Unit – 1: Database Management System

Table of Contents

Characteristics of DBMS 2
- Data Redundancy 2
- Data Consistency 3
- Concurrent Access 3
- Query Language 3
- Security 3
- Transaction Management 3

Data Abstraction 3
- Physical Level 4
- Logical Level 4
- View Level 4

Data Independence 5
- Physical Data Independence 5
- Logical Data Independence 6

Architecture of DBMS 6
- 1 tire architecture 7
- 2 tire architecture 7
- 3 tire architecture 8

Types of DBMS User 10
- DBA (Database Administrator) 10
- End Users (Naive users) 10
- Application Programmers 11
- Sophisticated Users 11

Roles of Database Administrator (DBA) 11

Schema Design 11
Set Data Access 12
Assisting Application Programmer 12
Monitor Performance 12
Backup and Recovery 12
What is Data model? 12
Type of Data models 13
Hierarchical model 13
Network Model 13
Relational model 14
Entity Relationship Model 15
Object Oriented Database model 15
NoSQL Database model 16

Characteristics of DBMS

- Data bases Stores data into tables contains rows and columns.
- Various tables are tied together with relationship to make data meaningful.
- All Database Management Systems have some characteristics to improve performance, deal with issues like security, validation of data etc.

Data Redundancy

- Reduce Data Redundancy
- All DBMS provides some mechanism for data redundancy.
- Allows to reduce duplicate data.
- When duplicate data are stored, it may cause various problems like need more hard-disk space, data become inconsistent etc.

Data Consistency

- Data consistency is very important in every DBMS.
- In modern days data is accessed and modified so many times, hence it is important that data is in correct form.
- Most DBMS have some mechanism which can be used for data consistency.

Concurrent Access

- Allows multiple users to use database and also maintain data consistency.

Query Language

- Provides some sort of query language which can be used to read, write and modify data stored in the database.
- Most DBMS system supports query language known as SQL (Standard Query Language).
- SQL is standard query language which is used by most RDMBS.

Security

- DBMS also provides a way to manage security of data.
- We can create different types of users so each type of user only has particular access to data.
- We can also control the read/write access to the data.

Transaction Management

- Allow to better manage data integrity by using transaction management

Data Abstraction

- All Database Management System uses complex data structures to store and retrieve data.

- To make things easier for user, DBMS provides data abstraction, means it provides a way to separate of data accessed by user and how data is actually handled by the DBMS on the file system.
- This concept is known as data abstraction.

There are 3 levels of data abstraction

Physical Level

- This is lowest level of data abstraction. This actually represents how the data is actually stored on physical medium like hard disk.
- DBMS uses different data structures like stacks, queues, arrays, tree, graphs etc. to store the data.

Logical Level

- It is middle level (higher level compared to physical level)
- It describes what data is stored in the database and how different relationship between tables.

View Level

- It is highest level of abstraction.
- Also closest to user.
- Shows the relevant data inform of tables and views.
- Different user can have different view of database according to the rights given to each user.

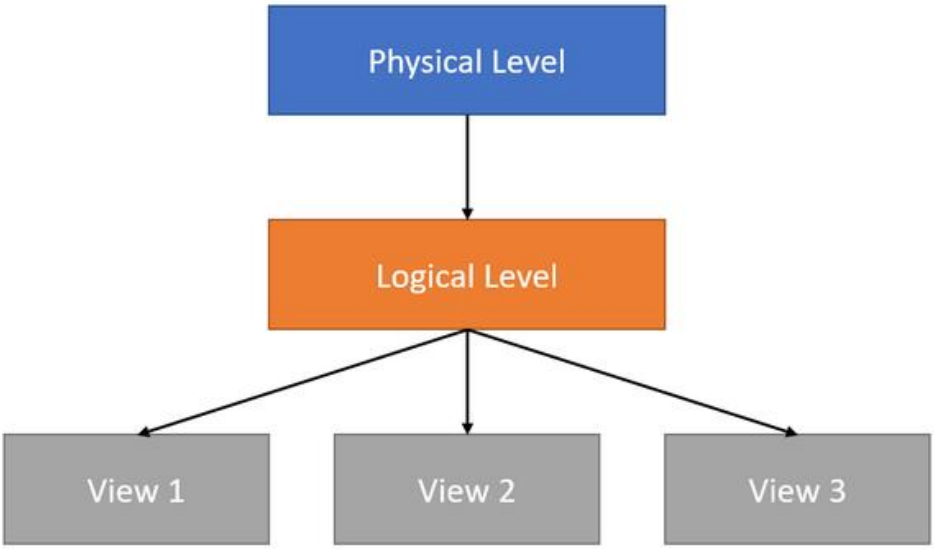

Fig. 1 - Levels of Data Abstraction (Hierarchy of Data Abstraction Levels)

Data Independence

Main purpose of Data Abstraction is to achieve data independence to save time and cost required when data is modified or altered.

Data Independence can be divided into two categories.

- Physical Data Independence
- Logical Data Independence

Physical Data Independence

- Physical data independence is to modify or alter the physical schema without any alteration of logical schema.
- The term 'Schema' refers to how data is structured in the database. We can say it is the blueprint of the data.

Below are some examples of Physical Data Independence where we change something at physical layer which should not affect the logical schema.

- Adding new storage devices to store more data.
- Modify the data structures used to store the data.

- Change of operating system
- Change of storage devices

Logical Data Independence

- It refers to modifying logical structure of database without affecting external schema or application using the database.
- The view of the data should not be affected by changes of database logical structure.
- It is little hard to implement as when logical structure change, many times view is also gets affected.

Below are some examples of Logical Data Independence where we change something at logical layer which shouldn't affect the view layer.

- Altering table structures
- Modifying columns datatypes
- Change relationship among tables.

Architecture of DBMS

- When designing a modern application, chances are that you will need a database to store data.
- There are many ways to architect software solutions that use a database
- Database architecture describes how a database management system (DBMS) will be integrated with your application.
- When designing application first decide which database and which type of database you want to use.
- Once you've decided the type of database you want to use, you can determine the type of architecture you want to use. Typically, these are categorized into single-tier or multi-tier applications.
- Generally, there are 3 types of architecture used in projects.
    - 1 tire architecture
    - 2 tire architecture
    - 3 tire architecture

1 tire architecture

- In 1-tier architecture, the database and any application interfacing with the database are kept on a single server or device
- Because there are no network delays involved, this is generally a fast way to access data.
- E.g., if some mobile application is using some database which is present in the mobile itself then this type of architecture, we can say that it is 1 tire architecture.

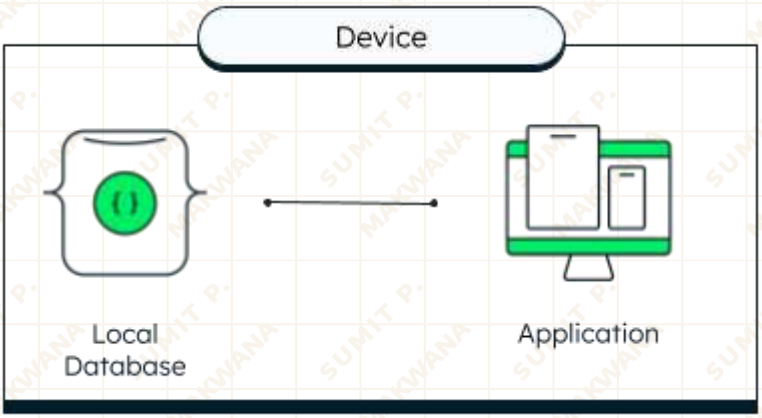

Fig 2.1 – 1 Tire Architecture

Image source - https://www.mongodb.com/basics/database-architecture

2 tire architecture

- 2-tier architectures consist of multiple clients connecting directly to the database. This architecture is also known as client-server architecture.
- This is mostly used with desktop application, where application is installed on multiple computers and is connected to a single server where database is hosted to access data.

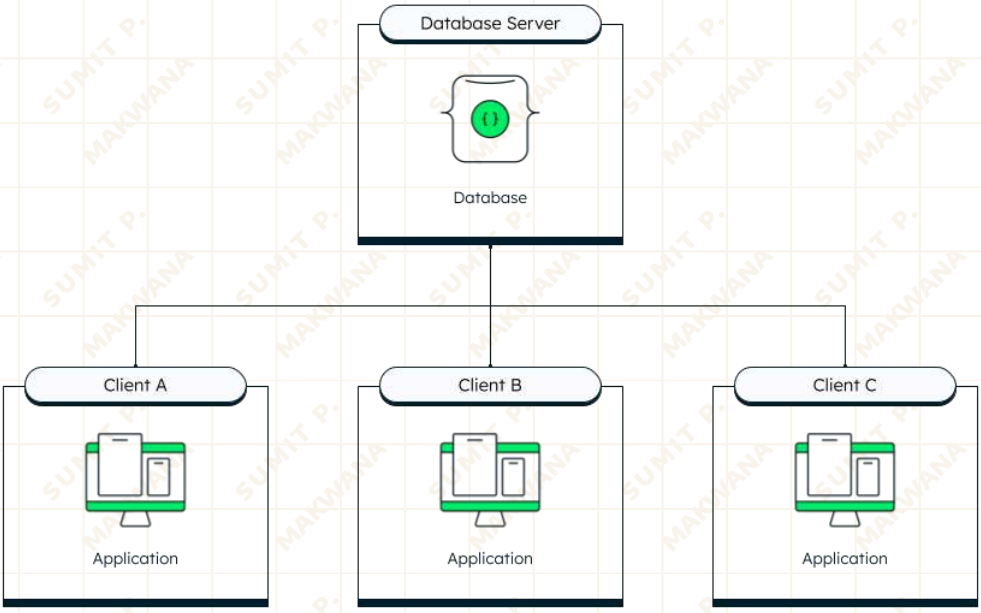

Fig 2.2 – 2 Tire Architecture

Image source - https://www.mongodb.com/basics/database-architecture

3 tire architecture

- Most modern application uses this type of architecture.
- In this architecture, the clients connect to a back end, which in turn connects to the database.
- Hence in this architecture application never directly connects to the database. Application will connect to back-end server and then back-end server.
- This type of approach has many benefits.
- It leads to fast development of application which can run on different platform like mobile app, web app, desktop app etc.
- Scalability – Because each layer operate independently it is easier to scale part of the application.

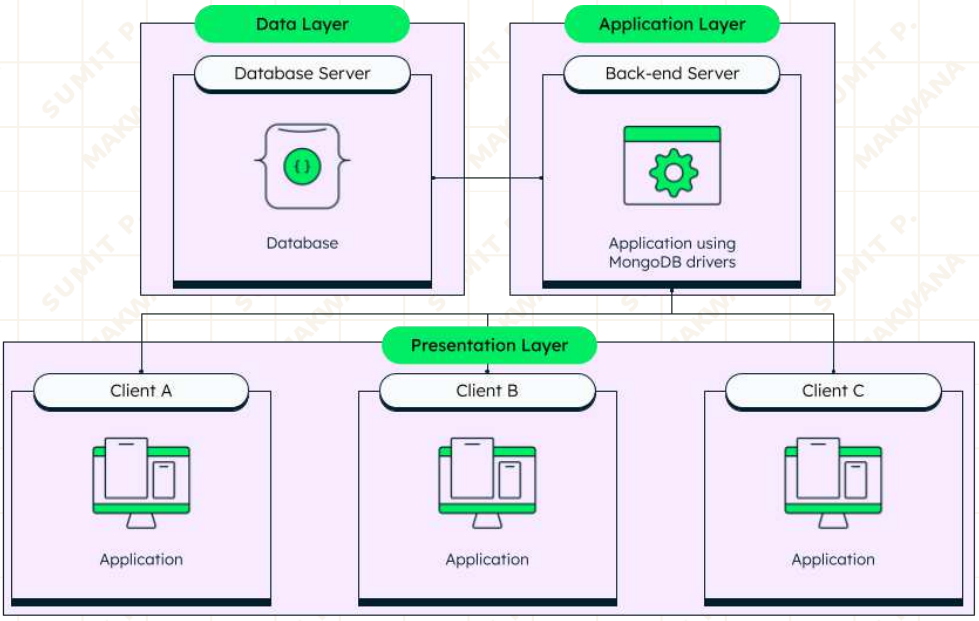

Fig 2.3 – 3 Tire Architecture

Image source - https://www.mongodb.com/basics/database-architecture

Presentation Tire

- In presentation tire is GUI which is offered to the end user, make up of mostly graphical interface, which decides how data is presented to user.
- It does not contain any business logic; it only deals how the data is represented to user.

Application Tire

- Also known as logic tire or middle tier.
- Information is sent/collected to/from the presentation layer
- Information is processed using business logic (rules of business is applied)
- It communicates with data tire using API calls.

Database Tire

- Data tire or database tire is the tire which as direct access to the database.
- Data tire provides different APIs to manipulate database information.
- These APIs are used by Application tire to access the database.

Types of DBMS User

Database users are the one who uses the database. Depending on the how user uses the database we can categorized users into multiple categories.

- DBA (Database Administrator)
- End Users (Naive users)
- Application Programmers
- Sophisticated Users

DBA (Database Administrator)

- Database Administrator (DBA) is a person/team who defines the schema and creates the database.
- There are various roles of DBA like defining schema of database, monitoring performance, taking care of backup / restore etc.

End Users (Naive users)

- End user is the person who use database but do not have any knowledge about how to operate or use the database.
- Let's say a user wants to transfer some amount from his account to another account. He doesn't need to know programming languages or query languages to do this.
- He is a naive user, so he goes to his Internet banking interface which is a previously written program. All he has to do is, add the account number from which he wants to transfer the money, add the amount of money he wants to transfer and

finally add the account number of the account he wants the money to be transferred to.

- After submission of this information, the program will write the query according to the information provided by the user and execute the query and the money will be transferred.

Application Programmers

- Application programmers are computer professionals who write application programs in various programming languages which use the database.
- Application programmers develops an interface to access that database which is generally used by end users.

Sophisticated Users

- Sophisticated users can be engineers, scientists, business analyst, who are familiar with the database.
- Sophisticated users are users who do not use any programs, instead they write the queries themselves or develop their own application to use database.
- Generally these types of user use database directly writing commands (SQL queries) to access data instead of using any application.
- E.g. Weather Analyst

Roles of Database Administrator (DBA)

Schema Design

- DBA defines the logical schema of the database. Schema refers to the logical structure of database which includes tables' structures, their relationship, views etc.
- Schema is designed according to the requirement of the project / organization for example which data needs to be stored.

Set Data Access

- Creates different types of users and user groups.
- Assign privilege to all users/user groups to set data access for each user.

Assisting Application Programmer

- DBA provides assistance to the application programmer how data can be accessed or manipulated to develop an application.
- Data Access
- Creates different types of users and set data access rights for each user.

Monitor Performance

- DBA checks and monitor the performance of the database, and try to achieve best performance.
- If the performance is subpar then DBA may change logical schema or design to ensure the performance.

Backup and Recovery

- Database is most important part of any application.
- Data stored in the database is most important and must not be lost in any situation.
- DBA periodically takes the back of the database depending on the requirement of the application.
- In case of failure like hardware failure, virus attack etc. DBA restore/recover the database to ensure smooth operation.

What is Data model?

- Database data model defines the logical structure of the database.
- Various data models are used to define the structure of the database when designing a database.
- It defines how the data will be organized and accessed.
- Conceptual tool to describe a database design.
- Shows the logical structure, relationships, constraints which decides how data will be stored and accessed.
- Most data models can be represent using a accompanying database diagram in UML.

Type of Data models

- Hierarchical model
- Network model
- Relational model
- Entity Relationship Model
- Object-oriented database model
- NoSQL model

Hierarchical model

- Organize the data in tree like structure.
- Where each record has single parent or root.
- Siblings are sorted in a particular order.
- Useful to represent real life relationships.
- This model was mostly used in IBM's Information Management System in early 60s and 70s.
- This model is not used much in present day.

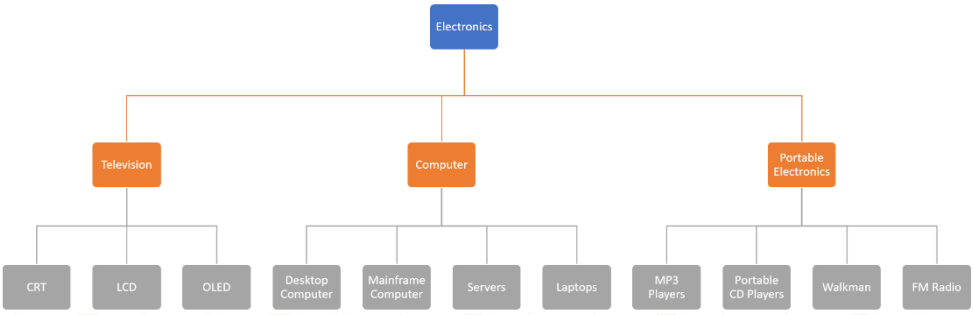

Fig 3 - Hierarchical model

Network Model

- Build on the top of Hierarchical model.
- Allows many to many relationships.
- In Hierarchical model any record only can have a single parent, but in Network model record can have multiple parents.
- Hence it allows complex relationships.

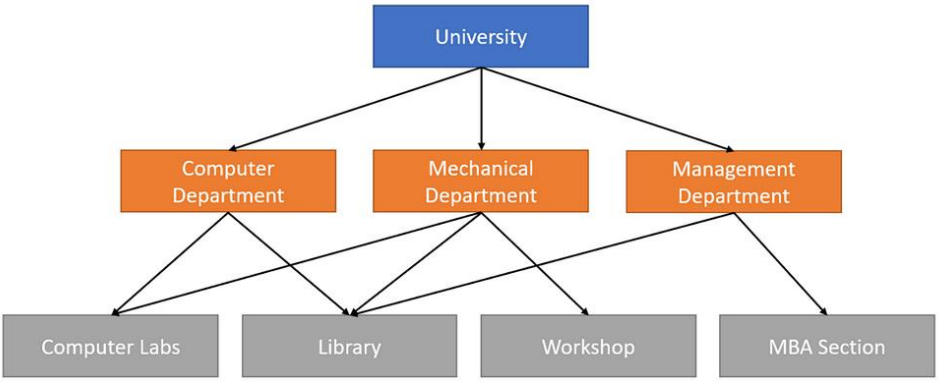

Fig 4 - Network Model

Relational model

- Most common model
- Arrange the data into tables.
- Each tables consists of rows and columns.
- Columns represents attributes.
- E.g. name, address, zip code, country etc.
- Rows represent a record. Rows also known as tuple.
- Tables are joined together using some common column.
- Most used model in current time.
- This type of database generally uses SQL (Structured Query Language) to read/write the data.

StudentNo	Name	Class
1001	Jay	DC1
1002	Ajay	DC2
1003	Vijay	DC3
1004	Ram	DC2

SubjectCo	Subject Name
09CE1001	DBMS
09CE1002	C#.NET
09CE1003	AWT

StudentNo	SubjectCode	Mark
1001	09CE1001	50
1001	09CE1002	60
1002	09CE1001	55
1002	09CE1002	65
1002	09CE1003	40
1003	09CE1003	95
1004	09CE1002	55

Relational Model to store student details, their marks and details of subject.

All tables are joined using common column

Fig 5 - Relational Model

Entity Relationship Model

- In this model, it captures the relationship between object that represents the real-world object known as entity.
- It defines how different entities are related to each other.
- After defining entities, attributes of each entity are defined.
- Attributes represents the data about the entity.
    - E.g. name, enrolment no, salary, address, organization etc.
- Generally relational data model is constructed from the ER Model.
- In ER model entity is represented using rectangle, attributes are represented using ellipse and relationships are represented using Dimond shape.

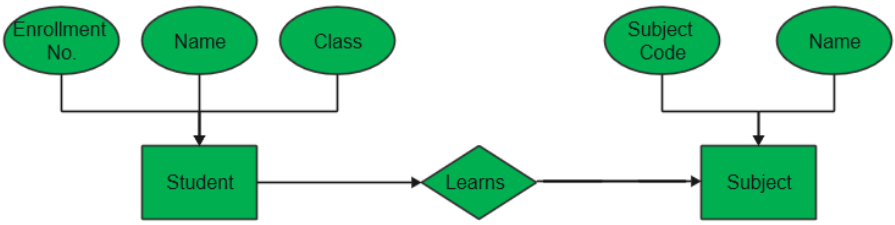

Fig 6 - Entity Relationship Model (E R Model)

Object Oriented Database model

- Object Oriented Database model defines a database as a collection of objects, and reusable component like we use in object-oriented programming.
- This model is used to solve the need of complex data storage.
- In object-oriented data model both data and their relationship are contained in a single structure known as object.
- In Object Oriented Data Model, data and their relationships are contained in a single structure which is referred as object in this data model.
- In this, real world problems are represented as objects with different attributes.
- All objects have multiple relationships between them.

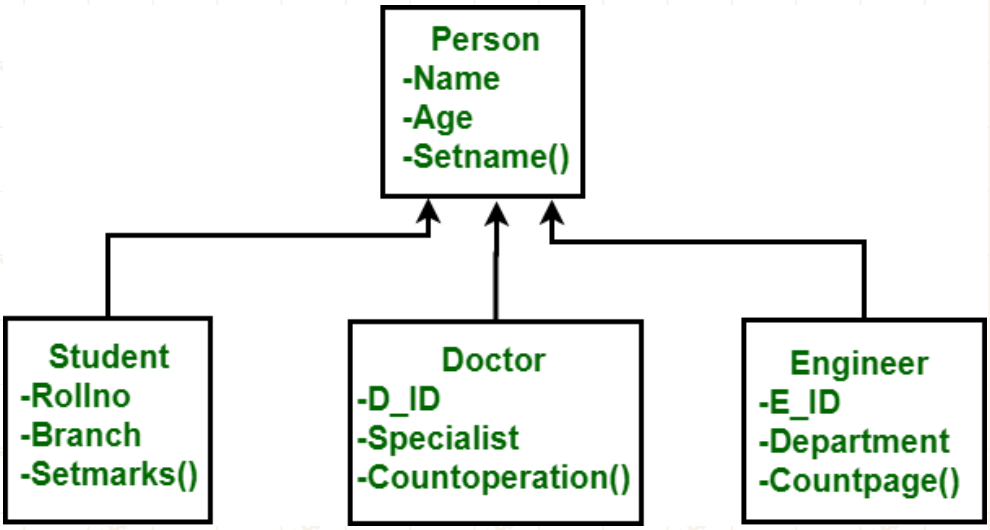

Fig 7 - Object Oriented Data Model
Image Source - https://www.geeksforgeeks.org/basic-object-oriented-data-model

NoSQL Database model

- NoSQL (Also known as not only SQL) database model uses non tabular approach and store data differently compared to relational model.
- NoSQL database model stores the data in many different ways like document, key-value pairs, graphs, wide-column etc.
- It provides flexible schema which can be scaled easily with large amount of data.
- Firebase, MongoDB are examples of NoSQL database model.