# Database Management System

## CSE2004 : Database System Concepts and Architecture

Dr. Nachiyappan S

VIT-Chennai

# Contents.....!

## Module-1

- History and motivation for database systems
- Characteristics of database approach.
- Actors on the scene and Workers behind the scene.
- Advantages of using DBMS approach
- Data Models, Schemas, and Instances.
- Three Schema Architecture and Data Independence.
- The Database System Environment.
- DBMS Components
- Centralized and Client/Server Architectures for DBMSs.
- Classification of database management systems.

# Text Books and References

## Text Books

- R. Elmasri & S. B. Navathe, Fundamentals of Database Systems, Addison Wesley, 7 th Edition, 2015
- Raghu Ramakrishnan,Database Management Systems,Mcgraw-Hill,4 th edition,2015

## References

- A. Silberschatz, H. F. Korth & S. Sudershan, Database System Concepts, McGraw Hill, 6 th Edition 2010
- Thomas Connolly, Carolyn Begg, Database Systems : A Practical Approach to Design, Implementation and Management,6 th Edition,2012
- Pramod J. Sadalage and Marin Fowler, NoSQL Distilled: A brief guide to merging world of Polyglot persistence, Addison Wesley, 2012.
- Shashank Tiwari, Professional NoSql, Wiley, 2011.

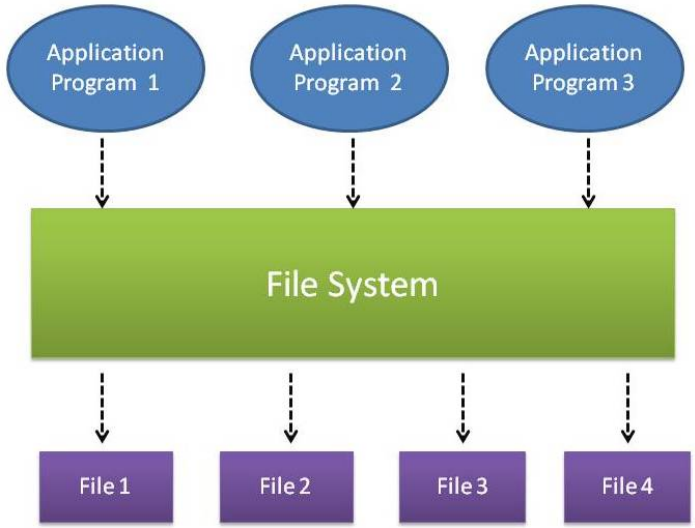
VIT Vellore Institute of Technology

# Outline

- Introduction
  1. File based approach
  2. Database approach
  3. Nuts and Bolts
- Database systems concepts
  1. Data model
  2. Three schema architecture – Data independence
  3. Database Schema, State and Instance
  4. DBMS languages
  5. Classification of DBMS
  6. Database users
- Centralized Database and its Merits & De-merits
- Client-Server Database and its Merits & De-merits

# File Based or Traditional Approach

Data is stored in one or more separate computer files
Data is then processed by computer programs - **Applications**




In traditional approach, information is stored in flat files which are maintained by the file system under the operating system's control.
Application programs go through the file system in order to access these flat files

# File Based Approach

**How data is stored in flat files**

- Data is stored in flat files as records.
- Records consist of various fields which are delimited by a space, comma, pipe, any special character etc.
- End of records and end of files will be marked using any predetermined character set or special characters in order to identify them

**Example**: Storing employee data in flat files.

```
102 Anil HR
103 Maya    Softwzare
104 Jijo  Purchase
```

# Problems with traditional approach for storing data

- **Data Security** : The data stored in the flat file(s) can be easily accessible and hence it is not secure.
  **Example:** Consider an online banking application where we store the account related information of all customers in flat files. A customer will have access only to his account related details. However from a flat file, it is difficult to put such constraints. It is a big security issue.

- **Data Redundancy** : In this storage model, the same information may get duplicated in two or more files. This may lead to higher storage and access cost. It also may lead to data inconsistency.
  Suppose the same data is repeated in two or more files. If a change is made to data stored in one file, other files also needs to be change accordingly.

# Problems with traditional approach for storing data

- **Data Isolation** : Data Isolation means that all the related data is not available in one file. Usually the data is scattered in various files having different formats. Hence writing new application programs to retrieve the appropriate data is difficult.

- **Program/Data Dependence** : In traditional file approach, application programs are closely dependent on the files in which data is stored. If we make any changes in the physical format of the file(s), like addition of a data field , etc, all application programs needs to be changed accordingly.

# Problems with traditional approach for storing data

- **Lack of Flexibility** : The traditional systems are able to retrieve information for predetermined requests for data. If we need unanticipated data, huge programming effort is needed to make the information available, provided the information is there in the files. By the time the information is made available, it may no longer be required or useful.
- **Concurrent Access Anomalies** : Many traditional systems allow multiple users to access and update the same piece of data simultaneously. However this concurrent updates may result in inconsistent data. To guard against this possibility, the system must maintain some form of supervision. But supervision is difficult because data may be accessed by many different application programs and these application programs may not have been coordinated previously.

# Introduction to DBMS

The difficulties like **Data Security, Data Dependency, Data Redundancy, Data Isolation, Lack of Flexibility and Concurrent Access Control** lead to the development of database systems.

**Database Management System** or **DBMS** in short refers to the technology of storing and retrieving users data with utmost efficiency along with appropriate security measures.

# Introduction to DBMS

**What is Database ?**

It is a collection of inter-related data which is used to retrieve, insert and delete the data efficiently. It is also used to organize the data in the form of a table, schema, views, and reports, etc.

**For example**: The college Database organizes the data about the admin, staff, students and faculty etc. Using the database, you can easily retrieve, insert, and delete the information

**Database Management System**

- It is a software which is used to manage the database. For example: MySQL, Oracle.
- It permits to perform operations like Database creation, storing data in it, updating data, creating a table in the database and a lot more.
- It provides protection and security to the database. In the case of multiple users, it also maintains data consistency.

VIT Vellore Institute of Technology

# Introduction to DBMS

DBMS allows users the following tasks:

- **Data Definition**: It is used for creation, modification, and removal of definition that defines the organization of data in the database.
- **Data Updation**: It is used for the insertion, modification, and deletion of the actual data in the database.
- **Data Retrieval**: It is used to retrieve the data from the database which can be used by applications for various purposes.
- **User Administration**: It is used for registering and monitoring users, maintain data integrity, enforcing data security, dealing with concurrency control, monitoring performance and recovering information corrupted by unexpected failure.

# Characteristics of DBMS

## Characteristics of DBMS

- It uses a digital repository established on a server to store and manage the information.
- It can provide a clear and logical view of the process that manipulates data.
- DBMS contains automatic backup and recovery procedures.
- It contains ACID properties which maintain data in a healthy state in case of failure.
- It can reduce the complex relationship between data.
- It is used to support manipulation and processing of data.
- It is used to provide security of data.
- It can view the database from different viewpoints according to the requirements of the user.

# DBMS vs. File System

| DBMS | File System |
|---|---|
| DBMS is a collection of data. In DBMS, the user is not required to write the procedures. | File system is a collection of data. In this system, the user has to write the procedures for managing the database. |
| DBMS gives an abstract view of data that hides the details. | File system provides the detail of the data representation and storage of data. |
| DBMS provides a crash recovery mechanism, i.e., DBMS protects the user from the system failure. | File system doesn't have a crash mechanism, i.e., if the system crashes while entering some data, then the content of the file will lost. |
| DBMS provides a good protection mechanism. | It is very difficult to protect a file under the file system. |
| DBMS contains a wide variety of sophisticated techniques to store and retrieve the data. | File system can't efficiently store and retrieve the data. |
| DBMS takes care of Concurrent access of data using some form of locking. | In the File system, concurrent access has many problems like redirecting the file while other deleting some information or updating some information. |

# Advantages of DBMS

- **Controls database redundancy**: It can control data redundancy because it stores all the data in one single database file and that recorded data is placed in the database.
- **Data sharing**: In DBMS, the authorized users of an organization can share the data among multiple users.
- **Easily Maintenance**: It can be easily maintainable due to the centralized nature of the database system.
- **Reduce time**: It reduces development time and maintenance need.
- **Backup**: It provides backup and recovery subsystems which create automatic backup of data from hardware and software failures and restores the data if required.
- **Multiple user interface**: It provides different types of user interfaces like graphical user interfaces, application program interfaces

# Disadvantages of DBMS

- **Cost of Hardware and Software**: It requires a high speed of data processor and large memory size to run DBMS software.
- **Size**: It occupies a large space of disks and large memory to run them efficiently.
- **Complexity**: Database system creates additional complexity and requirements.
- **Higher impact of failure**: Failure is highly impacted the database because in most of the organization, all the data stored in a single database and if the database is damaged due to electric failure or database corruption then the data may be lost forever.

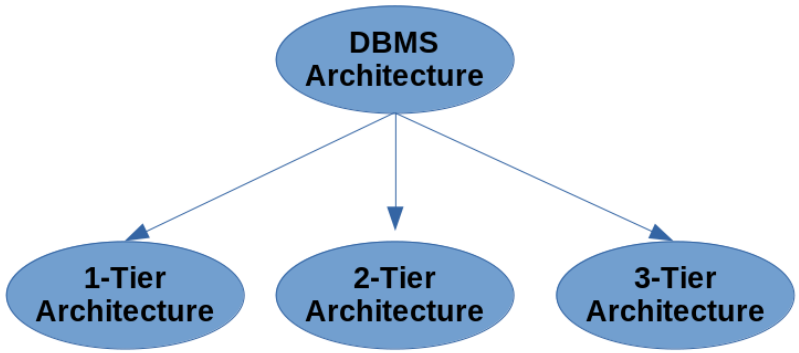
# DBMS Architecture

The DBMS design depends upon its 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.

DBMS architecture depends upon how users are connected to the database to get their request done.



# 1-Tier Architecture

- In this architecture, the database is directly available to the user. It means 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.
- The 1-Tier architecture is used for development of the local application, where programmers can directly communicate with the database for the quick response.

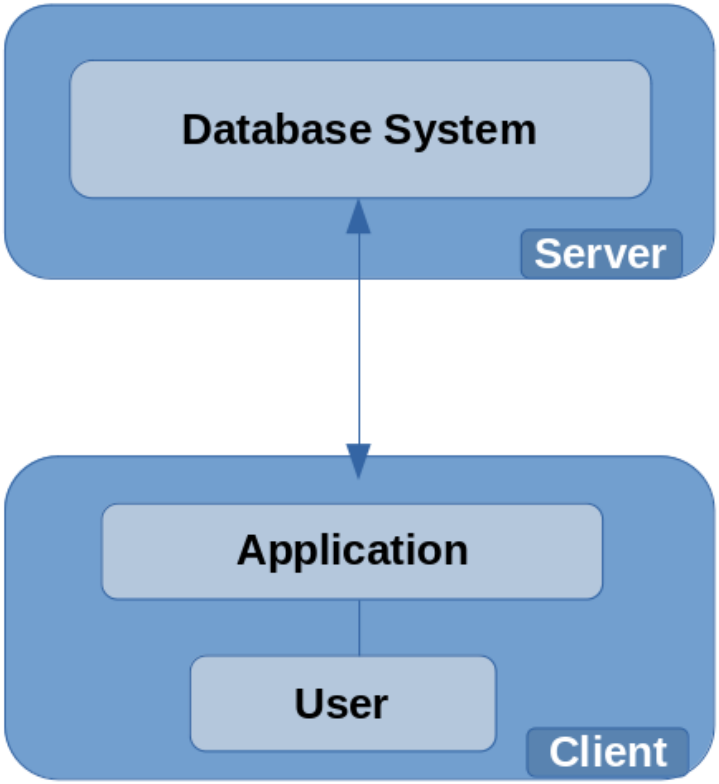
# 2-Tier Architecture

- It is same as basic client-server. But Applications on the client end can directly communicate with the database at the server side. For this interaction, API's like: ODBC, JDBC are used.
- The user interfaces and application programs are run on the client-side.
- The server side is responsible to provide the functionalities like: query processing and transaction management.
- To communicate with the DBMS, client-side application establishes a connection with the server side.

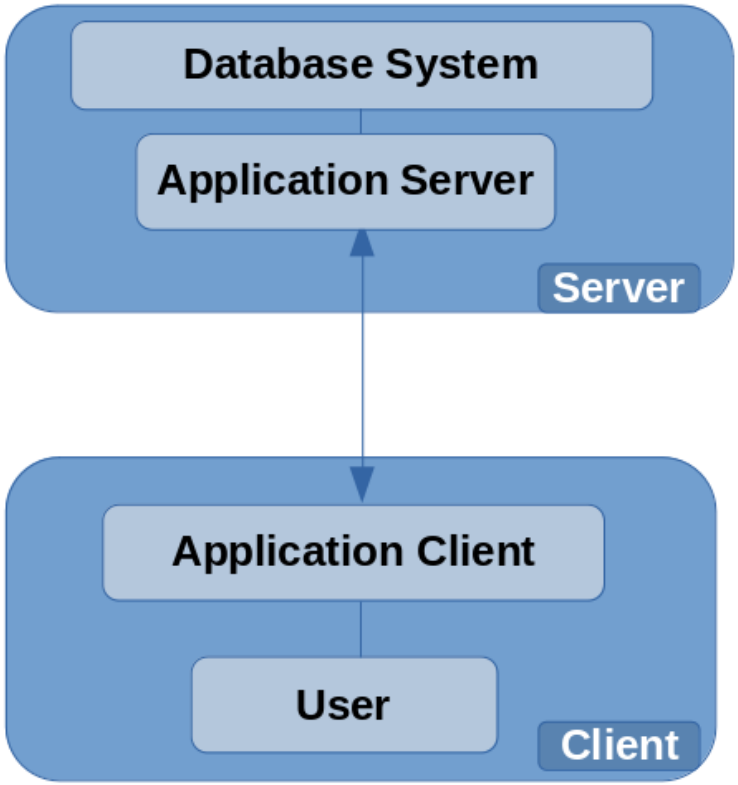
# 3-Tier Architecuter

- The 3-Tier architecture contains another layer between the client and 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.
- 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.
- The 3-Tier architecture is used in case of large web application.

# 2-Tier and 3-Tier Architectures




2-Tier Architecture



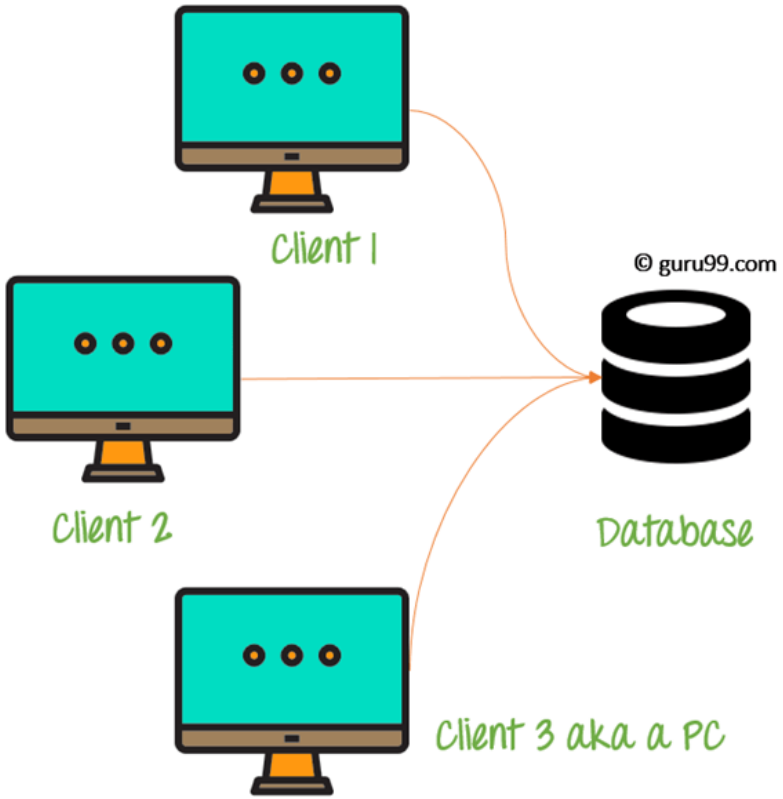

3-Tier Architecture

# Tier Architecutures

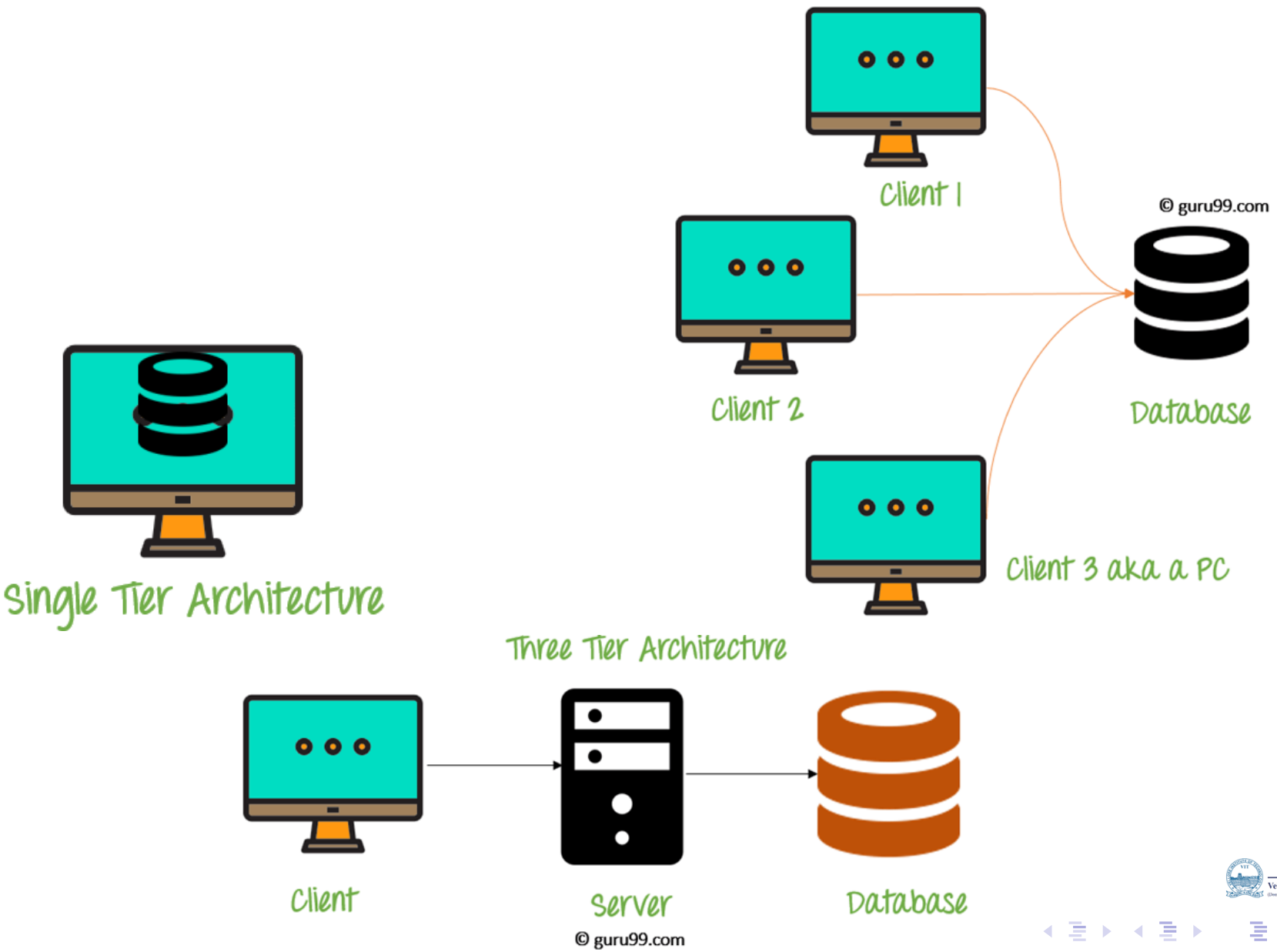
Single Tier Architecture

# Tier Architecutures

Single Tier Architecture

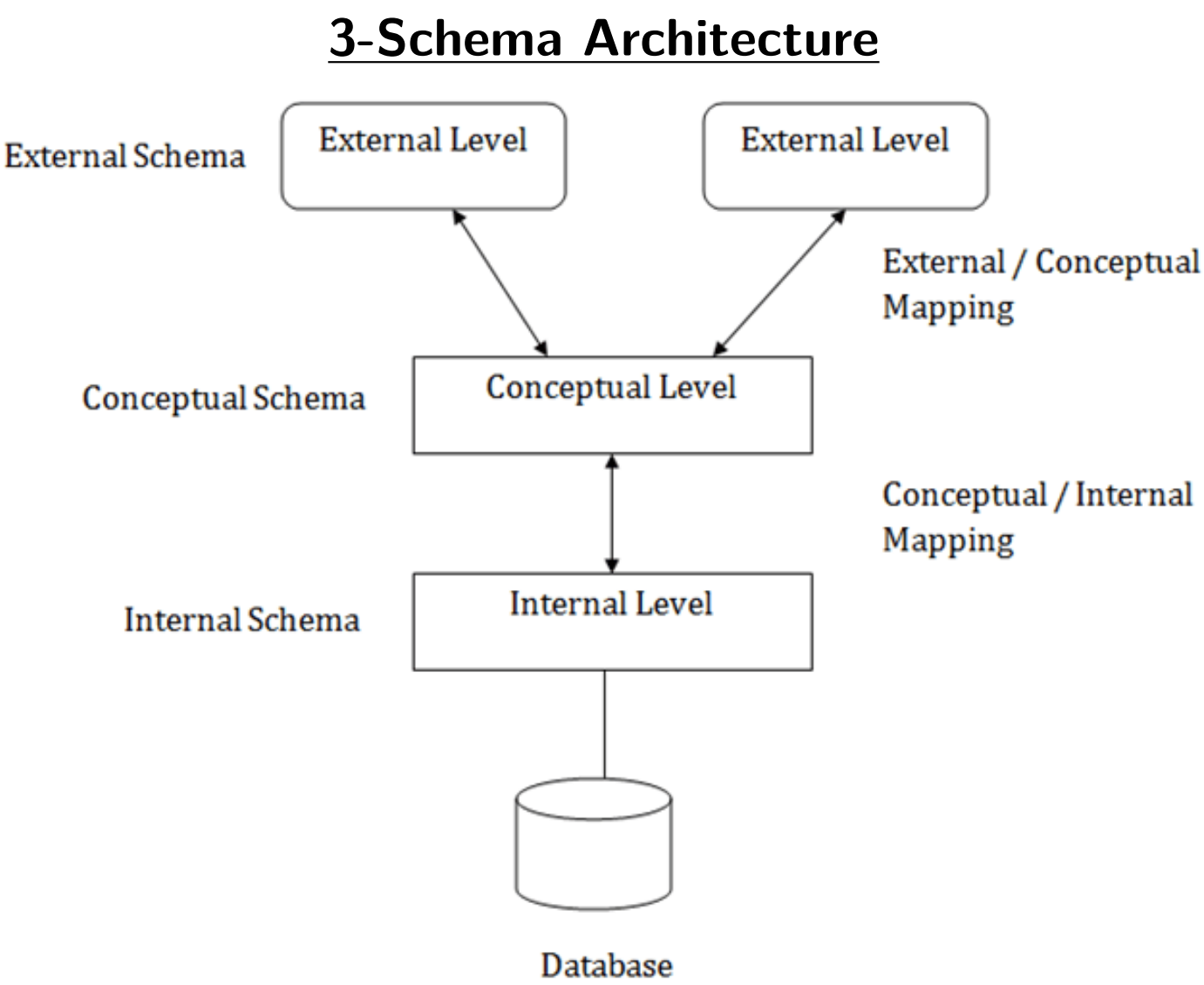


# Tier Architecutures



# 3-Schema Architecture

- It is also called ANSI/SPARC architecture or 3-level architecture.
- It is used to describe the structure of a specific database system.
- It is also used to separate the user applications and physical database.
- It contains 3-levels. It breaks the database down into three different categories.
    - **Internal Level**: Actual PHYSICAL storage structure and access paths.
    - **Conceptual or Logical Level**: Structure and constraints for the entire database
    - **External or View level**: Describes various user views


3-Schema Architecture
External Schema
External Level
External Level
External / Conceptual Mapping
Conceptual Schema
Conceptual Level
Conceptual / Internal Mapping
Internal Schema
Internal Level
Database

# 3-Schema Architecture

1. **Internal level/Schema** : It defines the physical storage structure of the database. It is a very low-level representation of the entire database. It contains multiple occurrences of multiple types of internal record. In the ANSI term, it is also called "stored record".
   **Facts about Internal schema**:
   - The internal schema is the lowest level of data abstraction
   - It helps you to keeps information about the actual representation of the entire database. Like the actual storage of the data on the disk in the form of records
   - The internal view tells us what data is stored in the database and how
   - It never deals with the physical devices. Instead, internal schema views a physical device as a collection of physical pages

# 3-Schema Architecture

1. **Conceptual Schema/Level** : It describes the Database structure of the whole database for the community of users.
   It hides information about the physical storage structures and focuses on describing data types, entities, relationships, etc.
   This logical level comes between the user level and physical storage view. However, there is only single conceptual view of a single database.

   **Facts about Conceptual schema:**

   - Defines all database entities, their attributes, and their relationships
   - Security and integrity information
   - In the conceptual level, the data available to a user must be contained in or derivable from the physical level

# 3-Schema Architecture

**Facts about External schema:**

1. **External Schema/Level** : It describes the part of the database which specific user is interested in & hides the unrelated details of the database from the user.

   There may be "n" number of external views for each database.

   Each external view is defined using an external schema, which consists of definitions of various types of external record of that specific view.

   An external view is just the content of the database as it is seen by some specific particular user.

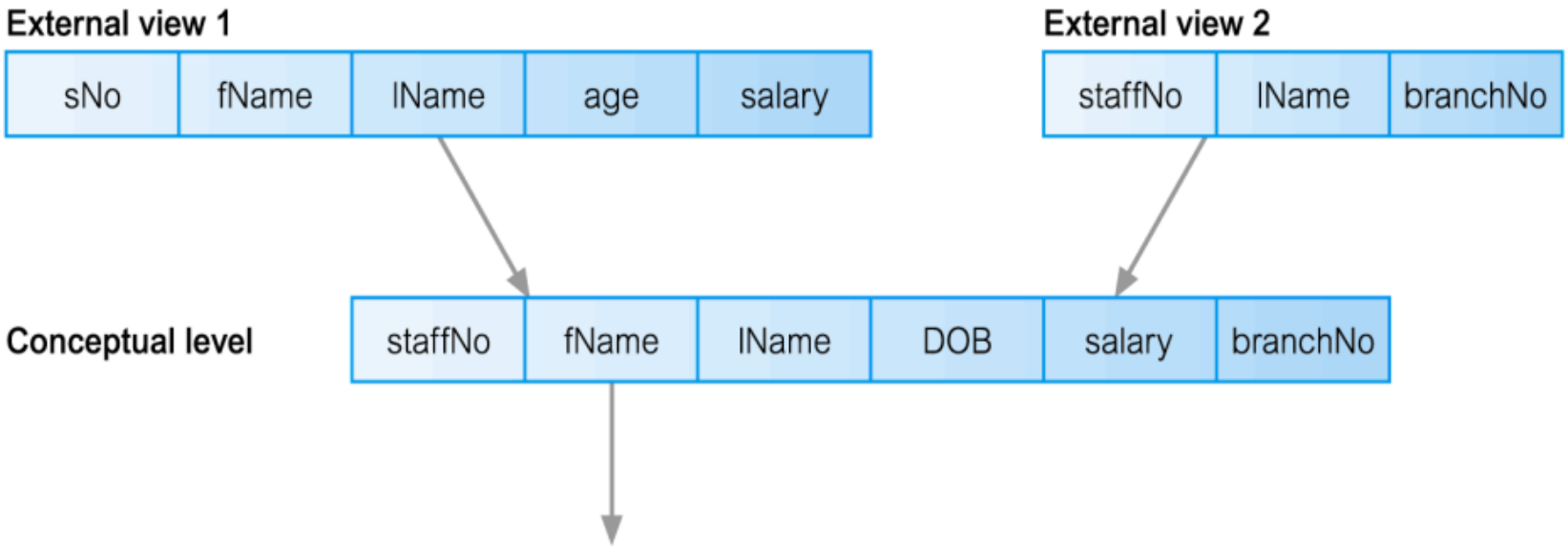
   **For example,** a user from the sales department will see only sales related data.

# Goals of 3-Schema Architecture

Here, are some Objectives of using Three schema Architecture:

- An external level is only related to the data which is viewed by specific end users.
- This level includes some external schemas.
- External schema level is nearest to the user
- The external schema describes the segment of the database which is needed for a certain user group and hides the remaining details from the database from the specific user group

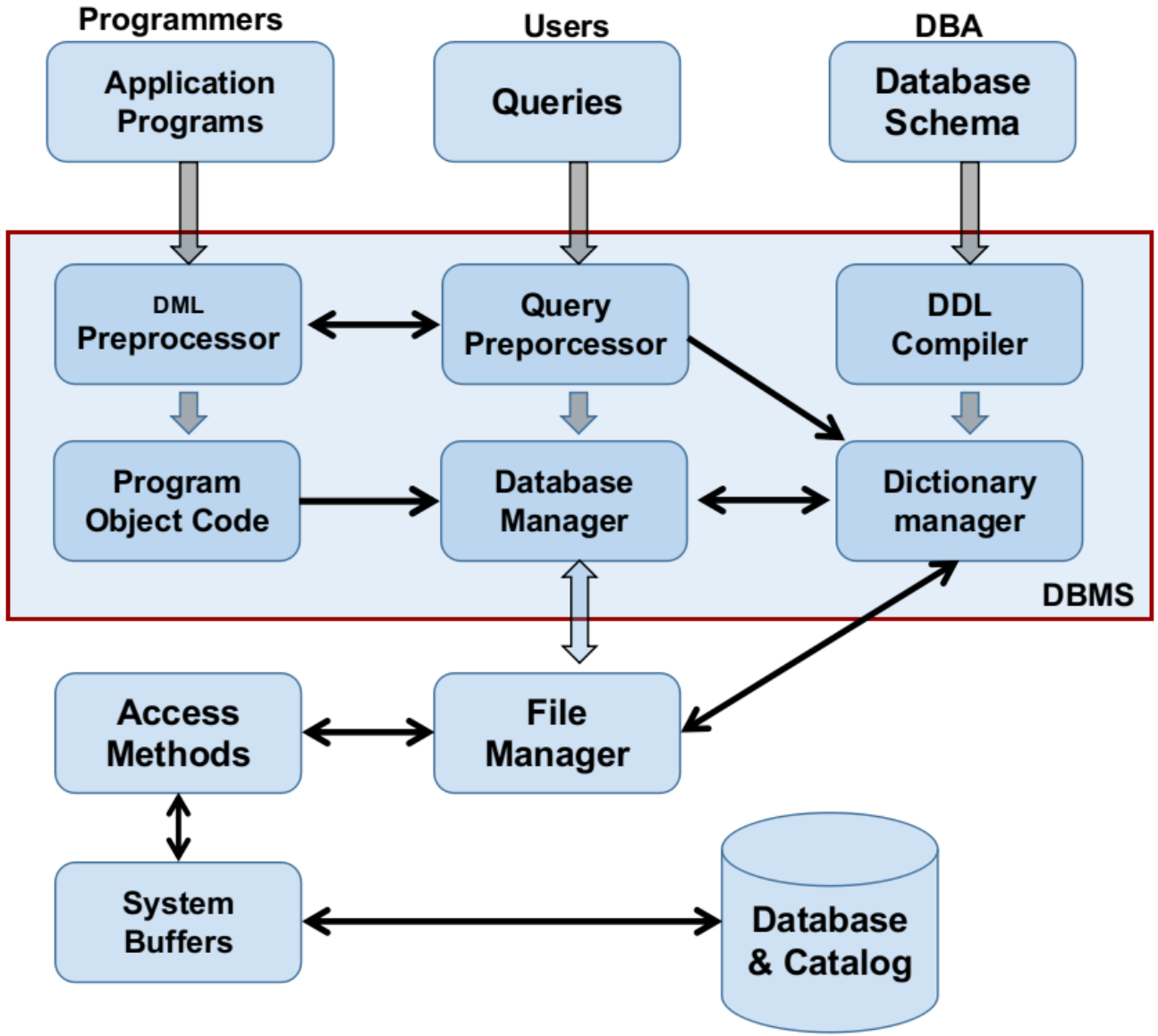
# Examples - Schema Architecture




**Internal level**

```
struct STAFF {
        int staffNo;
        int branchNo;
        char fName [15];
        char lName [15];
        struct date dateOfBirth;
        float salary;
        struct STAFF *next;                 /* pointer to next Staff record */
};
index staffNo; index branchNo;              /* define indexes for staff */
```

# DBMS Components



# DBMS Components

- DML : DML processor must interact with the query processor to generate the appropriate code
- DDL interacts with Data Dictionary/ System Catalog
- System Catalog : It is a collection of tables and views that contain important information about a database. It is available for each database. It defines the structure of the database.
  For example, the DDL (data dictionary language) for all tables in the database is stored in the system catalog.
- Query processor : It transforms user queries into a series of low level instructions. It is used to interpret the online user's query and convert it into an efficient series of operations in a form capable of being sent to the run time data manager for execution. It uses the data dictionary to find the structure of the relevant portion of the database and uses this information in modifying the query and preparing and optimal plan to access the database

# DBMS Components

- Data Dictionary : It contains all the information about the database. As the name suggests, it is the dictionary of all the data items. It contains description of all the tables, view, materialized views, constraints, indexes, triggers etc.
-

# DBMS - Data Models

It define how the logical structure of a database is modeled.

They are fundamental entities to introduce abstraction in a DBMS.

They define how data is connected to each other and how they are processed and stored inside the system.

The very first data model could be flat data-models, where all the data used are to be kept in the same plane.

Earlier data models were not so scientific, hence they were prone to introduce lots of duplication and update anomalies

# Categories of Data Models

- **Conceptual (high-level, semantic) data models**: Provide concepts that are close to the way many users perceive data. (Also called entity-based or object-based data models.)
- **Physical (low-level, internal) data models**: Provide concepts that describe details of how data is stored in the computer.
- **Implementation (representational) data models**: Provide concepts that fall between the above two, balancing user views with some computer storage details.

# Data Independence

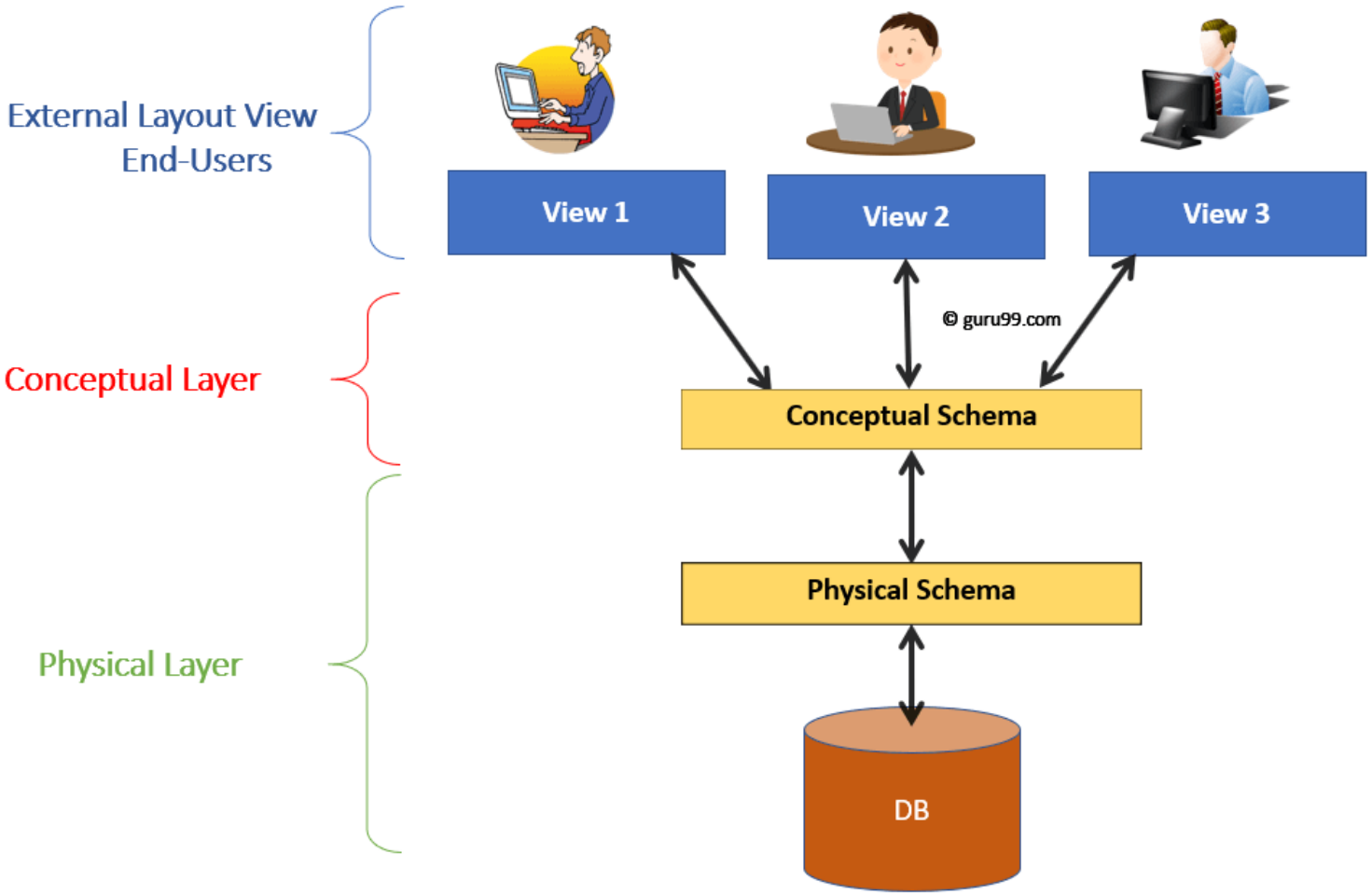
Data Independence is defined as a property of DBMS that helps you to change the Database schema at one level of a database system without requiring to change the schema at the next higher level.

Data independence helps you to keep data separated from all programs that make use of it.

**Types of Data Independence**

- Physical Data Independence
- Logical Data Independence

# Levels of Database



# Levels of Database

Consider an Example of a University Database. At the different levels this is how the implementation will look like:

| Type of Schema | Implementation |
| --- | --- |
| External Schema | **View 1**: Course info(cid:int,cname:string)<br>**View 2**: studeninfo(id:int. name:string) |
| Conceptual Shema | `Students(id: int, name: string, login: string, age: integer)`<br>`Courses(id: int, cname.string, credits:integer)`<br>`Enrolled(id: int, grade:string)` |
| Physical Schema | • Relations stored as unordered files.<br>• Index on the first column of Students. |

# Physical Data Independence

**Physical Data Independence :** Change internal schema without having to change conceptual schema.

i.e Can easily change the physical storage structures or devices with an effect on the conceptual schema.

It helps you to separate conceptual levels from the Internal/Physical levels.

It allows you to provide a logical description of the database without the need to specify physical structures.

It is achieved by the presence of the internal level of the database and then the transformation from the conceptual level of the database to the internal level.

# Example of Physical Data Independence

Due to Physical independence, any of the below change will not affect the conceptual layer.

- Using a new storage device like Hard Drive or Magnetic Tapes
- Modifying the file organization technique in the Database
- Switching to different data structures.
- Changing the access method. Modifying indexes.
- Changes to compression techniques or hashing algorithms.
- Change of Location of Database from say C drive to D Drive

# Logical Data Independence

**Logical Data Independence** is the ability to change the conceptual scheme without changing :

- **External views**
- **External API or programs**

Any change made will be absorbed by the mapping between external and conceptual levels.

**When compared to Physical Data independence, it is challenging to achieve logical data independence.**

Due to Logical independence, any of the below change will not affect the external layer.

- Add/Modify/Delete a new attribute, entity or relationship is possible without a rewrite of existing application programs
- Merging two records into one
- Breaking an existing record into two or more records

# Importance of Data Independence

- Helps you to improve the quality of the data
- Database system maintenance becomes affordable
- Enforcement of standards and improvement in database security
- You don't need to alter data structure in application programs
- Permit developers to focus on the general structure of the Database rather than worrying about the internal implementation
- It allows you to improve state which is undamaged or undivided
- Database incongruity is vastly reduced.
- Easily make modifications in the physical level is needed to improve the performance of the system.

# Physical vs. Logical Data Independence

| Logical Data Independence | Physical Data Independence |
|---|---|
| It is mainly concerned with the structure or changing the data definition. | Mainly concerned with the storage of the data |
| It is difficult as the retrieving of data is mainly dependent on the logical structure of data. | It is easy to retrieve. |
| You need to make changes in the Application program if new fields are added or deleted from the database. | A change in the physical level usually does not need change at the Application program level. |
| Compared to Logic Physical independence it is difficult to achieve logical data independence. | Compared to Logical Independence it is easy to achieve physical data independence. |

# Physical vs. Logical Data Independence

Cont....!

| Logical Data Independence | Physical Data Independence |
|---|---|
| Modification at the logical levels is significant whenever the logical structures of the database are changed. | Modifications made at the internal levels may or may not be needed to improve the performance of the structure. |
| Concerned with conceptual schema | Concerned with internal schema |
| Example: Add/Modify/Delete a new attribute | Example: change in compression techniques, hashing algorithms, storage devices, etc |

# Summary of Data Independence

- Data Independence is the property of DBMS that helps you to change the Database schema at one level of a database system without requiring to change the schema at the next higher level.
- Two levels of data independence are 1) Physical and 2) Logical
- Physical data independence helps you to separate conceptual levels from the internal/physical levels
- Logical Data Independence is the ability to change the conceptual scheme without changing
- When compared to Physical Data independence, it is challenging to achieve logical data independence
- Data Independence Helps you to improve the quality of the data

# Historical Development and Database Technology

- **Early Database Applications:** Hierarchical and Network Models (in mid 1960's).
- **Relational Model based Systems:** Researched and experimented with in IBM and the universities (in 1970).
- **Object-oriented applications:** OODBMSs (in late 1980's and early 1990's )
- **Data on the Web and E-commerce Applications**: using new standards like XML (eXtended Markup Language).

# Schema, Instance and State

- **Database Schema**: The description of a database.
- **Schema Diagram**: A diagrammatic display of (some aspects of) a database schema.
- **Schema Construct**: A component of the schema or an object within the schema, e.g., **STUDENT, COURSE**.
- **Database Instance**: The actual data stored in a database at a particular moment in time. Also called database state (or occurrence).
- **Database State:** Refers to the content of a database at a moment in time.
- **Initial Database State**: Refers to the database when it is loaded
- **Valid State**: A state that satisfies the structure and constraints of the database.
- **Distinction**
  - The database schema changes very infrequently.
  - The database state changes every time the database is updated.
  - Schema is also called **Intension**, whereas state is called **Extension**

# DBMS Languages

- **Data Definition Language (DDL)** allows the DBA or user to describe and name entities, attributes, and relationships required for the application plus any associated integrity and security constraints
- **Data Manipulation Language (DML)** provides basic data manipulation operations on data held in the database
- **Data Control Language (DCL)** defines activities that are not in the categories of those for the DDL and DML, such as granting privileges to users, and defining when proposed changes to a databases should be irrevocably made.
- **Low Level or Procedural DML**: allow user to tell system exactly how to manipulate data (e.g., Network and hierarchical DMLs)
- **High Level or Non-procedural DML**(declarative language): allow user to state what data is needed rather than how it is to be retrieved
(e.g., SQL, QBE)

# DBMS Data Model Classification

- **Based on the data model used:**
  - Traditional: Relational, Network, Hierarchical.
  - Emerging: Object-oriented,Object-relational.
- **Other classifications:**
  - Single-user (typically used with micro- computers) vs. multi- user (most DBMSs).
  - Centralized (uses a single computer with one database) vs. distributed (uses multiple computers, multiple databases)

# History of DataModels

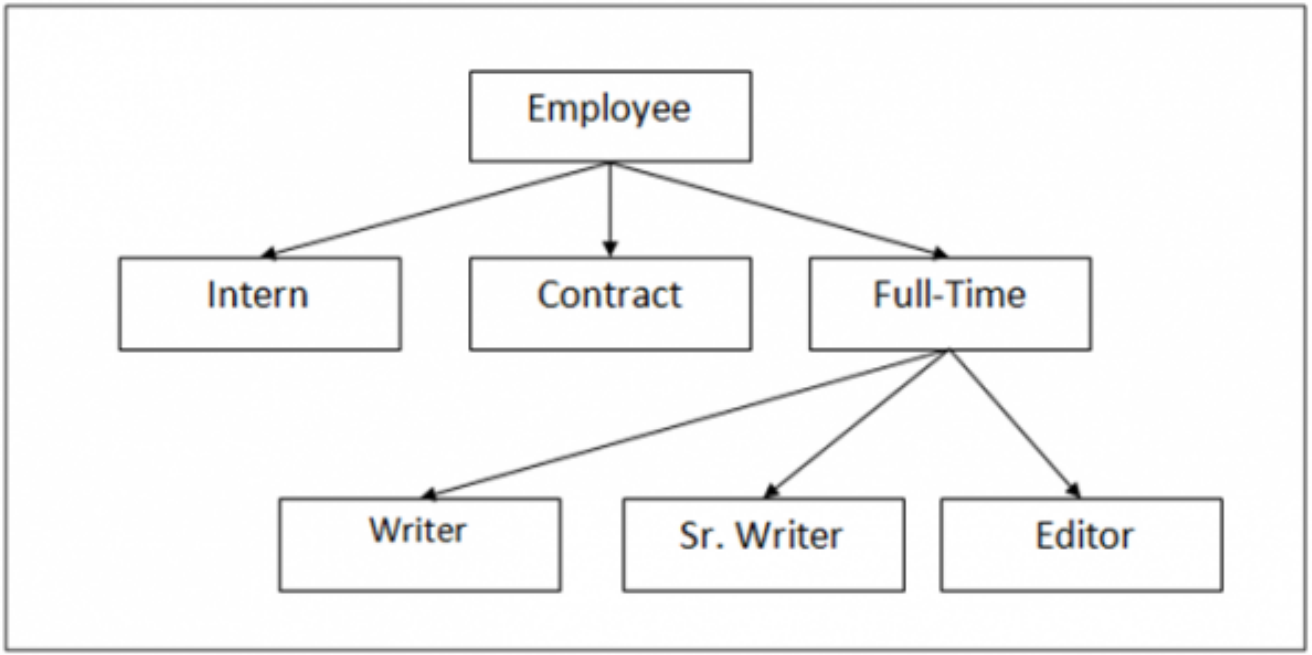
- **Relational Model:** proposed in 1970 by E.F. Codd (IBM), first commercial system in 1981-82. Now in several commercial products (DB2, ORACLE, SQL Server, SYBASE, INFORMIX).
- **Network Model:** the first one to be implemented by Honeywell in 1964-65 (IDS System). Adopted heavily due to the support by CODASYL (CODASYL - DBTG report of 1971). Later implemented in a large variety of systems - IDMS (Cullinet - now CA), DMS 1100 (Unisys), IMAGE (H.P.), VAX -DBMS (Digital Equipment Corp.).
- **Hierarchical Data Model:** implemented in a joint effort by IBM and North American Rockwell around 1965. Resulted in the IMS family of systems. The most popular model. Other system based on this model: System 2k (SAS inc.)

# History of DataModels

- **Object-oriented Data Model(s):** several models have been proposed for implementing in a database system. One set comprises models of persistent O-O Programming Languages such as C++ (e.g., in OBJECTSTORE or VERSANT), and Smalltalk (e.g., in GEMSTONE).
  Additionally, systems like O 2, ORION (at MCC - then ITASCA), IRIS (at H.P.- used in Open OODB).
- **Object-Relational Models:** Most Recent Trend. Started with Informix Universal Server. Exemplified in the latest versions of Oracle-10i, DB2, and SQL Server etc. systems.

# Hierarchical Data Model

- In Hierarchical Model, a hierarchical relation is formed by collection of relations and forms a tree-like structure.
- The relationship can be defined in the form of parent child type.
- One of the first and most popular Hierarchical Model is Information Management System (IMS), developed by IBM

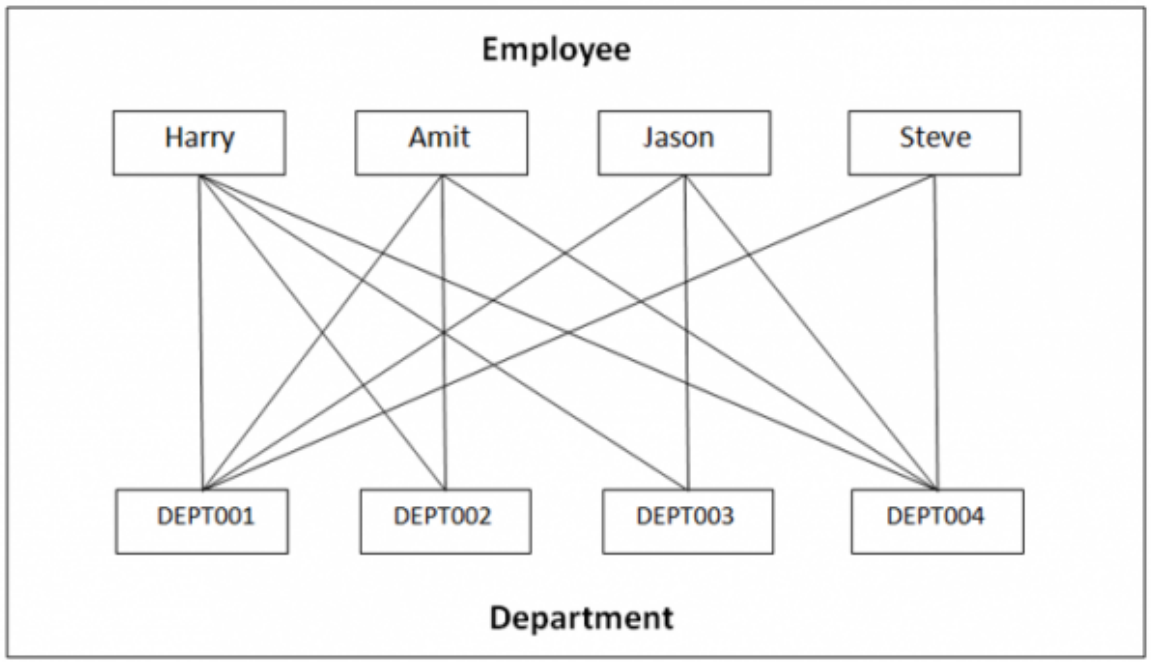


# Merits and De-Merits of Hierarchical Data Model

- **Merits**
  - The design of the hierarchical model is simple.
  - Provides Data Integrity since it is based on parent/ child relationship
  - Data sharing is feasible since the data is stored in a single database.
  - Even for large volumes of data, this model works perfectly.
- **De-Merits**
  - Implementation is complex.
  - This model has to deal with anomalies like Insert, Update and Delete.
  - Maintenance is difficult since changes done in the database may want you to do changes in the entire database structure.

# Network Data Model

- The Hierarchical Model creates hierarchical tree with parent/ child relationship, whereas the Network Model has graph and links.
- The relationship can be defined in the form of links and it handles many-to-many relations. This itself states that a record can have more than one parent.

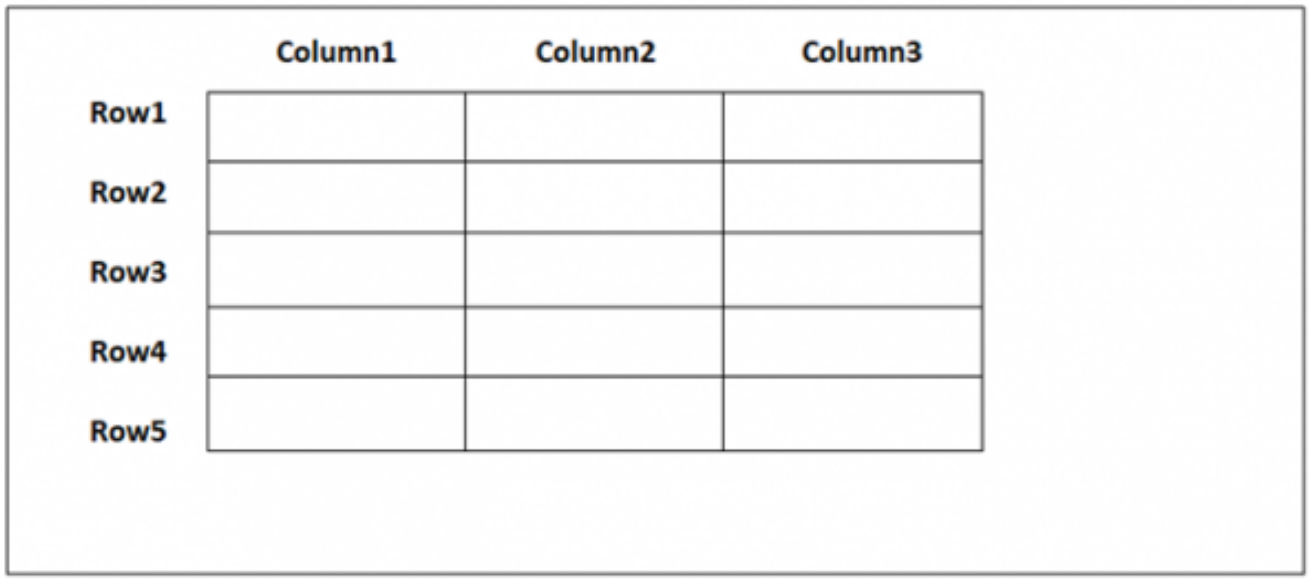


# Merits and De-Merits of Network Model

- **Merits** :
  - Easy to design the Network Model
  - The model can handle one-one, one-to-many, many-to-many relationships.
  - It isolates the program from other details.
  - Based on standards and conventions.
- **De-Merits** :
  - Pointers bring complexity since the records are based on pointers and graphs.
  - Changes in the database isn't easy that makes it hard to achieve structural independence.

# Relational Data Model

- A relational model groups data into one or more tables. These tables are related to each other using common records.
- The data is represented in the form of rows and columns i.e. tables:



# Relational Data Model

- **Example** : Let us see an example of two relations <**Employee**> and <**Department**> linked to each other, with DepartmentID, which is **Foreign Key** of <**Employee**> table and Primary key of <Department> table.

<Employee>

| EmployeeID | EmpName | DepartmentID |
|---|---|---|
| E09 | Emily | D001 |
| E04 | Tom | D002 |
| E11 | Emma | D003 |

<Department>

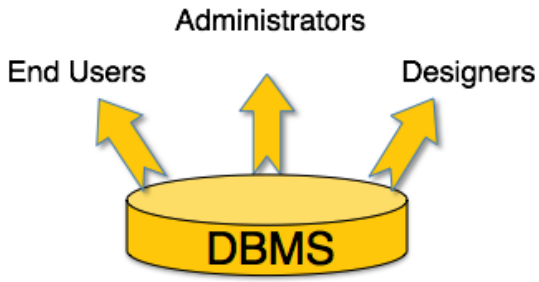
| DepartmentID | DeptName | DeptZone |
|---|---|---|
| D001 | Finance | North |
| D002 | Operations | East |
| D003 | Marketing | West |

# Merits and De-Merits of Relational Model

- **Merits**:
  - It does not have any issues that we saw in the previous two models i.e. update, insert and delete anomalies have nothing to do in this model.
  - Changes in the database do not require you to affect the complete database.
  - Implementation of a Relational Model is easy.
  - To maintain a Relational Model is not a tiresome task.
- **De-Merits** :
  - Database inefficiencies hide and arise when the model has large volumes of data.
  - The overheads of using relational data model come with the cost of using powerful hardware and devices.

# Roles in DBMS

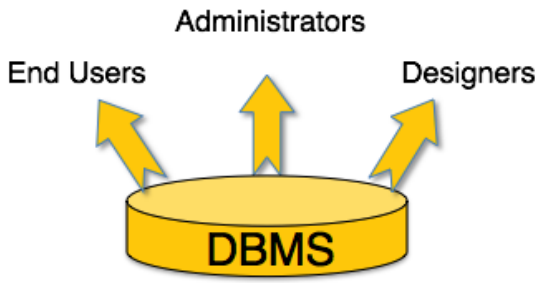
DBMS has users with different rights and permissions who use it for different purposes. Some users retrieve data and some back it up. The users of a DBMS can be broadly categorized as follows :




- **Administrators** : One who maintains the DBMS and are responsibility's are :
  - Administrating the Database
  - To look after its usage and by whom it should be used and Create access profiles for users.
  - Apply limitations to maintain isolation and force security.
  - System license, required tools, and Other software and hardware related maintenance

# Roles in DBMS

DBMS has users with different rights and permissions who use it for different purposes. Some users retrieve data and some back it up. The users of a DBMS can be broadly categorized as follows :




- **Designers** : are the group of people who actually work on the designing part of the database. They keep a close watch on **what data should be kept and in what format**. They identify and design the whole set of entities, relations, constraints, and views.
- **Users** : End users are those who actually reap the benefits of having a DBMS. End users can range from simple viewers who pay attention to the logs or market rates to sophisticated users such as business analysts.

# DBMS Interface

- Stand-alone query language interfaces.
- Programmer interfaces for embedding DML in programming languages:
  - Pre-compiler Approach
  - Procedure (Subroutine) Call Approach
- User-friendly interfaces:
  - Menu-based, popular for browsing on the web
  - Forms-based, designed for users
  - Graphics-based (Point and Click, Drag and Drop etc.)
  - Natural language: requests in written English
  - Combinations of the above
- Interfaces for the DBA:
  - Creating accounts, granting authorizations
  - Setting system parameters
  - Changing schema's or access path

# Database System Utilities/Tools

To perform certain functions such as:

- Loading data stored in files into a database. Includes data conversion tools.
- Backing up the database periodically on tape.
- Reorganizing database file structures.
- Report generation utilities.
- Performance monitoring utilities.
- Other functions, such as sorting, user monitoring, data compression, etc.

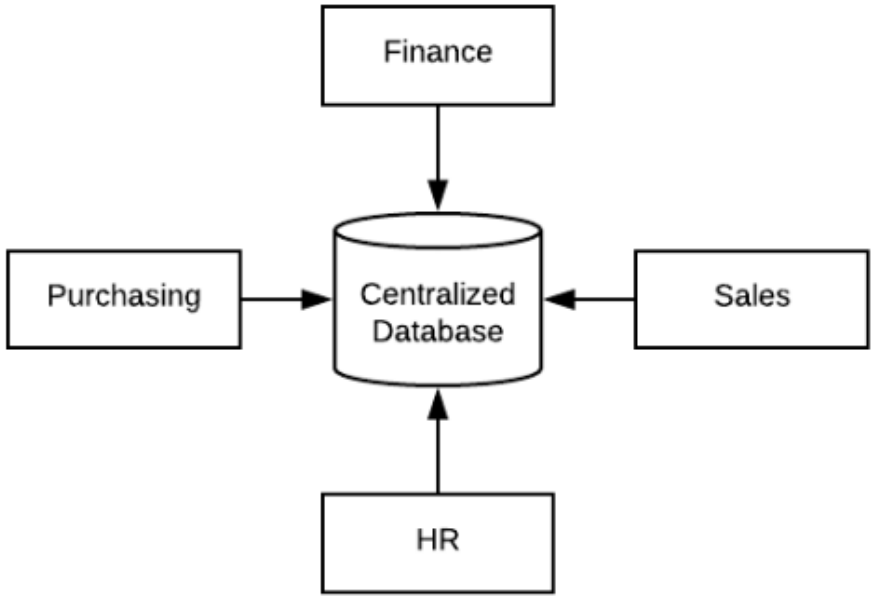
# Database System Utilities/Tools

- **Data dictionary / repository**:
  - Used to store schema descriptions and other information such as design decisions, application program descriptions, user information, usage standards, etc.
  - Active data dictionary is accessed by DBMS software and users/DBA.
  - Passive data dictionary is accessed by users/DBA only

- Application Development Environments and CASE (Computer-Aided Software Engineering) tools:
  **Examples** – Power builder (Sybase), Builder (Borland)

# Centralized Database Management System

A centralized database is stored at a single location such as a mainframe computer. It is maintained and modified from that location only and usually accessed using an internet connection such as a LAN or WAN. The centralized database is used by organisations such as colleges, companies, banks etc.



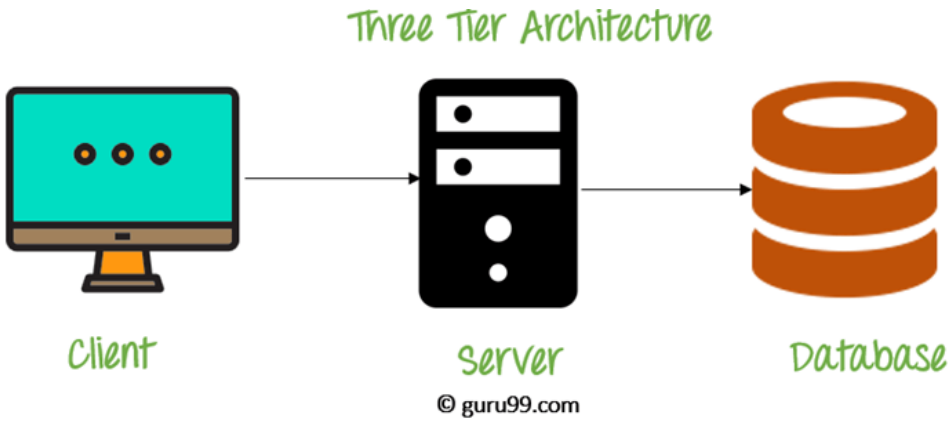
# Merits and De-merits of Centralized DBMS

- **Advantages** :
  - The **Data Integrity** is maximized as the whole database is stored at a single physical location. It is easier to coordinate the data and it is as accurate and consistent as possible.
  - The **Data Redundancy** is minimal in the centralized database. All the data is stored together and not scattered across different locations. So, there is no redundant data available.
  - Since all the data is in one place, there can be stronger security measures around it. So, It is much more secure.
  - Data is easily portable because it is stored at the same place.
  - It is cheaper than other types of databases as it requires less power and maintenance.
  - All the information can be easily accessed from the same location and at the same time.

# Merits and De-merits of Centralized DBMS

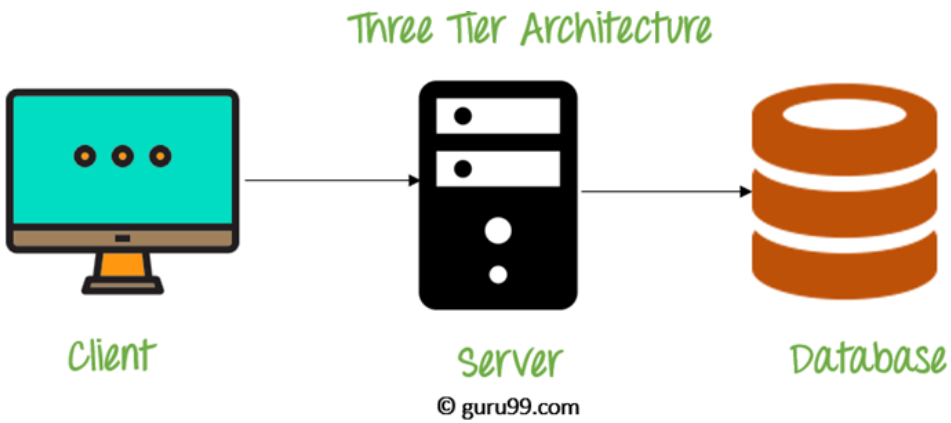
- **Disadvantages** :
  - Since all the data is at one location, it takes more time to search and access it. If the network is slow, this process takes even more time.
  - There is a lot of data access traffic for the centralized database. This may create a bottleneck situation.
  - Since all the data is at the same location, if multiple users try to access it simultaneously it creates a problem. This may reduce the efficiency of the system.
  - If there are no database recovery measures in place and a system failure occurs, then all the data in the database will be destroyed.

# Client-Server Architecture in DBBMS




- The client–server model is a distributed application structure that partitions tasks or workloads between the providers of a resource or service, called servers, and service requesters, called clients.
- Clients and Servers communicate over a computer network on separate hardware, but both client and server may reside in the same system.
- A server host runs one or more server programs which share their resources with clients.

# Client-Server Architecture in DBBMS




- A client does not share any of its resources, but requests a server's content or service function.
- Clients therefore initiate communication sessions with servers which await incoming requests.
- Examples of computer applications that use the client–server model are Email, network printing, and the World Wide Web.

# Merits and De-merits of Client-Server

- **Merits** :
  - **Centralization** – Access, Resources, and Data Security are controlled through server
  - **Scalability** – Any element can be upgraded when needed
  - **Flexibiltiy** – New Technology can be easily integrated into the system
  - **Interoperabilty** – All components work together.
- **DeMerits** :
  - **Dependability** – When Servers goes down, operations will cease
  - **Lack of Mature Tools** - To administrate
  - **Lack of Scalability** – Network OS are not vary scalable.
  - **Higher than anticipated Cost**.
  - **Network Congestion**.

# Thanks