super Key:** Unique
- **Candidate Key:** Unique + Minimal

4. Step 3: Primary Key

Definition: Primary Key = **Selected Candidate Key**. Table ke multiple candidate keys mein se jise main key ke roop mein select kiya jata hai, woh Primary Key hoti hai.

Primary Key ki Main Properties ⭐

Property	Meaning
Unique	Do records ki same Primary Key nahi ho sakti.
NOT NULL	Primary Key kabhi bhi empty ya NULL nahi ho sakti.
One Primary Key	Ek table mein normally ek hi Primary Key hoti hai.
Identifies Record	Har tuple ko uniquely identify karti hai.

> 🧠 **Candidate Key vs Primary Key Relation:**
> • **Candidate Key:** Jo Primary Key ban sakti hai (Eligible keys).
> • **Primary Key:** Jo Candidate Keys mein se select ho gayi.

5. Step 4: Alternate Key

Definition: Candidate Keys mein se jo key Primary Key ke liye **select nahi hui**, usse Alternate Key kehte hain.

Example: Student Table mein Candidate Keys hain: `Roll_No` aur `Email`.
Agar humne **Primary Key = Roll_No** select kar liya, toh bachi hui candidate key (`Email`) ko:
⇒ **Email = Alternate Key** ✓

6. Step 5: Foreign Key

Definition: Foreign Key ek table ka aisa column hota hai jo kisi doosri table ki Primary Key ko refer karta hai.

> 🔗 **Simple Hinglish:** Foreign Key → Do tables ke beech relationship establish karti hai.

Example: Student & Marks Tables

1. Student Table:

Student_ID (Primary Key)	Name
101	Abhishek
102	Rahul
103	Amit

2. Marks Table:

Mark_ID (Primary Key)	Student_ID (Foreign Key)	Marks
1	101	80
2	102	75
3	103	90

- Foreign Key ka unique hona zaroori nahi hota.
- Foreign Key NULL values accept kar sakti hai (unless constrained).

7. 🔑 One-Liner Memory Trick Summary

Key Type	One-Liner Memory Trick
Super Key	Uniquely identify kare = Super Key
Candidate Key	Minimal Super Key = Candidate Key
Primary Key	Selected Candidate Key = Primary Key
Alternate Key	Candidate Key jo select nahi hui = Alternate Key
Foreign Key	Do tables ko connect kare = Foreign Key

DBMS QUICK REVISION NOTES

ER Model & Database Architecture

Complete Conceptual Guide with ER Diagrams, Symbols & Terms

1. ER Model (Entity-Relationship Model) Kya Hai?

ER Model database ko design karne ka **Conceptual Model** hai.

> 💡 **Formula:** ER Model = Entities + Attributes + Relationships

Example: `Student ── Enrolls ── Course`
Yahan database banane se pehle ye decide kiya jata hai ki kaun-kaun si entities hain aur unke beech kya relationship hai.

A. Entity

Real-world object jiske baare mein data store karna hai.

Example: College database mein **Student**, **Teacher**, aur **Course** Entities hain.
🧠 **Trick:** Entity = Thing / Real-World Object

B. Attribute

Entity ki property / characteristic.

Example: **Student** Entity ke Attributes: `Student_ID`, `Name`, `Age`, `Email`.

C. Relationship ⭐⭐⭐

Do ya more entities ke beech connection/association.

Example: `Student ── Enrolls ── Course`
• **Student** → Entity
• **Course** → Entity
• **Enrolls** → Relationship (Student enrolls in Course)

2. Cardinality Ratio ⭐⭐⭐

Cardinality batati hai ki ek entity ka kitne records ke saath relationship ho sakta hai.

> 💡 **Simple Language:** Cardinality = "Kitne records interconnected hain?"

Cardinality	Meaning	Example
1:1	One → One	Person → Passport
1:M	One → Many	Department → Employees
M:1	Many → One	Employees → Department
M:N	Many → Many	Student → Course

🧠 **STET Memory Trick:**

- **1:1** → एक से एक
- **1:M** → एक से बहुत
- **M:1** → बहुत से एक
- **M:N** → बहुत से बहुत

3. ER Diagram Symbols ⭐⭐⭐

ER Diagram banane ke liye use hone waale primary & advanced symbols:

Symbol	Represents	Example / Description
Rectangle ▭	**Entity**	Student, Course
Oval ⭕	**Attribute**	Name, Age
Diamond ◇	**Relationship**	Enrolls, Works_In
Double Oval ⭕⭕	**Multivalued Attribute**	Phone No. (Multiple numbers per student)
Double Rectangle ▭▭	**Weak Entity**	Entity jiska apna primary key nahi hota (e.g., Dependent)
Underlined Attribute Attr	**Key Attribute**	Primary Key (e.g., Student_ID)
Dashed Oval ◦⭕	**Derived Attribute**	Age (calculated from DOB)

4. Schema vs Instance

Schema: Database ka fixed structure/design (tables, attributes, keys, aur relationships kaise arranged hain).

Instance: Kisi particular point of time par database mein actual stored data.

> 🧠 **One-Liner Trick:**
> • **Schema:** Database bana kaisa hai? (Structure)
> • **Instance:** Abhi database mein data kya hai? (Actual Data)

3-Level Architecture Connection ⭐⭐⭐

Schema Level	Meaning
External Schema	Particular user's view (User View)
Conceptual Schema	Complete logical structure of whole database
Internal Schema	Physical storage structure on disk

5. Relational Terms Overview

DBMS Term	Meaning	Example
Relation	Table	Student Table
Tuple	Row / Record	`(101, Abhishek, 22, B.Tech)`
Attribute	Column / Field	`Student_ID`, `Name`, `Age`
Domain ⭐	Set of allowed/permitted values for an attribute	If Age range is 17 to 30 → Domain = {17, 18, ..., 30}

6. 🔑 Quick Revision Summary

Concept	One-Liner Memory Trick
Entity	Thing / Real-world Object (Rectangle)
Attribute	Property / Characteristic (Oval)
Relationship	Connection between entities (Diamond)
Cardinality	Kitne records connect ho sakte hain? (1:1, 1:M, M:1, M:N)
Schema	Database ka fixed Structure
Instance	Actual data at a specific point of time
Domain	Allowed values ka set for a column

DBMS QUICK REVISION NOTES — PART 2

Integrity Constraints & Normalization Guide

1NF, 2NF, 3NF, BCNF, Partial & Transitive Dependencies & Anomalies

1. DBMS Integrity Constraints ⭐⭐⭐

Constraint Type	Main Rule & Meaning	Example & Trick
Domain Constraint	Attribute mein sirf valid / allowed values hone chahiye.	Age → 18 to 60. (If Age = 150 ❌ **Invalid**) 🧠 *Trick: Valid Values*
Entity Integrity Constraint	Primary Key ki value kabhi bhi **NULL** nahi ho sakti.	`Student_ID` Primary Key hai → `NULL` value not allowed. 🧠 *Trick: Primary Key ≠ NULL*
Referential Integrity Constraint	Foreign Key mein aisi value nahi ho sakti jo Parent Table ke Primary Key mein exist na karti ho.	Child table foreign key value must match parent primary key. 🧠 *Trick: No Orphan Records*
Key Constraint	Key ka primary purpose records ko **uniquely identify** karna hai.	Primary key uniquely identifies each row. 🧠 *Trick: Unique Identification*

2. Normalization & Database Anomalies

Normalization: Database ko properly organize karna taaki **Data Redundancy** (duplicate data) aur **Anomalies** ko eliminate/reduce kiya ja sake.

> 💡 **One-Liner:** Normalization = Redundancy Kam + Database Organized

Database Anomalies (Problems during Data Operations)

Anomaly Type	Description
Insertion Anomaly	Naya data insert/add karne mein constraint ya problem hona.
Update Anomaly	Same data multiple places par store hone par ek jagah change miss ho jana.
Deletion Anomaly	Data delete karne par doosra zaruri data unexpectedly loss ho jana.

3. First Normal Form (1NF) ⭐⭐⭐

Rule: Har cell mein sirf **ONE / ATOMIC** value honi chahiye. Ek cell ke andar multiple values ya repeating groups nahi ho sakte.

❌ Un-normalized Table (Not in 1NF)

Student_ID	Name	Phone
101	Abhishek	**9876, 8765** (Multiple Values)
102	Rahul	9999

✅ Converted Table (In 1NF)

Student_ID	Name	Phone
101	Abhishek	9876 ✅ **Single Value**
101	Abhishek	8765 ✅ **Single Value**
102	Rahul	9999 ✅ **Single Value**

🧠 **1NF Memory Trick:** "One Cell = One Value" (Atomic Values Only)

4. Partial Dependency & 2NF (Second Normal Form) ⭐⭐⭐

Partial Dependency Definition: Jab koi non-key attribute Composite Primary Key ke *sirf ek part* par depend kare.

2NF Rule: `1NF + No Partial Dependency` (Har non-key attribute poori primary key par depend karna chahiye).

Example Problem: Composite Key = `Student_ID + Course_ID`

- `Student_ID → Student_Name` ❌ **Partial Dependency (Aadhi Key)**
- `Course_ID → Course_Name` ❌ **Partial Dependency (Aadhi Key)**
- `Student_ID + Course_ID → Marks` ✅ **Full Dependency**

Solution: Decompose into 3 Tables

1. **Student Table:** `Student_ID (PK), Student_Name`
2. **Course Table:** `Course_ID (PK), Course_Name`
3. **Enrollment Table:** `Student_ID, Course_ID, Marks` ✅ **2NF Satisfied**

🧠 **2NF Trick:** 2NF = Partial Dependency Hatao (Aadhi Key Par Dependency ❌)

5. Transitive Dependency & 3NF (Third Normal Form) ⭐⭐⭐

Transitive Dependency Definition: Jab ek non-key attribute kisi doosre non-key attribute par depend kare (indirectly Primary Key par depend ho).

3NF Rule: `2NF + No Transitive Dependency` (Non-key → Non-key dependency nahi honi chahiye).

> **Transitive Dependency Chain:**
> `Student_ID (PK) → Dept_ID (Non-key) → Dept_Name (Non-key)`
> Here `Dept_Name` depends on `Dept_ID` through `Student_ID` ❌**Transitive Dependency**

Solution: Decompose into 2 Tables

1. **Student Table:** `Student_ID (PK), Student_Name, Dept_ID`
2. **Department Table:** `Dept_ID (PK), Dept_Name` ✅**3NF Satisfied**

> 🧠 **3NF Trick:** 3NF = Transitive Dependency Hatao (Beech Ka Non-Key ❌)

6. BCNF (Boyce-Codd Normal Form) ⭐⭐⭐

Definition: BCNF is a stricter version of 3NF. Iska main rule hai: **"Every Determinant must be a Candidate Key"**.

> 💡 If `A → B`, then **A** MUST be a Candidate Key.

Feature	3NF	BCNF
Strictness	Less Strict	More Strict (Strict 3NF)
Rule	No Transitive Dependency	Every Determinant = Candidate Key
Exception Allowed?	Allowed if RHS is a prime attribute	No Exception Allowed

7. 🔑 STET Fast Revision One-Liners

Topic / Normal Form	STET One-Liner Memory Trick
Normalization	Data redundancy aur anomalies reduce/eliminate karna
1NF	Every cell must contain a **single atomic value**
Partial Dependency	Non-key attribute depends on part of composite key
2NF	1NF + **No Partial Dependency** ("Aadhi Key ❌")
Transitive Dependency	Non-key attribute depends on another non-key attribute
3NF	2NF + **No Transitive Dependency** ("Beech Ka Non-key ❌")
BCNF	Every **Determinant = Candidate Key**
Anomalies	Insertion, Update, Deletion Operations mein Problems

DBMS Constraints & Normalization

STET Exam Revision Notes & Memory Tricks

Database Management | STET Preparation | Quick Memory Guide

1. DBMS Constraints

Domain Constraint

Kisi attribute mein sirf **valid/allowed values** honi chahiye.

- **Example:** Age → 18 to 60
- Agar Age = 150 ✗, to valid domain ke bahar hai.

> **Trick:** Domain Constraint = Valid values

Entity Integrity Constraint★★★

Primary Key NULL nahi ho sakti.

Student_ID	Name
101	Abhishek
NULL ✗	Rahul

* Student_ID Primary Key hai, isliye NULL allowed nahi.

Referential Integrity Constraint

Foreign Key mein aisi value nahi honi chahiye jo parent table mein exist hi nahi karti.

Key Constraint

Key ka purpose records ko **uniquely identify** karna hai.

2. Normalization & Anomalies

Normalization kya hai?

Database ko properly organize karna taaki **data redundancy (duplicate data)** aur **anomalies** ko reduce kiya ja sake.

Anomalies: Database mein data ko add, change ya delete karte waqt hone wali problems.

Redundancy ka Example:

Student_ID	Name	Course	Teacher
101	Abhishek	DBMS	Sharma

102	Rahul	DBMS	Sharma
103	Amit	DBMS	Sharma

Yahan DBMS aur Sharma baar-baar repeat ho rahe hain. Normalization mein database ko multiple related tables mein divide karke redundancy reduce karte hain.

> **One-line memory:** Normalization = Redundancy kam + Database organized

3. Normal Forms (1NF, 2NF, 3NF, BCNF)

First Normal Form — 1NF★★★

1NF ka main rule: Har cell mein sirf **ONE / ATOMIC value** honi chahiye (No repeating groups).

✗ Example: Not in 1NF

Student_ID	Name	Phone
101	Abhishek	**9876, 8765 ✗**
102	Rahul	9999

Problem: Phone ke ek cell mein 2 values hain (9876, 8765). Ye atomic nahi hai.

✓ 1NF mein convert karne ke baad:

Student_ID	Name	Phone
101	Abhishek	**9876**
101	Abhishek	**8765**
102	Rahul	**9999**

> **1NF Memory Trick:** "One Cell = One Value" ✓ | Every cell contains a single atomic value + no repeating groups.

Second Normal Form — 2NF

2NF = 1NF + No Partial Dependency

Non-key column ko Primary Key ke poore combination par depend karna chahiye.

Partial Dependency kya hai?
Jab koi non-key column, composite Primary Key ke sirf *ek part* par depend kare, to use Partial Dependency kehte hain.

Example: Primary Key = `Student_ID + Course_ID`

- `Student_ID → Student_Name` **(Partial Dependency ✗)**
- `Course_ID → Course_Name` **(Partial Dependency ✗)**
- `Student_ID + Course_ID → Marks` **(Full Dependency ✓)**

2NF Conversion (Table Division):

- **Student Table:** (Student_ID, Student_Name)

- **Course Table:** (Course_ID, Course_Name)
- **Enrollment Table:** (Student_ID, Course_ID, Marks)

> **2NF Memory Trick:** 2NF = 1NF + No Partial Dependency (Aadhi key par dependency ❌)

Third Normal Form — 3NF

3NF = 2NF + No Transitive Dependency

Non-key column kisi doosre non-key column par depend nahi hona chahiye (Direct Primary Key → Non-key dependency honi chahiye).

Transitive Dependency kya hai?
Jab ek non-key attribute kisi doosre non-key attribute par depend kare, aur indirectly Primary Key par depend ho (Chain dependency).
`Student_ID → Department_ID → Department_Name` ✗

3NF Solution (Table Division):

- **Student Table:** (Student_ID, Student_Name, Department_ID)
- **Department Table:** (Department_ID, Department_Name)

> **3NF Memory Trick:** 3NF = No Transitive Dependency (Beech ka non-key ❌)

Boyce-Codd Normal Form — BCNF★★★

Definition: BCNF mein **har determinant ek Candidate Key होना चाहिए**.

Agar `A → B`, to A (Determinant) ka Candidate Key hona zaroori hai.

3NF vs BCNF Comparison:

3NF	BCNF
Transitive dependency remove karta hai	Determinant must be Candidate Key
Less strict	More strict (Stricter than 3NF)

BCNF Solution Example:

Original Table: (Student, Course, Teacher) jahan `Teacher → Course` lekin Teacher candidate key nahi tha.
Divide into **Teacher_Course Table** (Teacher, Course) jahan Teacher Candidate Key hai, aur **Student Table**.

> **One-Line Rule:** BCNF = "Every determinant must be a Candidate Key."

4. STET Fast Revision — One-Liners

Topic	STET One-Liner / Rule
Normalization	Database redundancy aur anomalies ko reduce karne ke liye data organize karna
1NF	Every cell must contain a single atomic value (One cell = One value)
2NF	1NF + No Partial Dependency

Topic	STET One-Liner / Rule
Partial Dependency	Non-key attribute depends on part of a composite key
3NF	2NF + No Transitive Dependency
Transitive Dependency	Non-key attribute depends on another non-key attribute
BCNF	Every determinant must be a Candidate Key
Insertion Anomaly	Data insert/add karne mein problem
Update Anomaly	Same data ko multiple places par update karne ki problem
Deletion Anomaly	Data delete karne par unwanted data loss

🧠 **Main Memory Formula:**
1NF → Atomic | 2NF → Partial hatao | 3NF → Transitive hatao | BCNF → Determinant = Candidate Key

DBMS: Concurrency Control & 2PL

Complete Quick Revision Notes (Hinglish + STET Exam Focus)

Database Management Systems | STET Prep | Transaction Management

1️⃣ Concurrency Control

Definition: Concurrency Control is the process of managing multiple transactions running at the same time so that the database remains correct and consistent.

Simple Hinglish: Jab ek time par multiple transactions chal rahi ho, unko properly manage karna = *Concurrency Control*.

🏦 Practical Banking Example

Maan lo Rahul ke bank account me balance hai: **₹10,000**

Ab ek hi time par 2 transactions chalti hain:

- **Transaction A:** Rahul se ₹3,000 withdraw.
- **Transaction B:** Rahul se ₹2,000 withdraw.

❌ Without Concurrency Control (Problem)

```
A reads balance → ₹10,000
B reads balance → ₹10,000
A updates balance → ₹7,000
B updates balance → ₹8,000
```

Result: Final balance galat ho jayega (₹7,000 ya ₹8,000).
Actual Expected: ₹10,000 - ₹3,000 - ₹2,000 = **₹5,000**

✅ With Concurrency Control (Solution)

```
Transaction A → Updates ₹10,000 - ₹3,000
A Completes → Balance = ₹7,000
Transaction B → Updates ₹7,000 - ₹2,000
B Completes → Balance = ₹5,000
```

Result: DBMS ensure karta hai ki transactions coordinate karke safely execute hon. Correct Balance = **₹5,000**

> 🧠 **Ekdum Simple Definition:**
> Concurrency Control = Multiple transactions ko safely manage karna.

🔒 Database Locking & Lock Types

Locking Concept: Jab ek transaction data use kar rahi ho, tab doosri conflicting transaction ko temporarily rokna (waiting state me dalna).

Transaction A → Working 🔒 | Transaction B → Waiting ⏳
A Complete → COMMIT → Unlock 🔓 → Transaction B → Working

Important Lock Types

Shared Lock (S-Lock) ⭐⭐⭐

Purpose: Read karne ke liye lock.

Multiple transactions ek saath READ lock le sakti hain kyunki sirf data dekhna hota hai, modify nahi karna hota.

```
A → READ (S-Lock) ✅
B → READ (S-Lock) ✅
```

Exclusive Lock (X-Lock) ⭐⭐⭐

Purpose: Write / Update karne ke liye lock.

Jab ek transaction X-Lock leti hai, toh kisi doosri transaction ko Read ya Write kisi ki permission nahi hoti.

```
A → WRITE (X-Lock) 🔒
B → READ / WRITE ❌ (Wait ⏳)
```

🔥 Shared vs Exclusive Lock Comparison

Feature / Property	Shared Lock (S-Lock)	Exclusive Lock (X-Lock)
Purpose	Read Data 👀	Write / Update Data ✍️
Multiple Transactions?	Multiple Reads Allowed	Only One Transaction
Other Transaction Read	✅ **Allowed**	❌ **Not Allowed**
Other Transaction Write	❌ **Not Allowed**	❌ **Not Allowed**

🧠 **STET EXAM TRICK & KEY LINES**

- **S = Shared = See / Read** 👀 (Multiple transactions can read, but cannot modify).
- **X = Exclusive = Write / Change** ✍️ (One transaction gets exclusive access to modify the data).

🌱 Two-Phase Locking (2PL) Protocol ⭐⭐⭐⭐⭐

Definition: Two-Phase Locking (2PL) is a concurrency control protocol where every transaction acquires and releases locks in two distinct phases: **Growing Phase** and **Shrinking Phase**.

Simple Hinglish: Transaction pehle saare required locks leti hai (Grow karti hai), aur uske baad locks chhodti hai (Shrink karti hai).

Phase 1 — Growing Phase 🌱

Is phase mein transaction locks acquire karti hai.
✅ **Lock lena allowed** | ❌ **Lock release karna NOT allowed**

`Transaction Start → S-Lock 🔒 → X-Lock 🔒 → X-Lock 🔒`

Phase 2 — Shrinking Phase 📉

Is phase mein transaction locks release karti hai.

❌ **New lock lena NOT allowed** | ✅ **Lock release karna allowed**

`Unlock 🔓 → Unlock 🔓 → Transaction End`

> 💡 **IMPORTANT 2PL TAKEAWAY FOR STET**
>
> - 2PL ensures **Serializability** in database transactions.
> - Once a transaction releases its first lock, it enters the Shrinking Phase and can **never acquire any new lock**.

DBMS Quick Revision Notes

DDL, DML & SQL JOINs — Quick Reference & Exam Guide (STET / Competitive Exams)

DDL Data Definition Language

Definition: DDL is used to define or change the structure of a database.

> 💡 **Simple Hinglish:** DDL = Database ya Table ka structure banana ya change karna.

Command	Meaning
CREATE	Table / Database banana
ALTER	Structure change (modify) karna
DROP	Table / Database complete delete karna
TRUNCATE	Table ka saara data remove karna (Structure rehta hai)

Example:

```
CREATE TABLE Student (
    id INT,
    name VARCHAR(50)
);
```

➔ Yahan hum table ka structure bana rahe hain → **DDL** ✅

DML Data Manipulation Language

Definition: DML is used to insert, update, and delete data in a table.

> 💡 **Simple Hinglish:** DML = Table ke andar ke data ko change/manipulate karna.

Command	Meaning
INSERT	Naya data add karna
UPDATE	Existing data change/modify karna
DELETE	Specific ya saara data remove karna

JOINs SQL JOIN Concepts

> 💡 **JOIN ka Simple Meaning:** Do ya do se adhik tables ke related data ko common column ke basis par combine karna.

Source Tables (Base Data):

👨‍🎓 Student Table (LEFT)

ID	Name	Dept
1	Rahul	D1
2	Amit	D2
3	Ravi	D3

🏢 Department Table (RIGHT)

Dept	Department
D1	Computer
D2	Mechanical
D4	Civil

1. INNER JOIN — "Jo Match kare, wahi do" ⭐

Student: D1 ✅, D2 ✅, D3 ❌ | Department: D4 ❌

Name	Department
Rahul	Computer
Amit	Mechanical

> 🧠 **याद रखो: INNER = Sirf MATCHING data**

2. LEFT JOIN — "Left ko pura rakho" ⭐

Left (Student) ke sabhi records rahenge, chahe department match ho ya nahi.

Name	Department
Rahul	Computer
Amit	Mechanical
Ravi	*NULL*

> 🧠 **याद रखो: LEFT JOIN = Left table ka SAB data + matching right data**

3. RIGHT JOIN — "Right ko pura rakho" ⭐

Right (Department) ke sabhi records rahenge. Civil (D4) Student me nahi hai isliye Name = NULL.

Name	Department
Rahul	Computer
Amit	Mechanical
NULL	Civil

🧠 याद रखो: **RIGHT JOIN = Right table ka SAB data + matching left data**

4. FULL OUTER JOIN — "Dono ko pura rakho" ⭐

Student ke sabhi records + Department ke sabhi records.

Name	Department
Rahul	Computer
Amit	Mechanical
Ravi	*NULL*
NULL	Civil

🧠 याद रखो: **FULL JOIN = Left ka sab + Right ka sab**

STET SPECIAL STET Exam ke liye Quick 4 Lines Summary 🚀

INNER JOIN = Matching only

LEFT JOIN = All Left

RIGHT JOIN = All Right

FULL JOIN = All Both