Contents

- Introduction to DBMS
- DBMS?
- DBMS Applications
- Database systems vs. File Processing Systems
- Drawbacks of using file systems to store data
- Advantage of DBMS over File system

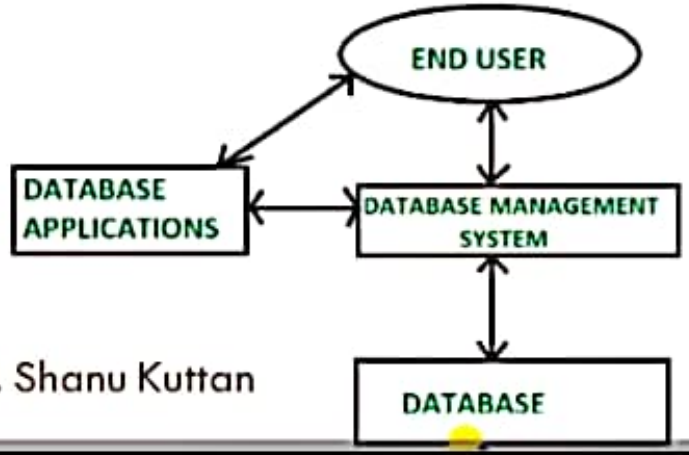
Introduction to DBMS

- Data: any fact that can be recorded
  - E.g. Text, Number, images, audio, video, speech, map
- Database (DB): Collection of interrelated data
  - Traditional Database (TDB): Text and Number
  - Multimedia Database (MDB): Video, speech, song, movie
  - Geographic Information System (GIS): Images of earth
  - Real-time Database (RDB): Production, Supermarket -varying products data
  - Data Warehouse (DW): Huge and Historical data
- Database Management System (DBMS): Set of Programs or software used to define, store and manipulate the database
- DB+DBMS=DBS (Database System)

by Prof. Shanu Kuttan

Database Management System (DBMS)

- DBMS contains information about a particular enterprise.
- DBMS is :
  - A Collection of interrelated data
  - A Set of programs to access the data
  - An environment that is both *convenient* and *efficient* to use
- DBMS stores data in such a way that it becomes easier to retrieve, manipulate and produce information



DBMS Applications

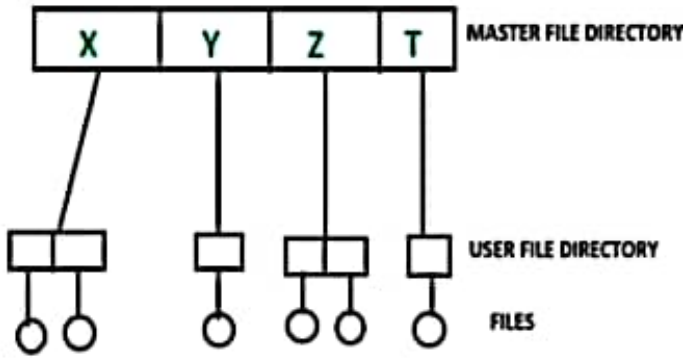
- Banking: all transactions
- Airlines: reservations, schedules
- Universities: registration, grades
- Sales: customers, products, purchases
- Manufacturing: production, inventory, orders, supply chain
- Human resources: employee records, salaries, tax deductions
- Online Retailers: Order tracking
- Many more..

Database System (DBS)

- Database can be very large and it touches all aspects of our lives.
- Database Systems (DBS) are designed to manage large bodies of information
- Management involves:
  - Defining structures for storage of information
  - Providing mechanisms for the manipulation of information
- In addition, DBS must ensures the safety of the information stored.

Database System vs. File Processing System

- In the early days, database application were built directly on top of file systems
- **1. File System :**
  - A File Processing system is a software that stores and manage files in computer hard-disk
    - It allows access to single files or tables at a time.
    - Data is directly stored in set of files.
    - It contains flat files that have no relation to other files.
    - File systems consists of different files which are grouped into directories. The directories further contain other folders and files.
    - E.g. NTFS(New Technology File System), EXT(Extended File System).



- **2. DBMS**
  - A Database Management System (DBMS) is a software that allows users to efficiently define, create, maintain and share databases.
    - E.g. Oracle, MySQL, MS SQL server, IBM DB2, MS Access, dBASE, FoxPro, SQLite, PostgreSQL, MariaDB

Drawbacks of using file systems to store data

- Data redundancy and inconsistency
  - Multiple file formats, duplication of information in different files
  - Data redundancy leads to data inconsistency
  - All copies may not be updated properly
- Difficulty in accessing data
  - Need to write a new program to carry out each new task
- Data isolation
  - data are scattered in multiple files and in different formats
  - difficult to write new application programs to retrieve the appropriate data
- Dependency on application program
  - changing files would lead to change in application programs
- Integrity problems
  - Data may be required to satisfy Integrity constraints (e.g. account balance > 0)
  - Difficult to add new constraints or change existing ones

by Prof. Shanu Kuttan

Continued…

- Atomicity of updates
  - Atomicity of a transaction refers to "All or nothing", which means either all the operations in a transaction executes or none
  - Failures may leave database in an inconsistent state with partial updates carried out
    - E.g. transfer of funds from one account to another should either complete or not happen at all
- Concurrent access by multiple users
  - Concurrent accessed needed for performance
  - Uncontrolled concurrent accesses can lead to inconsistencies
    - E.g. two people reading a balance and updating it at the same time
- Security problems
  - Data should be secured from unauthorised access
  - Hard to provide user access to some, but not all, data.

"Database systems offer solutions to all the above problems"

Advantage of DBMS over File system

- **No redundant data:** Redundancy removed by data **normalization.** No data duplication saves storage and improves access time.
- **Data Consistency and Integrity:** the root cause of data inconsistency is data redundancy, since data normalization takes care of the data redundancy, data inconsistency also been taken care of as part of it
- **Data Security:** It is easier to apply access constraints in database systems so that only authorized user is able to access the data. Each user has a different set of access thus data is secured from the issues such as identity theft, data leaks and misuse of data.
- **Privacy:** Limited access means privacy of data.
- **Easy access to data :** Database systems manages data in such a way so that the data is easily accessible with fast response times.
- **Easy recovery:** Since database systems keeps the backup of data, it is easier to do a full recovery of data in case of a failure.
- **Flexible:** Database systems are more flexible than file processing systems.

S.NO.	File System	DBMS
1.	File system is a software that manages and organizes the files in a storage medium (HDD) within a computer.	DBMS is a software for managing the database.
2.	Redundant data can be present in a file system.	In DBMS there is no redundant data.
3.	It doesn't provide backup and recovery of data if it is lost.	It provides backup and recovery of data even if it is lost.
4.	There is no efficient query processing in file system.	Efficient query processing is there in DBMS.
5.	There is less data consistency in file system.	There is more data consistency because of the process of normalization.
6.	It is less complex as compared to DBMS.	It has more complexity in handling as compared to file system.
7.	File systems provide less security in comparison to DBMS.	DBMS has more security mechanisms as compared to file system.
8	It is less expensive than DBMS.	It has a comparatively higher cost than a file system.

Contents

- View of Data
  - Levels of Abstraction
  - Schemas and instances
  - Data Independence
  - Three-Level DBMS Architecture

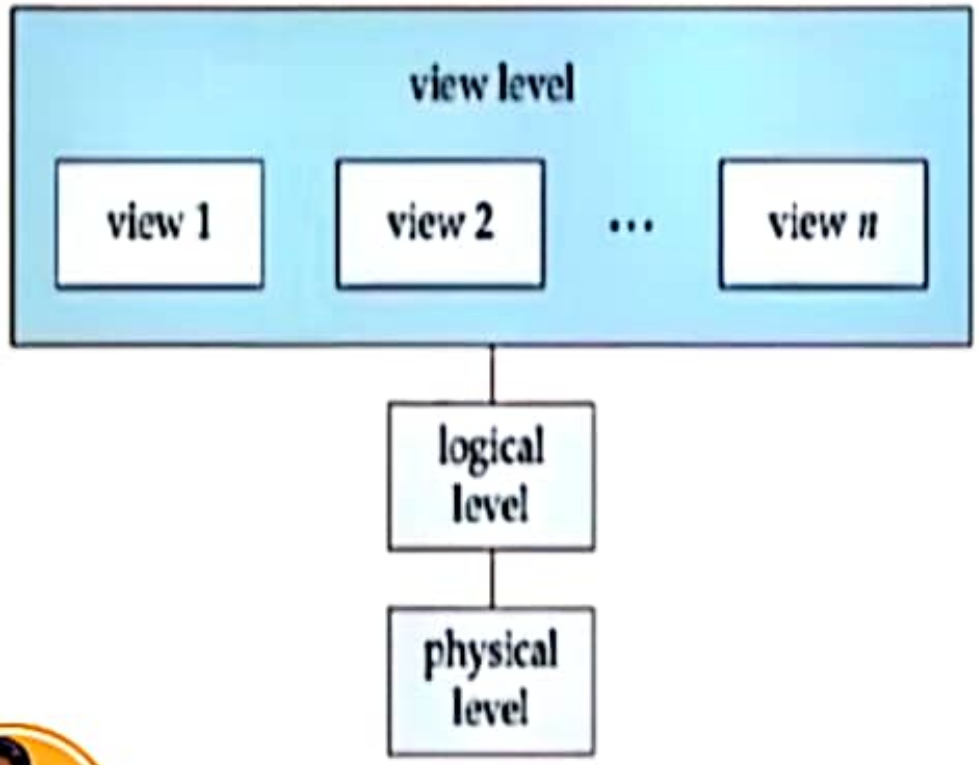
View of Data

- Database systems are made-up of complex data structures.
- *To ease the user interaction with database, the developers hide internal irrelevant details from users and* provides abstract view of data to users. *This process of hiding irrelevant details from user is called* ***Data Abstraction.***
- **Abstraction** is one of the main features of database systems.

Levels of abstraction

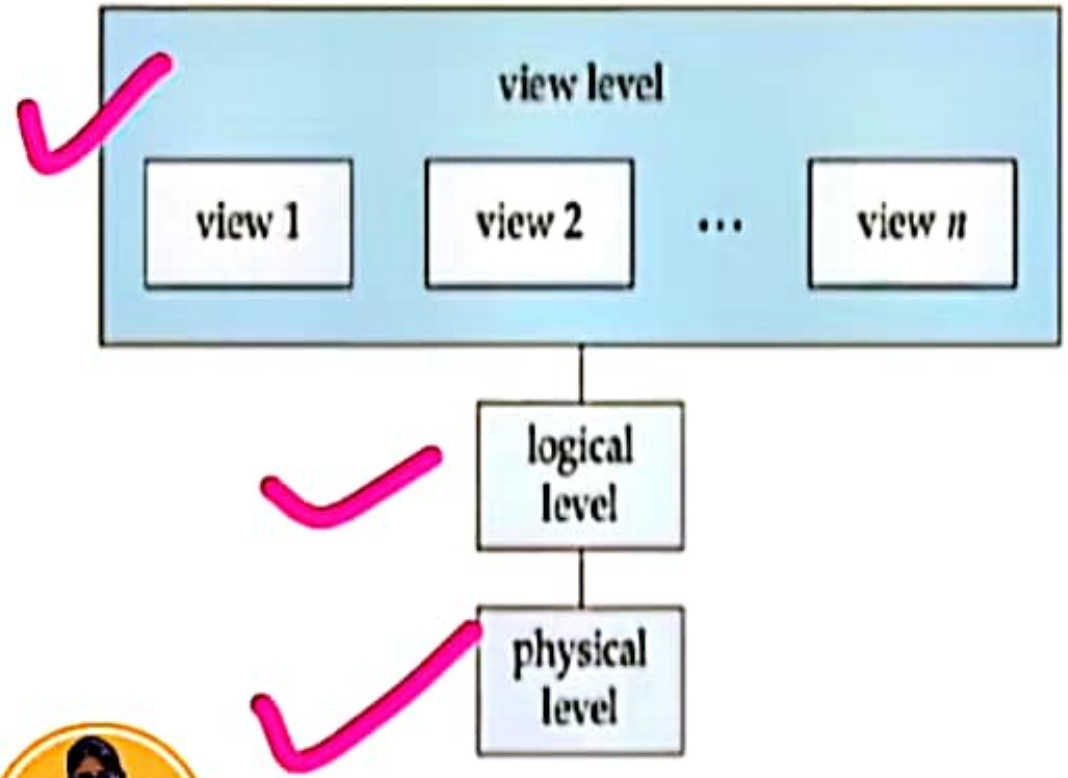
- **Physical level (Internal level):**
  - This is the lowest level of data abstraction.
  - It describes **how** data is actually stored in database. You can get the complex data structure details at this level.
- **Logical level (Conceptual level):**
  - This is the middle level of 3-level data abstraction architecture.
  - It describes **what** data is stored in database and **what** relationships exist among those data.
- **View level (External level):**
  - Highest level of data abstraction.
  - This level describes only part of the entire database i.e. it describes the user interaction with database system via application programs that hide details of data types.

Three Levels of data abstraction



- At **view level**, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.
- At the **logical level** these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented.
- The programmers generally work at this level because they are aware of such things about database systems.
- At **physical level** these records can be described as blocks of storage (bytes, gigabytes, terabytes etc.) in memory.
- These details are often hidden from the programmers.

Three Levels of data abstraction



- At **view level**, user just interact with system with the help of GUI and enter the details at the screen, they are not aware of how the data is stored and what data is stored; such details are hidden from them.
- At the **logical level** these records can be described as fields and attributes along with their data types, their relationship among each other can be logically implemented.
- The programmers generally work at this level because they are aware of such things about database systems.
- At **physical level** these records can be described as blocks of storage (bytes, gigabytes, terabytes etc.) in memory.
- These details are often hidden from the programmers.

BHANU KUTTAM CSE CLASSES

Example:

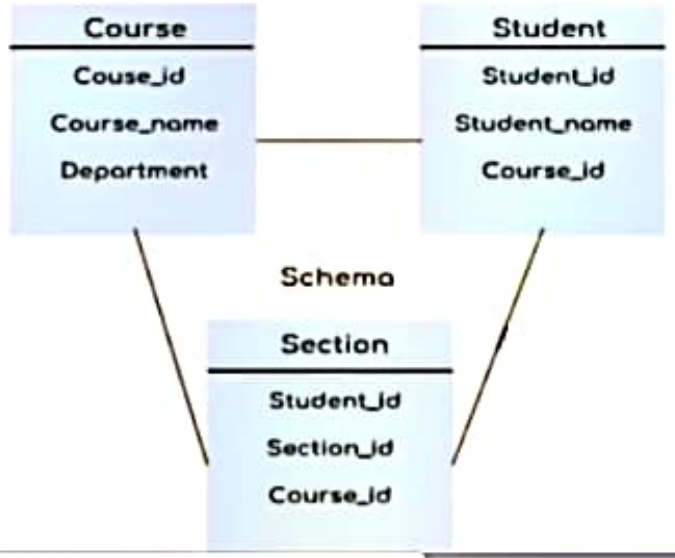
	Customer	Manager
External View	View1 AC_Name Amount	View2 AC_No AC_Name Type
Logical / Conceptual View	AC_No AC_Name Amount Type	Numeric(15) Character(20) Numeric(15) Character(10)
Physical/ Internal View	Stored-acc Account # Type	length =60 type-bytes(15) offset(10)

Schemas and Instances

- Schemas and Instance are similar to types and variables in programming languages

Schema: Overall design of a database is called the Schema

- For example: Given a schema shows the relationship between three tables: Course, Student and Section. The diagram only shows the design of the database, it doesn't show the data present in those tables. Schema is only a structural view (design) of a database as shown in the diagram below.



Schemas types...

- Schema is of three types:
  - Physical schema: It describes database design at physical level i.e. physical structure of database
    - how the data stored in blocks of storage is described at this level.
  - logical schema: It describes database design at logical level i.e. logical structure of database
    - **Programmers** and **Database Administrators** work at this level, at this level data can be described as certain types of data records gets stored in data structures, however the internal details such as implementation of data structure is hidden at this level (available at physical level).
  - Sub-schema (or view schema): It describes different views of database
    - This generally describes end user interaction with database systems.

Instances....

- **Instance**: The data stored in database at a particular moment of time is called instance of database.
  - For example, lets say we have a single table student in the database, today the table has 100 records, so today the instance of the database has 100 records. Lets say we are going to add another 100 records in this table by tomorrow so the instance of database tomorrow will have 200 records in table. In short, at a particular moment the data stored in database is called the instance, that changes over time when we add or delete data from the database.

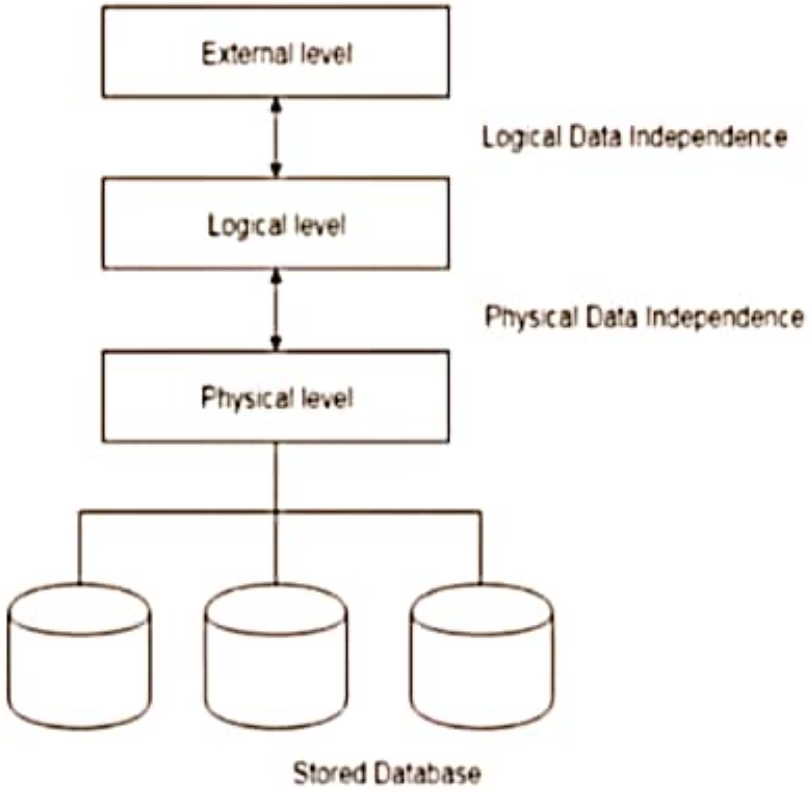
Database schema defines the ***variable declarations along with type definition*** in tables that belong to a particular database; the **value of these variables** at a moment of time is called the **Instance** of that database.

Data Independence

- **Data Independence:** The ability to modify the schema at one level of the database system without altering the schema at the next higher level.
- Two types of Data Independance:

  1. **Logical Data Independence:** ability to modify the Logical schema without changing the external view and application

  2. **Physical Data Independence:** ability to modify the Physical schema without changing the Logical schema



Three Level DBMS Architecture

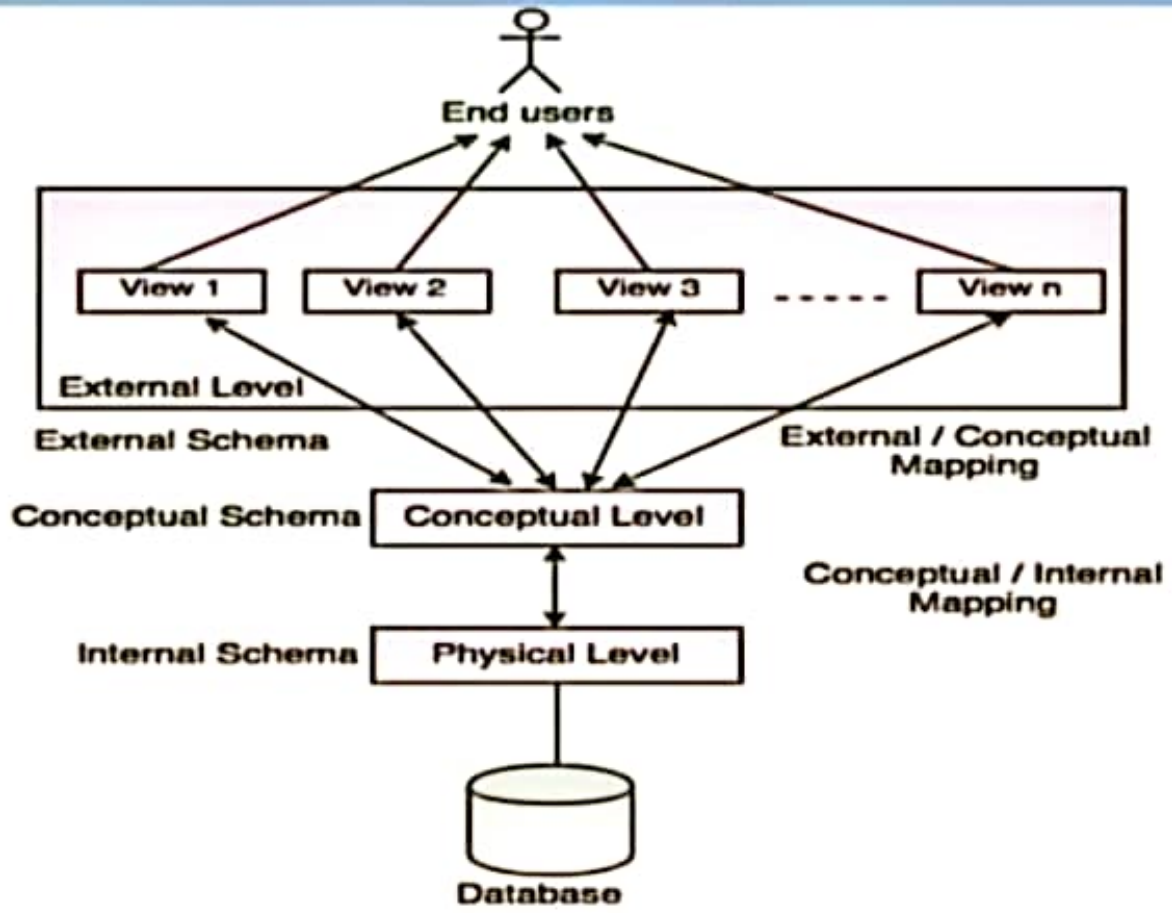


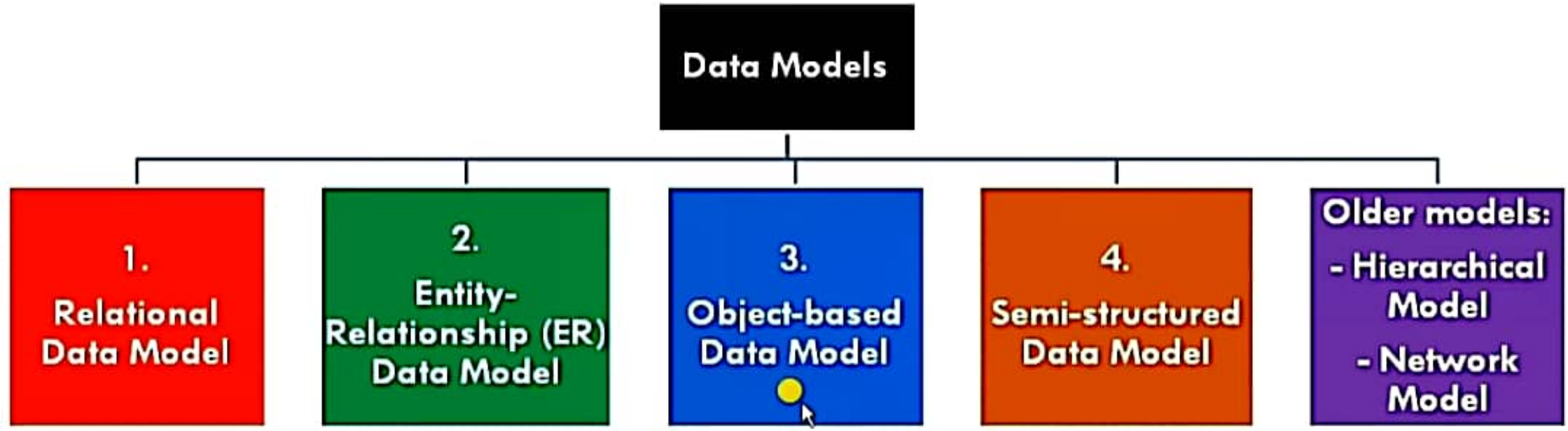
Fig. Three Level Architechture of DBMS

Data Models in DBMS

Data Model gives us an idea that how the final system will look like after its complete implementation.

- A **Data Model** in DBMS, is the concept of tools that are developed *to summarize* the *description of the database.*
- It defines how the logical structure of a database is modeled
- A **Data Model** is collection of conceptual tools for describing:
  - Data
  - Data relationships
  - Data semantics and
  - Consistency constraints
  - It describes the design of a database at each level of data abstraction.
  - It defines how data is connected to each other and how they are processed & stored inside the system

Types of Data Models



(We focus in this course)

(Mainly for database design)

(Object-oriented and Object-relational)

(XML)

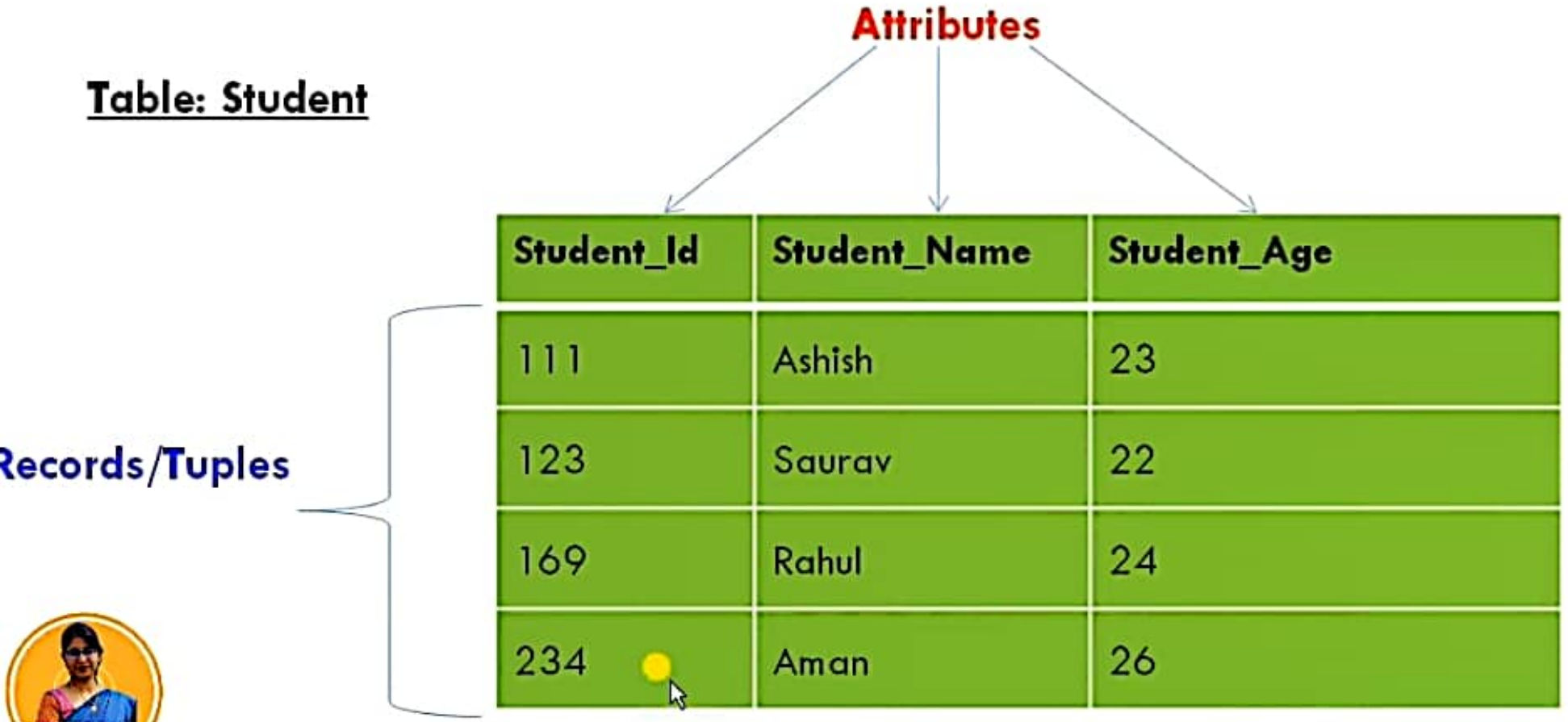
1. Relational Model

- **Most widely used model** by commercial data processing applications
- It uses collection of **tables** for representing **data and the relationships** among those data
- Data is stored in **tables** called ***Relations***
- Each table is a group of **column and rows**, where column represents **attribute** of an entity and rows represents **records** (or **tuples**).
  - ***Attribute* or *field*:** Each **column** in a relation is called an **attribute.** The values of the attribute should be from the same domain.
    - *Example*: we have different attributes of the *student* like Student_Id, Student_Name, Student_Age, etc.
  - ***Tuple* or *Record*:** Each **row** in the relation called **tuple.** A tuple defines a collection of attribute values. So each row in a relation contains unique values.
    - *Example*: each row has all the information about any specific individual like the first row has information about student Ashish.

This model was initially described by Edgar F. Codd, in 1969.

Example: Relational Model

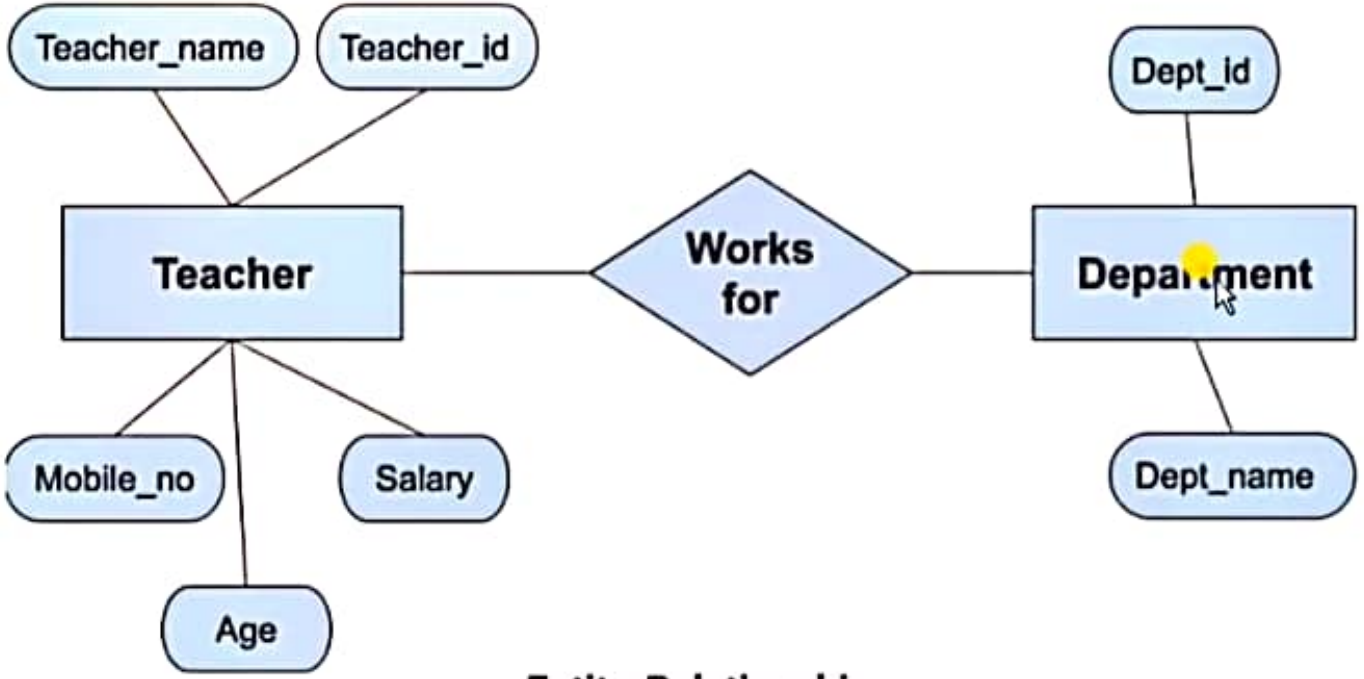
Table: Student



2. Entity-Relationship (ER) Model

- ER Model is a **high-level data model diagram**
- ER model **describes the structure of a database** with the help of a diagram, which is known as Entity Relationship Diagram (ER Diagram).
- An ER model is a **design or blueprint of a database** that can later be implemented as a database.
- It is based on the notion of real-world entities and relationships among them
- ER diagram has the following three components:
  - ***Entities:*** Entity is a real-world thing or object. It can be a person, place, or even a concept.
    - *Example:* Teachers, Students, Course, Building, Department, etc are some of the entities of a School Management System.
  - ***Attributes:*** An entity contains a real-world property called attribute. This is the characteristics of that attribute.
    - *Example:* The entity teacher has the property like teacher id, name, salary, age, etc.
  - ***Relationship:*** Relationship tells how two attributes are related.
    - *Example:* Teacher works for a department.

Example: ER Model



Entity-Relationship Model

- The **entities** are **Teacher** and **Department.**
- The **attributes** of ***Teacher*** entity are Teacher_Name, Teacher_id, Age, Salary, Mobile_Number.
- The **attributes** of ***Department*** entity are Dept_id, Dept_name.
- The two entities are connected using the **relationship**. Here, each teacher **works for** a department.

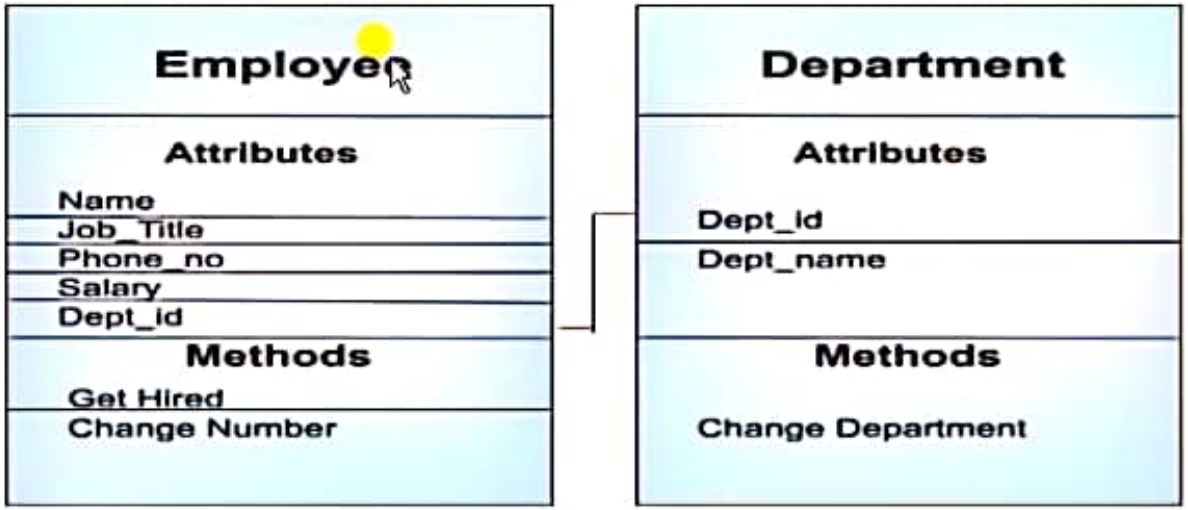
3. Object-based Data Model

- Two types:
  1. **Object-oriented data model**
  2. **Object-relational data model**

1. Object-oriented data model : An extension of the ER model with notions of functions, encapsulation, and object identity, as well.

- In this model, both the **data and relationship** are present in a single structure known as an **object.**
- Two are more objects are connected through **links.** We use this link to relate one object to other objects.

Example: Object-oriented data model



Object_Oriented_Model

- Here, two **objects** Employee and Department.
- All the **data and relationships** of each object are contained as a **single unit.**
- The **attributes** like Name, Job_title of the employee and the **methods** which will be performed by that object are **stored as a single object.**
- The two objects are **connected through a common attribute** i.e. the Department_id and the communication between these two will be done with the help of this common id i.e. the Department_id

2. Object-relational data model: It is a combination of the object-oriented data model and relational data model

- This model was built to fill the gap between object-oriented model and the relational model.
- It has many advanced features like complex data types that can be formed using the existing data types.
- The problem with this model is that this can get complex and difficult to handle. So, proper understanding of this model is required.

4. Semi-structured Data model

- Semi-structured model is an evolved form of the relational model.
- The **semi-structured data model** allows the data specifications at places where the *individual data items of the same type may have different sets of attributes.*
- In this model, some entities may have missing attributes while others may have an extra attribute.
- This model gives flexibility in storing the data. It also gives flexibility to the attributes.
  - ***Example:*** If we are storing any value in any attribute then that value can be either atomic value or a collection of values.
- The **Extensible Markup Language (XML)** is widely used for representing the semi-structured data.
  - In XML we can create use tags and use different mark ups to describe the data.

Example: XML or JSON

<student 1>
<Roll. No.>.....</Roll. No.>
<Name>.....</Name>
<Class>.....</Class>
<Age>.....</Age>
</student 1>

<student 2>
<Name>.....</Name>
<Class>.....</Class>
<Age>.....</Age>
</student 2>

Older Data Models

- **Hierarchical Data Model**
- **Network Data Model**

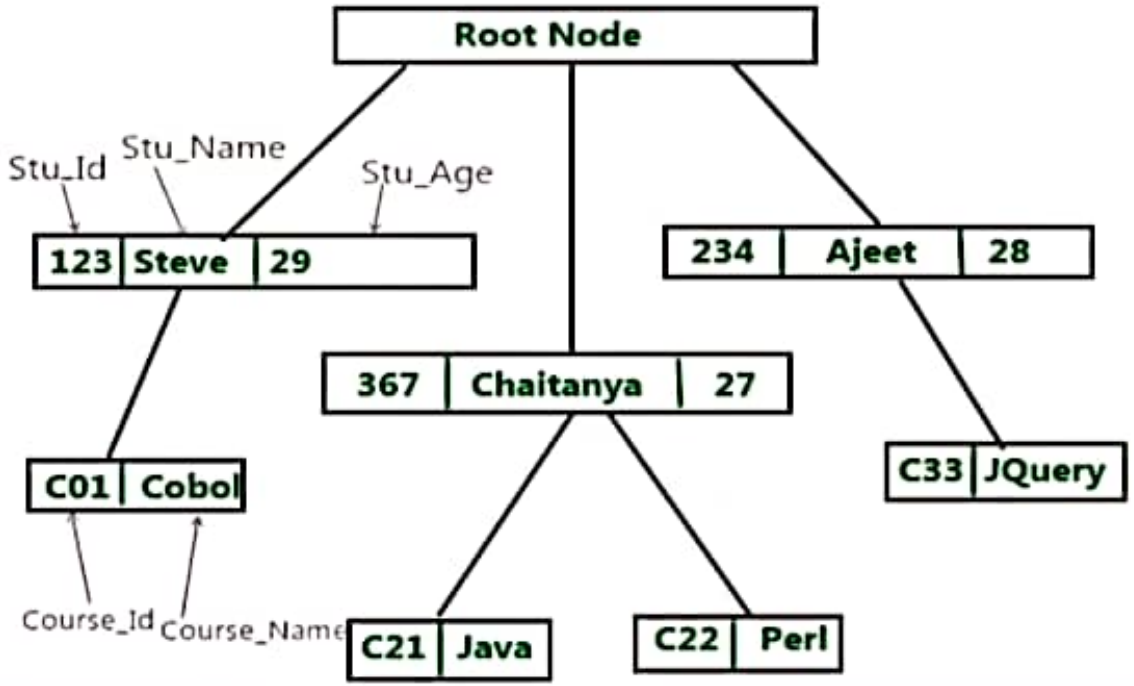
  - Hierarchical Data Model and Network Data Model preceded the Relational data model
  - But today they are replaced by Relational data model

Example: Hierarchical Model

Lets we have few **students** and few **courses:**

- a course can be assigned to a single student only,
- however a student take any number of courses

so this relationship becomes one to many.



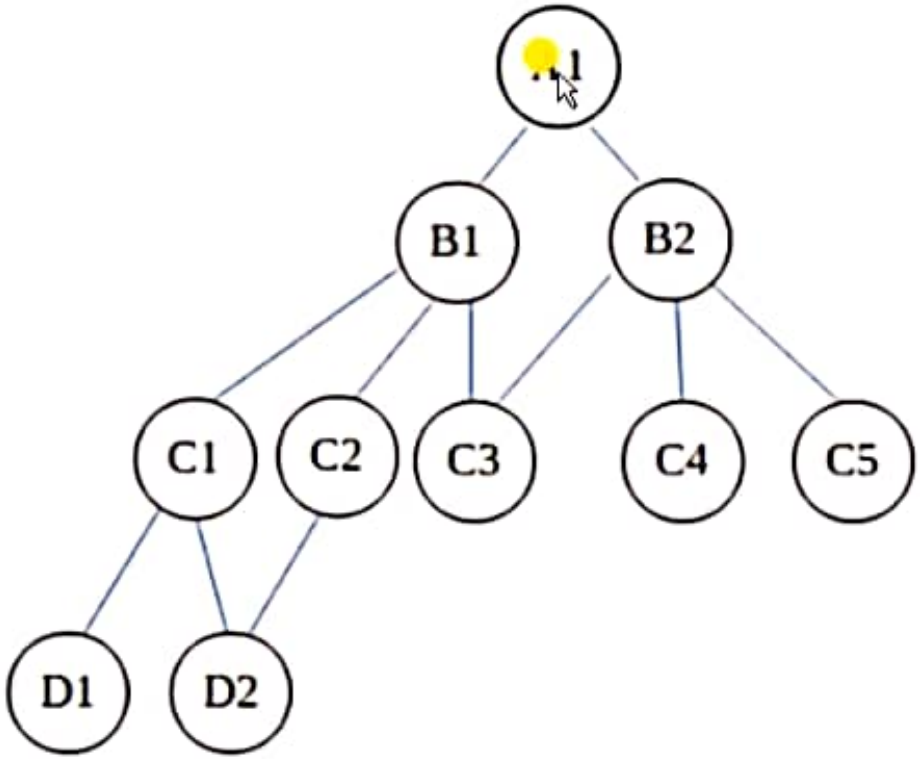
Hierarchical Model

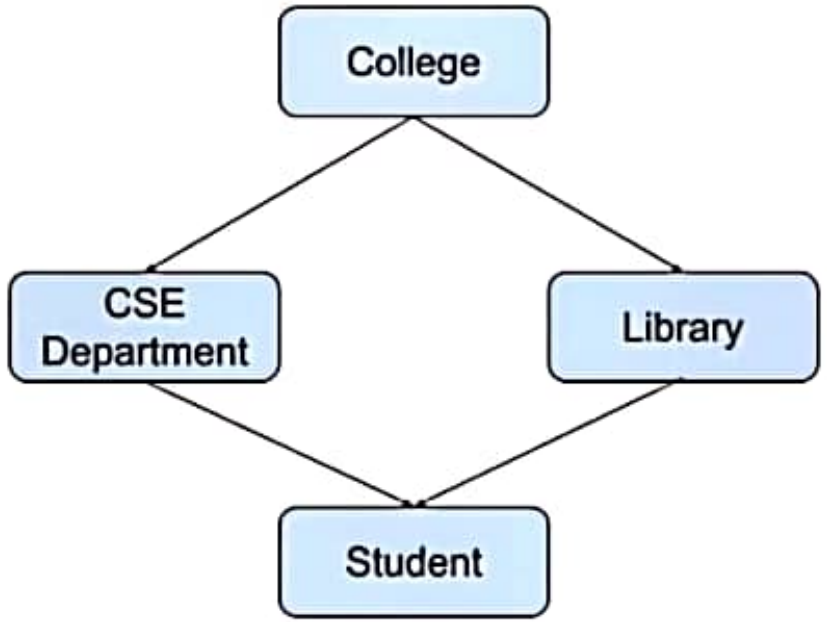
- It was the first DBMS model.
- In **hierarchical model**, data is organized into a **tree like structure** with each record is having one parent record and many children.
- The **main drawback** of this model is that, it can have only **one to many** relationships between nodes.
- Hierarchical models are **rarely used** now.

Network Model

- This model is an extension of the hierarchical model. It was the most popular model before the relational model.
- **Network Model** is same as hierarchical model except that it has **graph-like structure** rather than a tree-based structure and are allowed to have more than one parent node.
- It supports **many-to-many** data relationships.
- This was the most widely used database model, before Relational Model was introduced.

Example: Network Model





Network Model

Example: the node student has two parent nodes i.e. CSE Department and Library. This was earlier not possible in the hierarchical model.

Database languages

- A **DBMS** must provide appropriate languages and interfaces for each category of users to express database queries and updates.
- **Database Languages:**
  - communicates with the database
  - used to create and maintain database on computer.
- Mainly two **types** of Database Languages :
  - Data Definition Language (DDL): to specify database schema
  - Data Manipulation Language (DML): to express database queries and updates
- In practical, DDL and DML are not separate languages, instead they are the parts of a single database language such as widely used SQL (Structured Query Language)

Data Definition Language (DDL)

- **DDL** (stands for **D**ata **D**efinition **L**anguage) is used for specifying the **database schema.**
- Used by the DBA and database designers to specify the conceptual schema of a database
- DDL is used for creating tables, schema, indexes, constraints etc. in database.

  Eg: **create table** *account* (
  account-number **char**(10),
  balance **integer**)

- DDL is used to store the information of **metadata** like the number of tables and schemas, their names, indexes, columns in each table, constraints, etc.
  - DDL compiler generates a set of table templates stored in a *data dictionary*.
  - *Data dictionary contains Metadata (i.e. data about data)*
    - *Database Schema*
      - *Table name*
      - *Column names & types*
    - *Integrity Constraints*
      - *Primary key*
    - *Authorization*
      - *Who can access what*

DDL Commands in SQL:

- **CREATE:** used to create the database instance
- **ALTER:** used to alter the structure of the database
- **DROP:** used to delete the database instance
- **TRUNCATE:** used to remove all records from a table
- **RENAME:** used to rename database instances
- **Example**:

```
create table department
    (dept_name char(20),
    building char(15),
    budget numeric(12,2));
```

Note: These commands **either defines or update the database schema** that's why they come under Data Definition language.

Data Manipulation Language (DML)

- **DML** (stands for **D**ata **M**anipulation **L**anguage) is used for accessing and manipulating data in a database.
- It allows users to insert, update, delete and retrieve data from the database.
- **DML** is also known as **Query Language**
  - A **query** is a statement requesting the retrieval of information
- Two classes of DML languages:
  - Procedural DMLs/High-level DMLs:
    - User specifies **what** data is required and **how** to get those data
    - Procedural DML is embedded into a high-level programming language like java, etc.
    - Eg: PL/SQL
  - Declarative (Non-Procedural)DMLs/Low-level DMLs:
    - User specifies **what** data is required only; **without** specifying **how** to get those data
    - Declarative DMLs are usually easier to learn and use than procedural DMLs. However, since a user does not have to specify how to get the data, the database system has to figure out an efficient means of accessing data.
    - Eg: SQL

Data Manipulation Language (DML)

- **DML** (stands for **D**ata **M**anipulation **L**anguage) is used for accessing and manipulating data in a database.
- It allows users to insert, update, delete and retrieve data from the database.
- **DML** is also known as **Query Language**
  - A **query** is a statement requesting the retrieval of information
- Two classes of DML languages:
  - Procedural DMLs/High-level DMLs:
    - User specifies **what** data is required and **how** to get those data
    - Procedural DML is embedded into a high-level programming language like java, etc.
    - Eg: PL/SQL
  - Declarative (Non-Procedural)DMLs/Low-level DMLs:
    - User specifies **what** data is required only; **without** specifying **how** to get those data
    - Declarative DMLs are usually easier to learn and use than procedural DMLs. However, since a user does not have to specify how to get the data, the database system has to figure out an efficient means of accessing data.
    - Eg: SQL

DML Commands in SQL:

- **SELECT:** used to retrieve data from a database.
- **INSERT:** used to insert data into a table.
- **UPDATE:** used to update existing data within a table.
- **DELETE:** used to delete all records from a table.

- **Example:**

```
select student_name, Student_age
from student;
```

Other Database Language

- Data Control Language (DCL)
- Transaction Control Language (TCL)

Data Control language (DCL)

- **Data Control Language (DCL)** is used to control privilege in Databases i.e. DCL is used for granting and revoking user access on a database (Authorization)
- To perform any operation in the database, such as for creating tables, sequences, or views, we need privileges.
- In particular, it is a component of Structured Query Language (SQL)
- **DCL Commands**:
  - **GRANT** – To give user access privileges to a database
  - **REVOKE** – To take back permissions from the user
- The operations for which privileges may be granted to or revoked from a user or role apply to both the Data definition language (DDL) and the Data manipulation language (DML),

Transaction Control Language (TCL)

- **TCL (Transaction Control Language)** commands are used to manage transactions in the database.
- TCL is used to run the changes made by the DML statement.
- The changes in the database that we made using DML commands are either performed or rollbacked using TCL.
- **TCL Commands:**
  - **COMMIT**: to save the transaction on the database
  - **ROLLBACK**: to restore the database to original since the last Commit.
  - **SAVEPOINT**: Savepoint command is used to temporarily save a transaction so that you can rollback to that point whenever necessary.

Database Access from Application Programs

- **SQL**: widely used non-procedural, commercial language
- Example: Find the name of the instructor with ID 22222

```
select   name
from     instructor
where    instructor.ID = '22222'
```

- **SQL** is NOT a Turing equivalent language which means that everything that need to be computed cannot be computed in SQL
- **SQL** alone is not very powerful; because there are some computations that cannot be obtained by any SQL query. Such computations must be written in a host language, such as C, C++, Java with embedded SQL queries that access the data in the database
  - Application Programs are programs that are used to interact with the database in this fashion. Eg: Banking system are programs that generate: Payroll checks, Debit accounts, Credit Accounts or transfer funds between accounts
  - To access the database, DML statements need to be executed from the host language
- **Application Programs generally access the database through one of the 2 ways:**
  1. APIs (ODBC/JDBC) which allows SQL Queries to be sent to database
  2. Language extensions to allow embedded SQL

Application Programs generally access the DB through one of the 2 ways

1. *By providing an Application Programming Interface i.e API (the set of procedures) that can be used to send DDL & DML statements (SQL Queries) to the database and retrieve the results* i.e. **APIs (ODBC/JDBC) which allows SQL Queries to be sent to database**
   - Eg: 1. ODBC (Open Database Connectivity) standard
     - It is created by Microsoft in 1992 that is used by Windows software applications, using C/C++ language, to access databases via SQL
     2. JDBC (Java Database Connectivity) standard
     - It is created by Sun Microsystems in 1997 that is used by JAVA application, using only JAVA Language, to access databases via SQL
2. *By extending the host language syntax to embed DML calls within the host language program* i.e **Language extensions to allow embedded SQL**
   - Eg: C,C+, JAVA with embedded SQL queries
   - Usually special character (like EXEC) prefaces DML Calls, and a Preprocessor called the DML Pre-Compiler, that converts the DML statements to normal procedure calls in the host language

Database Users and Administrators

- People who work with a database can be
  - Database Users
  - Database Administrators (DBAs)

Database Users

- **Database Users** are the persons who interact with the database and take the benefits of database.
- Users are differentiated by the way they expect to interact with the system.
- Four types of DB users:
  1. **Naive users/Native users/End users**
  2. **Application programmers**
  3. **Sophisticated users**
  4. **Specialized users**

Types of Database Users

1. **Naive Users/Native Users/End Users**: They are the unsophisticated users who use the existing application to interact with the database.
   - *Example: People accessing database over the web, bank tellers, clerical staffs, etc.*
2. **Application Programmers**: They are the computer professionals who write the application programs. They interact with system through DML queries.
   - *For example: Writing a C program to generate the report of employees who are working in particular department, will involve a query to fetch the data from database. It will include a embedded SQL query in the C Program.*

Types of Database Users ...

3. **Sophisticated Users**: They interact with the system by writing SQL queries directly through the query processor (like SQL) without writing application programs.
   - *Example: Analysts, who submits SQL queries to explore data in the DBMS.*
4. **Specialized Users**: They are also sophisticated users who write specialized database applications that do not fit into the traditional data processing framework. They are the developers who develop the complex programs to the requirement.
   - *Example: Computer-Aided Design (CAD) Systems, Expert Systems, Knowledge Based System, etc. that store complex data types (graphics and audio data) & environment modelling systems*

Types of Database Users ...

3. **Sophisticated Users**: They interact with the system by writing SQL queries directly through the query processor (like SQL) without writing application programs.
   - *Example: Analysts, who submits SQL queries to explore data in the DBMS.*
4. **Specialized Users**: They are also sophisticated users who write specialized database applications that do not fit into the traditional data processing framework. They are the developers who develop the complex programs to the requirement.
   - *Example: Computer-Aided Design (CAD) Systems, Expert Systems, Knowledge Based System, etc. that store complex data types (graphics and audio data) & environment modelling systems*

Database Administrators (DBAs)

- DBA is a person or group that is responsible for supervising both the database and the use of the DBMS.
- Database Administrators (DBAs) coordinate all the activities of the database system.
- They use specialized software to store and organize data
- They have all the permissions

DBAs Tasks

- Schema definition
- Storage structure and access method definition
- Schema and physical organization modification
- Specifying integrity constraints
- Granting user authority to access database
- Monitoring performance & responding to changes in requirements
- Routine Maintenance
- Acting as liaison with users
- Backing up and restoring databases

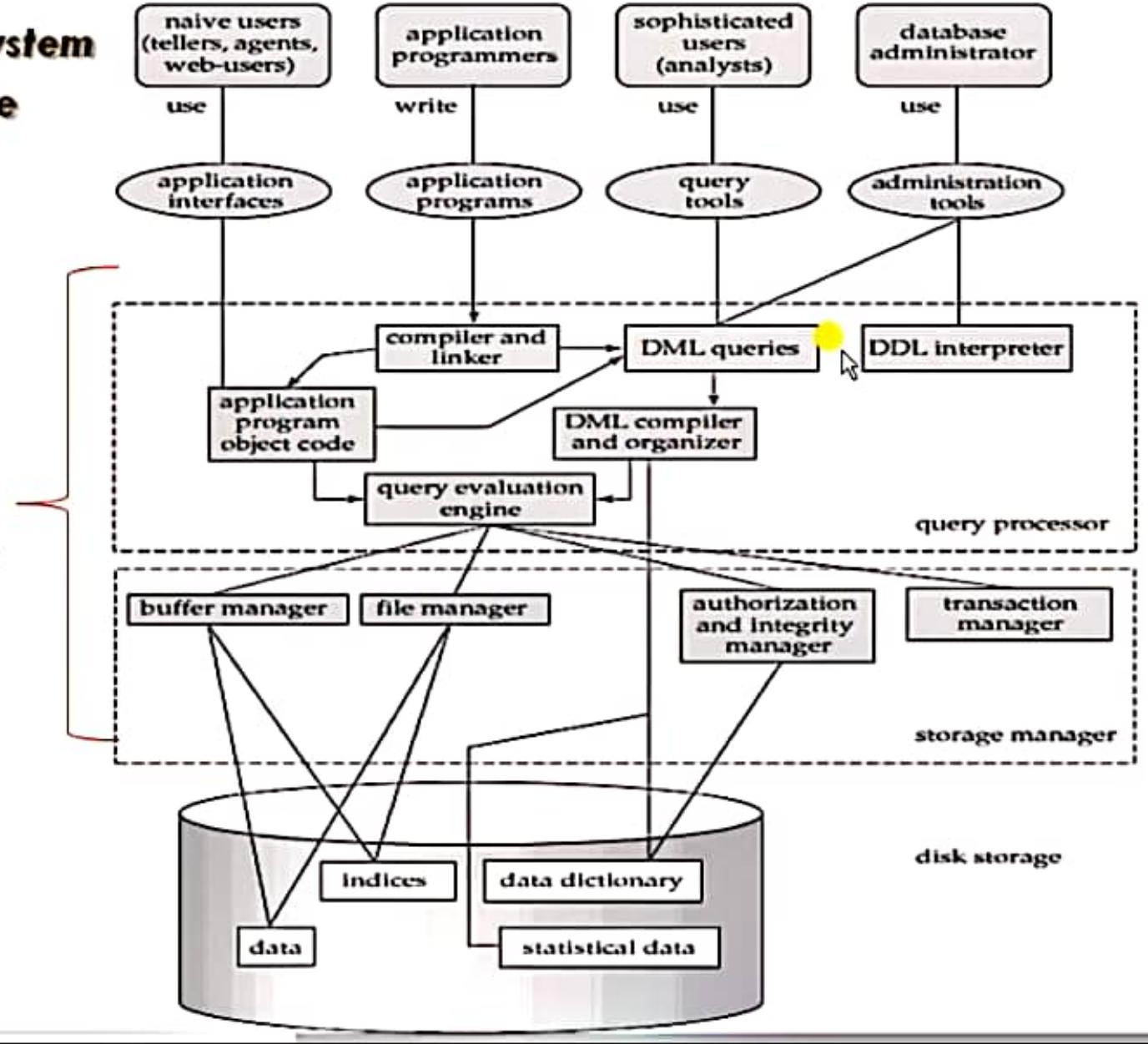
DBMS

- **Database Management System (DBMS)** is a software that allows access to data stored in a database and provides an easy and effective method of:
  - Defining the information
  - Storing the information
  - Manipulating the information
  - Protecting the information from system crashes or data theft
  - Differentiating access permissions for different users
- **DBMS (Database Management System)** acts as an **interface between the user and the database.** The user requests the DBMS to perform various operations such as insert, delete, update and retrieval on the database

Database System Structure

- **Database System Structure is partitioned into modules for different functions.**
- Some functions (e.g. file systems) may be provided by the *operating system*.
- The database system is divided into two functional components:
  1. **Query Processor**
  2. **Storage Manager**

Database System Structure



DBMS

DBMS acts as an interface between the user and the database.

1. The ***Users*** issues a queries using a particular database language like SQL

2. The ***Query Processor*** processes the queries or programs with the help of its components

3. The ***Storage Manager*** access the stored data from the **disk**

4. **DBMS** returns the results to the user

1. Query Processor

- Query processor helps the database system, simplify and facilitate access to data.
- The work of query processor is **to execute the query successfully**
  - It interprets the requests (queries) received from end user via an application program into instructions
  - It also executes the user request which is received from the DML compiler

Query Processor components:

- **DDL Interpreter**
  It interprets the DDL statements (like schema definition) into a set of table containing meta data (data about data) stored in a data dictionary
- **DML Compiler**
  It translates the DML statements (like select, insert, update, delete) into low level instruction (Object code), so that they can be executed.
  - A query can usually be translated into any of a number of alternative evaluation plans that all give the same result. The DML compiler also performs **query optimization**, that is, it picks the lowest cost evaluation plan from among the alternatives.
  - **Query optimization** determines the efficient way to execute a query with different possible query plans.
- **Embedded DML Pre-compiler**
  It processes DML statements embedded in an application program into procedural calls.

Query Evaluation Engine
It executes the low level instruction (object code) generated by DML Compiler.

2. Storage Manager

- ***Storage Manager*** is a program that provides an ***interface*** ***between the data stored in the database and the queries received.***
- The storage manager is responsible for the interaction with the *file manager*.
  - The raw data are stored on the disk using the **file system**, which is usually provided by a conventional *operating system*.
- The storage manager **translates** the various DML statements into low-level file-system commands
- Thus, It is responsible for updating, storing, deleting, and retrieving data in the database
- It is also known as ***Database Control System***. It maintains the consistency and integrity of the database by applying the constraints and executes the DCL statements.

Storage Manager components:

- **Authorization Manager and Integrity Manager**
  It checks the authority of users to access data and checks the integrity constraints when the database is modified.
- **Transaction Manager**
  It ensures that the database remains in a consistent (correct) state despite system failures, and that concurrent transaction executions proceed without conflicting
- **File Manager**
  It manages the allocation of space on disk storage and the data structures used to represent information stored on disk
- **Buffer Manager**
  It is responsible for fetching data from disk storage into main memory, and deciding what data to cache in main memory.

Disk Storage

The ***Storage Manager*** implements the following data structures as part of the physical system implementation:

- **Data Files**
  It stores the database itself
- **Data Dictionary**

  It contains the information about the structure of any database. Or, It stores metadata about the database, in particular the schema of the database.
- **Indices**
  It provides faster retrieval of data item.
- **Statistical Data**

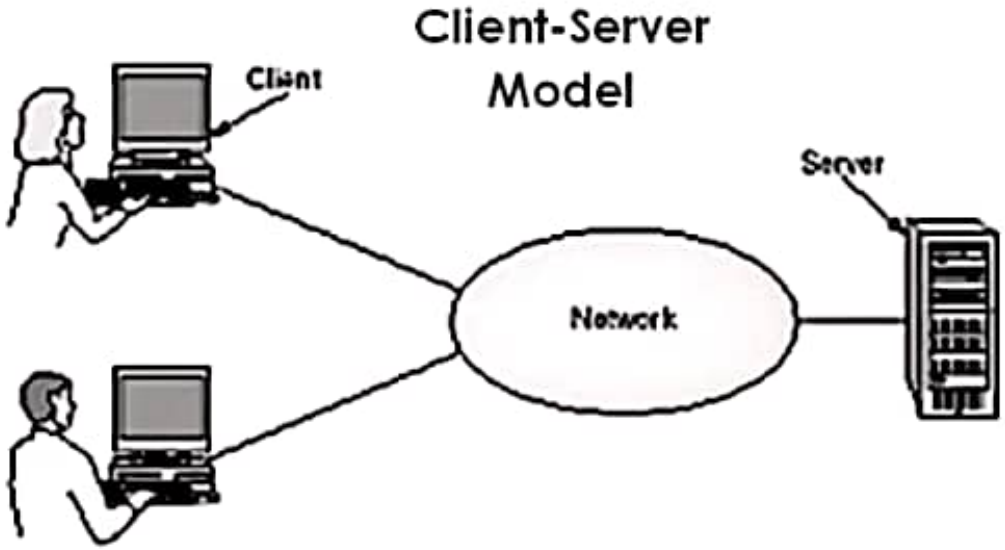
  It stores statistical information about the data in the database. This information is used by the ***query processor*** to select efficient ways to execute a query

DBMS Architecture

- The design of a DBMS depends on its **architecture.**
- **DBMS architecture** helps in design, development, implementation, and maintenance of a database.
- A database stores critical information for a business. Selecting the correct Database Architecture helps in quick and secure access to this data.
- ***DBMS architecture depends upon***
  - ***the underlying computer system on which database system runs***
  - ***how users are connected to the database to get their request done***

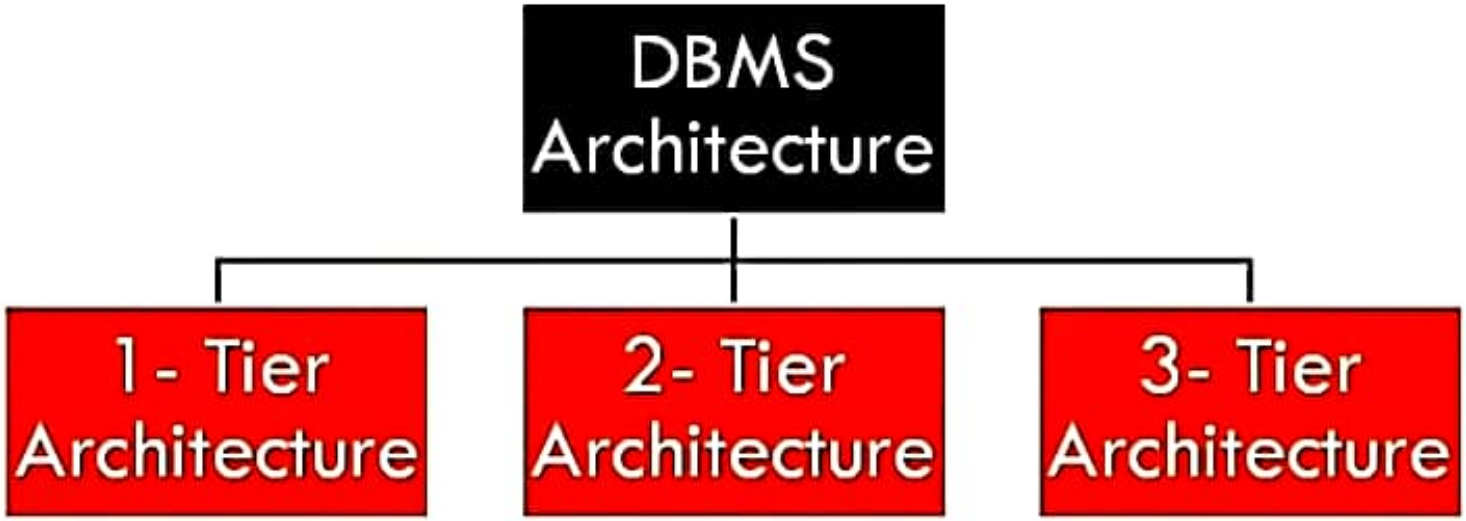
Client/Server architecture?

- The ***basic client/server architecture*** is used to deal with a large number of PCs, web servers, database servers and other components that are connected with networks.
- The client/server architecture consists of many PCs and a workstation which are connected via the network. **Client** request the server for a service. **Server** provides the requested service to the client.

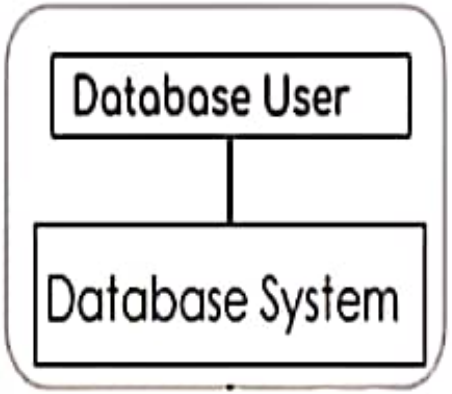


Types of DBMS Architecture

- Database architecture can be seen as a ***single tier*** or ***multi-tier***



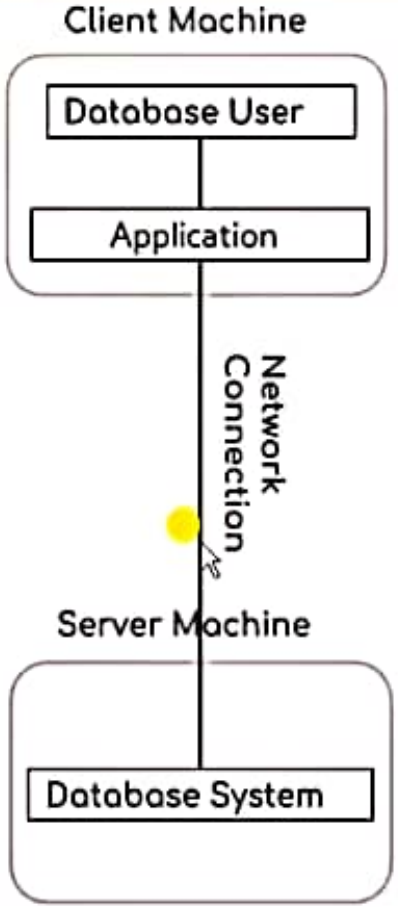
Single or 1-tier Architecture



1-tier Architecture

- In **single** or **1 tier architecture,** the database is directly available to the user. The Client, Server, and Database all reside on the same machine i.e. the user can directly sit on the DBMS and uses it.
- Any changes done here will directly be done on the database itself. It doesn't provide a handy tool for end users.
- No network connection is required to perform the action on the database
- **1 tier architecture is used**
  - where data does not change frequently and where no multiple user is accessing the system
  - for development of the local application, where programmers can directly communicate with the database for the quick response.
  - But such architecture is rarely used in production.

2-Tier Architecture



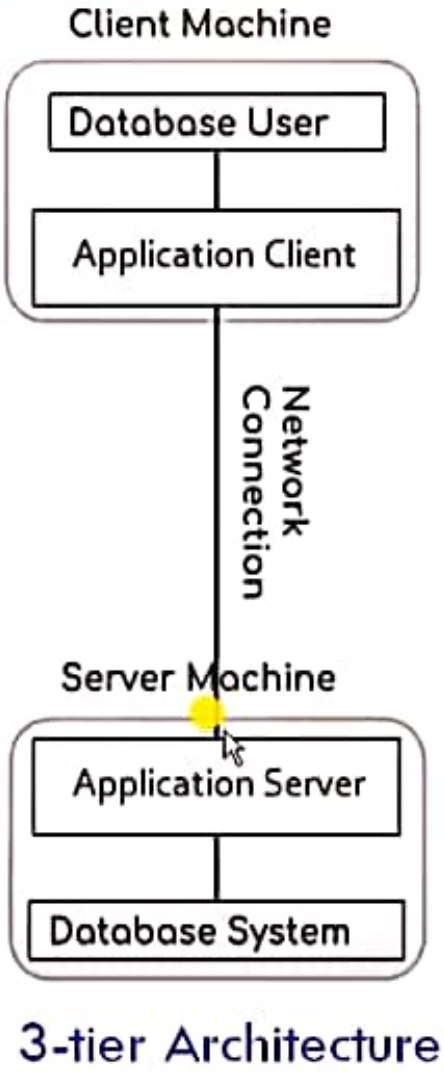
2-tier Architecture

- It is same a **client-server**
- In the **2-tier architecture**, **applications** on the *client end* can directly communicate with the ***database*** at the *server side*.
- An ***Application Programming Interfaces (APIs)*** like **ODBC & JDBC** are used by client side program to call the DBMS
- The user interfaces and application programs are run on the ***client-side***
- To communicate with the DBMS, ***client-side application*** establishes a ***connection*** with the ***server side***
- The ***server side*** is responsible to provide the functionalities like: query processing and transaction management.

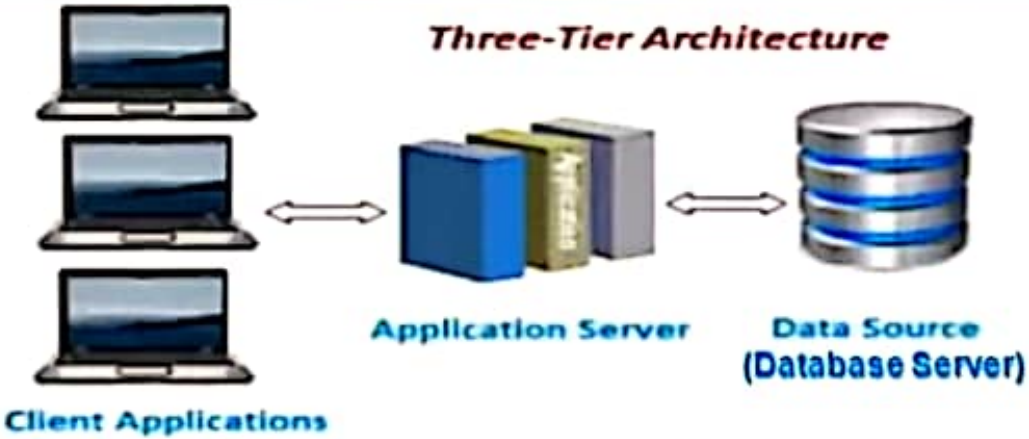
2-Tier Architecture ...

- The **2-tier architecture is used** inside any organization, where multiple clients accessing the database server directly
- **Example:** Railway reservation from counter, where clerk as a client accesses the railway server directly.
- **Advantage :**
  - *Direct* and *faster* communication
  - *Maintenance* and *understanding* is easier
  - *Compatible* with existing systems
- **Disadvantage:**
  - *Scalability* i.e. It gives poor performance when there are a large number of users
  - *Less secure* as client can access the server directly

3-Tier Architecture

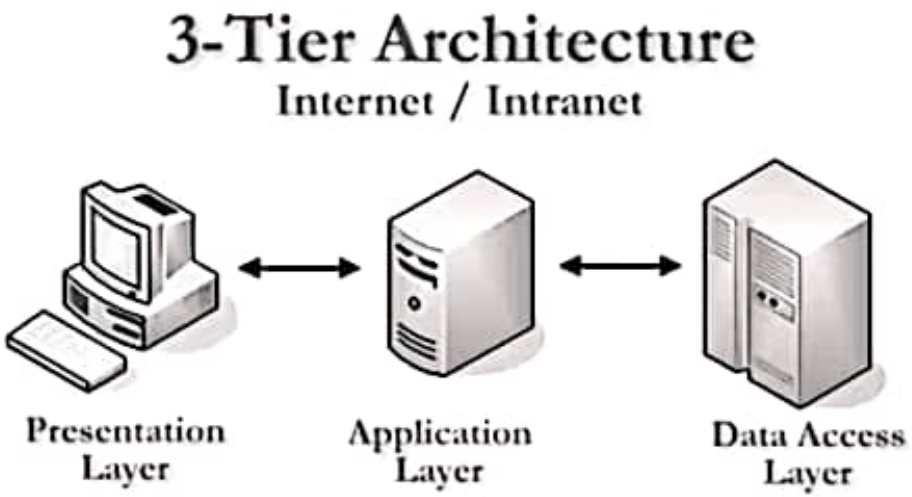


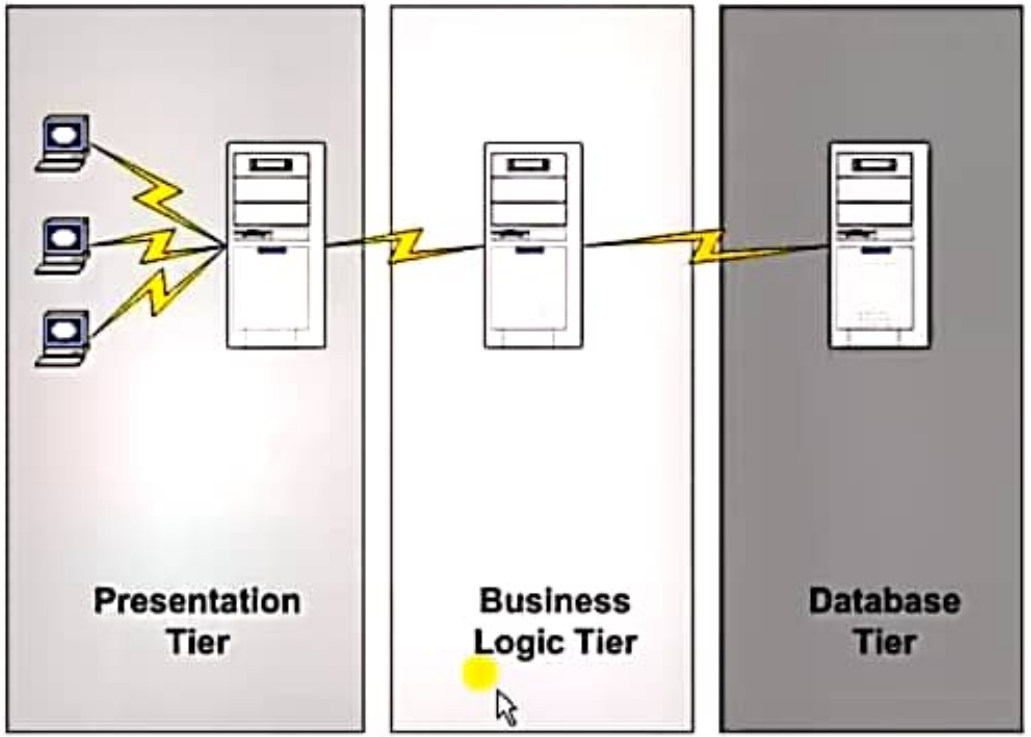
3-tier Architecture



- The **3-Tier architecture** contains *another layer of* ***Application Server*** between the ***client*** & ***server***. In this architecture, client can't directly communicate with the server.
- The ***application on the client-end*** interacts with ***an application server*** which further communicates with the ***database system*** and then the query processing and transaction management takes place.
- This **intermediate layer of Application Server** acts as a **medium** for exchange of partially processed data between **server** and **client.**
- ***End user*** has no idea about the existence of the database beyond the ***application server.*** The ***database*** also has no idea about any other user beyond the ***application.***

- In 3-tier architecture, an application is virtually split into three separate logical layers





3-Tier Architecture...

- The **3-Tier architecture** is **used** in **large web applications**
- It is the **most popular** DBMS architecture.
- **Advantages:**
  - **Enhanced scalability** due to distributed deployment of application servers. Now, individual connections need not be made between client and server.
  - **Data Integrity** is maintained. Since there is a middle layer between client and server, data corruption can be avoided/removed.
  - **Security** is improved. This type of model prevents direct interaction of the client with the server thereby reducing access to unauthorized data.
- **Disadvantages:**
  - **Increased complexity** of implementation and communication. It becomes difficult for this sort of interaction to take place due to presence of middle layers.