

# Week 1 Notes - DBMS

---

*Prof. Partha Pratham Das, IIT KGP*

*Notes by Adarsh (23f2003570)*


## Course Overview (00:25:39)

---

1. DBMS contains information of an enterprise
  - i. It can be RDBMS, or Document DB
2. RDBMS: Oracle, MSSQL, MySQL, PostgreSQL, IBM DB2, SQLite, MariaDB, CockroachDB
3. Problems with data in file system:
  - i. inconsistencies, redundancies
  - ii. slow to access, find data
  - iii. lack of data integrity constraints (you have to write it in code)
    - a. hard to maintain integrity constraints
  - iv. data isolation: hard to separate data and keep them in consistent
  - v. Unavailability of ACID properties
    - a. Atomicity
    - b. Consistency
    - c. Isolation
    - d. Durability
  - vi. Every file system has a limit on the size of the file:
    - a. NTFS has 16TB file size bound, 256TB Volume Size
    - b. FAT32 is 4GB, 2TB Volume Size
    - c. exFAT is 16 EB  $10^{18}$  bytes, 128PB Volume Size
    - d. This means there is an upper bound on the rows a file can have. DBMS does not seem to have this issue..virtually unlimited rows can be had in a DBMS subject to memory and volume size.. You need to choose the FileSystem carefully
  - vii. Advantages of DBMS over FS
    - a. Ease of recovering data
    - b. Faster (due to indexing)
    - c. No headache of concurrency
    - d. Data Consistency
    - e. Easier Security and Access Control
    - f. Transaction Control !!! Remember all or nothing ?
  - viii. Requirements:
    - a. Set Theory

- b. Demorgans law, union, intersection, complement, difference, cartesian product
- c. Membership: reflexive, symmetric, asymmetric, transitive, total
- d. Functions: injective, surjective, bijective, composition, inverse
- e. Proposition Logic
  - a. truth tables
  - b. conjunction (and), disjunction (or), negation (not), implication, equivalence
  - c. Closure under operations
- f. Predicate logic
- g. predicates
- h. quantification
  - a. existential
  - b. universal
- i. Data Structures
- j. Array
- k. List
- l. Binary Search Tree
  - a. Balanced Tree
- m. B-Tree
- n. HashMap/Map
- o. Object Oriented Analysis and Design

Week No.	Topics	
Week 1	Course Overview, Introduction	
Week 2	Basic Structured Query Language	
Week 3	Advanced Structured Query Language	
Week 4	Relational Algebra, Entity Relationship Model	
Week 5	Normal Forms and Functional Dependency	
Week 6	Normal Forms and Functional Dependency	
Week 7	Application Development	
Week 8	Storage Management	
Week 9	Indexing and Hashing	
Week 10	Transactions	
Week 11	Backup and Recovery	
Week 12	Query Optimization, Conclusion	

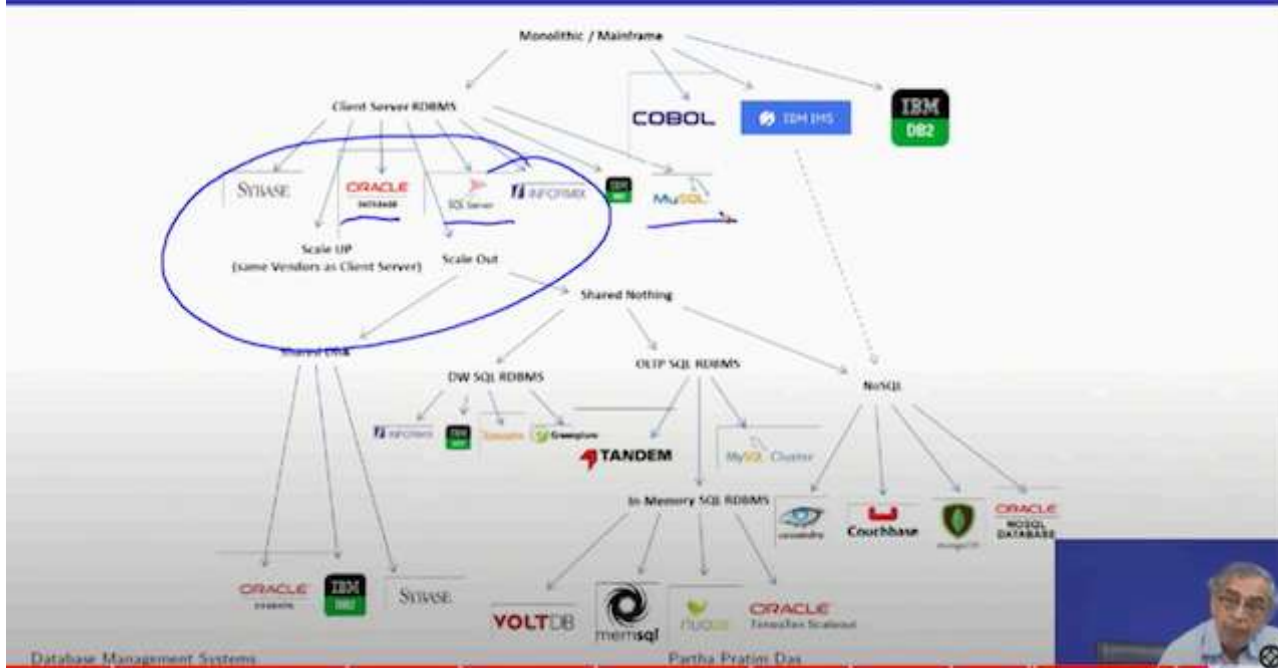


Application Programmer  
DBA / Designer

## Why DBMS/1 (00:28.57)

Electronics Data Management Parameters

1. Durability
2. Scalability
3. Security
4. Retrieval
5. Ease To Use
6. Consistency
7. Efficiency
8. Cost



## Why DBMS/2 (00:29.02)

Github: CSV based Banking system

## File Vs DBMS

Comparison		
Parameter	File Handling via Python	DBMS
<u>Scalability</u> with respect to amount of data	Very difficult to handle insert, update and querying of records	In-built features to provide high scalability for a large number of records
<u>Scalability</u> with respect to changes in structure	Extremely difficult to change the structure of records as in the case of adding or removing attributes	Adding or removing attributes can be done seamlessly using simple SQL queries
<u>Time of execution</u>	In seconds	In milliseconds
<u>Persistence</u>	Data processed using temporary data structures have to be manually updated to the file	Data persistence is ensured via <u>automatic, system induced mechanisms</u>
<u>Robustness</u>	Ensuring robustness of data has to be done manually	Backup, recovery and restore need minimum manual intervention
<u>Security</u>	Difficult to implement in Python (Security at OS level)	User-specific access at database level
<u>Programmer's productivity</u>	Most file access operations involve extensive coding to ensure persistence, robustness and security of data	Standard and simple built-in queries reduce the effort involved in coding thereby increasing a programmer's throughput
<u>Arithmetic operations</u>	Easy to do arithmetic computations	Limited set of arithmetic operations are available
<u>Costs</u>	Low costs for hardware, software and human resources	High costs for hardware, software and human resources

## Introduction to DBMS/1 (00:24.27)

## Levels of abstraction

1. Physical Level defines how the record is stored on Hard disk. Basically Data Structures at the core
2. Logical Level defines how data is stored in DB. Defines relationship, fields
3. View Level:
  - i. Hides datatype
  - ii. Hides information like salary
4. Schema and Instance
  - i. Schema is the way the data is organized
  - ii. Data is the value of the data
  - iii. Schema:
    - a. Physical Schema: Overall physical structure of DB
    - b. Logical Schema: overall logical structure of DB
      - a. ex: Customer schema: {name, customer\_id, account\_id, aadhar\_id, mobile}
      - b. ex: Account Schema: {account\_number, account\_type, interest\_rate, balance}
    - c. Looks like schema is the definition of the table... `create table` . It is the structure of the table
  - iv. Instance is the actual values of the table... from the lecture it appears to be collection of the table records.
  - v. instances can be added/removed but schema remains the same unless `alter table`
  - vi. `Attributes` are field names in a schema.. like `account_number`, `account_type`

## 1. Physical Level (Lowest Level of Abstraction)

- **Description:** The physical level describes **how the data is actually stored** in the database. It involves complex data structures and file organization methods used to store data on storage devices, such as hard drives or SSDs.
- **Focus:** It focuses on the **efficiency** and **optimization** of data storage and retrieval, ensuring that the database runs smoothly and handles large volumes of data.
- **Example:** At this level, data could be stored in the form of index files, B-trees, or hash functions, and the system defines how to store and access these files on disk.

## 2. Logical Level (Middle Level of Abstraction)

- **Description:** The logical level defines **what data is stored** in the database and the **relationships among those data**. It describes the database structure as a whole in terms of tables, records, and fields, without going into details of how these elements are physically stored.
- **Focus:** This level focuses on the **structure of the entire database** as seen by the database administrators and developers. It involves the design of schemas that define entities, attributes, and relationships, typically using data models like the relational model.
- **Example:** A table named "Employees" with columns such as "EmployeeID", "Name", and "Salary" is part of the logical schema. It defines the content and structure of data but not how it is stored.

### 3. View Level (Highest Level of Abstraction)

- **Description:** The view level describes **only part of the entire database** that is relevant to a particular user or group of users. It provides multiple perspectives or "views" of the database, ensuring that users can access the data they need without seeing the complexity of the database.
- **Focus:** It focuses on **user interaction** and **data security** by restricting access to only the necessary data. Different users might have different views based on their role and permissions, such as salespeople seeing customer information without access to sensitive employee data.
- **Example:** A bank manager might have access to a view showing customer balances and transaction histories, while a bank teller might only see customer names and account numbers.

#### Summary of the Three Levels

- **Physical Level:** How data is stored physically (internal storage).
- **Logical Level:** What data is stored and its structure (conceptual structure).
- **View Level:** How data is presented to users (external views).

This separation of abstraction levels ensures **data independence**, meaning that changes to the physical storage don't affect the logical structure, and changes to the logical structure don't affect the views seen by users.

#### Logical Schema vs. Physical Schema:

1. **Logical Schema:** Defines the structure of the database at a logical level. It includes the tables, relationships, constraints, and other elements that define how data is organized and related to each other.
2. **Physical Schema:** Refers to how the data is stored on the storage medium. It involves details like file structures, indexing, and access paths.

- **Instance**

- The actual content of the database at a particular point in time
- Analogous to the value of a variable

Name	Customer ID	Account #	Aadhaar ID	Mobile #
Pavan Laha	6728	917322	182719289372	9830100291
Lata Kala	8912	827183	918291204829	7189203928
Nand Prabhu	6617	372912	127837291021	8892021892

- Customer Instance
- Account Instance

Account #	Account Type	Interest Rate	Min. Bal.	Balance
917322	Savings	4.0%	5000	7812
372912	Current	0.0%	0	291820
827183	Term Deposit	6.75%	10000	100000





# Data Independence

## 1. Physical data independence

- i. What Prof. Das is saying is you should be able to modify the physical underlying file format and apply migration without setting the logical table on fire. Like a new version of a DBMS could have a better performing underlying physical file format
- ii. He is saying any changes to the Physical schema should not change the logical level or view level (as a consequence of logical independence) of the DBMS
- iii. Physical data independence allows database administrators to optimize storage and access methods (e.g., through indexing, partitioning, or clustering) to improve performance without needing to modify how applications interact with the database.

## 2. Logical Data Independence says that if you modify the Logical Level, the View should not change

- i. Logical data independence ensures that changes made to the logical schema, such as adding new fields, changing data types, or modifying relationships, do not necessitate changes to the application programs that use the data.
- ii. logical data independence is a fundamental principle that ensures flexibility and stability in database design and management, allowing changes to be made to the logical schema without affecting the applications and users relying on the data.

## 1. Physical Data Independence

- **Definition:** Physical data independence refers to the ability to change the **physical schema** without affecting the **logical schema** (and by extension, the application programs or users interacting with the data). In other words, changes to how the data is stored (e.g., storage structures, access methods, or file organization) do not require changes to the logical structure of the database.
- **Importance:** This is essential because administrators may need to optimize or modify how data is stored (for example, by changing indexing methods, compressing data, or switching to a new storage device) to improve performance or efficiency, but these changes should not affect the logical design of the database.
- **Example:** Suppose a company changes the way it physically stores employee records from a sequential file format to an indexed file format. As long as the logical structure (e.g., the Employee table with columns for EmployeeID, Name, Salary) remains unchanged, applications querying the Employee table should continue to work without modification.

## 2. Logical Data Independence

- **Definition:** Logical data independence refers to the ability to change the **logical schema** without having to change the **external schema** or user views. In other words, changes to the logical design of the database (such as adding new fields or tables, changing relationships, etc.) should not require changes to how users or applications access the data.

- **Importance:** This is crucial because organizations often need to modify their data models (e.g., adding new fields to accommodate new business requirements) without affecting existing applications that depend on specific views of the data.
- **Example:** A company may decide to add a new attribute, "PhoneNumber", to the Employee table. With logical data independence, existing applications that access only "EmployeeID" and "Name" will not be affected, even though the logical schema has changed.

## Summary

- **Physical Data Independence:** Changes to the **physical storage** of data do not affect the **logical schema**.
- **Logical Data Independence:** Changes to the **logical schema** do not affect the **user views** (external schema) or application programs.

## Importance of Data Independence

- **Flexibility:** Data independence allows database systems to be flexible and adapt to changing requirements without significant disruption to the existing system.
- **Cost-Efficiency:** It reduces the cost and effort required to modify the database when performance optimizations or new features are added.
- **Maintenance:** Database administrators and developers can focus on maintaining different parts of the system (physical storage, logical structure, or user views) independently, making database management more straightforward.

By promoting separation between levels of abstraction (physical, logical, and view), data independence ensures that the database system remains scalable, maintainable, and adaptable to future needs.

## Data Model

Is a description of the following

1. Data (fields, attributes, data types)
2. Data relationships (like how is this table related to other tables?)
3. Data semantics (like meaning of this table)
4. Data Constraints (like what kind of values can a field/attribute take?)
5. Relational Algebra !!

## DDL and DML

1. Data Definition Language
  - i. Basically the notation to define schema
  - ii. `create table` uses it
  - iii. primary/foreign keys are defined in it



- iv. You can also add ACL (Access Control List) to decide who can read/write these tables.
- v. integrity constraints
- vi. DDL compiler generates a set of table templates stored in Data Dictionary .
- vii. All these schema information is also stored in a table. this table is called Data Dictionary
- viii. Data Dictionary contains Database Schema, Integrity Constraints, Authorizations and ACL's

## 2. Data Manipulation Language or Query Language

- i. insert, update, delete, select
- ii. Two classes of language
  - a. Pure basically its some mathematical way of querying the DB
    - a. Relational Algebra
    - b. Tuple relational Algebra
    - c. Domain relational Algebra
    - d. This could be fast, but hard to craft
    - e. These are Pure Query Languages
  - b. Commercial
    - a. Something like SQL (Structured Query Language)
    - b. SQL is a DDL and a DML as it can create database schemas and also query the database tables.

## SQL (Structured Query Language)

- 1. SQL is not turing complete language like C, C#
- 2. SQL could be embedded in High Level Languages
- 3. You can use a JDBC/ODBC driver.. these drivers use the Pure Query syntax

## Good design of DB relations and bad designs

- 1. Basically Prof. Das is talking of not duplicating data by separating entities and keeping a relation between them
- 2. Normalizing a DB (Normalization Theory)

## Introduction to DBMS/2 (00:29.12)

---

## Database Design

Database design has 2 components

- 1. Logical Design
  - i. What is the schema?
  - ii. What are the attributes/fields? What data-type ?

iii. What relational schema ?

- a. Flat schema ?
- b. How to normalize it?
- c. FK, PK ? Indices ?

## 2. Physical Design

- i. How do you layout the files, the b-trees, transactions in memory

The Database engine can change the physical design while retaining the logical design. This could be possible by physical design conversion.

Database Migration refers to changing the Logical Design Schema and/or migrating old data

## Good or Bad design ?

Database Design (2)

• Is there any problem with this relation?

ID	name	salary	dept_name	building	budget
22222	Einstein	95000	Physics	Watson	70000
12121	Wu	90000	Finance	Painter	120000
32343	El Said	60000	History	Painter	50000
45565	Katz	75000	Comp. Sci.	Taylor	100000
98345	Kim	80000	Elec. Eng.	Taylor	85000
76766	Crick	72000	Biology	Watson	90000
10101	Srinivasan	65000	Comp. Sci.	Taylor	100000
58583	Califert	62000	History	Painter	50000
83821	Beardt	92000	Comp. Sci.	Taylor	100000
15151	Mozart	40000	Music	Packard	80000
33456	Gold	87000	Physics	Watson	70000
76543	Singh	80000	Finance	Painter	120000

1. look's like dept\_name and building are categorical variables, budget is numeric and all three are redundant. Redundant means a tuple of all the three are identical and repeated!

i. Also looks like the above variables are tuples.

ii. can you group all these 3 fields/attributes into a single table?

2. Can ID be indexed for faster retrieval?

3. You need to factor it 2 tables.

i. Two ways to do this

- Entity Relationship Model (ER Model)
- Normalization Theory

## Entity Relation Model

1. used for planning and designing the logical layer of the DBMS

## Object Relational Data Models

1. Relational Model: this applies to RDBMS. They are flat, and composed of primary types, atomic values (atom values like char, varchar, int, datetime, string).

i. Do not compose it with a composite type 🚫

ii. In DocumentDB's you can have composite types

2. Object Relational Model: These are models that you write in your code. They map to Relational Models.

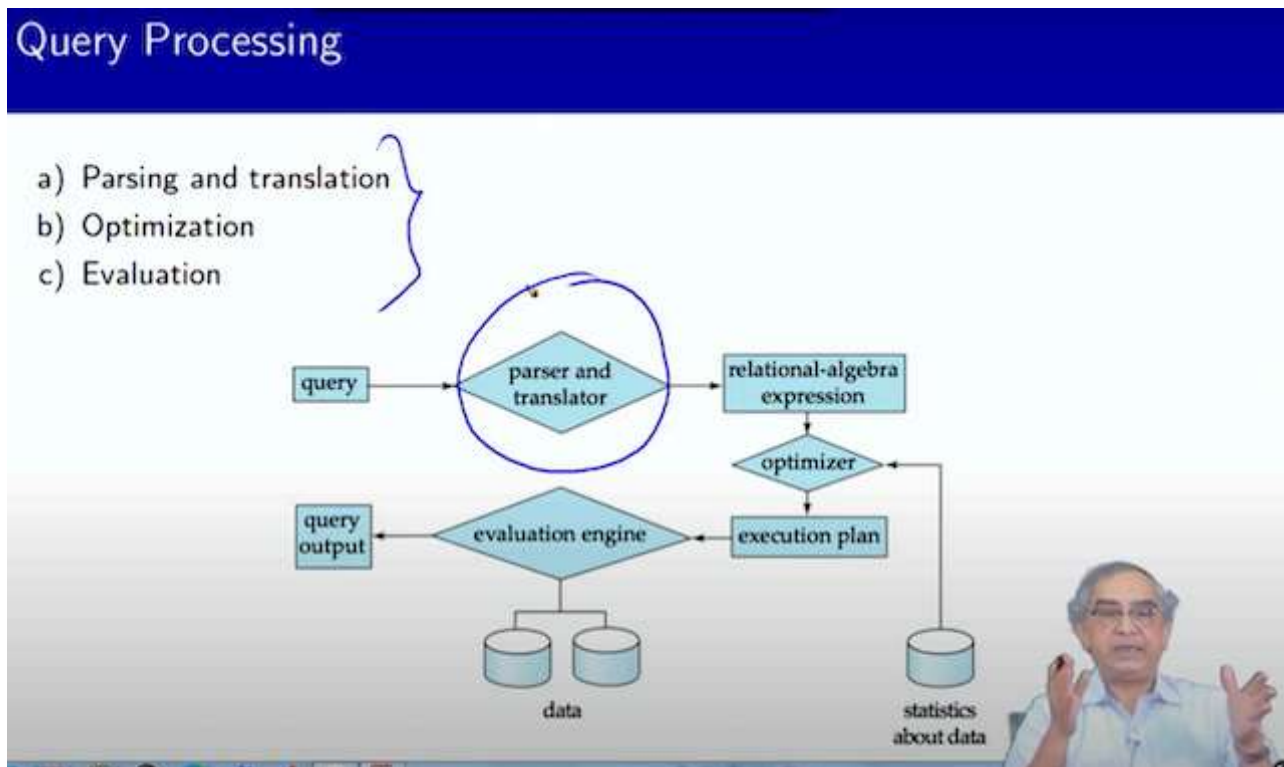
- i. These can have complex/composite types. Like Person, Company.
- ii. Preserves the basic Relational Model foundation.

## Storage Management

1. This is a layer, or interface that does Physical Level jobs.
2. uses the Operating System File calls to efficiently store and retrieve data
  - i. efficiency means block writes (after transaction) are preferred they are faster of the disk is a magnetic seeking disk or if it's a SSD, it would improve the life (lesser writes is a happy SSD).. ignore this
  - ii. Issues (just ignore for now, will be covered later)
    - a. Storage Access
    - b. File Organization
    - c. Indexing and Hashing

## Query Processing

1. Parsing and translation
2. Optimization
3. Evaluation



1. Whatever query language you use (SQL or say a custom one) has to be translated to the Pure /mathematical/relational algebra expression.
2. The relative algebra expression is optimized and an execution plan is made.

3. Evaluation Engine basically records execution plan vs execution timing for the Statistics about query
  - i. Optimizer uses Statistics about query engine to chose Execution Plan Strategy

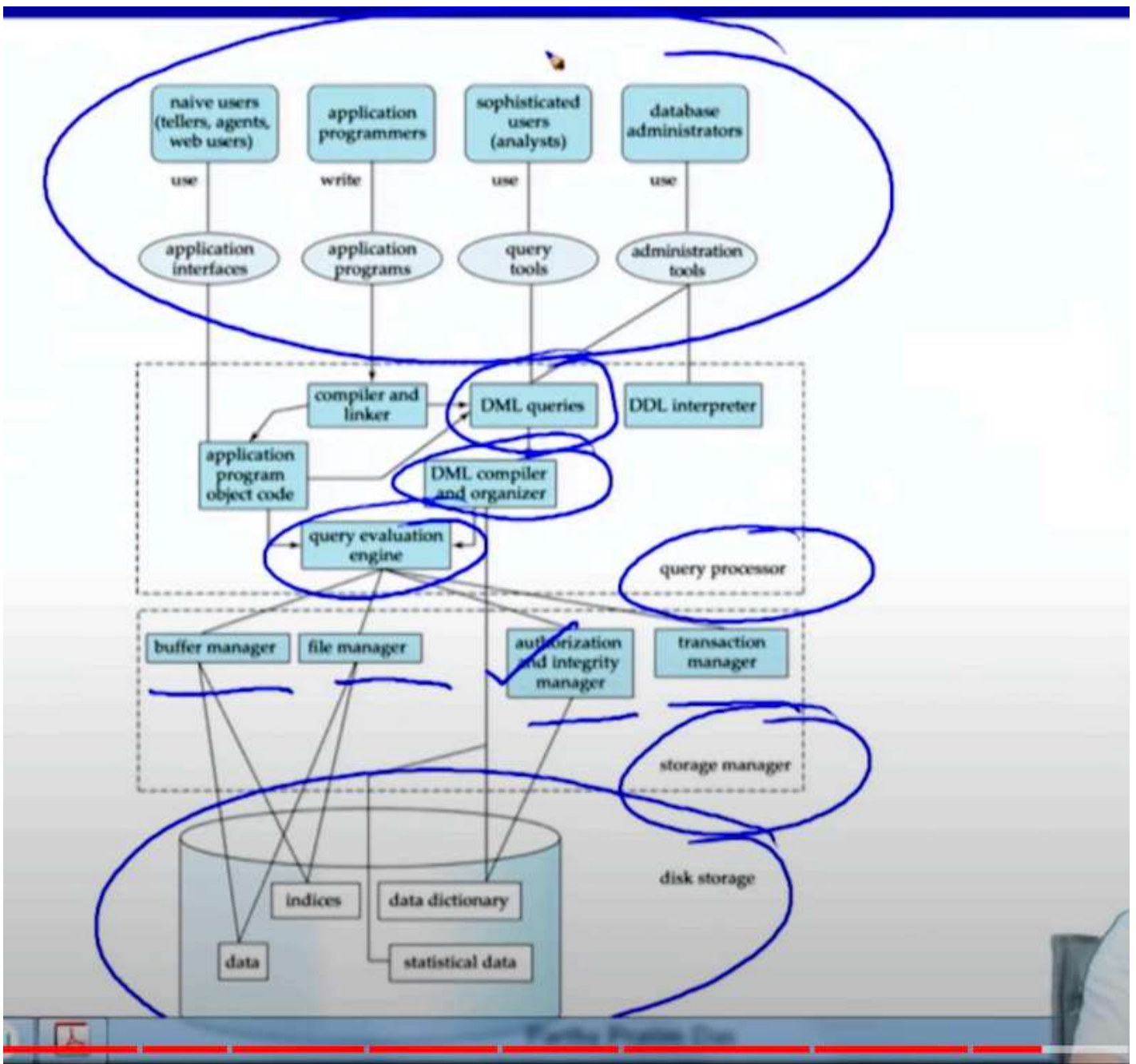
## Transaction Management

1. Transaction is a collection of operations, where success is achieved if all the operations succeed.
  - i. If even 1 operation fails, the transaction fails and a rollback is done. The state of the record is NOT changed
  - ii. If all operations within the transaction block succeeds, then the state of the record is modified.
  - iii. Transaction ensures that your DB is always consistent.
  - iv. Think of it as "All or Nothing"

## Concurrency Transactional Manager

1. Controls concurrent transactions
  - i. Uses Concurrency Control Mechanisms of the host OS.
  - ii. Makes sure that your data is consistent!
  - iii. Transactions include multiple R-DB tables!
  - iv. Please research further on this concept, because the same concept can turn up in distributed systems (like in your field of work)
  - v. Prof: uses the word `serialize` in the train case. He says
    - a. A gets the berth or B gets the berth - but not both A and B
    - a. Somehow lock the **entity** in question!
    - b. So the R-DB system somehow manages to achieve the above. This is Concurrency Control

## Database Systems Internal



## Graph Databases

1. Designed to model and store relationships directly as first-class citizens, allowing for fast and efficient querying of complex relationships. Operations like traversing relationships are optimized for performance.
2. They are Schema-less or have a flexible schema, allowing you to easily add new types of relationships and nodes without disrupting existing data or requiring a significant schema redesign
3. Perform complex queries involving multiple levels of relationships (e.g., finding friends of friends) efficiently. Query performance tends to be consistent and predictable, even with deeply nested relationships. Good luck with RDBMS
4. Handle complex and highly interconnected data structures with ease.
5. Can scale effectively to handle large volumes of interconnected data and relationships, maintaining performance even as the data grows.

# XML - Extensible Markup Language

1. Defined by WWW (W3C)
2. Hierarchical in nature
3. You can define your own tags!
4. Can be queried using forward XML parsing, XML DOM load, XPATH
5. Text based - so easy to transmit across internet
6. Serializable - Objects to XML and vice versa

## Parts of a Database

---

### 1. Hardware

- **Description:** The physical devices used in a database system, such as computers, storage devices (e.g., hard drives, SSDs), servers, and network equipment.
- **Role:** Hardware provides the **infrastructure** to store and process the data and execute the necessary database operations.
- **Example:** The database server where the database is installed, the disk on which data files are stored, or a network interface used to connect clients to the database.

### 2. Software

- **Description:** The software includes the **Database Management System (DBMS)** itself, along with any related applications or utilities that help manage the database.
- **Components:**
  - **DBMS Software:** The core software responsible for managing the database, handling queries, ensuring transaction processing, maintaining data integrity, etc.
  - **Operating System:** The system software that interacts with the hardware and runs the DBMS.
  - **Database Applications:** Programs that interact with the database system for various tasks, like data entry or report generation.
- **Role:** Software handles the **data manipulation, query processing, storage management**, and ensures **data security** and **integrity**.
- **Example:** Oracle, MySQL, PostgreSQL, or SQL Server are examples of DBMS software. An application like a CRM tool might use the DBMS to store customer data.

### 3. Data

- **Description:** The most critical component of a database system, data refers to the actual information stored within the database, which can be processed, analyzed, and retrieved.
- **Structure:** The data is usually organized in a specific format (e.g., tables in a relational database) based on the **database schema**.



- **Types of Data:**
  - **User Data:** The actual business data, such as customer details, orders, transactions, etc.
  - **Metadata:** Data about the data, such as table definitions, column types, constraints, etc.
  - **Indexes:** Additional structures that improve the speed of data retrieval.
- **Role:** Data is at the core of the system and is the main reason for the system's existence.
- **Example:** In an employee database, the actual records of employees (names, IDs, salaries) are the user data, while the schema defining the structure (tables, fields) is the metadata.

## 4. Users

- **Description:** The users are people or systems that interact with the database. They can be classified based on their roles and the tasks they perform.
- **Types of Users:**
  - **Database Administrators (DBAs):** Responsible for managing the database system, including database design, performance tuning, security, backup, and recovery.
  - **Database Designers:** Define the structure of the database and design the schema that supports the application requirements.
  - **End Users:** People who interact with the database via applications, usually without knowing the details of how the data is stored or managed.
  - **Application Programmers:** Developers who write programs that interact with the database to fetch or modify data.
- **Role:** Users interact with the database to **query, modify, design, or maintain** the database, depending on their roles.
- **Example:** A sales manager using an inventory management system is an end user, while a database administrator may manage backups and performance tuning.

## 5. Procedures

- **Description:** Procedures refer to the set of **rules, instructions, or protocols** that guide the design, use, and maintenance of the database. These procedures ensure the database operates efficiently and securely.
- **Role:** Procedures ensure that the database system functions according to policies, including **backup protocols, recovery procedures, security policies, and database design guidelines**.
- **Example:** A procedure could involve the steps for performing a regular backup of the database or protocols for how users should query the database without causing performance issues.

## 6. Database Languages

- **Description:** Database languages are used to interact with the database system. The most common language is SQL (Structured Query Language), which allows users to query and manipulate data.
- **Types of Database Languages:**

- **Data Definition Language (DDL):** Defines the database structure, such as creating, altering, and deleting tables (e.g., `CREATE` , `ALTER` , `DROP` commands).
- **Data Manipulation Language (DML):** Handles the retrieval, insertion, modification, and deletion of data (e.g., `SELECT` , `INSERT` , `UPDATE` , `DELETE` commands).
- **Data Control Language (DCL):** Manages access control, specifying who can access and manipulate data (e.g., `GRANT` , `REVOKE` commands).
- **Transaction Control Language (TCL):** Manages transaction behavior (e.g., `COMMIT` , `ROLLBACK` ).
- **Role:** These languages allow users and administrators to interact with the database, modify its structure, query data, and manage access.
- **Example:** A `SELECT` query to retrieve employee data or a `CREATE TABLE` command to create a new table in the database.

## 7. Database Schema

- **Description:** The database schema defines the logical structure of the database, describing how the data is organized and the relationships between different pieces of data.
- **Types of Schemas:**
  - **Physical Schema:** Describes the physical storage of data (how data is stored in memory or on disk).
  - **Logical Schema:** Describes the structure of the data (e.g., tables, fields, and relationships).
  - **View Schema:** Defines different views of the data for various users or applications.
- **Role:** The schema serves as a blueprint for how the database is structured and managed, providing the framework for both data storage and retrieval.
- **Example:** The schema could define a relational model with tables for "Employees", "Departments", and "Salaries", along with the relationships between them.

## 8. Query Processor

- **Description:** The query processor translates user queries (typically written in SQL) into efficient execution plans that the database system can use to retrieve or modify data.
- **Components:**
  - **Query Parser:** Parses and checks the syntax of the user's query.
  - **Query Optimizer:** Optimizes the query by generating an efficient execution plan, considering factors like indexes, available memory, and data distribution.
  - **Execution Engine:** Executes the query based on the optimized plan and retrieves or modifies data as requested.
- **Role:** The query processor ensures that user queries are processed efficiently and accurately.
- **Example:** When a user submits a query like `SELECT * FROM Employees WHERE Department = 'HR'` , the query processor determines the best way to retrieve the data (e.g., using an index).

## 9. Storage Manager

- **Description:** The storage manager is responsible for managing how data is stored on disk, retrieved, and updated.
- **Components:**
  - **Buffer Manager:** Manages data in memory, minimizing the number of disk I/O operations.
  - **File Manager:** Manages space allocation on disk and keeps track of data files.
  - **Transaction Manager:** Ensures that database transactions are processed in a way that maintains data consistency and integrity, especially in cases of concurrent transactions.
  - **Authorization and Integrity Manager:** Enforces security and integrity constraints.
- **Role:** It handles the interaction between the **logical data model** and the **physical storage**.
- **Example:** When data is inserted or updated, the storage manager writes the changes to disk in an efficient manner and ensures they can be recovered if the system crashes..