Lecture Delivery Blueprint (LDB) – Database Management System

Subject Name :DBMS(BCSC-1003)
Syllabus : 6 Modules
Delivery Breakdown: Modules and No of Lectures

Topic / Module	No of lectures
Understanding the class and Context setting	2
Module 1: **Introduction:** An Overview of Database Management System, Database System Vs File System, Database System Concept and Architecture, Data Model Schema and Instances, Data Independence, Database Language and Interfaces (DDL, DML, DCL), Database Development Life Cycle (DDLC) with Case Studies. **:** ER Model Concepts, Notation for ER Diagram, Mapping Constraints, Keys, Specialization, Generalization, Aggregation, Reduction of an ER Diagram to Tables, Extended ER Model.	8
Module 2: **Relational Data Model and Language;** Relational Data Model Concepts, Integrity Constraints, Entity Integrity, Referential Integrity, Keys Constraints, Primary Key, Foreign Key, Candidate Key, Super Key, Domain Constraints, Relational Algebra.	6
Module 3: **Database Design & Normalization:** Functional Dependencies, Canonical Cover, Normal Forms, First, Second, Third Normal Forms, BCNF, Lossless Join and Dependency Preserving Decomposition, MVD and 4th Normal Form, JD and 5th Normal Form, Inclusion Dependence.	6
Module 4: **Transaction Processing Concept:** Transaction System, Testing of Serializability, Serializability of Schedules, Conflict & View Serializable Schedule, Recoverability, Recovery from Transaction Failures, Log Based Recovery, Deadlock Handling.	6
Module 5: **Concurrency Control Techniques:** Concurrency Control, Locking Techniques for Concurrency Control, 2PL, Time Stamping Protocols for Concurrency Control, Validation Based Protocol.	5
Module 6: **File Organization & Distributed Database:** Indexing, Structure of Index files and Types, Dense and Sparse Indexing. **Distributed Database:** Introduction of Distributed Database, Data Fragmentation and Replication.	5

Course Outcome	**Outcome:** After the completion of the course, the student will: • CO1: Understand the concept of database management systems and Relational database. • CO2: Identify the various data model used in database design. • CO3: Design conceptual models of a database using ER modeling for real life applications and construct queries in Relational Algebra. • CO4: Create and populate a RDBMS for a real life application, with constraints and keys using SQL. • CO5: Select the information from a database by formulating complex queries in SQL. • CO6: Analyze the existing design of a database schema and apply concepts of normalization to design an optimal database. • CO7: Discuss indexing mechanisms for efficient retrieval of information from a database. • CO8: Discuss recovery system and be familiar with introduction to web database, distributed databases.
Learning / study Materials	**Text Book:** Elmasri and Navathe, “Fundamentals of Database Systems”, 6th Edition, Addison Wesley,2010. **References Book:** Korth, Silbertz and Sudarshan, “Database Concepts”, 5th Edition, TMH,1998. **Website:** **https://www2.cs.sfu.ca/CourseCentral/354/zaiane/material/notes/contents.html**

Placement Relevance	**Company Name**	**DBMS Topics Frequently Asked**	**Relevance Level**
	TCS	ACID, Transactions, Normalization, ER Diagrams	★★★★☆
	Infosys	Keys, Joins, Normalization, SQL Queries	★★★★☆
	Accenture	ER Model, Functional Dependencies, SQL Basics	★★★☆☆
	Wipro	Transactions, Locking, Schedules	★★★★☆
	Capgemini	SQL, Normal Forms, Recovery Techniques	★★★☆☆

	Cognizant (CTS)	ACID, Normalization, Relational Algebra	★★★★☆
	IBM	Indexing, Query Optimization, Isolation Levels	★★★★★
	Oracle	SQL Advanced, Views, Triggers, Transactions	★★★★★
	Amazon (AWS)	Scalability, NoSQL vs RDBMS, Indexing	★★★★★
	Google	Distributed DBMS, CAP Theorem, Transactions	★★★★★
	Microsoft	Stored Procedures, Triggers, Query Optimization	★★★★★
	Flipkart	Transaction Control, Query Optimization	★★★★☆
	Paytm	SQL Queries, Indexing, Locking	★★★★☆
	HCL	Normalization, Transactions, Relational Algebra	★★★☆☆
	L&T Infotech	Keys, ER Models, SQL Joins	★★★☆☆

Topic Weightage Summary

DBMS Topic	Relevance Across Companies
SQL Queries (JOIN, GROUP BY, Subqueries)	Very High
Normalization & FDs	High
Transactions & Concurrency	High
ER Diagrams & Relational Models	Medium
Indexing & Optimization	Medium-High (Product Companies)
Views, Triggers, Stored Procedures	Medium

Project / Research Relevance	1. **Real-World Data Handling**: Enables efficient design and management of large-scale, real-time data for industries like banking, healthcare, and e-commerce. 2. **Data Integrity & Security**: Focuses on ensuring accurate, consistent, and secure data—crucial for mission-critical applications. 3. **Optimization & Performance**: Research leads to query optimization, indexing, and transaction management for faster system performance. 4. **Emerging Technologies**: Forms the backbone for modern tech such as Big Data, Cloud Computing, IoT, and AI-driven analytics. 5. **Data Modeling**: Enhances logical and physical design through ER modeling, normalization, and schema design. 6. **Transaction & Concurrency Control**: Ensures reliable multi-user access and ACID compliance in real-time systems. 7. **Recovery & Fault Tolerance**: Promotes robust design using log-based recovery, checkpoints, and backups for failure resilience. 8. **Scalability Solutions**: Supports research in distributed databases, NoSQL systems, and horizontal scaling. 9. **Research Gateway**: Opens doors to interdisciplinary research involving data mining, machine learning, and information retrieval.
Learning Approach	- Follow any book – from the beginning - Go thru as many solved examples, activity sheets kind of exercise - Programming Practice – spend time on solving queries (specially on Joins, nested & subqueries) , don't google for quick answer - Attempt lot of MCQs and Questions on coding platforms; and start building profiles

Module 1 : Lecture Plan

Module Learning Objectives	By the end of the modules, learners will be able to:
	Module 1: Introduction to DBMS **Learning Objectives:** • Understand the fundamental concepts and importance of a DBMS. • Differentiate between Database Systems and File-based Systems. • Describe the components of DBMS architecture (1-tier, 2-tier, 3-tier). • Explain data models, schemas, instances, and data independence. • Understand the purpose and syntax of DDL, DML, and DCL. • Explore the Database Development Life Cycle (DDLC) through case studies. **Module 2: Relational Data Model and ER Modeling** **Learning Objectives:** • Understand the structure and principles of the Relational Data Model. • Learn ER Model concepts: entities, attributes, relationships. • Apply ER diagram notation, identify keys, and mapping constraints. • Understand generalization, specialization, and aggregation in ER models.

	• Convert ER diagrams into relational • schemas. Understand the formal query languages: Relational Algebra and Tuple/Domain Relational Calculus. • Apply selection, projection, join, union, intersection, and difference operations. • Develop complex queries using nested operations. • Analyze and optimize relational expressions. **Module 3: Database Design & Normalization** **Learning Objectives:** • Understand the purpose of normalization and its role in database design. • Identify and eliminate data anomalies using 1NF, 2NF, 3NF, BCNF. • Apply multi-valued dependency concepts to reach 4NF. • Perform schema refinement based on functional dependencies. **Module 4-5: Transaction Management and Concurrency Control** **Learning Objectives:** • Understand the concept of transactions and the ACID properties. • Analyze the problems due to concurrent transaction execution. • Learn concurrency control techniques: locking, time-stamp, and validation-based protocols. • Determine conflict serializability and view serializability. • Apply techniques for deadlock prevention and recovery. • Understand database failures and recovery techniques. • Apply log-based recovery mechanisms (Undo/Redo). • Explore checkpointing and buffer management. **Module 6: File Organization & Distributed Database** **Learning Objectives:** • Understand the architecture and design of distributed databases. • Learn data fragmentation, replication, and consistency models. • Understand the architecture and concept of Indexing. • Learn about index files and types of indexing.

Lec. No.	Content	Pre-Reading Topics / Ques (Before Class)	Context Setting / Teaching Strategy	Detailed Delivery Approach (What & How to Teach)	Thought-Provoking Questions	Post-Class Activities / Readings / Assignments	Actual Delivery dates and detailed remarks
1	An Overview of Database Management System, File System vs DBMS	- Read: Navathe 6th Ed, Ch. 1, Page 4–10, 17-21 -Definition of Data and Database -Need for Database Management Systems (DBMS) - Think of 3 real-life systems where data is stored and accessed? - Play short clip: Lec-3: File System vs DBMS | Disadvantages of File System	Quick brainstorming: - Where do you think databases are used in daily life? -Imagine you're running an online store like Amazon. How do you keep track of millions of products, customer data, orders, and delivery details? You can't possibly store all that in Excel files or notepads!" - How does your University keep track of your marks and attendance? - Where do you think Instagram stores user profiles, photos, likes?	- Definition of Data, Information, Database, DBMS - Importance of Data Management -Applications of DBMS - Compare DBMS and File System using real-life examples (college records, hospital data)	- Why do we need a DBMS when data can be stored in spreadsheets? - Why is a DBMS used instead of a traditional file system for managing data? -Why does a file system lead to more data redundancy compared to a DBMS? - How would a DBMS handle the challenges of storing and managing data for a real-time traffic monitoring system?	- In a school's file-based system, student address information is stored separately in each department (e.g., library, admin, hostel). Sometimes, the same student has different addresses in each file. - Question: What problem does this scenario highlight, and how can a DBMS solve it? -An HR department maintains employee records using multiple Excel files. When an employee is promoted, the data is updated in one	

		| DBMS Advantages (youtube.com)	- PPT + Board			file but not in others. - Question: What kind of issue is this? Would a DBMS be helpful here? Why?	
2	Database System Concept and Architecture	1. What do you understand by architecture in software? 2. Have you heard of physical and logical data? 3. Why do users need different views of the same data? Read TB, Navathe pp. 38-47 - Play short clip: Lec-4: 2 tier and 3 tier Architecture with real life examples | Database Management System	Introduce 3-schema architecture with functionality of physical and logical data along with suitable example. - PPT + Board	-Schema definition & structure -Explain 3-tier DBMS architecture: external, conceptual, internal. - DBMS components. -Instance	Why DBMS architecture make systems flexible and secure? What are the security implications of exposing the internal schema to users? Can you simulate schema mapping between user view and actual data? Why is conceptual schema crucial for schema evolution?	-Draw the 3-schema architecture of a student management system. -Simulate schema transformation for a banking system across all three levels. .	
3-4	Data Model, Schema and Instances, Data Independence,	1. What is a model? Can you give an example? 2. What do schema and instance mean in general life? 3. Can you name some types of databases? Read TB, Navathe pg. 39-46 - Play short video clips: Lec-6: Three Schema Architecture | Three Level of Abstraction | Database Management System - YouTube Lec-7: What is Data Independence | Logical vs. Physical Independence | DBMS	-Use real-world examples (e.g., blueprint vs building). --Explain models with examples. -PPT + Board	-Need of Modelling -Explain data models: hierarchical, network, relational. -Explain schema-instance concept. - data independence -Logical vs physical data independence.	-Why is data independence critical? Can it be fully achieved? - Imagine you are designing a new social media app. Would you use a relational database or a different model (document, graph, etc.) and why?	-Identify schemas in Google Sheets or DBMS. -Model a small database using the hierarchical model, and then convert it into relational.	

5	Database Language and Interfaces (DDL, DML, DCL),	-SQL language components: DDL, DML, DCL - Read TB, Elmasri & Navathe, Sec.4.2 - Silberschatz et al., Ch.4	- Use a case study: "Building a University database system" -PPT + Board -Demo of SQL Commands	-Procedural vs Non Procedural Languages overview -SQL categories with examples. -DDL vs DML vs DCL. -Show simple CREATE, INSERT, GRANT statements.	- Why should we be careful while using the DROP command? -Why do we need the INSERT command in a database? -Why is it important to use the WHERE clause with the UPDATE command?	- Write the SQL statement to create the World table with the following schema: i. name (VARCHAR, primary key) ii. continent (VARCHAR) iii. area (INT) iv. population (INT) v. gdp (BIGINT) -Insert the following data into the World table: i. ('Afghanistan', 'Asia', 652230, 25500100, 20343000000) ii. ('Albania', 'Europe', 28748, 2831741, 12960000000) iii. ('Algeria', 'Africa', 2381741, 37100000, 188681000000) iv. ('Andorra', 'Europe', 468, 78115, 3712000000) v. ('Angola', 'Africa', 1246700, 20609294, 100990000000) -Write a query to retrieve all the data from the World table.	
5	Database Development Life Cycle (DDLC) with Case Studies.	1. What are the steps to build a software application? 2. Have you ever worked on a project? What steps did you follow? 3. Why is planning necessary before building a database? Read TB, Navathe 282-304	-Link SDLC with DBMS DDLC. -Present in stages with banking app example. -PPT + case study (University database development)	-DDLC Definition & overview - Phases of DDLC: - Database Planning - Requirements Collection and Analysis - Conceptual Design - Logical Design - Physical Design -Testing - Implementation - Monitoring and Maintenance -Importance of user feedback in iterative DB design -Use mini case study: University DB.	- At which stage does failure hurt most? Why? -Which DDLC phase most affects scalability of the system? -Why prototyping affect the traditional DDLC stages? -Compare the DDLC for an online vs. offline inventory system.	-Design a database roadmap for a food delivery app using DDLC stages. -Compare two industry-based DDLC workflows (e.g., Oracle vs. Microsoft). -Analyse a failed project and identify which DDLC phase was weak. -Design a simple DB for Book My Show (refer online movie ticket booking System).	
6-8	ER Model Concepts, Notation for ER Diagram,	-What is an entity and what is an attribute? -Can you draw a relationship	Use a classroom example: College database (students, faculty, courses)	-Entities, Attributes, Relationships, Constraints. Mapping	-Why use ER diagrams? Can we design without them?	-Design an ER model for a version-controlled document management	

	Mapping Constraints, Reduction, Extended ER Model	diagram of your family or school system? -What do terms like 1:1 and 1:N mean? Navathe pp. 203-205 - Play short video clips: Lec-14: Introduction to ER model | ER Model क्या है	-Visuals for Notations -White Board +PPT -Mini case study (Employee Management System) - PPT + Board	constraints (1:1, 1:N, M:N) -Reduction of ER diagram into Relations -Generalization -Specialization -Association/Aggregation	-How can an ER model handle time-varying relationships? -Can ER modeling be used for non-relational databases? How? -Why mapping constraints affect schema performance?	system. -Add recursive and ternary relationships to a basic ER model. - Design an ERD for an e-commerce website. The website has multiple products, each identified by a unique product ID, and has attributes like name, price, and quantity in stock. Customers have unique customer IDs and attributes such as name, email, and shipping address. Customers can place multiple orders, and each order can include multiple products. Each product can belong to multiple orders.	
9	Relational Data Model Concepts	-Go thru: https://youtu.be/Q45sr5p_NmQ?si=vuAmh_t8D13dj0xj - In the context of relational databases, what exactly is a **relation**? How does it relate to a **table**? - What are the key differences between a *relational model* and other data models like *hierarchical* or *network*?	Quick brainstorming:- "Where do you think data is stored when you order something from Amazon?" -"How does your university portal retrieve your attendance or marks instantly?" - Basic Terminology: Relation, Tuple, Attribute, Domain - PPT + Board	- **Show examples** of simple tables: -Student records -Shopping cart - From Hierarchical/Network Models to Relational Model (E. F. Codd's contribution). -Real-life analogies: Table = Relation, Row = Tuple, Column = Attribute.	-Why do we need a formal data model like the relational model when file-based systems already exist for storing data? -In a world of unstructured and semi-structured data, is the relational model still relevant? Why or why not?	**Task**: Design a relation schema for a University database storing student, course, and enrollment information. **-Requirements**: List attributes, their domains	
10	Integrity Constraints, Entity Integrity, Referential Integrity	-Read TB Navathe ,pg 89-92 - Can you think of a situation where missing or incorrect data might lead to a problem in a real-world application?	- "Can you think of a situation where incorrect or missing data caused a problem in real life (wrong bill, duplicate entries, etc." -PPT + Board	-Concept Introduction -Integrity Constraints: Definition & Types -real-world analogies (e.g., passport number, employee ID, etc.)	-"What happens if your Aadhaar or Roll Number is missing in a college database?" - "Consider a database for Library with Books, Members, and BorrowRecords. What constraints would you add?"	Discussion Forum: "Why is it risky to allow NULLs in primary keys or invalid foreign keys?" -Practical -Write SQL scripts to create tables with entity	

		- What are integrity constraints in a DBMS? -Why are constraints necessary in relational databases?		- Primary Key - Not NULL constraint - Foreign Key constraint		and referential integrity -Test constraint violations	
11-12	Keys Constraints, Primary Key, Foreign Key, Candidate Key, Super Key, Domain Constraints	Go thru: https://youtu.be/p3yJZH8_bsc?si=GQf29u5RyMiihMcR - What is a constraint in the context of a database? -Why are constraints important in relational databases? - How is a primary key chosen among multiple candidate keys?	"Imagine managing a college database that tracks students, courses, and enrollments. How do we uniquely identify each student? How do we ensure a student cannot enroll in a non-existent course?" - Visualize with a sample table (Student(RollNo, Name, Email, Phone)) -PPT +Board	-Keys Overview -Role in identifying tuples - Types of Keys - Unique + Not NULL - Composite Keys - Minimal Super Key - Multiple candidate keys - Selection of Primary Key from candidate keys -Foreign Key - Tabular comparison of Key Types	- Why do we need multiple types of keys (primary, candidate, super)? Would one universal key type sufficient in all scenarios? Why or why not? -How can the choice of a primary key impact the performance of queries, especially in large-scale databases?	**Case:** A company uses a **flat Excel sheet** to store employee records, and faces issues with **duplicate entries**, **inconsistent department names**, and **orphan records** (managers not in employee list). **Question:** How would relational constraints (PK, FK, domain constraints) solve these problems?	
13-14	Relational Algebra	Read :TB, Navathe page 174-176 Go thru: https://youtu.be/76v3gRns28U?si=Oc9WhTdBRBx9JOnJ - What is the goal of relational algebra in the context of databases? -What is the difference between **procedural** and **declarative** query languages? -Can relational algebra be used to retrieve information from a database without SQL?	- If your phone contacts were a database, how would you find only your family members who live in Delhi? Prompt students to think in terms of filters (SELECT) and attributes (PROJECT). -PPT + Board	- each operation with relational tables. - **Basic Operations** Selection (σ),Projection (π),Renaming (ρ),Cartesian Product (×) -**Set Operations** Union (∪),Set Difference (−),Intersection (∩) -**Advanced Operations** Join (⋈)**:** Natural, Theta, Equi-Join, Division (÷), Assignment Operation - **Derived Operations** Semi-join, Outer joins (Left, Right, Full) -Use simple schemas like STUDENT(SID,	- Why do we need SELECT when we already have filters in SQL? - Can UNION be performed on relations with different schemas? - Why is Cartesian Product not preferred in real systems? - Why is JOIN preferred over Cartesian Product?	Given a database schema: Student(sid, name, age) Course(cid, cname) Enrolled(sid, cid, grade) Write relational algebra expressions for: -List names of students older than 20. -Get names of students enrolled in any course. -Get names of students and their course names.	

				Name, Dept) and COURSE(CID, Name, Credits). Show step-by-step derivation of queries.			
15	Functional Dependencies	Go thru: https://youtu.be/fZ41WtisQgo?si=Zkitywc-oMDO1nta - Give one real-life example where one attribute determines another. -Why do you think functional dependencies are important in reducing redundancy? -When a dependency "violated" in a relation instance?	-Imagine a student database where every student has a **unique roll number**. If I know a roll number, I can get the **student name, department, and year**. But if I only know the name, can I be sure of the department? - Why is redundancy a problem in databases? Can you guess what might cause it? -PPT + Board	- Introduce **functional dependency** using real-life examples: E.g., "StudentID → StudentName" means knowing StudentID gives us exactly one StudentName. -Formal Notation: X → Y -Full Functional Dependency -Partial Dependency -Transitive Dependency - Armstrong's Axioms - Use a simple **inference tree** to show how one FD leads to another.	- Why you approach discovering candidate keys just from a list of functional dependencies? - Why Armstrong's axioms be used to prove or disprove a functional dependency between attributes? - Why are functional dependencies not always obvious from the data in a relation?	-Given R(A, B, C) and FDs like A → B, B → A, A → C, AB → C, decide which are valid. -Analyze a real-world scenario (e.g., College DB or Hospital DB) -Identify entities and attributes -Derive functional dependencies	
16	Canonical Cover	Read : page 342-344 RB, Korth - What is the purpose of minimizing a set of functional dependencies? - How do extraneous attributes affect a functional dependency?	- Imagine you're cleaning up a cluttered toolbox with duplicate tools doing the same job. To organize it efficiently, you want just one of each kind that covers all tasks nothing more, nothing less. That's exactly what we do with functional dependencies **using a** canonical cover. -Concept Introduction - Explain → Explore → Apply -PPT + Board	-Definition of Canonical Cover -Why it is needed -Rules to Minimize FDs -Steps to compute Canonical Cover -Applications **Additional** -Present a table of FDs and show which are redundant using arrows/flowcharts. -Show what happens when redundant attributes or FDs are not removed.	- Why do we even need a canonical cover when we already have a complete set of functional dependencies? -In real-world schema design, is computing the canonical cover truly necessary, or is it an academic exercise?	- You are designing a schema for a hospital management system. Given a list of functional dependencies (some of which are non-canonical), analyze and: -Convert them into canonical form -Identify redundancies in the original set -Reflect on the risks of using the non-canonical form in real-world scenarios	

17-18	Anomalies, Normal Forms, First, Second	Read : page 501-504 TB, Navathe Go thru: https://youtu.be/UF0UHCX-z0E?si=dBHZ5qmMTB7lT7mF - Why do we need to make tables simple and organized? - What do you think **"normal form"** means in databases? - What does the word **"anomaly"** mean in general?	- You are managing student data at a college. In your table, if one student enrolls in multiple courses, do you repeat their name every time? What happens when a student changes their address? Do you update it everywhere? What if you miss one place? - Worksheet: Identify anomalies + Need of normalizing a given relation. -PPT + Board	- Update, Insert, Delete Anomaly - Real-world examples (e.g., student-course registration) - Concept of Normalization - 1NF -2NF - Converting to 2NF (decomposition) - Summarize steps: Find FD → Identify keys → Check anomalies → Normalize -Mind Map: Build a visual flowchart on board from Anomalies → 1NF → 2NF	- Why do insertion, deletion, and update anomalies occur in a database? -Can you provide a real-world example for each? - Is it always necessary to normalize a database if no anomalies have occurred yet? - Suppose you're working with a legacy database that violates 2NF. What steps would you take before recommending restructuring?	- Relation: Course(Course_ID, Course_Name, Instructor_ID, Instructor_Name, Department) FDs: 1. Course_ID → Course_Name 2. Instructor_ID → Instructor_Name, Department Check if it's in 1NF and 2NF. If not, decompose into relations in 2NF with clear reasoning.	
19-20	Third Normal Forms, BCNF, MVD and 4th Normal Form, 5th Normal Form	Read TB, Navathe Page 523, 529-531 Go thru: https://youtu.be/OTCuykFHBeA?si=hVJ92ZyLOQUYCrv- - If a relation has a composite candidate key, what type of **partial dependency** might occur? - What would happen if we ignore **multivalued dependencies** in schema design? - Why do we break tables into smaller ones?	-Imagine you're building a database for an e-commerce site. Everything works well until you start getting strange duplicate records and inconsistent updates. Despite applying 1NF and 2NF, problems still arise. Why? -Distribute a table schema with FDs and ask to: -Identify candidate keys -Apply each NF step-by-step till 4NF -Justify decompositions - PPT + Board	-3NF: no transitive dependency - Define BCNF - Show difference from 3NF - Define MVD - Examples of MVD in real life - Problems caused by MVD - Define 4NF - 4th NF vs BCNF -5NF	- Why is 3NF often considered "good enough" in practical database design? - Why is it possible for a relation to be in 3NF but not in BCNF? Can you give an example? - Why is it essential to identify MVDs when designing schemas beyond BCNF? -	- Given a relation: **Employee(EmpID, DeptID, DeptLocation, ProjectID)** and dependencies: EmpID → DeptID DeptID → DeptLocation EmpID →→ ProjectID a)Identify which normal form this relation violates. b) Decompose it into 4NF relations. - **Task:** Normalize the given relation to 3NF, BCNF, and 4NF (if possible). R(StudentID, CourseID, Instructor, Hobbies) FDs: StudentID → Instructor, CourseID → Instructor	

						, StudentID ↠ Hobbies - A relation can be in 3NF and not in BCNF. (T/F)	
21-22	Lossless Join and Dependency Preserving Decomposition, JD and, Inclusion Dependence	Read TB, Navathe page 552-553 - What is a lossless join decomposition ? Why is it essential? - What is a join dependency (JD)**?** - What is Fifth Normal Form (5NF)**?**	- Imagine you're designing a university system where student data, course enrolment and faculty allocations are spread across different tables. How do you ensure when you split this data, nothing is lost or misinterpreted—and you can always reconstruct the original? -PPT +Board	- Lossless Join: Introduction & Theory - Dependency Preservation - JD -Examples & Applications - Inclusion Dependency - Meaning & Applications	- Is it always necessary to preserve all functional dependencies in a decomposition? In what scenarios might some FD loss be tolerated or even preferred? - Why are join dependencies rare in practice? Can you design a realistic business case where 5NF is essential?	A company stores data in a table with columns: Employee, Project, and Location. They split the table into three separate tables: Employee-Project Project-Location Employee-Location Later, when they join these three tables, they get combinations of Employee, Project, and Location that never actually existed. Why do incorrect combinations appear after the join? How can this be fixed using normalization?	
23	Transaction System	Go thru: https://youtu.be/wHUOeXbZCYA?si=Pqds2xF6dCEf5S2- Video Ref: 8.1 Introduction To Transaction and Concurrency Control in DBMS (youtube.com) Read: Navathe Ch. 21, Page 744-754. - Define the term **"consistency"** in the context of database transactions. - Can you list activities that involve transactions?	Imagine you're booking a train ticket online. You've selected your seat, entered payment details, and hit confirm. The payment gets deducted, but the seat isn't booked due to a system crash. What happened? - Sample case studies (banking, e-commerce) -Visuals/diagrams of transaction systems -PPT + Board	-Introduction to Transaction Processing, SQL examples like BEGIN TRANSACTION, UPDATE, COMMIT, ROLLBACK. -Transaction States (Life Cycle), Use a state transition diagram. -ACID Properties -Use real-world analogies (e.g., bank transfer example).	- If transaction atomicity is not enforced, how could it lead to data anomalies? - Suppose a power outage occurs midway through a transaction. Which transaction state and recovery mechanism would be triggered, and how would the system ensure data integrity? - In modern cloud-based databases, eventual consistency is often preferred over strong consistency. How does this decision challenge the traditional ACID principles of transactions?	- Simulate a transaction scenario where multiple operations are grouped under a transaction. Use BEGIN, COMMIT, and ROLLBACK operations. - **Task**: Given a set of operations, identify: Where the transaction starts, fails, or commits T1: Read(X); X = X – 50; Write(X) T2: Read(Y); Y = Y + 100; Write(Y); Commit	

24-25	Serializability, Testing of Serializability, Serializability of Schedules	Read: Navathe 6th Ed, Ch. 21, Page 759-767, - What do you understand by the term **schedule** in the context of transaction processing? - What makes a schedule **serializable**? - Difference between **conflict serializability** and **view serializability** ? Video Ref: Lec-83: Conflict Equivalent Schedules with Example | Transaction concurrency and Control | DBMS (youtube.com) Lec-84: Conflict Serializability | Precedence Graph | Transaction | DBMS (youtube.com)	- Imagine you're at a supermarket with three self-checkout machines. Three customers (transactions) are scanning their items (operations). Ideally, the final record (database state) should be the same as if they had taken turns one by one. What if two people scanned the same item at the same time? Would the bill be correct? - Provide students with a schedule of operations. - List the conflicts. - Draw the graph. - Determine serializability. - PPT + Board	-Need for serializability - Serial schedule vs non-serial schedule - Serializable schedule -Conflict serializability - View serializability -Conflict operations (Read-Write, Write-Read, Write-Write) -Precedence graph (serializability graph) - Cycle detection -**Method**: Step-by-Step Demo + Hands-on Practice **Activities**: -Explain how to build a precedence graph. -Assign 2-3 schedules to build precedence graphs and test for cycles.	-Why is serializability considered a key criterion for ensuring correctness in concurrent transaction execution? Can a schedule be correct without being serializable? -Imagine you're a DBMS designer optimizing for high concurrency. Would you prioritize view serializability or conflict serializability?	- Consider the schedule: T1: R(X), W(Y) T2: R(Y), W(Z) T3: R(Z), W(X) a) Construct the precedence graph. b) Is the schedule conflict serializable? Why or why not? -	
26-27	Recoverability, Recovery from Transaction Failures, Log Based Recovery	Read: Navathe 6th Ed, Ch. 21, Page 751–754 Video Ref: Lec-80: Irrecoverable Vs Recoverable Schedules in Transactions | DBMS (youtube.com) - What does the term "recoverability " refer to in the context of transactions in a DBMS? - List the typical reasons for transaction	- Imagine you're transferring ₹10,000 from your savings to your friend's account using a banking app. Suddenly, your phone loses internet connection. The money is debited from your account but your friend doesn't receive it. What just happened? How does the bank fix it? -PPT + Board	-Recoverability -Recovery from Transaction Failures -Log-Based Recovery (Undo/Redo Protocols, Checkpoints) -Present a sample log and demonstrate:Crash point -What operations will be redone/undone -Use WAL rules to validate logs	-Why is "cascadeless schedule" preferred over a "recoverable schedule"? Can you provide a practical example where the difference between the two becomes critical? -Why is Write-Ahead Logging (WAL) critical for ensuring database consistency? What could go wrong if it's not followed?	**-T1: Read(A); Write(A);** **T2: Read(A); Write(A); Commit;** **T1: Commit;** ☐ Is the schedule **recoverable**? ☐ Is it **cascadeless**? Justify. -	

		failure in a database system? - What is a **log** in DBMS and how is it used during recovery?					
28	Deadlock Handling.	Read page 674-676 RB, Korth - What is a *deadlock* in the context of DBMS transactions? - Can you think of any real-world situations where two or more parties are stuck waiting for each other to act first? -	- Have you ever been stuck in a traffic jam where no car could move because each one was waiting for another to move first? - What if Transaction A is waiting for a lock held by B, and B is waiting for a lock held by A? -PPT + Board	- Definition & Conditions - Deadlock -Deadlock Handling in DBMS -wait-die scheme -wait-wound scheme -real life examples (eg IRCTC, Airlines reservation)	- A bank system requires both high consistency and high availability. Which deadlock handling technique would you recommend and why? - Students submit answers, which are auto-evaluated and updated in a shared result database. During high-load periods (e.g., end of session), some submissions seem "stuck". **Prompt**: Could this be a deadlock scenario? If so, how would timeout vs wait-die prevention strategies affect system performance and student experience?	- Consider the following transactions: T1: lock(A), lock(B) T2: lock(B), lock(C) T3: lock(C), lock(A) All transactions are running concurrently. **Question:** a) Identify whether a deadlock will occur. b) Propose a deadlock detection and resolution strategy.	
29-30	Concurrency Control	Go thru: https://youtu.be/h7Ucn5bqipQ?si=PlAJ_bg88zW2kj0h Read: Navathe 6th Ed, Ch. 22, Page 777 - What do you understand by the term *concurrency* in a database system? -What kind of techniques or mechanisms do you think might be used to control concurrency in databases?	- You are withdrawing money from an ATM while your partner is depositing money into the same account. What could go wrong if both actions are processed simultaneously without coordination? - a simple example where two transactions lead to a lost update. Share your schedule. -PPT + Board	- Concept of Concurrency control - Problems due to concurrent execution -Need for concurrency control -Lost update, dirty read, unrepeatable read. Phantom read -Overview of Lock-Based Protocols - Timestamp Ordering & Validation-Based Protocols - **Activity**: Animated demonstration of lost update & dirty read problems	- Is it possible to achieve both high concurrency and strict isolation simultaneously? Why or why not? - Imagine you are tasked with designing a database system for a high-frequency trading application. Which concurrency control technique would you choose and why?	**T1: R(A), W(A), R(B)** **T2: R(A), W(A), R(B), W(B)** - Identify if this schedule leads to a **lost update** or other concurrency issue. Justify your answer. - Design and simulate a transaction schedule involving at least three transactions that leads to a **deadlock**. Identify how this deadlock can be prevented using any **deadlock prevention strategy**.	

31	Locking Techniques for Concurrency Control, 2PL	Go thru: https://youtu.be/44Uc6ohuOAk?si=FiOfy44Ertagvakv YouTube Ref: 8.25 Lock Based Protocol in DBMS Concurrency Control Part-1 (youtube.com) 8.26 Properties of Lock Based Protocols in Concurrency Control Part-2 (youtube.com) Read: Navathe 6th Ed, Ch. 22, Page 778-779, Page 780–784 - What do you think a “lock” does in a database? - Why should a transaction lock data before using it? - What is the difference between read access and write access in terms of their effect on data consistency?	- **Scenario:** Imagine a library where multiple people want to read the same rare book, but only one copy is available. Q1: What problems might arise if two people try to read and annotate the same book at the same time? Q2: How can the library ensure fairness and consistency when multiple users want to access the same book? -Relate to exclusive/shared access and the need for locking in DBMS. PPT + Board	- Introduce the basic idea of locks -Use real-world analogy (e.g., locking a file cabinet while accessing documents). -Explain **Binary Locks** (locked/unlocked), **Shared (S) & Exclusive (X) locks**. -structure of 2PL and its variants -**Timeline Diagrams**: Show lock acquisition & release over time. -*Growing Phase*: Only acquire locks -*Shrinking Phase*: Only release locks -**Strict 2PL** (release X-locks only after commit) **-Rigorous 2PL** (release all locks after commit)	-If a transaction is long-running and holds several locks, what risks does it pose to the system? How can these risks be minimized without violating consistency? -Under 2PL, all locks must be acquired before any are released. Can you relate this to real-life examples (e.g., airline ticket booking or banking)?	**-** Analyze the following schedule and determine whether it can be generated by a system that uses **Strict 2PL**. Justify your answer. **T1: Lock-S(A), R(A), Lock-X(B), W(B), Unlock(A), Unlock(B)** **T2: Lock-S(B), R(B), Lock-X(A), W(A), Unlock(A), Unlock(B)**	
32-33	Time Stamping Protocols for Concurrency Control, Validation Based Protocol.	Read: Navathe 6th Ed., Ch. 22, Page 788–791, Page 794–795 - What do you understand by the terms "read-timestamp" and "write-timestamp"? - What does it mean to “validate” a transaction in the context of databases? Read: Navathe 6th Ed., Ch. 22, Page 788–791.	- Imagine a system with multiple transactions running simultaneously. What could go wrong if there's no control over their execution order? - What might happen if multiple transactions are allowed to execute without any locks? Can we still maintain consistency? -PPT + Board	**-**Concept of Timestamps -Real-life analogy: queue tokens in a bank or timestamps on emails. - Rules for Read and Write operations. -Use of Read_TS(X) and Write_TS(X). - Validation-Based Protocol / Optimistic Concurrency - Read Phase	- Why might a timestamp-based protocol reject a schedule that appears to be conflict-serializable? - Validation-based protocols assume transactions rarely conflict. What are the implications when this assumption fails?	**-** A system uses timestamp ordering for concurrency control. The following operations are issued: T1: r1(A), w1(A) T2: r2(A), w2(A) Assuming T1 has timestamp 1 and T2 has timestamp 2, explain step-by-step how the system will handle these operations. - Given the read and write sets of two transactions,	

		YouTube Link: 8.21 Time Stamping Protocol in DBMS Part-1 (youtube.com) 8.22 Time Stamping Protocol in DBMS Part-2 (youtube.com)		-Validation Phase -Write Phase		apply the validation conditions to check whether both can commit: T1: Read Set = {A, B}, Write Set = {C} T2: Read Set = {C}, Write Set = {D} Assume T1 validates before T2.	
34-35	**File Organization:** Indexing, Structure of Index files and Types	Go thru: https://youtu.be/RgYU5r9A5TU?si=q7_DlXOzdaI7wb6r Lec-96: Types Of Indexes | Most Important Video on Indexing - YouTube Read: Navathe 6th Ed., Ch. 18, Page 631–634 - Define indexing in the context of databases. - What are the different types of file organization techniques you are familiar with? - Have you ever used a book index to find something? How does it compare to a database index?	- Have you ever used a book index to find something? How does it compare to a database index? - Today we will explore how databases organize files more efficiently using Indexing ,its structure and types. -PPT + Board	- need and purpose of indexing in file organization. - structure of index files. - Logical layout: Key and pointer. - various types of indexing - Single-level Index, Multi-level, Clustered vs Unclustered Index, Primary vs Secondary Index, **Additional** -Index blocks/pages. -B+ tree indexe Page access and disk block mapping. - Use a **library catalog system** or a **phone book** to explain indexing.	- Why is indexing considered a trade-off between speed and storage in file systems? Can you think of scenarios where indexing may not be beneficial? - If you were to build a custom indexing method for multimedia files (like images or videos), what factors would you consider compared to text-based indexes?	- Given a file with 1 million records and each block can hold 100 records: Calculate the number of index entries in a clustered index. -A file has **400,000 records** stored in blocks of **200 records each**. A block can store **50 index entries**. **(a)** How many first-level index entries are needed (sparse index)? **(b)** How many second-level entries are required to index the first-level index?	
36	Dense and Sparse Indexing.	Go thru: https://youtu.be/h2iKFWl1RFY?si=-vy4nrr6ctMiHpn2 Read: Navathe 6th Ed., Ch. 18, Page 633–635, 636-640	- A retail store maintains a database of product inventory sorted by product code. Queries often involve searching for product details based on product codes.	- **Activity**: Ask students – How does Google search show results so fast? -Use this analogy to connect with indexing in databases.	- Why is a sparse index not possible on an unordered file? - In what scenarios would a dense index be more efficient despite its higher storage cost?	- A file has 20,000 records. The block size is 1 KB, and each record is 100 bytes. -Construct a dense index and a sparse index. How many	

		- How does using an index improve the performance of data retrieval operations? - when to use dense or sparse indexing? - How do you think the structure of a dense or sparse index might affect disk I/O performance?	-Would a sparse index be sufficient if products are frequently accessed in sequential ranges (e.g., by category)? -How would a dense index support quick access to products that are updated or restocked frequently? **Strategy:** -Show a short demo of data retrieval *with and without* indexing using a simulated large text or spreadsheet. -Use analogies like a book index (dense) vs. chapter title list (sparse). - PPT + Board	-Primary, Secondary, Clustering indexes (brief) - Structure of dense index - Storage layout - Pros and cons - Structure of sparse index - Conditions for use - Pros and cons - Dense vs Sparse index comparison	- Imagine you are designing an indexing strategy for a search engine database with billions of records. What factors would influence your choice between dense and sparse indexing?	index entries will each contain? -A dense index has 500 entries pointing to 500 data blocks. Each index block can hold 50 entries. -How many index blocks are required? Repeat for sparse index if only 1 entry per 10 data blocks is used.	
37-38	Introduction of Distributed Database, Data Fragmentation and Replication.	Go thru: https://youtu.be/aoMOmSx5Zyw?si=uvjCphrJoOmRjFf7 L127: Types of Distributed Databases | Data Storage (Fragmentation, Replication) | Transparency (youtube.com) Read: Navathe 6th Ed., Ch. 25, Page 878–879, Page 894–900. - What do you understand by the term "distributed system"? Can you give a real-life example? - How do centralized databases differ from distributed databases in your opinion?	- Imagine you're using Google Docs, and you're typing something while your friend in another country is editing the same document. Ever wondered how both of you can work on the same file without conflict? That's made possible through the power of **distributed databases**. - Why do global companies like Amazon or Netflix not rely on a single centralized database? -What happens if a single server storing all your WhatsApp messages crashes? - PPT + Board	-Introduction -Architecture & Design Issues - Comparison: Centralized vs Distributed. -Types of distributed databases. -homogenous & hetrogenous -Data Fragmentation, Replication, Allocation - Types of fragmentation (horizontal, vertical, mixed). -Replication strategies: full, partial, none. -Allocation strategies and criteria. - Recovery & Reliability **Additional** -NoSQL overview	- When should a company opt for a distributed database over a centralized one? Explore with at least two real-world scenarios - Why fragmentation in a distributed system affect data security and access control mechanisms? - In a distributed system, can replication and partitioning coexist effectively? What trade-offs must be considered?	- A university has different campuses. Student data needs to be split such that each campus stores only the records of its students. What type of fragmentation fits this need? - In a healthcare system, lab staff only needs access to patient test results, while the admin needs address and billing information. How would vertical fragmentation help in this scenario? **Additional:** Choose one of the following distributed databases and write a short report (~500 words) covering its architecture, data fragmentation	

				- Real-world examples (MongoDB, Cassandra, DynamoDB).		strategy, and replication mechanisms: -Google Spanner -Apache Cassandra -Amazon DynamoDB	