


Database Management Systems

Dr. Dev Narayan Yadav
Department of CSE, NIT Rourkela
Email: yadavd@nitrkl.ac.in
Mo. – 8349869748
Room – CS204


Dr. Dev Narayan YadavDepartment of CSE, National Institute of Technology Rourkela





Course Overview

- ❑ **Subject Code:** *CS2008/CS2062 (Database Engineering / Database Management Systems (L-T-P:3-0-0, Credit: 3))*
- ❑ **Evaluation:**
 - *Teachers Assessment (20): CT1(5), CT2(5), Assignment (10)*
 - *Mid Term (30)*
 - *End Term (50)*
- ❑ **Essential Readings:**
 - J.D. Ullman, Principles of Database Systems, Computer Science Press, 1982.
 - A. Silberschatz, H.F. Korth, and A. Sudarshan, Database System Concepts, McGraw Hill, 5th ed, 2006.

Dr. Dev Narayan YadavDepartment of CSE, National Institute of Technology Rourkela


Syllabus	
<ul style="list-style-type: none"> ❑ Introduction to database systems: <ul style="list-style-type: none"> • Data Independence, Data Models, Levels of abstraction, structure of DBMS ❑ Data models and query languages <ul style="list-style-type: none"> • Relational Model, Relational Languages, Query Languages: Relational Algebra, Relational Calculus, SQL, QUEL, QBE, Integrity constraints, Aggregate operators, Embedded and Dynamic SQL. ❑ Database design: <ul style="list-style-type: none"> • E-R Model, Functional dependencies, Decomposition, Normalization, Multivalued dependencies. ❑ Concurrency control and recovery: <ul style="list-style-type: none"> • Transaction, Schedules, Lock based concurrency, • Lock management, Concurrency control without locking, Crash recovery- log, check • pointing, media recoveries 	
Dr. Dev Narayan Yadav	Department of CSE, National Institute of Technology Rourkela

Syllabus	
<ul style="list-style-type: none"> ❑ File Organization: <ul style="list-style-type: none"> • Storage, Buffer management, Disk Management, File organization techniques, Indexing. ❑ Query Processing and Optimization: <ul style="list-style-type: none"> • Query processing on various operations, Translating SQL queries, Estimating the cost. ❑ Advanced topics: <ul style="list-style-type: none"> • Database Security, Distributed databases design, Object Oriented database design & its implementation, Introduction to recent advances in database technology. 	
Dr. Dev Narayan Yadav	Department of CSE, National Institute of Technology Rourkela



Introduction to Database Engineering

Dr. Dev Narayan Yadav Department of CSE, National Institute of Technology Rourkela



Common Terminologies

- ❑ **Data:**
 - Fact, figures, statics etc. (e.g. 1, ABC, 19).
- ❑ **Record:**
 - Collection of related data Items.

Roll	Name	Age
1	ABC	19

Dr. Dev Narayan Yadav Department of CSE, National Institute of Technology Rourkela

Common Terminologies



❑ Table or Relation:

- Collection of related records.
- **Columns** are called: Fields, Attributes or Domain.
- **Rows** are called Tuples or Records.

Roll	Name	Age
1	ABC	19
2	DEF	22
3	XYZ	23

Common Terminologies



❑ Database:

- Collection of related relations/tables.

Roll	Name	Age
1	ABC	19
2	DEF	22
3	XYZ	23

Roll	Address
1	Rourkela
2	Delhi
3	Shillong

Roll	Year
1	I
2	II
3	III

Roll	Hall
1	CVR
2	SSB
3	MV

Components of Database

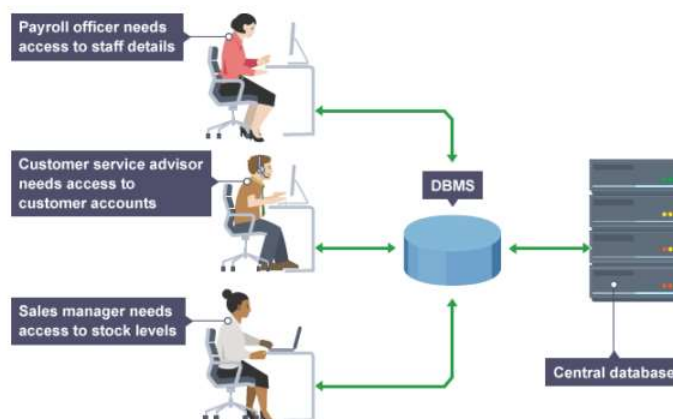
- ☐ Data:
 - Fact, figures, statics etc. (e.g. 1,2, ABC, 19AB).
- ☐ Hardware
 - Physical device where data is stored e.g. servers.
- ☐ Software's:
 - DBMS e.g. SQL server, Oracle.
- ☐ Users
 - End User
 - Database Administrators
 - Developers

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

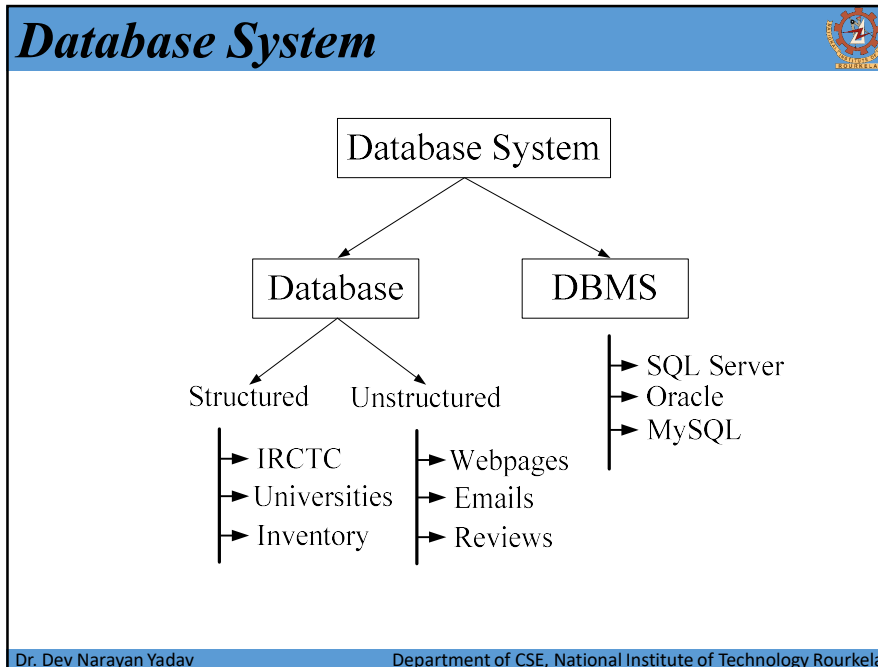
Components of Database

- ☐ A database in DBMS could be viewed by different peoples with different responsibilities.



Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela



Database System

- ❑ **What is database?**
 - An organized collection of inter-related data.
 - Can be stored manually or electronically.
- ❑ **What is DBMS?**
 - A software that allows user to store, retrieve and edit data in database.
- ❑ The combination of the DBMS and the data it manages is often referred to as a **“database system,”** or simply a **“database”**.

Dr. Dev Narayan Yadav Department of CSE, National Institute of Technology Rourkela

Applications



- ❑ Database touch all aspect of our lives
 - **Banking:** Transactions, Account holders, etc.
 - **Airlines:** reservation, schedules etc.
 - **Universities:** registration, grades, etc.
 - **Sales:** customer, products, purchases etc.
 - **Manufacturing:** production, inventory, orders, etc.
 - **Human Resource:** employee records, salaries, tax, etc.

File System v/s Database System



- ❑ *Example*
 - File System:
 - NTFS (windows), FAT32 (USB), ext4(Linux), APFS(macOS and iOS), HFS (distributed storage).
 - User_photos/photo1.jpg
 - Database System:
 - Relational: MySQL, PostgreSQL, Oracle Database.
 - NoSQL: MangoB, Cassandra, Redis.
 - SELECT * FROM
 - Hybrid
 - AWS: file system (to store large files), database to store metadata.

File System v/s Database System



□ Structure and Organization

- File System:
 - Data is stored in files within a directory hierarchy.
 - Organization: manual or custom schema.
 - Suitable for unstructured or semi structured data.

- Database System:
 - Structured format e.g. table, rows, columns.
 - Support relation between datasets.
 - Ideal for structured or organized data.

File System v/s Database System



□ Data Access and Retrieval

- File System:
 - Access using file path or API.
 - Limited support for advance query.
 - Slow for large dataset.

- Database System:
 - Advance query support like SQL.
 - Optimized indexing and searching capability.
 - Fast and efficient retrieving of data.

File System v/s Database System



❑ Performance and Scalability

- File System:
 - Depends on the file organization and file size.
 - Not inherently designed for concurrent access or scaling.
 - Better for small-scale, simple applications.
- Database System:
 - Supports concurrent read/write operations.
 - Scales well horizontally (distributed systems) or vertically (hardware upgrades).

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

File System v/s Database System




❑ Data Integrity and Consistency

- File System:
 - Minimal support for ensuring data consistency.
 - Developers must implement mechanisms for locking, versioning, and integrity checks.
- Database System:
 - Built-in support for data integrity (e.g., ACID properties).
 - Prevents data corruption during concurrent or transactional operations.

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

File System v/s Database System




❑ Security

- File System:
 - Relies on OS-level security (file permissions, access controls).
- Database System:
 - Supports robust security features (user roles, authentication, encryption).
 - Allows fine-grained access control at table, row, or column levels.

Dr. Dev Narayan Yadav Department of CSE, National Institute of Technology Rourkela

File System v/s Database System



❑ Backup and Recovery

- File System:
 - Simple to back up by copying files.
 - Recovery processes depend on manual intervention or custom scripts.
- Database System:
 - Built-in tools for backup and recovery.
 - Can perform incremental backups, ensuring minimal data loss.

Dr. Dev Narayan Yadav Department of CSE, National Institute of Technology Rourkela

File System v/s Database System



☐ ***Use Cases***

- File System:
 - Storing static assets like images, videos, or log files.
 - Simple or temporary storage needs.
 - Applications where data relationships are minimal or nonexistent.
- Database System:
 - Managing relational data (e.g., customer information, inventory).
 - Complex querying and reporting needs.
 - Applications requiring high scalability, consistency, and security.

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

Why Database?



- ☐ Data management.
 - Efficient storage and organization.
- ☐ Accessibility
 - Easy retrieval and manipulation.
- ☐ Security
 - Protect sensitive information.
- ☐ Scalability
 - Handle large amount of data.
- ☐ Enforcement of Standards, Data Independence etc.

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

Advantage of Database



- ☐ Data duplication eliminated
- ☐ Controlled redundancy
- ☐ Restricting unauthorized access
- ☐ Providing backup and recovery
- ☐ Providing Multiple User Interface
- ☐ Enforcing Security and integrity constraints
- ☐ Centralized Control and Data quality enhanced

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

Disadvantage of Database



- ☐ Problems associated with centralization
- ☐ Cost of software/hardware and migration
- ☐ Complexity of backup and recovery.

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

Data Models



- ❑ A **framework** that defines how data is stored, organized and manipulated in a database.
- ❑ In other word its a collection of conceptual tools for describing data, data relationships, data semantics, and consistency constraints.
- ❑ Data models provide the foundation for database design and operation.
- ❑ The choice of a data model depends on the type of data, complexity of relationships, and specific application needs.

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

Categories of Data Models



- ❑ **Conceptual Data Models:**
 - High-level models that describe the structure of the database in a way that is independent of the underlying DBMS or hardware.
 - **Purpose:** To represent the overall logical structure of the data.
 - **Example:** Entity-Relationship (ER) Model.
 - **Use Case:** Used during the initial database design phase to communicate with stakeholders

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

Categories of Data Models



☐ *Logical Data Models*

- Abstract representations of the data that define the structure in terms of tables, columns, keys, and relationships but are still independent of the physical implementation.
 - **Purpose:** To bridge the gap between the conceptual design and the physical implementation.
 - **Example:** Relational model, Object-oriented model.
 - **Use Case:** Used to design schemas that work across different DBMS platforms.

Categories of Data Models



☐ *Physical Data Models:*

- Detailed models that describe how the data will be stored in the database system.
 - **Purpose:** To optimize storage, performance, and data retrieval.
 - **Example:** Indexing structures, storage formats..
 - **Use Case:** Used during the actual database implementation phase.

Types of Data Models



☐ *Hierarchical Data Models*

- **Structure:** Data is organized into a tree-like structure with parent-child relationships.
- **Feature:** Parent → child, one-to-many relationship.
- **Example:** XML data representation.
- **Use Case:** Early file system.
- **Limitations:** Difficult to handle complex relationships.

Types of Data Models



☐ *Network Data Model*

- **Structure:** Data is represented as a graph with nodes and edges, allowing many-to-many relationships.
- **Key Features:** Flexible relationships; uses pointers to link records.
- **Example:** Integrated Data Store (IDS).
- **Use Case:** When data relationships are complex.
- **Limitation:** Complex to implement and manage.

Types of Data Models



❑ Relational Data Model

- **Structure:** Data is organized into tables (relations) with rows and columns.
- **Key Features:** Tables represent entities; primary keys and foreign keys establish relationships, uses Structured Query Language (SQL).
- **Example:** MySQL, PostgreSQL.
- **Use Case:** Widely used in modern applications.

Types of Data Models



❑ Document Data Model (NoSQL)

- **Structure:** Data is stored as documents (e.g., JSON, BSON).
- **Key Features:** Schema-less or flexible schema, supports hierarchical and nested data.
- **Example:** MongoDB.
- **Use Case:** Web applications, real-time analytics.
- **Limitation:** Lacks standard query languages like SQL.

Types of Data Models



❑ Key-Value Data Model (NoSQL)

- **Structure:** Data is stored as key-value pairs.
- **Key Features:** Simple and fast; each key uniquely identifies a value.
- **Example:** Redis, DynamoDB.
- **Use Case:** Caching, session management.
- **Limitation:** Not suited for complex queries or relationships.

Types of Data Models



Model	Structure	Relationship Type	Example DBMS	Use Case
Hierarchical	Tree	One-to-Many	IMS DB	Early file systems, XML data
Network	Graph	Many-to-Many	IDS	Complex data relationships
Object-Oriented	Objects	Complex (OO-based)	ObjectDB	CAD, multimedia databases
Document (NoSQL)	Documents	Flexible	MongoDB	Web apps, unstructured data
Key-Value (NoSQL)	Key-Value Pairs	None	Redis, DynamoDB	Caching, real-time data storage
Relational	Tables	Many-to-Many	MySQL, PostgreSQL	Modern applications, analytics

Schema and Instance



- ❑ A **database schema** defines the structure and organization of a database.
- ❑ It is a blueprint that specifies how data is stored, organized, and interrelated in the database.
- ❑ The schema includes details about tables, fields, relationships, constraints, and other elements required to define the database's design.
- ❑ **Instance:** Actual data stored in the database at a moment (Records stored in tables like Students and Courses).

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

Types of Schema



- ❑ **Physical Schema:**
 - Specifies how data is physically stored in the database.
 - Includes file structures, storage methods, and indexing.
- ❑ **Logical Schema:**
 - Defines the logical structure of the database, independent of physical storage.
 - Includes tables, columns, data types, relationships, and constraints.
- ❑ **View Schema (External):**
 - Specifies how data is presented to different users or applications.
 - Can include subsets of data or a customized format..

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

Key Component of Schema



❑ **Tables or Relations (Entities):**

- Represents the objects or concepts in the database.
- **Example:** Students, Courses, Enrollments.

❑ **Attributes (Fields or Columns):**

- Represents properties of entity.
- **Example:** Name, Age, Hostel, Roll.

❑ **Keys:**

- **Primary Key:** Uniquely identifies a record in a table (e.g., Roll).
- **Foreign Key:** Establishes relationships between tables.

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

Key Component of Schema



❑ **Constraints:**

- Define rules for data integrity and consistency.
- **Example:** NOT NULL, UNIQUE, FOREIGN KEY..


❑ **Relationships:**

- Define how tables are related (e.g., one-to-one, one-to-many, many-to-many).
- **Example:** Student → Enrollments (one-to-many), each students can enroll in many courses.

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela


DBMS Users



- ❑ **Naïve Users:**
 - Users who need not be aware of the presence of the database system or any other system supporting their usage.
- ❑ **Online Users:**
 - Users who may communicate with the database directly via an online terminal or indirectly via a user interface and application program.
- ❑ **Application Programmer:**
 - Professional programmers who are responsible for developing application program or user interface utilized by the naive and online users.

Dr. Dev Narayan YadavDepartment of CSE, National Institute of Technology Rourkela

DBMS Users



- ❑ **Database Administrator (DBA):**
 - Centralized control of the database is exerted by a person or group of persons who are referred to as the Database Administrator (DBA).
 - They are responsible for creating, modifying, and maintaining its three levels.

Dr. Dev Narayan YadavDepartment of CSE, National Institute of Technology Rourkela

DBMS Facilities



❑ **DDL (Data Definition Language):**

- Used for defining the conceptual scheme
- Used to specify additional properties of the data
- Provides facilities to specify constraints

❑ **DML (Data Manipulation Language):**

- Enables users to access or manipulate data as organized by the appropriate data model
- **Access Types:** Retrieval, Insertion, Deletion and Modification of Information.

❑ Apart from these, there exists **DCL (Data Control Language)** and **TCL (Transaction Control Language)**

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

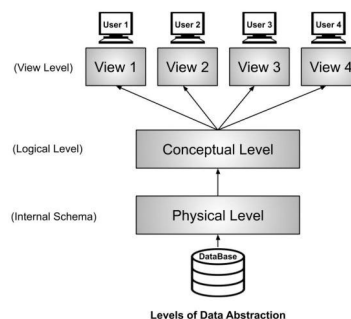
Levels of Abstraction



❑ **What do you mean by abstraction?**

❑ **Database abstraction?**

- Hide the complexity from users.
- **How?:** By separating concerns across different layers.



Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

Levels of Abstraction



❑ *Physical Level (Internal Schema)*

- Describe how the data are actually stored. It include details like file format, indexing, storage structure (e.g. B-Trees, Hash indexing etc.).
- **Purpose:** focus on optimization of storage and retrieval.
- **Example:** Data stored as binary files on an SSD with specific indexing techniques..
- **User Type:** Database administrators and system designers.

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

Levels of Abstraction



❑ *Logical Level (Conceptual Schema)*

- Describes what data is stored in the database and the relationships between data elements. Independent of physical storage details
- **Purpose:** Provides a unified view of the entire database for database designers..
- **Example:** Entities like Student, Course, and Enrollment with their attributes and relationships.
- **User Type:** Database designers and architects.

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

Levels of Abstraction



❑ View Level (External Schema)

- Describes how data is presented to specific users or groups. Hides unnecessary details and shows only what is relevant to the user.
- **Purpose:** Provides customized views to meet the needs of different users.
- **Example:** A student sees only their grades, while a professor sees grades for all students in their course.
- **User Type:** End-users, application developers.

Data Independence



- ❑ The ability to change the schema at one level of a database without affecting higher levels.
 - Ensures flexibility and adaptability.

❑ Types

- Logical Data Independence (conceptual level)
- Physical Data Independence (Internal Level)

Data Independence



□ Logical Data Independence

- The ability to modify the logical schema (conceptual level) without affecting the external schema (user views) or applications.
 - Adding a new column, changing structure of a table.
 - Application will be unaffected by schema changes.

Data Independence



□ Physical Data Independence

- The ability to modify the physical schema (internal level) without affecting the logical schema (conceptual level) or applications.
 - Moving data to different storage medium (SSD → Cloud).
 - Change indexing method.
 - Optimization at storage level without disturbing the logic or user interface.

Importance of Data Independence



❑ *Simplifies Maintenance*

- Changes can be localized to specific schema levels without affecting the entire system.

❑ *Flexibility*

- Easier to adapt to new requirements or optimize performance.

❑ *Scalability*

- Supports modifications needed as the database grows.

❑ *Data Abstraction:*

- Separates how data is stored from how it is accessed or viewed.

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

Real World Example



❑ *University Database*

- Physical Schema:
 - Data Stored in SSD with indexes for performance.
- Logical Schema:
 - Tables: Students, Courses, Enrollments.
 - Attributes: StdID, Name, CourseID, Grade
- External Schema
 - Students: view grades
 - Professor: view performance of their course.

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

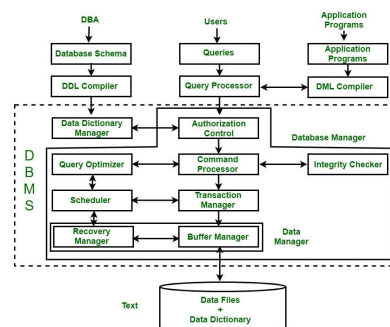
Real World Example

❑ University Database

- Logical Change
 - Adding new department table, adding new column i.e. phone number.
- Physical Change
 - Switching storage from local SSD to cloud.

Structure of DBMS

- ❑ The structure of a DBMS is designed to efficiently manage data storage, retrieval, and manipulation.
- ❑ It is typically organized into several layers, each responsible for specific tasks, ensuring data consistency, security, and abstraction.



DBMS Structure



❑ *User Level:*

- Interfaces for users to interact with the database.
- **Components:**
 - **Application Programs:** Custom applications developed for specific tasks.
 - **Query Tools:** SQL queries, reporting tools.
 - **User Interfaces:** Forms, dashboards, and GUIs.

DBMS Structure



❑ *Query Processor:*

- Interprets user queries and translates them into database operations.
- **Components:**
 - **DDL Compiler:** Processes Data Definition Language (DDL) commands to define schema.
 - **DML Compiler:** Processes Data Manipulation Language (DML) commands for CRUD (Create, Read, Update, Delete) operations.
 - **Query Optimizer:** Optimizes queries for efficient execution.

Components of DBMS Structure



❑ *Storage Manager:*

- Manages data storage and retrieval, ensuring security and consistency.
- **Components:**
 - **Authorization Manager:** Enforces user permissions.
 - **Transaction Manager:** Handles transactions to ensure ACID (Atomicity, Consistency, Isolation, Durability) properties.
 - **File Manager:** Manages physical files on storage devices.
 - **Buffer Manager:** Handles data caching between main memory and storage.

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

Components of DBMS Structure



❑ *Database Engine:*

- The core component that executes queries and performs database operations.
- **Key Functions:**
 - Indexing for fast data access.
 - Query execution plans.
 - Ensuring data consistency and concurrency.

Dr. Dev Narayan Yadav

Department of CSE, National Institute of Technology Rourkela

Components of DBMS Structure



❑ *Data Storage:*

- The physical layer where data is stored.
- **Components:**
 - **Primary Storage:** Volatile memory (RAM) for temporary data.
 - **Secondary Storage:** Persistent storage (HDDs, SSDs).
 - **Indexes:** Structures for faster data retrieval.

DBMS Architecture



❑ *Single-Tier Architecture:*

- Direct interaction between user and database.
- Database resides on the user's machine.
- **Use Case:**
 - Small-scale or personal applications.

DBMS Architecture



❑ *Two-Tier Architecture:*

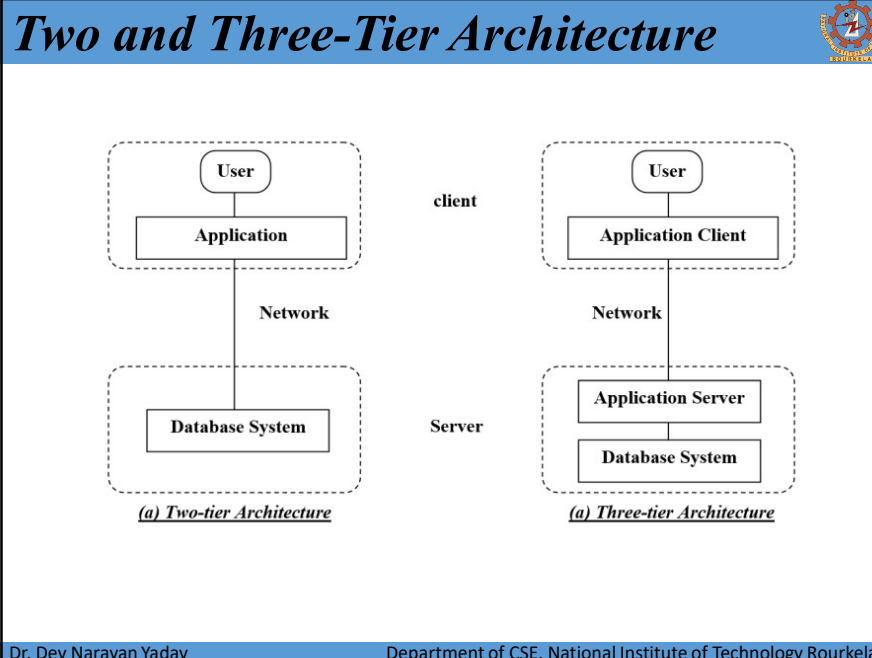
- Client-server model where the client interacts with the DBMS server.
- **Components:**
 - **Client:** User interface and application logic.
 - **Server:** Processes database queries.
- **Use Case:**
 - Medium-scale applications like desktop applications.

DBMS Architecture



❑ *Three-Tier Architecture:*

- Adds a middle layer between the client and the database server.
- **Components:**
 - **Presentation Layer:** User interface.
 - **Application Layer:** Business logic and query processing.
 - **Database Layer:** Data storage and retrieval.
- **Use Case:**
 - Large-scale, distributed applications like web-based systems.



Important Question

- ☐ Why database?
- ☐ Difference between file system vs database?
- ☐ Limitations of file system.
- ☐ Level of abstraction and its need?
- ☐ What do you mean by data independence?
- ☐ Roles of database administrator?